Erasmus University Rotterdam

Erasmus School of Economics

Master Thesis Econometrics and Management Science:

programme Business Analytics and Quantataive Marketing

Soft Isolation Forests

Sven Nieuwkerk

Student number: 542549

Abstract

In this thesis, soft splitting is proposed as an enhancement for the isolation-based anomaly detection method Isolation Forest (iForest). It is shown how a loss of information is created by the hard splitting used in that method and the Soft Isolation Forest (SIF) is proposed to address these shortcomings. To perform soft splitting, the relation between the splitting criterion in a node and the distance is set out and the concept soft isolation is defined. The algorithms of iForest are adapted with new hyperparameters and measures to perform soft splitting forming the SIF algorithm. For the evaluation multiple simulated and real world datasets are used. Soft splitting is found to improve predictive performance for datasets with a high number of attributes and with scattered anomalies. Predictive performance on datasets with few attributes and clustered anomalies decreases and computations times increase massively for all datasets.

Supervisor: Eoghan O'Neill

Second assessor: Mikhail Zhelonkin

Date final version: 25 July 2024

The content of this thesis is the sole responsibility of the author and does not reflect the view of the supervisor, second assessor, Erasmus School of Economics or Erasmus University.

Contents

1 Introduction

In anomaly detection the goal is to identify data points that do not follow the expected behaviour. These point are called anomalies or outliers in most research. Anomaly detection has applications in intrusion detection systems, detecting credit-card fraud, medical diagnosis, law enforcement and earth sciences (Aggarwal and Aggarwal, 2017). According to Aggarwal and Aggarwal (2017) a lot of outlier detection algorithms are model-based. This means that a profile of normal instances is created and is then used to identify instances that don't conform to this profile as anomalous. Liu et al. (2008) state that drawbacks of this approach are that these anomaly detectors are optimised to identify normal instances instead of anomalies and are constricted to low dimensional data and small data size because of the higher computational complexity.

Isolation Forest (iForest) is an anomaly detection method introduced by Liu et al. (2008) and differs from most earlier introduced anomaly detection methods by being isolation-based. Instead of building an entire model, isolation-based methods use the fact that anomalies are "few and different" to find the anomalies. In this case few means that anomalies are a small percentage of the total instances and and different means that they have values that are very different from those of normal instances. Using these properties, is it easier for isolation-based methods to isolate anomalies than normal instances, which allows them to detect the anomalies. As this no longer requires building a model for normal instances, isolation-based methods have several advantages over model-based methods. These include linear time complexity, small memory requirement and robustness to noise, irrelevant attributes and data density distribution (Cao et al., 2024). Furthermore, isolation-based methods are unsupervised, meaning that labels are only used to evaluate the model performance and thus not in the building of the model.

iForest determines an anomaly score for every instance based on how easy it is to isolate it using random splits. For this it uses an ensemble of Isolation Trees. In every node of an Isolation Tree, an attribute is selected together with a random splitting point for that attribute. Instances on one side of the splitting point are passed down to the left node, whilst the instances on the other side are passed down to the right node. When a node only consists of one instance that instance is considered isolated and that node is made a terminal node. The anomaly score of the iForest is the average of the path lengths in all the trees in the ensemble. Due to the earlier discussed properties of anomalies, these should be isolated earlier leading to smaller path lengths and are thus differentiable from normal instances.

This thesis proposes to extend the Isolation Forest with soft splitting. Instead of assigning an instance completely to the left or right side of a split, using soft splitting instances are distributed with weights over the nodes after a split. Irsoy et al. (2012) implemented soft splitting in decision trees and showed several advantages of soft splitting. Soft splitting increases the information used in every split. This leads to less complex and thus smaller trees. Furthermore, due to soft splitting the output had a smoother fit around splitting boundaries which leads to smaller biases at those places. These properties would be advantageous if they would also apply to Isolation Forests and could lead to an increased ability to detect anomalies. However, the introduction of soft splitting into a isolation algorithm gives a paradox. Using soft splitting, every instance is in every node with a certain weight, implicating that a node is never really isolated. This highlights a few problems in definition that arise when implementing soft splitting in Isolation Forests. Firstly, the distribution of weights over the nodes after the splits needs to be determined. Then, it needs to be determined when an instance is isolated. As all instances are present in every node, no instance will ever be the only instance in a node and thus trees can keep growing forever without a sufficient stopping criterion. Furthermore, when computing anomaly scores, one instance doesn't just have a single depth in one of the trees of the ensemble anymore. So a new method for calculating how easy it is to isolate a point needs to be studied.

In recent literature, there have been many developments related to Isolation Forests. Hariri et al. (2019) showed Isolation Forests can have biased anomaly scores due to axis-parallel splitting. In some cases this could even lead to ghost clusters, which are groups of point that collectively have a lower anomaly score than the points around them whilst being just as anomalous. It was concluded that Isolation Forests have trouble with more complex data structures. To solve this, the Extended Isolation Forest (EIF) is proposed, which uses random hyperplanes to split the data. At every node a new random slope and intercept vector are drawn for the hyperplane in that split. Using heatmaps of the anomaly score, it was shown that the anomaly scores are no longer biased along the axes.

Lesouple et al. (2021) proposed the Generalized Isolation Forest (GIF), which chooses the intercepts between the minimum and maximum of the projections of the datapoints on the normal vector. This was done to prevent the many empty branches that were created by the EIF, because many of the selected intercepts fell outside of the convex hull of the data point in the node. SCiForest, introduced by Liu et al. (2010), is another Isolation Forest method that uses hyperplanes instead of axis-parallel splitting. This implementation generates multiple hyperplanes in every node and then proceeds to chose the hyperplane that minimises the standard deviation in the children nodes. It is notable that this method also does not use completely random splits, but the split is guided towards splitting between different groups of data. A similar approach was used by Tokovarov and Karczmarek (2022), which extended regular Isolation Forests by replacing random splitting with a probabilistic approach. Instead of using a uniform distribution for the place of the splitting point, splits are made with a higher probability in places where data is more sparse. This again should increase the probability of splitting between different groups of data.

Further improvements to detect anomalies in complex data structures were made by Liu et al. (2024). They introduced a Layered Isolation Forest, where in each layer an Isolation Forest is made on a smaller subspace of the data. This is because, as the authors claim, on smaller spaces complex data structures become simpler and anomalies thus can be identified easier. Zhang et al. (2017) created a Local Sensitivity Hashing (LSH) framework for Isolation Forests, which also was able to detect anomalies in complex data structures. Furthermore, they showed some of the existing isolation based detection methods, including iForest, were special cases of their framework using a specific LSH family.

The introduction of soft splitting in the framework of isolation-based anomaly detection extends current research and as to the knowledge of the author has not been done before. The central question of this thesis is if soft splitting can improve the performance of Isolation Forests. The main improvement of using soft splitting could be the increase of information used in every node according to Irsoy et al. (2012). Due to this increase amount of information used in every split, the predictive performance of the algorithm could improve. This increase in performance can be a general improvement, but the increased information used in every split could possibly even solve problems the Isolation Forests have with recognising complex data structures, which was highlighted by Hariri et al. (2019) and Liu et al. (2024). Next to the predictive performance, soft splitting could also improve the computational performance of the algorithm. In Irsoy et al. (2012) it was also shown that the increased amount of information used in every split could lead to smaller trees. On the other hand are the computations in every node more complex, which potentially could lead to more computations. Especially if the extra information used in every split doesn't improve the decision by much, this could lead to a worse computational performance.

To find the effect of soft splitting in isolation based outlier detection, this thesis generalises the concept of hard splitting in Isolation Forests to a soft splitting framework. For this, it is shown that the distance from an instance to a splitting hyperplane is equivalent to how much the instance complies to the splitting criterion of that split. The concept of soft isolation is introduced to solve the problem of instances never being fully isolated using soft splitting. The Soft Isolation Forest (SIF) algorithm is introduced, extending the iForest algorithm with soft splitting. For this SIF algorithm a set of new hyperparameters are introduced to control the soft splitting process. In order to evaluate the performance of the SIF algorithm, two types of datasets are used. Simulated 2-dimensional datasets are used to visualise the effects of the newly introduced hyperparameters and to study the ability of the SIF algorithm to recognise complex data structures. Furthermore, a set of 13 real world datasets is used to evaluate the performance of the SIF algorithm. To evaluate the performance, the effects of the newly introduced parameters on the performance of the SIF algorithm are studied and benchmarks are set for these parameters. Then, the effects of the existing hyperparameters in the iForest algorithm are compared between the SIF algorithm and the iForest algorithm. Finally, the performance of the SIF algorithm using different measures for soft isolation is also studied.

In this thesis, it is found that soft splitting can improve the predictive performance on some datasets, especially ones with a high number of attributes and without clustered anomalies. However, on datasets with less attributes and clustered anomalies performance is most of the times worse than that of iForest. Furthermore, the computational performance declines massively compared to the iForest algorithm. The computation time can increases for most datasets by a factor of 100. This is caused by a big increase in the number of computations needed to perform soft splitting and an increase in the number of nodes used in the ensemble. Both predictive performance and computational performance are limited by the inability to optimally set hyperparameters, as hyperparameter optimisation in unsupervised outlier detection algorithms is still an active area of research (Ma et al., 2023). It is shown that the optimal setting of hyperparameters depend on both the settings of the other hyperparameters and on the dataset, which increases the difficulty of optimally setting the hyperparameters without optimisation. If the hyperparameters were to be optimised, both the predictive performance and computational performance of the SIF algorithm could be further improved, but that is left beyond the scope of this thesis. However, as these further improvements are only potential for now, soft splitting cannot be seen as an overall improvement from hard splitting without a better method to optimise the hyperparameters and enhancements to decrease the computation time. If these improvements can be made however, soft splitting shows potential to improve the performance for especially datasets with a high number of attributes.

This thesis is structured as follows: In Section 2 the soft splitting framework is formalised.

Section 3 proceeds to transform this framework into the SIF algorithm. The data and evaluation metrics used in this thesis are set out in Section 4. In Section 5 the effects of the hyperparameters on the performance of the SIF algorithm are studied and the performance of the SIF algorithm is compared to that of the iForest algorithm. Section 6 provides a discussion on the found results and concludes this this thesis.

2 Soft splitting for Isolation Forests

This section will proceed to explain how soft splitting works in the context of Isolation Forests. To do this, first some shortcomings of hard splitting in the Isolation Forest are pointed out. Then, it is shown how soft splitting can be used to overcome these shortcomings. Finally, new challenges that arise due to soft splitting are set out.

In a normal Isolation Forest, the trees are build by randomly partitioning the instances. Due to anomalies being few and different, they will be isolated more easily from the other instances and thus have shorter paths in the Isolation Trees (Liu et al., 2008). However, these random splits can be arbitrary, as a slightly different split could lead to a point being isolated earlier or later in the Isolation Tree. This could also lead to two points being isolated from each other whilst being really close together. High dimensional data could especially suffer from this, because another split in the same dimension could decrease the effect of a possibly badly placed split. For high dimensional data it is, however, less likely that another split is made in the same dimension. This problem is created by a loss of information that is introduced by hard splitting, as after the split instances are either on the left or right side of the split. All information about how much the instance complied with the splitting criterion in the split, is lost. Hard splitting also leads to large biases around decision boundaries, because as soon a scoring instance crosses the splitting criterion the output changes completely. However, a small shift of the instance should not result in a major change in output, especially since the place of the decision boundary is somewhat arbitrary.

A visualisation of the mentioned loss of information in a highly stylised setting can be seen in Figure 1. Here, three possible splits across the x-axis are shown together with 4 instances. It should be clear that an isolation-based algorithm aims to isolate both points A and D and wants to keep points B and C together. Despite the three splits shown being really close together, the results of the three splits are very different. Using split 1, the information saved reflects that point A complies less than the other points to the splitting criterion, but not that B and C also comply less to the

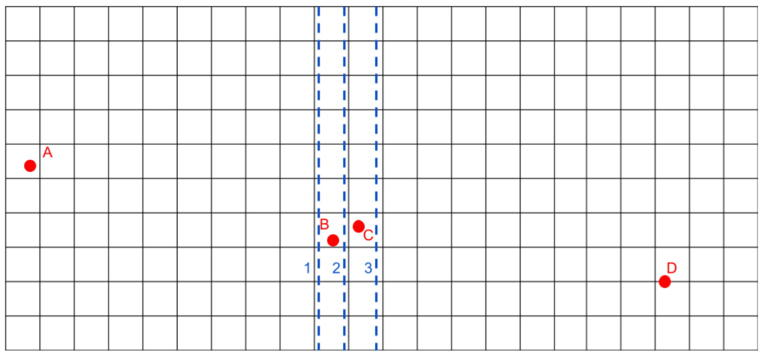


Figure 1: Visualisation of some shortcomings of hard splitting in Isolation Forrest

criterion than point D. For further nodes point B and C are recorded to have the same compliance to split 1 as point D whilst this is clearly not the case. Split 3 has the completely opposite effect, only switching the results for point A and point D. A small shift of the splitting hyperplane has caused the opposite information to be saved and either the information about the relation between point A or point D with points B and C is lost. Split 2 makes the loss of information even worse as the output gives that points A and B comply less to the splitting criteria than points C and D. However, it is not reflected that the compliances of points B and C to the split are almost equal, the compliance of point A is way more than those and the compliance of point D is way less. Using this split, not only is both the relation between point A and points B and C and the relation between point D and points B and C lost, but also the relation between point B and C themselves. This also shows the problems about the sharp decision boundaries as a small shift from point C to B, gives a completely different output for the instance, while the information known about the instance has barely changed. In practise, these shortcomings are mostly solved by the Isolation Forest being an ensemble method, meaning that the shortcomings of the multiple agents can be masked by each other. This means that by solving these shortcomings, Isolation Forests using soft splitting could potentially need a smaller ensemble to reach the same predictive performance.

To solve these shortcoming, this thesis introduces soft splitting in the Isolation Forest framework. Soft splitting was earlier implemented in decision trees by Irsoy et al. (2012). In soft splitting instances are no longer entirely in either the right split or the left split, but get a weight to be in either of the splits based on how much the instance complies to the splitting criterion of the split. Compared to hard splitting, this means that splits are no longer determined by a binary 0 or 1, but on a continuous scale from 0 to 1. An instance complies more to the split compared to another instance if it would satisfy a stricter version of the splitting criterion. For axis-parallel splitting this criterion is

$$x_q - p < o, \tag{1}$$

where q is the selected attribute, x_q the value of the instance X for attribute q and p the split point. o, which is mostly set to 0, is the strictness of the criterion with a lower value of o meaning a stricter criterion. It can easily be seen that an instance complying more to this split is geometrically equivalent to the instance having a smaller signed distance,

$$sigd(X, p, q) = x_q - p = sign(x_q - p) * d(X, p, q), \tag{2}$$

to the splitting hyperplane. The signed distance here is just a combination of the rotation and the distance of the instance towards the splitting hyperplane. This equivalence also holds when using oblique splitting, as in the EIF algorithm (Hariri et al., 2019). There the splitting criterion is

$$(\vec{x} - \vec{p}) \cdot \vec{n} < o, \tag{3}$$

where $\vec{p}$ is the splitting point and $\vec{n}$ is the normal of the splitting hyperplane, whilst the signed distance is

$$sigd(\vec{x}, \vec{p}, \vec{n}) = \frac{(\vec{x} - \vec{p}) \cdot \vec{n}}{||\vec{n}||}. \tag{4}$$

Here the length of the normal vector $\vec{n}$ is used to normalise the distance. This can be done as scaling with $||\vec{n}||$ on both sides in Equation 3 doesn't influence which instances satisfy the splitting criterion. So in this way, soft splitting can use the distance between an instance and the splitting hyperplane to determine the weight an instance should get after every split.

Figure 1 can now again be used to show how the loss of information is now solved. All three splits now reflect points B and C with really similar weights as they both have a similar distance to the splitting hyperplane, even if split 2 is used. Furthermore, both point A and point D have very different weights from point B and C, where with hard splitting splits 1 and 3 were only able to reflect the difference of either point A or point D to points B and C. Thus loss of information about the differences between the instances is completely solved. Furthermore, the large biases around decision boundaries are also solved. This can be seen as the output changes slowly as the distance of the instance to the splitting hyperplane increases, leading to smooth transitions around the decision boundaries. A small shift of an instance can no longer lead to big difference in output, but is now represented proportional to the distance of the shift. The shortcomings of hard splitting showed earlier in this subsection are resolved.

Soft splitting, however, does introduce some new challenges that need solutions before it can be implemented. Firstly, short splitting makes it impossible to isolate an instance from all other instances. This is because in every split, all instances will be present with a certain weight. This gives a paradox as the goal of the algorithm is to isolate every point from each other, whilst no instances can ever be isolated from each other. Therefore, the concept of soft isolation needs to be defined. A natural definition of this would be that an instance is softly isolated when the weight of an instance in a node is large compared to the weights of all other instances. In the most extreme case this would mean that an instance is isolated, when the proportion of the weight of that instance in the node goes to 1, while the proportion of the weights of the other instances go to 0. This would be equivalent with the definition of isolation when using hard splitting, showing that soft isolation is a more general concept. Therefore, soft isolation could possibly lead to a wider range of options to detect outliers. For example, not the the isolation of a single instance could be the goal, but rather the isolation of a group of instances to detect grouped anomalies. The importance of detecting clustered anomalies using Isolation Forests is emphasised in Liu et al. (2010).

Another implication of every instance being in every node is that in theory a node is never empty and thus can always be further expanded upon. Although, in most nodes at some point one instance will be isolated, the weights might already be so little that the output is no longer influenced by the results of these nodes. Therefore, to save computation power, branches with little weight should be pruned early. A final implication of every instance being present in every node is that the area in which the splitting intercepts of the hyperplanes are selected is the same for every node and does not shrink when proceeding further into the tree. With hard splitting this does happen as the area in which the intercepts can be selected is bounded by the instances present in the node. This allows Isolation Forests to zoom in on the small differences between the instances left in the node and differentiate between them. Using soft splitting this is harder all intercepts are chosen on a global level, significantly decreasing the probability of choosing a splitting point that would differentiate between two close instances. Furthermore, if a new split falls completely or mostly outside of an already made split, no new information will be revealed by the split. These splits will thus not improve the predictive performance, whilst increasing the computation time.

The evaluation stage of the algorithm also needs to be slightly changed to incorporate soft splitting. This is because as every point is in every split, instances no longer have a single depth. A natural way to get a single depth for a point in a tree, is to take the weighted average with the weights the point has in every terminal node. Notable is that following this method more information is used in the computation of the anomaly scores. In hard splitting an instance follows a single path through a certain number of splits, s, to a terminal node. In soft splitting the instances follows all the possible paths with a certain weight. This means that if it reaches the same depth of the hard splitting path in all the paths, it uses 2^{s-1} splits. This again, could especially be useful in a high-dimensional dataset, as the increased number of splits increases number of attributes the attributes and thus information used to compute the anomaly score, using more of the available data.

3 Methodology

In this section, the method for performing soft splitting in Isolation Forests, called Soft Isolation Forest (SIF), is introduced, as laid out by the previous section. The methodology of this algorithm can be split up in three algorithms, like in most research in Isolation Forests, where the first two algorithms are part of the training phase and the final algorithm is part of the scoring phase. The first algorithm is trivial as it combines the different agents of the Soft Isolation Forest into an ensemble. The second algorithm concerns the building of an individual Isolation Tree using soft splitting. The third and final algorithm computes the weighted path length of an instance. The algorithms are adaptations of the iForest algorithms of Liu et al. (2008), which they resemble very closely. To perform soft splitting, new functions and parameters are introduced to measure soft isolation and to transform the distance in a split into a weight,.

3.1 Building of Soft Isolation Forest and Soft Isolation Trees

The algorithm for building the ensemble of the Soft Isolation Forest, given in Algorithm 1, barely changes from the original algorithm for building the iForest. The only notable change is that the maximum tree height limit l is no longer set to ceiling($\log_2 \psi$), but to another function h of the subsample size ψ. This is because splitting is slightly different from the original algorithm, making it unclear if the logarithm still gives a good relation between subsample size and the maximum tree height.

Soft splitting does affect the algorithm for building an Isolation Tree. The new algorithm can be seen in Algorithm 2. This algorithm uses axis-parallel splitting, similarly to the original iForest algorithm by Liu et al. (2008). However, this can easily be adapted for oblique splitting. This adaption is not studied further in this thesis, but it can be found in Appendix A.

Algorithm 1 SIF

Input: X - input data, t - number of trees, ψ - sub-sampling size, i - stopping threshold for isolation measure, o - stopping threshold for empty nodes

Output: A set of t Soft iTrees

```
1: Initialize Forest
2: Set height limit $l = h(\psi)$
3: for $i = 1$ to $t$ do
4:     $X' \leftarrow$ sample($X$, $\psi$)
5:     Forest $\leftarrow$ Forest $\cup$ Soft Isolation Tree($X'$, 0, $l$, $i$, $o$)
6: end for
7: return Forest
```

Algorithm 2 Soft Isolation Tree

Inputs: X - input data, w - weight of every data point, e - current tree height, l - height limit, i -stopping threshold for soft isolation, o - stopping threshold for empty nodes, k - base steepness of the kernel

Output: a Soft Isoltaion Tree

```
1: if $e \geq l$ or $isolation(w) \leq i$ or $sum(w) < o$ then
2:     return $exNode\{Weights \leftarrow w\}$
3: else
4:     Let $Q$ be the list of attributes in X
5:     Randomly select an attribute $q \in Q$
6:     Randomly select an intercept point $p$ between the in the range of $q$ in $X$
7:     $w_l \leftarrow g(X, q, n, k)$
8:     $w_r \leftarrow 1 - w_l$
9:     return $inNode\{Left \leftarrow$ Soft Isolation Tree$(X, w * w_l, e+1, l, i, o, k)$,
10:                    $Right \leftarrow$ Soft Isolation Tree$(X, w * w_r, e+1, l, i, o, k)$,
11:                    $Attribute \leftarrow q$,
12:                    $Intercept \leftarrow p\}$
13: end if
```

The first change of Algorithm 2 compared to the iForest algorithm is the criterion which determines whether a node is a terminal node on row 1. Here, $isolation(w) \leq i$ has replaced $|X| \leq 1$ as criterion for a point to be isolated. $isolation$ is a function that measures the level of soft isolation, as defined in Section 2, in a node and i the threshold to determine the level of soft isolation needed for the node to be made a terminal node. A possible choice for $isolation$ could be an impurity measure. These are used in decision trees to decide the best possible split by measuring how homogeneous a split is. The misclassification error is the simplest impurity measure and the impurity measure that is closest to the original definition of isolation. It is computed as

$$\text{misclassification}(w) = 1 - \max_{i \in X} p_i., \tag{5}$$

where p_i is the relative weight of instance i compared to the weight of all instances in X

$$p_i = \frac{w_i}{\sum_{j \in X} w_j}. \tag{6}$$

Thus, the misclassification error measures the isolation of the most isolated instance and using this measure an instance is softly isolated when the relative weight of the instance is higher than $1-i$. If i would be set to 0, an instance would have to be completely isolated to reach soft isolation and the isolation criterion would be the same as when using hard splitting. The Gini-impurity and Entropy are more sophisticated impurity measures. They can be computed following

$$\text{Gini}(w) = 1 - \sum_{i \in X} {p_i}^2, \tag{7}$$

$$\text{Entropy}(w) = -\sum_{i \in X} p_i \log_2 p_i. \tag{8}$$

Both these use the weights of all instances in a node to determine how homogeneous the weights in the node are distributed. As these impurity are more sophisticated, they might be able to improve the detection of isolated instances. This could improve both the predictive performance through more accurate depths and the computational performance through smaller trees. Furthermore, as the weights of all instances are used to measure the level of isolation, they could potentially be used to better detect the presence of clustered anomalies. For setting the threshold for soft isolation it is important to note that all of these measures for soft isolation have different ranges of potential values. All three have a lower bound of 0, which is the most strict form of isolation. The upper bound however differs between them. For the misclassification error the upper bound is always 1, whilst for the Gini-impurity and the Entropy it depends on the subsample size ψ. The upper bound of the Gini-impurity is $1 - \frac{1}{\psi}$ and the upper bound of the Entropy is $\log_2 \psi$.

In the first row $sum(w) < o$ is also added to stop the algorithm developing further on empty nodes. The sum of weights w is used as a measure for how many instances are left in the node and the threshold o determines how much weight should be left in a node to not be cut off. By adding this condition, the algorithm is not able to infinitely expand on already empty nodes. There are other alternatives to determine whether a node is empty. For example, the instance with the highest weight could be used instead of the sum of weights. The threshold o could also be improved. In the current implementation the threshold is static, but it could use the depth of the node e. The expected weight of every instance decreases by a factor of 2 for every extra level of depth. By incorporating this decrease in the threshold, the weights of the instances in the node can be checked against their expected weights to see if the weights are low enough to be considered empty.

In rows 5 and 6 the splitting attribute and point are determined, which together form the splitting hyperplane. This does not change from the original algorithm. However, as explained in Section 2 contrary to the hard splitting algorithm the area in which the splitting point is selected does not shrink. Thus, the children of a node can have overlapping areas on earlier split attributes and can make the exact same split as the parents. Some enhancements to the algorithm are possible to shrink the selecting area. Firstly, the area in which the intercepts is selected in row 6 can be bounded by the splits in the parent nodes. Another option is to only use the area spanned by instances that have a weight higher than a certain threshold. Then the area in which the intercept is selected only uses relevant instances. Furthermore, other enhancements could also serve the same purpose. However, as this thesis will not use any enhancement to shrink the area in which the splitting area is selected, it will not proceed to dive further into this topic.

In rows 7 and 8 the hard splits of the iForest algorithm are replaced by a gating function g. The gating function g assigns weights between 0 and 1 to determine how much a point is in the split. As discussed in Section 2, these weights should use the signed distance between the instance and the splitting hyperplane as a measure of how much an instance complies to to splitting criterion. The distance should be transformed to weights between 0 and 1, where points with bigger distances to the split should be closer to either 0 or 1, depending on their rotation towards the split. Furthermore, the weights in the right and the left split should sum up to 1 meaning that a change in distance should have the same effect on the right and left split. This implies this transformation should be symmetric around a signed distance of 0 and the weight at that distance is 0.5 for both sides of the split. For this, a sigmoid function can be used. A sigmoid function that is often used in literature is the logistic function. The logistic function is defined as

$$f(x) = \frac{L}{1 + \exp(-k(x - x_0))}, \tag{9}$$

where L is the number the function approaches if x goes to $+\infty$, x_0 the midpoint of the function and k the steepness of the function. As f goes to 0 when x goes to $-\infty$, the logistic function with $L = 1$ and $x_o = 0$, satisfies all conditions. Thus, using the signed distance in equation 2, the gating function g can be written as

$$g(X, p, q, k) = \frac{1}{1 + \exp(-k * sigd(X, p, q))}. \tag{10}$$

The gating function determines the weights of the split using the distance of the instances to the splitting hyperplane and the steepness of the kernel k. k determines how fast the weights go to 0 or 1 for higher distances to the splitting hyperplane. As k is constant, every split has the same importance in computing the final weight of an instance. Alternatively, k could be scaled for different properties of the split. For example, k could be scaled using the tree depth e, to make later splits have a greater importance. This, however, is not adapted in this paper.

The scaling of different attributes directly influences the weights via the signed distance. Splits on attributes with a larger variance therefore always have more extreme weights. To resolve this, every attribute should be standardised before using the algorithm. Equivalently, k could also be scaled with the standard deviation of the attribute in every split.

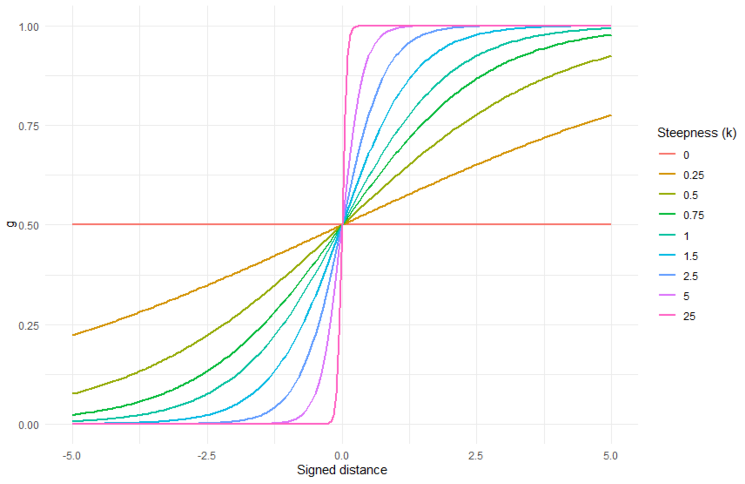


Figure 2: Plots of the allocated weight for a distance of an instance for different levels of steepness

The weights produced by the gating function g for different steepnesses of the kernel and distances is shown in Figure 2. The standardisation of the data also helps with the interpretation of the weights given at a certain distance. The weights can be compared to the proportion of values that falls under the standard normal distribution at that distance. If the centre of the standard normal distribution is at the mean of the attribute, a signed distance of 1 means that 84.2% is expected to comply more to the splitting criterion and 15.8% to comply less. At this distance a steepness k of 0.5 gives a weight of 0.62, a steepness of 1 gives a weight of 0.73, a steepness of 2 gives a weight of 0.88 and a steepness of 5 already a weight of 0.99. Thus, by setting the steepness of the kernel k, the ratio between the weight distributed at a distance and the proportion of the data present at that distance can be set. The small example also shows how quickly the weights rise for higher levels of the steepness of the kernel and it is clear that the weights distributed to both nodes converge to hard splitting as k goes to infinity. In the figure it can already be seen that the weights distributed to the nodes at k set to 25 already come very close to hard splitting.

Using this gating function, the algorithm can compute the weights on both the left and right side of the split in rows 7 and 8. The weights for the right side can be calculated by subtracting the calculated weights from 1, as the weights on both sides of the split should sum to 1. The algorithm then proceeds by calling the same function recursively for the left and right child, whilst saving the splitting attribute and intercept for the scoring phase.

3.2 Computing the anomaly score

In the iForest algorithm the anomaly scores are based on the path length of a point in different Isolation Trees. However, using soft splitting instance no longer have a single path length, as explained in Section 2. Instead, the weighted path length can be used. In Algorithm 3 this is computed. The only changes of this algorithm are that on rows 6 and 7 the weights for the left and right side of the split are computed using gating function g and that the returned value on row 8 is now the weighted path length of both children, instead of only the path length of one those nodes. On row 2, $c(n)$ is still added to the returned tree depth as compensation for the early termination when the maximum tree height is reached. As Isolation Trees with hard splitting have an equivalent structure to Binary Search Trees the expected path length $c(n)$ of an Isolation Tree with a n training instances can be computed in the same way as the expected path length of an unsuccesful search in a Binary search Tree. This computation is

$$c(n) = \mathbb{1}_{n>2} 2H(n-1) - (2(n-1)/n), \tag{11}$$

where H(i) is the harmonic number estimated by $log(i)$+0.5772156649 (Euler's constant). Therefore

Algorithm 3 WeightedPathLength

```
Input: $x$ - an instance, $T$ - an iTree, $e$ - current path length; to be initialized to zero when first called, $k$ - base steepness of the kernel
Output: path length of $x$
1: if $T$ is an terminal node then
2:     return $e$ + c(sum(T.Weights))
3: end if
4: $q \leftarrow T$.Attribute
5: $p \leftarrow T$.Intercept
6: $w_l \leftarrow g(x, q, p, k)$
7: $w_r \leftarrow 1 - w_l$
8: return $w_l$ · WeightedPathLength($x$, $T$.left, $e + 1$, $k$) + $w_r$ · WeightedPathLength($x$, $T$.right, $e + 1$, $k$)
```

$c(n)$ can be used to estimate the remaining path length of a branch that is terminated early by the tree height limit. However, using soft splitting the assumption of equivalence to a Binary Search Tree no longer holds. This can for example be seen, if the threshold for empty nodes o is set to a low level. Then empty nodes will keep being expended on, which would lead to higher expected path lengths than when using hard splitting and thus then will be predicted by the Binary Search. Thus, $c(n)$ will underestimate the expected remaining path length for nodes. Cortes (2021) showed that more variations of Isolation Forests that ore often used in literature, suffer from an inaccurate expected path length.

The computation of the anomaly scores of instance x is the same as in the iForest algorithm, given by

$$s(x, \psi) = 2^{-\frac{\frac{1}{t}\sum(h(x))}{c(\psi)}}, \tag{12}$$

where t is the number of trees in the forest and $h(x)$ is the weighted path length of point x. Here, $c(\psi)$ is the expected path length of a tree trained with ψ training instances. This estimation is again biased. However, in this case the expected path length is only used to scale the anomaly scores. Therefore, it will only impact the interpretation of the anomaly scores and not the performance.

4 Data and evaluation metrics

This section gives an overview of the datasets and evaluation metrics used in this research. This research uses a combination of both real world datasets and simulated datasets, which is a common approach in the anomaly detection literature. The most important evaluation metric is the Area Under The Curve (AUC) of the Receiver Operating Charecteristics (ROC) curve.

4.1 Real world datasets

In this research 13 real world datasets are used to evaluate the performance of the Soft Isolation Forest. These datasets are obtained from Shebuti (2016). Table 1 gives an overview of different datasets with their properties. The number of instances can differ between almost 600000 and around 500. Most datasets have around 5 numerical attributes with two datasets having more (36 and 274 respectively. Anomaly percentage can differ between very low percentages under 1% for the larger datasets to 35% for some of the smaller datasets. Of these datasets, Http and Annthyroid are identified by Liu et al. (2010) to only have clustered anomalies, whilst Satellite, Pima, Breastw and Ionosphere are identified to have mostly scattered anomalies. Here, clustered anomalies are anomalies that have close proximity to each other, whilst scattered anomalies are anomalies that do not have a close proximity to both other outliers and the normal instances.

4.2 Simulated datasets

In addition to the real world datasets, this research uses some simulated datasets. These are used to show how different structures of the data affect the output of the algorithm. As this will be done using visualisations, like in Hariri et al. (2019) and Liu et al. (2024), these datasets are 2-dimensional. Three different datasets are simulated. In the first the data follow the shape of a sinusoid, in the second a spiral and in the third the normal data consist of multiple small clusters, called small blobs. The sinusoid is generated using

$$y = 10 \cdot \sin\left(\frac{2\pi}{5} \cdot x\right) + \epsilon, \tag{13}$$

where x varies from 0 to 5. ϵ represents noise following a Normal distribution with mean 0 and standard deviation of 2. In total 3000 points are generated with x-values evenly spread out between 0 and 5. The spiral is generated using the equation for a spiral in polar coordinates, $r = \frac{1}{2\pi} \cdot \theta$,

	Number of instances	Number of attributes	Anomaly percentage
Http (KDDCUP99)	567497	3	0.4%
ForestCover	286048	10	0.9%
Smtp (KDDCUP99)	95156	3	0.03%
Shuttle	49097	9	7.2%
Mammography	11183	6	2.3%
Annthyroid	7200	6	7.4%
Satellite	6435	36	31.6%
Thyroid	3772	6	2.5%
Cardio	1831	21	9.6%
Pima	768	8	34.9%
Breastw	683	9	35.0%
Arrhythmia	452	274	14.6%
Ionosphere	351	33	35.9%

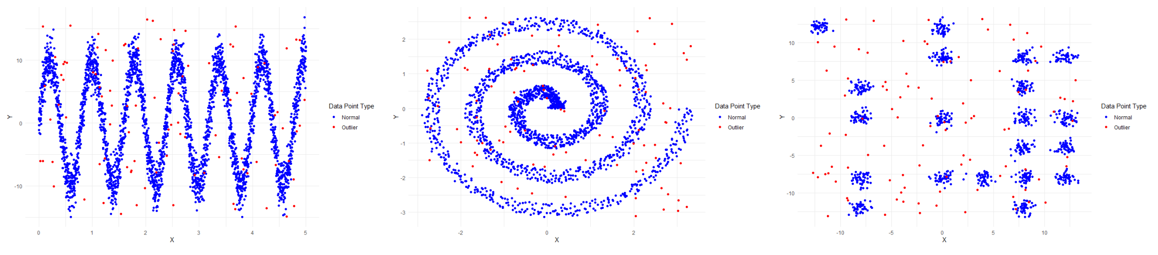
Table 1: Overview of the datasets containing some properties of the dataset

which gives the following formulas for the x- and y-coordinates

$$x = (r + \epsilon_x) \cdot \cos(\theta), \tag{14}$$

$$y = (r + \epsilon_y) \cdot \sin(\theta). \tag{15}$$

Here θ varies from 0 to $3 \times 2\pi$ and ϵ_x and ϵ_y are both distributed uniformly between 0 and 0.4. In total 2000 data points are generated evenly spread out over the range of theta. Finally, the small blob dataset is simulated using a multivariate normal distribution with covariance matrix $\Sigma = \begin{bmatrix} 0.25 & 0 \\ 0 & 0.25 \end{bmatrix}$ and a different mean for every cluster. In total there are 20 clusters and each cluster has 50 data points. In all three datasets 100 outliers are generated, where the x values of the outliers is uniformly distributed between the minimum and the maximum value of x in that dataset, whilst the y values are uniformly distributed between the minimum and maximum value of y. The scatterplots of these dataset can be seen in figure 3.



(a) Sinusoid dataset (b) Spiral dataset (c) Small blobs dataset

Figure 3: Scatterplots of the simulated datasets with the outliers in red the outliers and the normal data in blue

4.3 Evaluation metrics

The most important evaluation metric to measure the predictive performance of Isolation Forests is the Area Under Curve (AUC) of the Receiver Operating Characteristics (ROC) Curve. The ROC curve sets out the True Positive Rate (TPR) of a binary model against its False Positive Rate (FPR). The TPR is the ratio of which the model predicts actual positive, or non-anomalous, instances to be positive and can thus be calculated as

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{16}$$

where TP is the number of positives that are predicted to be positive, or True Positives, whilst FN is the number of positives that are predicted to be negative, or False Negatives. On the other hand, the FPR is the ratio of which the model predicts negative instances to be postives and thus is calculated as

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}}, \tag{17}$$

where FP is the number of negatives that are predicted to be positive, or False Positives, whilst TN is the number of negatives that is predicted to be negative, or True Negatives. A good performing model would have a high TPR, whilst having a low FPR.

Isolation Forests give an anomaly score to every instance, where an instance with a higher score has a higher probability to be anomalous, or negative. A threshold needs to be set to determine which instances are anomalous. By setting the threshold, instances over the thresold are predicted to be negative and under the threshold to be positive, thus determining the TPR and FPR. When the threshold rises, more instances are predicted to be positive as more instances fall under the threshold and thus the TPR and the FPR both rise. This means that a model is better when the TPR rises quicker than the FPR. To measure this, the ROC curve plots the combinations of

TPR and FPR set by all different thresholds against each other. The AUC value of this curve then represents the performance of the model. A model that randomly chooses between predicting positive or negative will have an AUC-value of 0.5

Next to the predictive performance of the model, the computational performance of the method is very important. Computation time for Isolation Forest can roughly be divided into two parts. The first part is the size of the trees, as computation time will increase as the number of nodes in a tree increases. This also has a big impact in the amount of storage the algorithm takes. The second part is the number of computations per node. When using soft splitting, these obviously increase as for every instance a weight needs to be calculated in every node.

5 Results

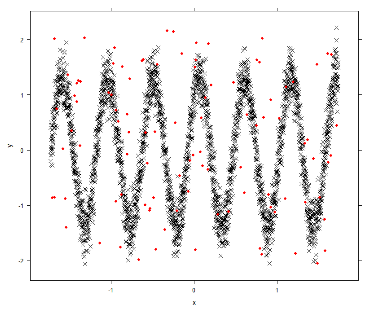
In this section the results of this research are set out. For this, first the simulated datasets are used together with heatmaps of the anomaly score to study how the output of the SIF algorithm differs from the iForest algorithm. Using these simulated datasets and heatmaps, the effects of the newly introduced parameters, the threshold for isolation and the steepness of the kernel, are set out. This section proceeds to study the effect of different levels of these newly introduced parameters on the performance of the model for different levels of subsample size and tree height limit. Then this is also done for the threshold for empty nodes. Benchmarks are then set for these new parameters based on the acquired results. These benchmarks will then be used to further study the performance of the SIF. Optimisation of hyperparameters in unsupervised outlier detection models is still an ongoing field of research according to Ma et al. (2023) and most literature in Isolation Forests use such benchmarks for the different parameters (see for example Liu et al. (2024)). After benchmarks for the new parameters are set, the effect of the subsample size, tree height limit and number of trees is compared to the effect of these in the iForest algorithm. Then, performance of the SIF algorithm is compared for a large number of real world datasets to the iForest algorithm in order to determine the global performance of both algorithms. Finally, a comparison of the performance of the SIF algorithm is made between different measures for soft isolation.

5.1 Visualisation of anomaly scores SIF

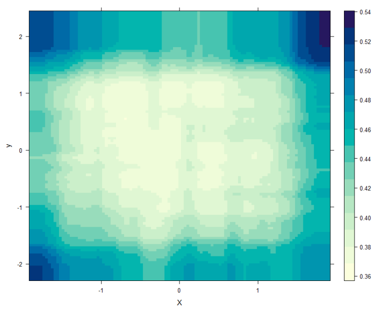
This subsection focuses on visualising the outputted anomaly scores of the SIF algorithm. These visualisations are first used to compare the outputted anomaly score maps between the SIF algorithm and the iForest algorithm. Then they are used study the effects of the two most influential new parameters, the threshold for soft isolation and the steepness of the kernel. For these visualisations, heatmaps of the outputted anomaly scores of the simulated 2-dimensional datasets are used.

In literature, it is widely established that the iForest algorithm has a limited capability to recognise complex structures in datasets (Hariri et al., 2019; Cao et al., 2024). In the heatmaps of anomaly scores, the complex structures, like sinusoid or spirals, can no be recognised, whilst they are very present in the data. Furthermore, in the outputted anomalies scores sometimes ghost cluster are created. Ghost clusters are regions where there are higher anomaly scores, without there actually being more datapoints in the dataset in those regions. One of the reason these can occur because of the symmetry of surrounding clusters together with axis-parallel splitting. In Section 2 the increased information used in every split when using soft splitting, is raised as a potential solution for these problems. Another visual implication of the use of soft splitting in SIF should be that the decision boundaries should be less present in the heatmaps of the outputted anomaly scores. To study these effects the anomaly scores on the 3 simulated datasets are computed for both the SIF and iForest algorithms and these anomaly scores are shown using heatmaps. For both SIF and iForest, the number of trees in the ensemble is set to 50, the subsample size to 64 and the maximum tree height to 8. Additionally for SIF the steepness of the kernel is set to 50, the measure for soft isolation used is the misclassification error, the threshold for soft isolation is 0.3 and the threshold for empty nodes is set to 1.

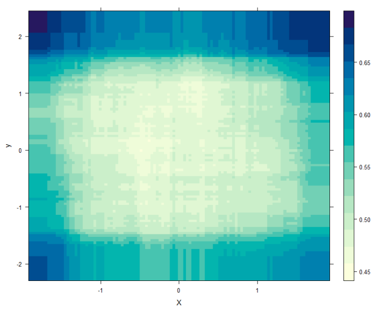
In Figure 4 the scatterplots of the normalised datasets can be seen together with the heatmaps of the SIF and iForest algorithms for those datasets. It can clearly be see that the ability of the SIF algorithm to recognise complex data structures has not improved very much. Comparing Subfigures 4b and 4c the sinusoid might be slightly more recognisable for the SIF algorithm than for the iForest algorithm. However, the spiral form is not recognisable in both Subfigures 4e and 4f. The ghost cluster problem, which can clearly be seen in some places of Subfigure 4i is also not entirely solved by applying soft splitting. In Subfigure 4h some of the ghost clusters have disappeared. However, in the empty spaces between the clusters anomaly scores have generally increased. This seems to be caused by the smoother transition between decision boundaries, which makes it harder to differentiate between clusters and the empty spaces. The SIF algorithm does better recognise the most extreme cluster of the dataset in the top left. Although the implementation of soft splitting does not have a big effect on the recognition of complex data structures, the smoothing out of the decision boundaries can clearly be seen. In the heatmaps of iForest the straight, axis-parallel decision



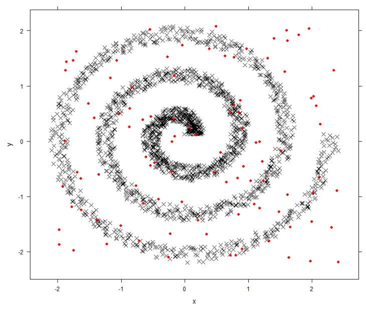
(a) Scatterplot of the normalised Sinusoid dataset



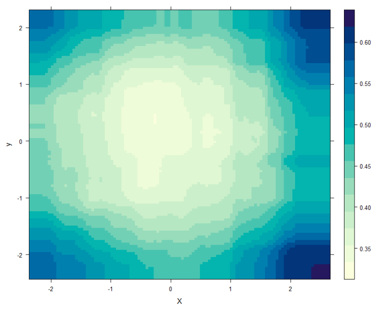
(b) Heatmap of anomaly scores SIF for Sinusoid dataset



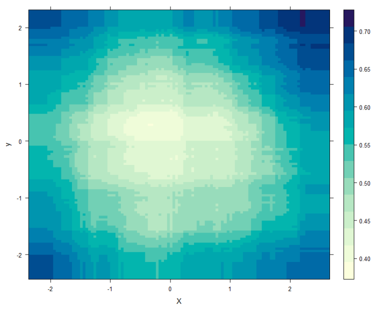
(c) Heatmap of anomaly scores iForest for Sinusoid dataset



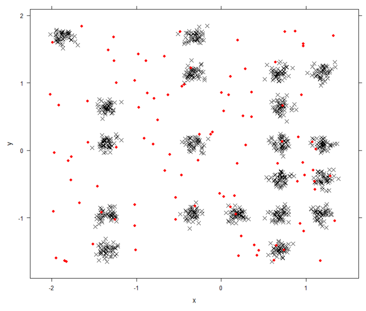
(d) Scatterplot of the normalised spiral dataset



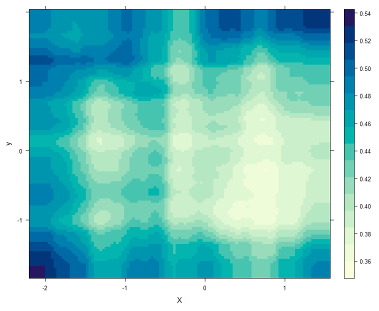
(e) Heatmap of anomaly scores SIF for Spiral dataset



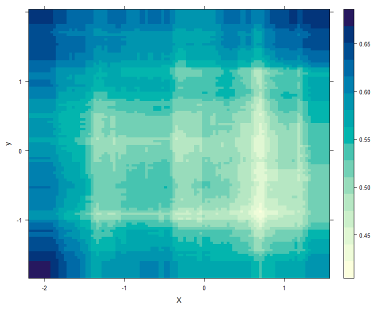
(f) Heatmap of anomaly scores iForest for Spiral dataset



(g) Scatterplot of Small blobs dataset

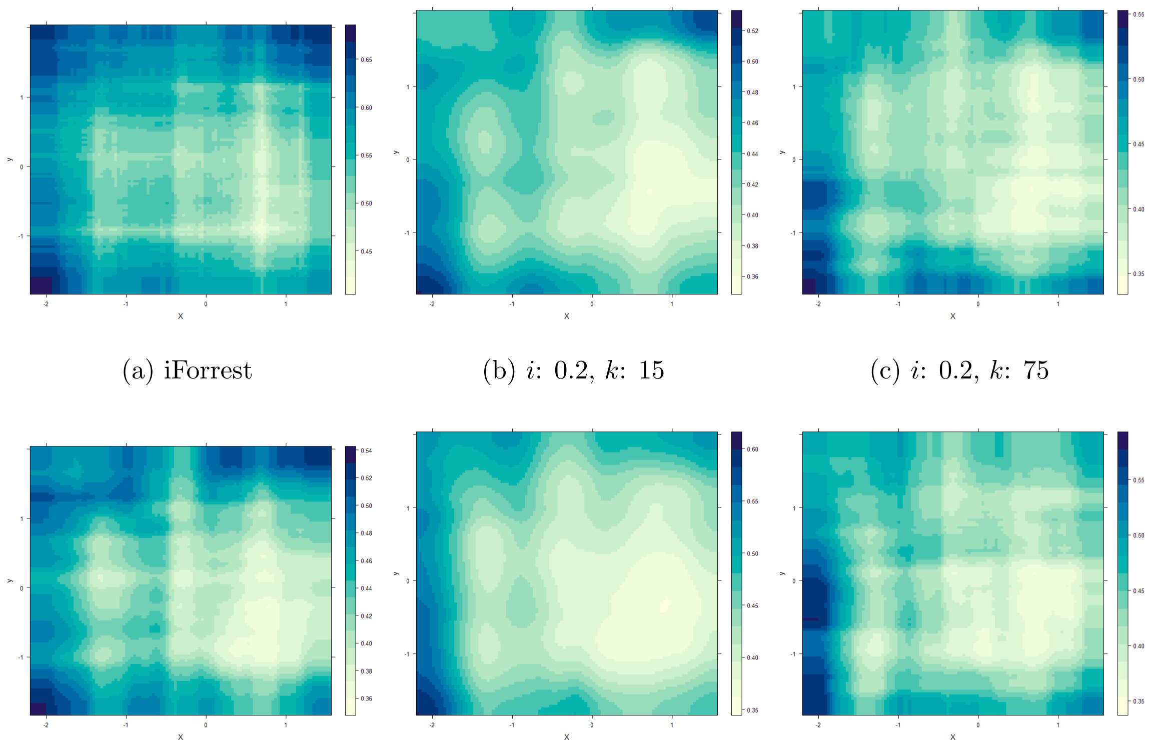


(h) Heatmap of anomaly scores SIF for Small blobs dataset



(i) Heatmap of anomaly scores iForest for Small blobs dataset

Figure 4: The scatterplots of three different simulated datasets with the heatmaps of anomaly scores that the SIF and iForest algorithms assign to these datasets.



(a) iForrest (b) i: 0.2, k: 15 (c) i: 0.2, k: 75

(d) i: 0.3, k: 50 (e) i: 0.8, k: 15 (f) i: 0.8, k: 75

Figure 5: Heatmaps of the anomaly scores in the dataset with multiple clusters for different settings of the threshold for soft isolation i and the steepness of the kernel k

boundaries are clearly visible compared to the heatmaps produced by the SIF algorithm. There, especially away from the edges of the data, the changes in score behave much more organically.

To further study the changes between the iForest algorithm and the SIF algorithm, the effect of two most influential newly introduced parameters is studied. These are the threshold for isolation i and the steepness of the kernel k. This study will only use the Small blobs as the algorithms are able to recognise the structure of this dataset, in contrast to the Sinusoid and the Spiral datasets. Heatmaps for different combinations of these parameters are shown in Figure 5. Here Subfigures 5a and 5d are the heatmaps of the iForest and SIF algorithms in Figure 4, whilst the other 4 subfigures are new heatmaps of the SIF algorithm using different values for the threshold for soft isolation and steepness of the kernel.

The effect of the steepness of the kernel can best be seen when comparing Subfigures 5b and 5e with Subfigures 5c and 5f. For lower values of the steepness of the kernel, the anomaly scores seem to be more homogeneous. The anomaly scores are smoothed out and decision boundaries are barely visible. This is different for higher values of the steepness of the kernel, where the anomaly values are way less smooth and decision boundaries are more visible. From these heatmaps, it can be seen that smooth decision boundaries are not always a positive. In this case, because the anomaly scores can only change gradually, a lot of empty space has a high anomaly score just because it is between regions with a high density of data points. Comparing Subfigures 5b and 5c with Subfigures5e and 5f the effects of different values of the threshold for isolation i can be found. The main effect of this parameter seems to be that anomaly scores become more focused around real data points as the setting of this parameter becomes stricter. Regions with a high density of points are differentiated better from regions with a low density of points. Finally the heatmap of the iForest algorithmin Subfigure 5a most closely resembles the heatmap in Subfigure 5c. This is as expected because as explained in section 3, an iForest can be seen as an SIF, where the threshold for isolation i approaches 0 and the steepness of the kernel k approaches infinity.

5.2 Effect of newly introduced parameters on performance of SIF algorithm

In this subsection the effects of the three newly introduced hyperparameters on the performance of the SIF algorithm are studied. As the effects of these parameters might depend on the settings of the subsample size and the maximum tree height, the results are set out for different levels of those parameters. Furthermore, the optimal values of the threshold for soft isolation and the steepness of the kernel are expected to possibly depend on each other, whilst the threshold for empty nodes is expected to by independent from those. Therefore, firstly the threshold for soft isolation and the steepness of the kernel are studied, keeping the threshold for empty nodes fixed. Then, the effect of the threshold for empty nodes is studied, keeping these other two hyperparameters fixed. The number of trees in the ensemble is kept constant at 50, as this parameter is not expected to influence the effects of the newly introduced hyperparameters and the misclassification error is used as measure for soft isolation.

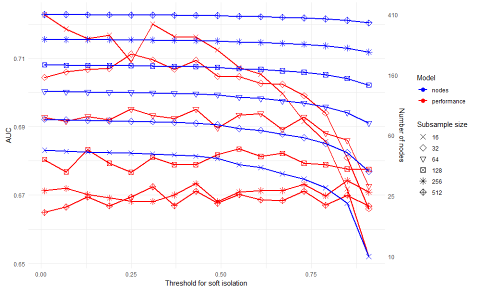
5.2.1 Effect of the threshold for soft isolation and steepness of the kernel on performance

In order to do study the effects of the threshold for soft isolation and the steepness of the kernel, the performance of SIF is computed for different levels of subsample size, tree height limit, threshold for soft isolation and steepness of the kernel. The values for subsample size used are 16, 32, 64, 128, 256 and 512. For tree height limit, values increasing from 1 to 10 with steps of 1 are used together with 12 and 15. The threshold for soft isolation uses values from 0.01 to 0.91 with steps of 0.06. Finally, the steepness of the kernel increases on a exponential scale as differences are much larger for small values of the kernel. Therefore the steepness is set to e^s, with s increasing from -1 to 0 with steps of 0.25 and then from 0 to 5 with steps of 0.5. The threshold for empty nodes to 1. These are set to fixed values as they are not expected to influence the effects of the threshold for soft isolation and the steepness of the kernel on predictive performance. The results are the average across 5 folds in cross-validation. In Figures 6 and 7 and the performance of the model is set out for either the threshold for soft isolation or steepness and the kernel in combination with either the subsample size or the tree height limit for both the Satellite and the Annthyroid datasets. The performance is here measured in predictive performance (AUC) and the size of the decision tree (average number of nodes of a tree in the ensemble). Results are averaged out over the the two hyperparameters that are not in the graphs.

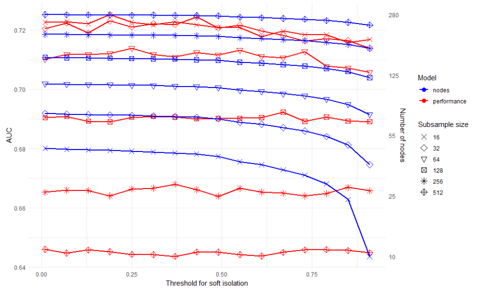
Firstly the effect of of the threshold for soft isolation is studied. In Subfigures 6a and 6b it can be seen that the predictive performance is stable for lower values of the isolation threshold for all subsample sizes. However, in the Satellite dataset this performance breaks down when the threshold becomes a considerable proportion of the subsample size. This can already be seen for subsample sizes 16, 32 and 64 already. As the highest threshold for soft isolation used is 0.91, whilst the theoretical maximum is 1, the other subsample sizes could break down for higher values of the threshold. This also could explain why the predictive performance does not seem to break down for the Annthyroid dataset, as the breakdown point might be higher for this dataset. In both datasets and for all subsample sizes the number of nodes decreases as the threshold rises. The probable cause for this is that branches can be terminated earlier as nodes do not need to be entirely isolated anymore.

The threshold for soft isolation does not seem to have different effects for different levels of maximum tree height, as can be seen in Subfigures 6c and 6d. The predictive performance drops for higher levels of the isolation threshold for both the Satellite and the Annthyroid datasets. This seems to be caused by the worse performance of models with low subsample sizes for higher thresholds, as the performance shown in these figures are the average over all subsample sizes. Furthermore, the predictive performance of all tree height limits seems to be less stable for the Annthyroid dataset, which is probably caused by the smaller range of the AUC-values for that dataset. The number of nodes again deceases for higher level of the threshold.

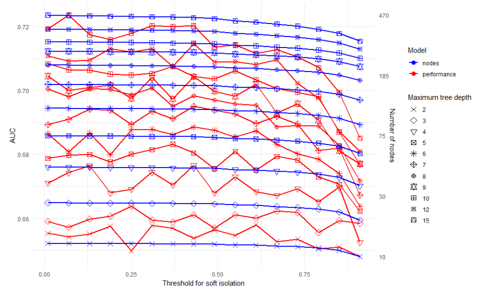
Overall, the optimal value of the threshold for soft isolation seems to be relatively high, to



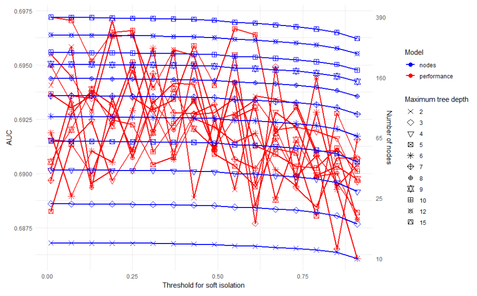
(a) Effect of the isolation threshold for different subsample sizes in the Satellite dataset



(b) Effect of the isolation threshold for different subsample sizes in the Annthyroid dataset



(c) Effect of the isolation threshold for different maximum tree heights in the Satellite dataset



(d) Effect of the isolation threshold for different maximum tree heights in the Annthyroid dataset

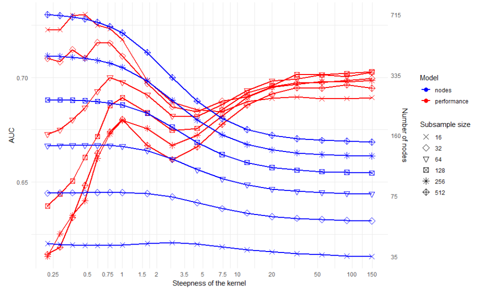
Figure 6: The effect of the isolation threshold on the AUC and number of nodes for different values of the subsample size and the maximum tree height. The number of nodes is shown on a logarithmic scale.

reduce the size of the trees. However, the threshold should be lower than the breakdown point for the subsample size used. This is interesting as the optimal value for the threshold seems to go away from 0, the setting for the threshold of the iForest algorithm. This seems to suggest that there is a added benefit in adding this parameter to the algorithm. This finding also seems to be somewhat in contrast to the findings of the last subsections, where the heatmaps seemed to imply that a stricter setting of the threshold for soft isolation could have a very positive impact on detecting anomalies.

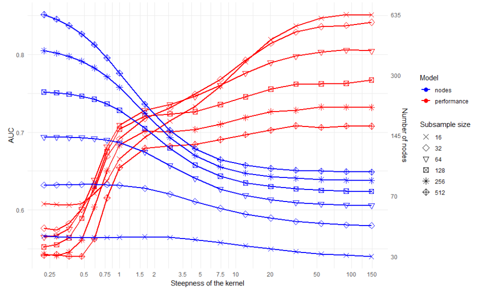
The steepness of the kernel affects the performance less homogeneously than the isolation threshold. This can clearly be seen in Subfigure 7a. For the Satellite dataset, the AUC-values increase for all subsample sizes until the steepness reaches a certain level. This level seem to be a bit lower for low subsample sizes (around 0.75) than for higher subsample sizes (around 1). After this, the predictive performance drops as the steepness goes to around 3 and then rises to a stable level for higher levels of steepness. For the lower subsample sizes the predictive performance is better when the algorithm uses a lower steepness of the kernel, whilst for higher subsample sizes the performance is best for higher levels of steepness. For the Annthyroid dataset, steepness of the kernel affects the predictive performance differently, as seen in Subfigure 7b. Predictive performance increases for low levels of the steepness, just as for the Satellite dataset, but instead of decreasing from around 1, it steadies off before increasing again. This clearly shows that a higher steepness is preferred for this dataset, which implies that soft splitting does not improve performance. This is different from the Satellite dataset, where for some subsample sizes the performance for low levels of the steepness, thus where soft splitting is used, actually outperform the algorithms using a higher steepness. For both datasets, the number of nodes in the trees decreases as the steepness goes up. This is to be expected as keeping all other parameters equal, an algorithm with a higher steepness has a bigger difference between weights given to all instances and will thus need less splits to reach the isolation threshold.

The maximum tree height has little influence on how the steepness of the kernel affects both the predictive performance and the number of nodes in the tree. In the Satellite dataset, as seen in Subfigure 7c, predictive performances are almost equal for all maximum tree heights for lower levels of the steepness, whilst the same increasing effect can be seen for higher levels of the steepness. For the Annthyroid dataset in Subfigure 7d the predictive performance are even almost the same for every maximum height for every level of steepness of the kernel. In both datasets the number nodes decreases as the steepness rises. Only for the highest maximum height the number of nodes first slightly increases until the steepness reaches around 1 and then also decreases.

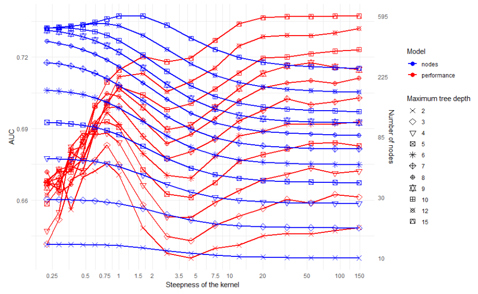
The optimal value for the steepness of the kernel is not easy to determine. For the number of nodes in the tree and thus the computational efficiency of the model, it is clearly better to set the steepness as high as possible. In the Annthyroid dataset this is also the case for the predictive performance. For the Satellite dataset, however, purely looking at predictive performance the optimal value of the steepness is around 1 depending on the subsample size. This implies that soft splitting, which happens at low levels of the steepness, has different effects in different datasets. In Section 2 it was pointed out that soft splitting could be beneficial especially in datasets with many attributes. This could possibly partly explain the results found in this subsection as the Satellite dataset has far more attributes than the Annthyroid dataset (36 attributes against 6). Further, in Section 4 it was pointed out that the Annthyroid dataset mostly contains clustered outliers, whilst the Satellite dataset contains mostly scattered outliers.



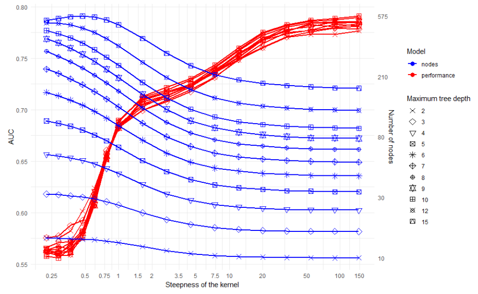
(a) Effect of the steepness of the kernel for different subsample sizes in the Satellite dataset



(b) Effect of the steepness of the kernel for different subsample sizes in the Annthyroid dataset



(c) Effect of the steepness of the kernel for different maximum tree heights in the Satellite dataset

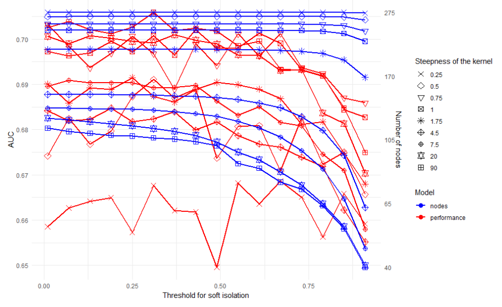


(d) Effect of the steepness of the kernel for different maximum tree heights in the Annthyroid dataset

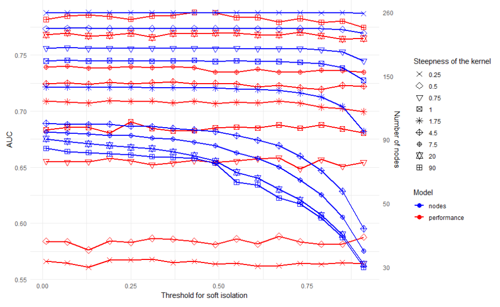
Figure 7: The effect of the steepness of the kernel on the AUC and number of nodes for different values of the subsample size and the maximum tree height. The number of nodes and the steepness of the kernel are shown on a logarithmic scale.

Finally, in Figure 8 the effects of the isolation threshold and the steepness of the kernel on each other are studied. In both datasets the predictive performance stays stable at every level of steepness for lower thresholds. In the Annthyroid dataset the predictive performance stays stable for higher levels of the isolation threshold. However, in the Satellite dataset the AUC values drop when the threshold reaches a certain level. This breakpoint is lower for higher values of the steepness. This is very similar to how the isolation threshold impacted the performance together with the subsample size. The number of nodes also decreases for higher levels of the isolation threshold, just as had been found before. This shows optimal levels of both parameters affect each other.

The differences in optimal values between the datasets stresses the need for the importance of optimisation of the hyperparameters when using the SIF algorithm. Further, as the optimal hyperparameter setting does not only depend on the dataset, but also on the setting of other



(a) Satellite dataset



(b) Annthyroid dataset

Figure 8: The predictive performance and number of nodes in a tree for different levels of steepness of the kernel and isolation threshold. The number of nodes is shown on a logarithmic scale. The values for the steepness of the kernel are rounded and not all steepness's from earlier experiments are used.

hyperparameters. This could for example be seen in the optimal isolation threshold that both depends on the subsample size and the steepness of the kernel. The need for further tuning is also amplified by the fact the the goal of the algorithm is two-sided. On the one hand the predictive performance should be as good as possible while on the other hand the computational performance is important. Depending on the balance between the importance of these goals, different combinations of hyperparameters could be optimal.

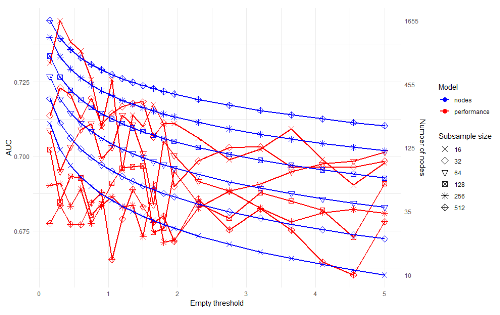
However, this thesis set out to determine the potential benefit of soft splitting in Isolation Forests. For this, the algorithm does not have to be completely optimised. To show the potential benefit of soft splitting, the main goal of this thesis is to achieve a good predictive performance with settings of the hyperparameters so that the algorithm clearly uses soft splitting. Therefore, the benchmark for the steepness of the kernel that will be used in the remainder of the results will be 1. This level is around the optimal in both datasets for the lower values of the steepness. In the Annthyroid dataset, the predictive performance can increase further by setting the steepness higher, but the algorithm would then more resemble hard splitting. The threshold for soft isolation is set to 0.5, as that seem to be high enough to decrease the number of nodes a little bit at this steepness, but not high enough to make the performance of the algorithm decrease.

5.2.2 Effect of threshold for empty nodes on performance

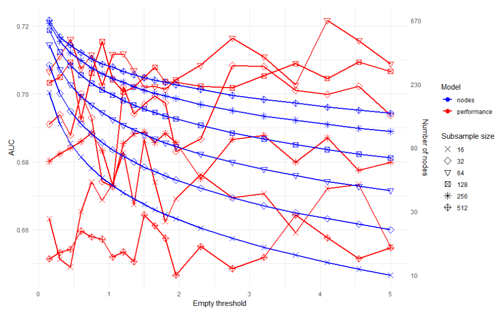
Next to the soft isolation threshold and the steepness of the kernel, this thesis also introduces the threshold for empty nodes. This parameter is meant to cut off branches that are mostly empty to improve computational performance. However, it should be ensured that branches are not cut off too early, which could impact the predictive performance. To find the optimal setting for this threshold, the predictive performance and number of nodes in the model are again set out for different levels of the subsample size and maximum tree height in Figure 9. The subsample sizes and maximum tree heights used, are the same as in the last section. The threshold for empty nodes varies between 0.15 and 5 with steps of 0.15 between 0.15 and 1.95 and steps of 0.45 between 2.3 and 5. The results are the average across 5 folds in cross-validation. The performance measures are averaged out over all values of the hyperparameter that are not shown in the figure. The number of trees is again set to 50, as this parameter is not expected to have a big impact on the results. The steepness of the kernel is set to 1 and the threshold for soft isolation to 0.5, as discussed in last subsection.

In the Satellite dataset, the AUC values stay stable for the higher subsample sizes, as can be seen in Subfigure 9a. For the lower subsample sizes, however, this is not the case. Here, after the threshold for empty nodes reaches a certain level the predictive performance drops. This is already around 0.75 for a subsample size of 16 and around 1.75 for a subsample size of 32. As the threshold is a larger proportion of the total weight for lower subsample sizes, it is logical that the algorithm suffers more from a higher threshold when it uses a lower subsample size to train. The number of nodes decreases as the threshold goes up, which is as expected. This effect seems to be equal for all different subsample sizes. For the Annthyroid dataset, shown in Subfigure 9b, the impact of the threshold on performances for the different subsample sizes seems to be the same. The only difference is that performance does not seem to be affected by the rising threshold for all subsample sizes. It is possible that the performance only drops for higher levels of the subsample size.

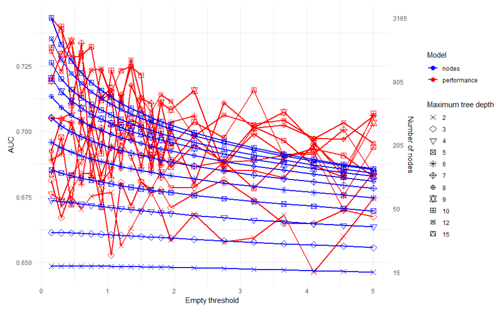
In Subfigure 9c it can be seen that for the Satellite dataset that the threshold barely has an effect on both the predictive performance and the number of nodes for low maximum tree height. This is logical as nodes are already terminated by the maximum tree height before they can reach the threshold. As the maximum tree height increases, the number of nodes in the trees start to decrease quicker for increases in the threshold. Furthermore, the predictive performance also start to slowly decreases, which is probably caused by the decreasing performance for lower subsample sizes. The influence of the maximum tree height in the Annthyroid is the same, which can be seen in Subfigure 9d. In this dataset the decrease in nodes is also more present for higher maximum tree heights. Here, the threshold again does not seem to influence the predictive performance, which also was found earlier.



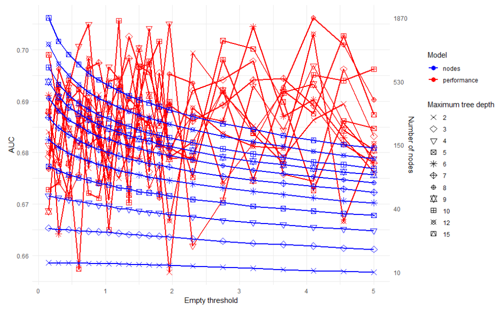
(a) Effect for different subsample sizes on Satellite dataset



(b) Effect for different subsample sizes on Annthyroid dataset



(c) Effect for different maximum tree heights on Satellite dataset



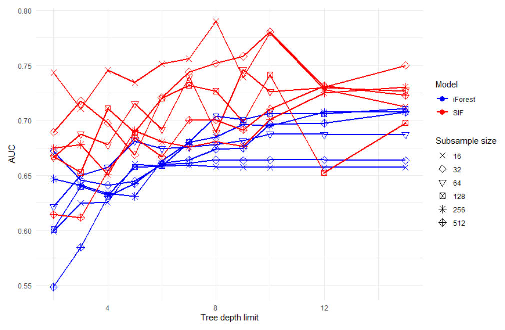
(d) Effect for different maximum tree heights on Annthyroid dataset

Figure 9: The effect of different thresholds for empty nodes on the AUC and number of nodes for different levels of the subsample size and maximum tree height. The number of nodes is shown on a logarithmic scale.

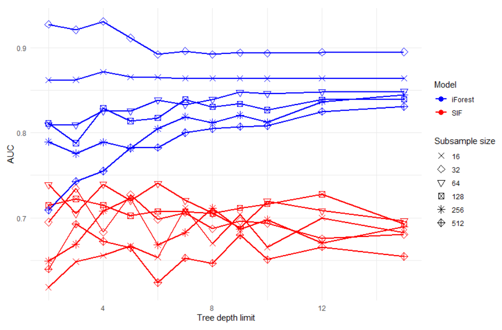
Thus, the optimal level of threshold for empty nodes seems to be as high as the given subsample size allows it. Higher values of this parameter increase the computational performance of the algorithm, but when the threshold reaches a certain level, determined by the subsample size, it starts to impact the predictive performance. The threshold for empty nodes also has a larger influence on the performance when the algorithm uses a higher maximum tree height. For the remainder of this thesis, the threshold will be set to 0.5 to make sure the predictive performance is not impacted. However, this value could be set closer to the breakpoint, especially for higher subsample sizes.

5.3 Comparison effects existing parameters between SIF and iForest

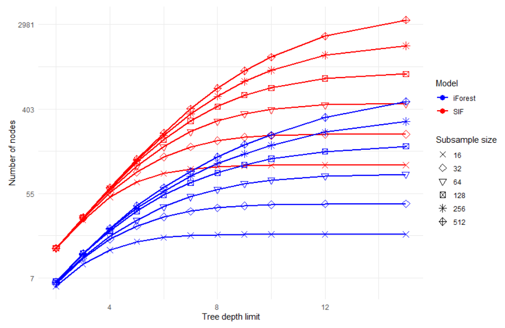
In this subsection the effects of the subsample size, the maximum tree height and the number of trees in the ensemble on the performance of the SIF algorithm are studied. Furthermore, these effects are compared to the effects that these parameters have on the iForest algorithm. For the



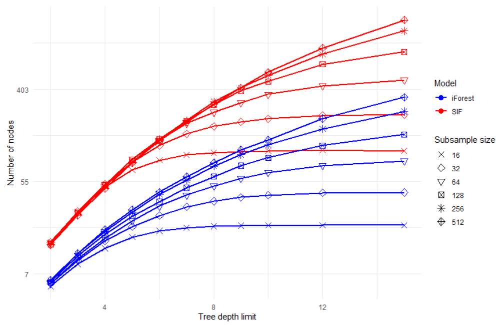
(a) AUC Satellite dataset



(b) AUC Annthyroid dataset



(c) Number of nodes Satellite dataset



(d) Number of nodes Annthyroid dataset

Figure 10: The AUC and average number of nodes for different combinations of the subsample size an tree height limit

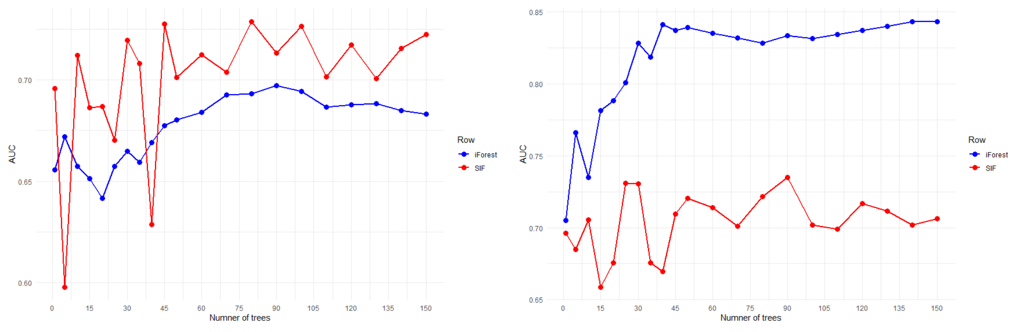
the benchmarks for the newly introduced hyperparameters set in the last subsection are used. The predictive and computational performances for different subsample sizes and three height limits of the SIF algorithm and the iForest algorithm for the Satellite and the Annthyroid datasets can be seen in Figure 10.

In Subfigure 10a it can be seen that for the Satellite dataset, the predictive performance of the SIF algorithm goes up as the subsample size goes down. This is different from the iForest algorithm, where the predictive performance rises as the subsample size goes up. Furthermore, it can be seen that the optimal maximum tree height is higher for the SIF algorithm than for the iForest algorithm. In the iForest algorithm, the performance can no longer increase for a certain maximum tree height because all instances in the subsample are already separated from each other. For the SIF, this is however not the case and the predictive performance keeps increasing for longer as the maximum tree height increases. In the Annthyroid dataset, the relation between subsample size and performance is a bit more complicated, as can be seen in Subfigure 10b. The predictive performance of the SIF is best for the middle subsample sizes, 64 and 128, and decreases when the subsample either decreases or increases. This slightly resembles the optimal subsample sizes for the iForest algorithm in this dataset. There the optimal subsample size is 32 and the predictive performance also decreases for both higher and lower values of the subsample size. The maximum tree height has does not seem to have an effect on the predictive performance of the SIF algorithm in this dataset.

In Subfigures 10c and 10d it can be seen that the number of nodes in a SIF tree grow both for higher tree height limits and higher subsample sizes, as is to be expected. The nodes in SIF are higher compared to the iForest algorithm, but the growth is similar between the two algorithms. It is notable, that for the Satellite dataset the number of nodes for low subsample sizes in the SIF algorithm is close to or lower than the number of nodes for higher subsample sizes in the iForest algorithm. Thus using when using optimal subsample sizes for both SIF and iForest, for the Satellite dataset SIF outperforms iForest on predictive performance and achieves similar tree sizes. This shows that by using soft splitting extra information per split can be used to improve the predictive performance.

Overall, optimal subsample size and tree height limit are different for the SIF algorithm than for the iForest algorithm. The tree height limit of the SIF should be set higher, as SIF seems to benefit longer from increasing maximum tree heights. The optimal subsample size for SIF is very hard to set based on these datasets as for both datasets the optimal subsample sizes behave differently from each other and from the optimal subsample size in the iForest algorithm. However, the best predictive performances seem to be around a subsample size of 64. Furthermore, choosing a lower level of subsample size also improves the computational performance. The optimal subsample size can also greatly vary per dataset based on the size of the dataset. As both datasets here have very similar number of instances, this will not be a big problem here.

Finally, the number of trees needed in the ensemble for the predictive performance to converge is compared between SIF and iForest. For this the subsample size is set to 64 for SIF and 128 for iForest as those values were found to be around optimal. For iForest the tree height is set following the commonly used benchmark to $\lceil \log_2 \psi \rceil$, where ψ is the subsample size. As the optimal tree height limit for SIF should be higher, as found earlier in this section, this benchmark is changed to $\lceil 1.25 \cdot \log_2 \psi \rceil$, raising the tree height limit with 25% for every subsample size. In Figure 11 it can be seen that the predictive performance of SIF converges for a smaller number of trees in the ensemble for both datasets. In Section 2 this was already pointed out as a possible advantage of using soft splitting. Furthermore, for all of the results this section an ensemble of 50 trees was used



(a) Satellite dataset (b) Annthyroid dataset

Figure 11: The predictive performance of SIF and iForest set out against the number of trees used in the ensemble.

to test the properties of the different parameters in the SIF algorithm. This value seems to be high enough for the performance of the algorithm to converge, thus confirming the earlier found results.

The results of this subsection show that the optimal settings of hyperparameters change when using soft splitting compared to hard splitting. The optimal subsample size seems to be lower as the improved amount of information used in every split rises. The predictive performance of SIF suffers more from lower maximum tree heights. Finally, the SIF needs slightly less trees in the ensemble for the predictive performance to converge, which could possibly indicate individual SIF trees containing more complete information about the whole sample compared to iForest trees.

5.4 Comparison performance SIF and iForest on wide range of real world datasets

To evaluate the overall performance of the SIF algorithm, it is implemented on a wide range of real world datasets and the performance on these datasets is compared to the performance of the iForest algorithm. Here, the problem of hyperparameter setting, which was alluded to earlier, becomes very relevant. As optimisation of the hyperparameters is computationally very expensive, the findings of last subsection are used to make an estimate of the optimal hyperparameter setting. The effect of different hyperparameters on the performance is however not easy to estimate and depends on both the settings of other hyperparameters and the dataset itself. Therefore, the performance of the SIF algorithm is very likely not optimal on most datasets. The iForest algorithm has a smaller number of hyperparameters and its performance is thus less dependent on the settings of those. Furthermore, Liu et al. (2008) showed that for sufficiently high values of subsample size and number of trees predictive performance of the iForest algorithm converges.

	Subsample size	Tree height limit
Http (KDDCUP99)	256	10
ForestCover	256	10
Smtp (KDDCUP99)	256	10
Shuttle	256	10
Mammography	128	9
Annthyroid	64	8
Satellite	64	8
Thyroid	64	8
Cardio	64	8
Pima	64	8
Breastw	64	8
Arrhythmia	64	8
Ionosphere	64	8

Table 2: Overview of hyperparameters used in the SIF algorithm for different datasets.

Therefore, the suggested values of Liu et al. (2008) are used as setting for the hyperparameters of the iForest algorithm. These are a subsample size of 256, the tree height limit to $\lceil \log_2 \psi \rceil$ and the number of trees to 50. For SIF the number of trees is also set to 50 and the tree height limit to $\lceil 1.25 \cdot \log_2 \psi \rceil$, which is a bit wider than for iForest as found in Subsection 5.3. Furthermore, the steepness of the kernel is set to 1, the threshold for isolation to 0.5 and the threshold for empty nodes to 0.5, as suggested in Subsection 5.2. These parameters were set to optimise the predictive performance of SIF, whilst soft splitting is performed. Therefore, the added value of soft splitting can be found when comparing the SIF algorithm with the iForest algorithm. Finally the subsample size needs to be set. In Subsection 5.3 it was shown that SIF seems to perform better for a lower value of subsample size than iForest. However, this parameter should also depend on the number of instances in the dataset. Therefore the subsample size used in the SIF algorithm is scaled based on the number of instances in every dataset. A complete overview of the hyperparameter settings can be found in Table 2. As the hyperparameters for the SIF algorithm are set with a little bit more care than the benchmarks of the iForest, these settings might slightly favour the SIF algorithm. However, as optimisation of the SIF algorithm is far more complex than that of the iForest algorithm, the predictive performance of the SIF algorithm probably still has more upwards potential.

	AUC SIF	AUC iForest
Http (KDDCUP99)	0.995 ± 0.001	**0.999** ± 0.00004
ForestCover	0.783 ± 0.046	**0.847** ± 0.020
Smtp (KDDCUP99)	0.854 ± 0.046	**0.907** ± 0.034
Shuttle	0.991 ± 0.003	**0.996** ± 0.002
Mammography	**0.863** ± 0.013	0.856 ± 0.019
Annthyroid	0.709 ± 0.033	**0.812** ± 0.033
Satellite	**0.748** ± 0.059	0.685 ± 0.033
Thyroid	0.940 ± 0.030	**0.969** ± 0.011
Cardio	0.888 ± 0.019	**0.941** ± 0.011
Pima	**0.714** ± 0.049	0.672 ± 0.045
Breastw	**0.994** ± 0.004	0.988 ± 0.008
Arrhythmia	**0.797** ± 0.103	0.793 ± 0.086
Ionosphere	**0.864** ± 0.049	0.853 ± 0.020
Average	0.857 ± 0.099	**0.871** ± 0.106

Table 3: Average AUC for SIF and iForest over every fold with the standard deviation between the folds.

The predictive performance of both the SIF and iForest algorithms can be seen in Table 3. The SIF algorithm outperformed the iForest on 6 out of the 13 datasets, meaning the iForest algorithm outperformed the SIF algorithm on 7 out of 13 datasets. Furthermore, it can be seen that the differences between the performance of both algorithms is a bit smaller for datasets that favour iForest compared to those that favour SIF. This leads to a better average performance of the iForest algorithm. The datasets on which the SIF algorithm performs better mostly have less instances, more attributes and a higher anomaly percentage. Of these properties especially the number of attributes seems to be relevant, as pointed out by Section 2. The exceptions to this are the Cardio dataset in favour of the iForest algorithm (low number of instances, many dimensions and a reasonably high anomaly percentage), the Mammography dataset in favour of the SIF algorithm (high number of instances, low number of dimensions, low anomaly percentage) and to a lesser extend the Thyroid dataset in favour of the iForest algorithm (low number of instances). An explanation of these differences could be that the estimated hyperparameter settings fit better with the properties of some datasets than others. In this regard, it is interesting to see that the two datasets on which the hyperparameter settings are based (Satellite and Annthyroid) are the datasets that favours the SIF algorithm the most (Satellite) and the dataset that favours the iForest algorithm the most (Annthyroid). Further, the soft splitting parameters, steepness of the kernel and the isolation threshold, were mostly set to optimise the Satellite dataset as the optimal values for the Annthyroid dataset converged to hard splitting. Therefore, it is not surprising that the SIF favours datasets which have similar properties to the Satellite dataset, high number of attributes and high anomaly percentage. There might be better hyperparameter settings for the steepness of the kernel and the isolation threshold, that do fit the other datasets and do not converge to hard splitting. Especially, since many of the other datasets that favour iForest have predictive performance using SIF that are closer to the performance of iForest.

Another interesting explanation for the differences in performance between the algorithms, can be found in the performances on the datasets with clustered and scattered anomalies. The SIF algorithm performs better on all four datasets that were identified to have mostly scattered anomalies (Satellite, Pima, Breastw and Ionosphere), whilst iForest massively outperforms SIF on both datasets that only had clustered anomalies (Http and Annthyroid). This strongly implies that the SIF algorithm has problems recognising clustered anomalies. In Section 2 a possible cause of this has already been identified. The area in which the intersect of the splitting hyperplane is selected, is spanned by all instances in the training set. Thus, making a split to isolate a group of clustered anomalies from each other could be very hard for the SIF. If this group is never isolated from each other, the path lengths to these instances will be indifferentiable from normal instances. Possible enhancements to the algorithm to solve this problem are already proposed in Section 3. Shrinking the area in which the intercepts can be selected could, increase the probability clustered anomalies are isolated from each other by the splits. To make sure this separation is reflected in the outputted weights, the steepness of the kernel should probably scale with the area in which the intercept is selected. Furthermore, the Gini-impurity and Entropy could possibly better identify clustered anomalies than the misclassification error, as the weights of all instances instead of only one.

In Table 4 the average time for each fold the algorithm takes to build the trees in the ensemble and score the instances in the hold-out sample is shown. The computational performance of the iForest is far better than that of the SIF algorithm. For all datasets the average computation time is at least 100 times faster and for some datasets the iForest is even 500 times faster. This difference

	Time SIF	Time iForest
Http (KDDCUP99)	53.6028 ± 3.2081	**0.2960** ± 0.0155
ForestCover	58.3617 ± 1.4739	**0.1188** ± 0.0345
Smtp (KDDCUP99)	14.3352 ± 0.5903	**0.0525** ± 0.0028
Shuttle	11.5766 ± 0.1880	**0.0185** ± 0.0096
Mammography	1.4806 ± 0.1417	**0.0073** ± 0.0007
Annthyroid	0.7523 ± 0.0527	**0.0062** ± 0.0008
Satellite	0.8791 ± 0.0451	**0.0060** ± 0.0038
Thyroid	0.5413 ± 0.0228	**0.0039** ± 0.0001
Cardio	0.4530 ± 0.0376	**0.0033** ± 0.0002
Pima	0.3849 ± 0.0245	**0.0022** ± 0.0001
Breastw	0.4189 ± 0.0539	**0.0021** ± 0.0003
Arrhythmia	0.4037 ± 0.0520	**0.0040** ± 0.0027
Ionosphere	0.4551 ± 0.0560	**0.0038** ± 0.0027
Average	11.050 ± 20.487	**0.040** ± 0.084

Table 4: Average running time of the training and scoring for SIF and iForest over every fold with the standard deviation between the folds.

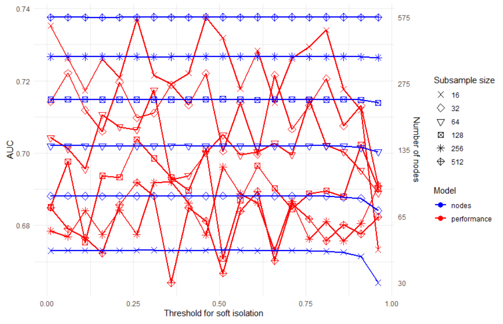
is already present for datasets with a low number of instances. This is while these datasets use a subsample size that is 4 times smaller for SIF compared to iForest. For these subsample sizes the number of nodes in the trees is comparable between the two algorithms, as was shown in Subsection 5.3. This shows that a large part of the extra computation time is caused by an increase of computations per node. Multiple causes for this increases of computations can be identified. Firstly, as every instance is in every node, both in training of the model and scoring of the test instances, the algorithm has to compare more instances to every splitting hyperplane. Secondly, the number of computations to compare an instance to the splitting hyperplane has increased. Using hard splitting, the algorithm only has to find the side of the split the instance is on. Using soft splitting, the exact distance of every instance to the splitting hyperplane needs to be determined and then also be transformed into a weight. Furthermore, during the training of the model more computations are needed to determine whether an instance is isolated or not. Finally, in the scoring phase, instead of every instance only taking on path, every instance travels down every possible path. So again are not only more computations needed to determine the weight of the instance after a split, but these computations also need to be made for an increased number of splits.

The computational performance could, however, be improved. The hyperparameters used were mostly picked to ensure the predictive performance of the model. Especially the threshold for isolation, threshold for empty nodes and maximum tree height were all set were set very conservatively for this end, whilst it was shown in Subsections 5.2 and 5.3 that these hyperparameters could influence the number of nodes and thus the computation time. If the hyperparameters were to be optimised, these could possibly be set better to also include optimising the computational performance. It should of course be noted that the act of optimising the hyperparameters would greatly raise computation times as optimisation would take a lot of computations. Furthermore, the number of trees in the ensemble for SIF is set equal to the number of trees in iForest, although Subsection 5.3 suggests that SIF might need a lower number of trees for the predictive performance to converge. Trimming down the number of trees could again lower the number of computations. There also could be other enhancements of the algorithm possible to further improve the computation time. An example of this could be the suggestion given in Section 3 to change the threshold for empty nodes to a more dynamic version in order to possibly cut off empty nodes more efficiently. Furthermore, shrinking the area in which the splitting intercept can be selected would also improve the computational performance, as pointed out in 2.

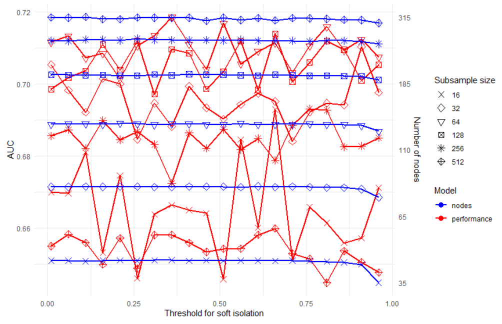
The performance of the iForest algorithm overall still seems to be better than that of the SIF algorithm. Soft splitting can be an improvement over hard splits for predictive performance, especially on datasets with many attributes and with mostly scattered anomalies. This however comes at the cost of a massively improved computation time, as the number of computations needed for soft splitting greatly increases. There is a good possibility that the predictive performance can further increase and the computation time can be decreased if the hyperparameters were to be optimised. That is however, left beyond the scope of this thesis.

5.5 Comparison measures of soft isolation

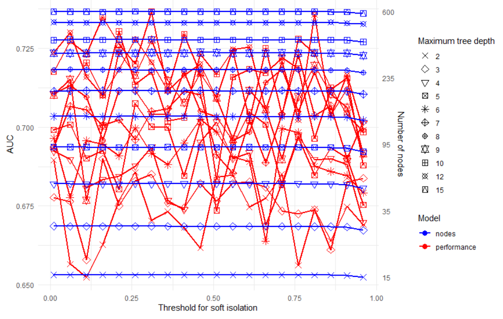
Until now, only the misclassification error was used as a measure for soft isolation. However, there are more possible options for this measure as lined out in Section 3. Furthermore, in the last subsection the Gini-impurity and Entropy were raised as possible solutions to solve the inability of SIF to detect clustered anomalies. Therefore, the performance of the SIF algorithm with these impurity measures is studied in this subsection and compared to the performance with the misclassification error. Before comparing the performance of these measure of soft isolation, firstly the threshold for soft isolation needs to be set for the new measures for soft isolation. Therefore, the effect of the threshold for soft isolation is again set out for different subsample sizes and maximum tree heights. However, the maximum value of both the Entropy and the Gini-impurity depends on the subsample size. To compare the effects of the threshold for soft isolation across different subsample sizes, the threshold is showed as fractions of the the maximum threshold for that subsample size. For this comparison the threshold for empty nodes is again set to 1, the steepness of the kernel 1 and the number of trees to 50.



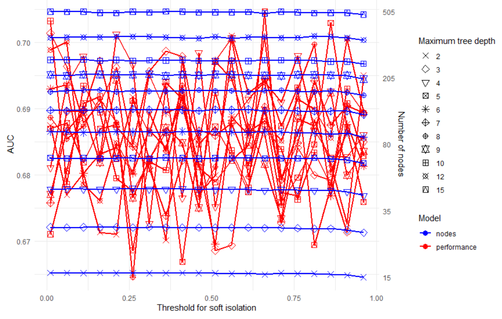
(a) Effect for different subsample sizes on Satellite dataset



(b) Effect for different subsample sizes on Annthyroid dataset



(c) Effect for different maximum tree heights on Satellite dataset



(d) Effect for different maximum tree heights on Annthyroid dataset

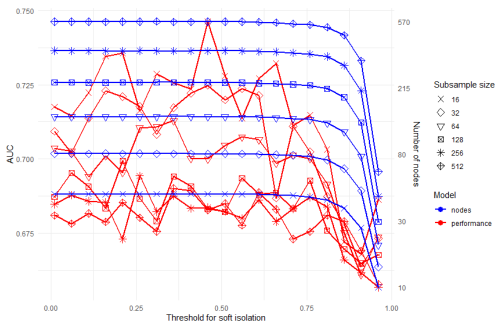
Figure 12: The effect of different values of the threshold for soft isolation on the performance of the SIF using the Gini-impurity as measure for soft isolation. The number of nodes is shown on a logarithmic scale, whilst the threshold for soft isolation is shown as a percentage of the maximum threshold for the subsample size

In Figure 12 the effect of the soft isolation threshold on the performance of the SIF using the Gini-impurity can be seen for different subsample sizes and maximum tree heights. From Subfigures 12a and 12b it can be seen that the different subsample sizes are mostly not affected by the thresholdfor soft isolation. For the Satellite dataset this however, is only until a certain breakpoint after which the predictive performance starts dropping. This breakpoint comes earlier for lower subsample sizes. This effect might not yet show for the Annthyroid dataset as the soft isolation threshold only goes up to 91% of the maximum threshold and the breakdown point might be closer to the maximum. For both datasets the number of nodes decreases slightly as the threshold for soft isolation increases. This effect is stronger for the lower subsample sizes. In Subfigures 12c and 12d it can be see that there is no clear effect of or difference between the different maximum tree heights. The predictive performance goes down slightly in the Satellite dataset for all maximum tree heights, but that is probably caused by the lower subsample sizes breaking down. Overall, the effects of the threshold for soft isolation seem very comparable to the effects when the misclassification error was used. Again, the threshold should be set as high as possible to decrease the number of nodes in the trees, but not high enough to make the predictive performance break down. For the Gini-impurity this seems to be around 75% of the maximum value of the threshold.

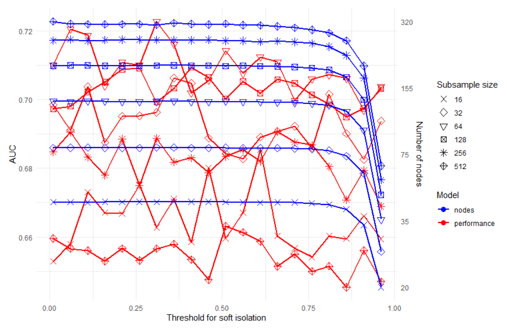
The effects of the soft isolation threshold when the Entropy is used, can be seen in Figure 13. These again follow the same patterns as for the Gini-impurity and the misclassification error. However the breakdown point seems to come earlier already around 60% and the drop off for both the predictive performance and the number of nodes is way bigger.

To compare the different measures for soft isolation the performances of the SIF algorithm with the different measures are set out against each other. As the use of a different soft isolation measure should only impact the soft isolation threshold, the same settings for the algorithm are used as in Subsection 5.4. This means the number of trees in the SIF is set to 50, the subsample size and the tree height limit used are given in Table 2, the steepness of the kernel is set to 1 and the threshold for empty nodes is set to 0.5. The soft isolation threshold is set to 0.5 for the misclassification error, to 75% of the maximum value of the threshold for the Gini-impurity and to 55% of the maximum value of the threshold for the Entropy.

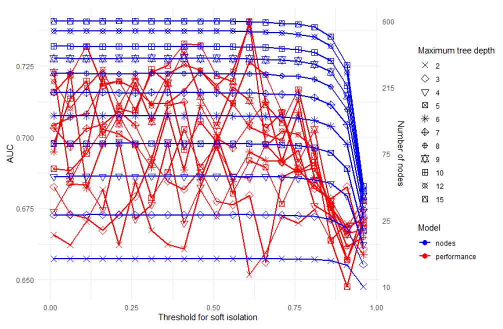
In Table 5 the predictive performance of the SIF using the different measures for soft isolation are given for the real-world datasets. There is not much of a difference between the measures as the misclassification error and the Gini-impurity outperform the others for 4 datasets and Entropy outperforms the other on 5 datasets. On average the misclassification error has the best predictive performance but the differences between all measures all small. Entropy seems to perform slightly better for datasets with a small number of instances and more attributes, whilst the Gini-impurity



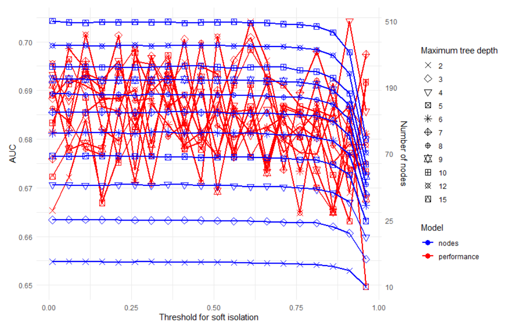
(a) Effect for different subsample sizes on Satellite dataset



(b) Effect for different subsample sizes on Annthyroid dataset



(c) Effect for different maximum tree heights on Satellite dataset



(d) Effect for different maximum tree heights on Annthyroid dataset

Figure 13: The effect of different values of the threshold for soft isolation on the performance of the SIF using the Entropy as measure for soft isolation. The number of nodes is shown on a logarithmic scale, whilst the threshold for soft isolation is shown as a percentage of the maximum threshold for the subsample size

performs slightly better for datasets with a large number of instances and less attributes. It is interesting that the extra information used by using the weights of multiple instances in the Gini-impurity and the Entropy does not lead to an increase in predictive performance. The Gini-impurity and Entropy also only seem to perform slightly better on one of the datasets with clustered anomalies, that being the Annthyroid. More sophisticated measures for soft isolation thus do not seem to be the solution to the inability of the SIF algorithm to detect clustered anomalies.

The computational performance of the SIF algorithm for the different soft isolation measures is given in Table 6. The Gini-impurity has the best computational performance on 6 of the 13 datasets, Entropy on 5 and the misclassification error on 2. On average, Entropy has the best computational performance. This is mostly caused by the computational performance on the Http dataset. As

	AUC Misclassification	AUC Gini	AUC Entropy
Http (KDDCUP99)	0.995 ± 0.001	**0.996** ± 0.0003	0.995 ± 0.00006
ForestCover	0.783 ± 0.046	**0.817** ± 0.027	0.810 ± 0.045
Smtp (KDDCUP99)	**0.854** ± 0.046	0.832 ± 0.075	0.810 ± 0.095
Shuttle	0.991 ± 0.003	0.991 ± 0.004	**0.991** ± 0.003
Mammography	0.863 ± 0.013	**0.864** ± 0.011	0.861 ± 0.010
Annthyroid	0.709 ± 0.033	**0.734** ± 0.043	0.723 ± 0.076
Satellite	**0.748** ± 0.059	0.714 ± 0.049	0.715 ± 0.047
Thyroid	**0.940** ± 0.030	0.940 ± 0.025	0.931 ± 0.029
Cardio	0.888 ± 0.019	0.881 ± 0.014	**0.890** ± 0.022
Pima	0.714 ± 0.049	0.710 ± 0.041	**0.715** ± 0.047
Breastw	**0.994** ± 0.004	0.994 ± 0.003	0.994 ± 0.004
Arrhythmia	0.797 ± 0.103	0.797 ± 0.087	**0.807** ± 0.086
Ionosphere	0.864 ± 0.049	0.861 ± 0.039	**0.872** ± 0.057
Average	**0.857** ± 0.103	0.856 ± 0.103	0.855 ± 0.103

Table 5: Predictive performance of the SIF algorithm using different measures for soft isolation for various real-life datasets

the computation time for this dataset is already way larger, this average is heavily skewed. The Gini-impurity and Entropy have a better computational performance, whilst the computation of the the misclassification error should be slightly easier. This seems to indicate that using these, the algorithm might be able to prune isolated branches earlier leading to smaller trees. However, the difference between the measures for soft isolation is still small.

Overall the choice of measure for soft isolation does not seem to have a big impact on both the predictive performance and the computational performance of the SIF algorithm. Finally, it should be noted that as all hyperparameters settings are the same in these experiments, except for the measure for soft isolation and the soft isolation threshold. As the soft isolation threshold, is picked somewhat arbitrarily, some of the differences between the measures could not be cause by the measure itself, but by the handpicked threshold.

	Time Misclassification	Time Gini	Time Entropy
Http (KDDCUP99)	53.6028 ± 3.2081	55.9468 ± 3.3510	**42.9951** ± 23.6633
ForestCover	58.3617 ± 1.4739	**55.6237** ± 0.7398	56.4846 ± 1.7201
Smtp (KDDCUP99)	14.3352 ± 0.5903	13.5745 ± 0.4483	**13.4521** ± 0.5045
Shuttle	11.5766 ± 0.1880	**11.4440** ± 0.3363	11.7521 ± 0.3897
Mammography	**1.4806** ± 0.1417	1.5906 ± 0.1572	1.5132 ± 0.0517
Annthyroid	0.7523 ± 0.0527	0.7583 ± 0.0579	**0.7204** ± 0.0297
Satellite	0.8791 ± 0.0451	0.8667 ± 0.0330	**0.8494** ± 0.0315
Thyroid	0.5413 ± 0.0228	0.5537 ± 0.0676	**0.5351** ± 0.0222
Cardio	**0.4530** ± 0.0376	0.4534 ± 0.0525	0.4932 ± 0.0548
Pima	0.3849 ± 0.0245	**0.3798** ± 0.0491	0.4112 ± 0.0498
Breastw	0.4189 ± 0.0539	**0.4067** ± 0.0314	0.4307 ± 0.0507
Arrhythmia	0.4037 ± 0.0520	**0.3865** ± 0.0254	0.4076 ± 0.0381
Ionosphere	0.4551 ± 0.0560	**0.4493** ± 0.0434	0.4695 ± 0.0419
Average	11.050 ± 19.683	10.956 ± 19.580	**10.040** ± 17.656

Table 6: Computational performance of the SIF algorithm using different measures for soft isolation for various real-life datasets

6 Conclusion

This thesis set out to determine the potential value of soft splitting in Isolation Forests. Due to soft splitting, the information used in every split can increase. This leads to several advantages in both training of the ensemble and scoring the test data. For datasets with many attributes and without clustered anomalies, this leads to an improvement of the predictive performance. However, the computational performance declines massively due to a the increased number of computations needed to perform soft splitting. This is a major drawback of using soft splitting as the low computational complexity is one of the biggest advantages of Isolation Forests over the traditional model-based anomaly detection methods.

A major drawback of this research is that the hyperparameters that are used to evaluate the performance of the SIF algorithm are not optimised. If the hyperparameters were to be optimised this could lead to both a better predictive and computational performance. A comprehensive analysis is done on the effect of all hyperparameters on the performance of the SIF algorithm. From this it is found that the optimal hyperparameters settings depend on both the characteristics of the dataset and the setting of other hyperparameters, further accentuating the need for a good tuning procedure. The hyperparameters used are set using this analysis and are set to study the effect of soft splitting above optimal predictive or computational performance.

Next to optimisation, future work should mostly focus on enhancements to improve the computational performance of the algorithm. In this thesis several suggestions are made for this. For example, a dynamic threshold for empty nodes is suggested, which could possibly terminate empty branches earlier. Furthermore, limitations on the area in which the splitting point is selected could be introduced to reduce the probability of making unnecessary or duplicate splits. The other major point of focus for future work should be the ability to detect clustered anomalies. A shrinking area in which the splitting point is selected, possibly combined with steepness of the kernel that is scaled by the size of this area, is again raised as a solution for this problem. Furthermore, possibilities to improve the predictive performance profiting from the more general definition of the algorithm could be found. For example, the level of soft isolation in a node , could possibly also be used to enhance the scoring by combining it with the depth of the node. Finally, soft splitting is a possible extension to almost all earlier extension of the iForest algorithm. Combinations of these could lead to interesting outcomes.

References

Aggarwal, C. C. and Aggarwal, C. C. (2017). *An introduction to outlier analysis*. Springer.

Cao, Y., Xiang, H., Zhang, H., Zhu, Y., and Ting, K. M. (2024). Anomaly detection based on isolation mechanisms: A survey. *arXiv preprint arXiv:2403.10802*.

Cortes, D. (2021). Revisiting randomized choices in isolation forests. *arXiv preprint arXiv:2110.13402*.

Hariri, S., Kind, M. C., and Brunner, R. J. (2019). Extended isolation forest. *IEEE transactions on knowledge and data engineering*, 33(4):1479–1489.

Irsoy, O., Yıldız, O. T., and Alpaydın, E. (2012). Soft decision trees. In *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*, pages 1819–1822. IEEE.

Lesouple, J., Baudoin, C., Spigai, M., and Tourneret, J.-Y. (2021). Generalized isolation forest for anomaly detection. *Pattern Recognition Letters*, 149:109–119.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2010). On detecting clustered anomalies using sciforest. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part II 21*, pages 274–290. Springer.

Liu, T., Zhou, Z., and Yang, L. (2024). Layered isolation forest: A multi-level subspace algorithm for improving isolation forest. *Neurocomputing*, 581:127525.

Ma, M. Q., Zhao, Y., Zhang, X., and Akoglu, L. (2023). The need for unsupervised outlier model selection: A review and evaluation of internal evaluation strategies. *ACM SIGKDD Explorations Newsletter*, 25(1):19–35.

Shebuti, R. (2016). Odds library https://odds.cs.stonybrook.edu.

Tokovarov, M. and Karczmarek, P. (2022). A probabilistic generalization of isolation forest. *Information Sciences*, 584:433–449.

Zhang, X., Dou, W., He, Q., Zhou, R., Leckie, C., Kotagiri, R., and Salcic, Z. (2017). Lshiforest: A generic framework for fast tree isolation based ensemble anomaly analysis. In *2017 IEEE 33rd international conference on data engineering (ICDE)*, pages 983–994. IEEE.

A Oblique splitting using soft splitting

The algorithm for building Soft Isolation Trees in 2 can easily be adapted into using oblique splitting instead of axis parallel splitting just as the EIF in Hariri et al. (2019). This adaptation is implemented in Algorithm 4. In the adapted algorithm, the process of making a note external in

Algorithm 4 Soft Isolation Tree

Inputs: X - input data, w - weight of every data point, e - current tree height, l - height limit, i - stopping threshold for isolation measure, o - stopping threshold for empty nodes, k - base steepness of the kernel

Output: a Soft Isoltaion Tree

```
1: if $e \geq l$ or $s(w) \leq i$ or $\max(w) < o(e)$ then
2:     return $exNode\{Weights \leftarrow w\}$
3: else
4:     Randomly select a normal vector $\vec{n} \in \mathbb{R}^{|X|}$ by drawing each coordinate of $\vec{n}$ from a standard
       Gaussian distribution
5:     Randomly select an intercept point $\vec{p} \in \mathbb{R}^{|X|}$ in the range of $X$
6:     distances $\leftarrow sigd(X, \vec{p}, \vec{q})$
7:     $k \leftarrow \frac{k}{sd(distances)}$
8:     $w_l \leftarrow g(X, \vec{n}, \vec{p}, k)$
9:     $w_r \leftarrow 1 - w_l$
10:    return $inNode\{Left \leftarrow \text{Soft\_iTree}(X, w * w_l, e+1, l, i, o, k),$
11:                     $Right \leftarrow \text{Soft\_iTree}(X, w * w_r, e+1, l, i, o, k),$
12:                     $Normal \leftarrow \vec{n},$
13:                     $Intercept \leftarrow \vec{p},$
14:                     $Steepness \leftarrow k\}$
15: end if
```

row 1 to 3 does not change. The oblique split in every node is determined by the vectors $\vec{n}$ and $\vec{p}$ which are determined in row 4 and 5. $\vec{n}$ is the normal vector of the hyperplane that gives the split. This can be simplified to axis-parallel splitting by having only one of the values of $\vec{n}$ be non-zero. Then the split is parallel to the attribute to which the non-zero value belongs. $\vec{p}$ is the intercept of the hyperplane, which is randomly chosen on the area of X.

An important change from the axis-parallel splitting is the scaling of k by the standard deviation of the signed distances in row 7. Although the data is standardised for every attribute, the data is not standardised for every possible splitting hyperplane. This means that the direction of the hyperplane $\vec{n}$ determines the weights of the instances after the split. To prevent this, the steepness of the kernel can be scaled with the standard deviation of the distances. This ensures that every effect of every split is on the weights is equal.

The weights can then be calculated with the new steepness of the kernel k as usual. This happens on row 8 of the algorithm, which also can reuse the distances calculated on row 6. A final change in the algorithm building trees using oblqiue splitting is that the steepness k is stored. This is because the scaling of the steepness of the kernel is determined by the training data and thus cannot be retrieved in the scoring phase unless it is stored. The scoring algorithm thus has to be slightly altered to include using the stored steepness of the kernel. This altered version can be seen in Algorithm 5.

Algorithm 5 WeightedPathLength

Input: x - an instance, T - an iTree, e - current path length; to be initialized to zero when first called

Output: path length of x

```
1: if $T$ is an external node then
2:     return $e$ + c(T.Weights)
3: end if
4: $\vec{n} \leftarrow T$.Normal
5: $\vec{p} \leftarrow T$.Intercept
6: $k \leftarrow T$.Steepness
7: $w_l \leftarrow g(x, \vec{n}, \vec{p}, k)$
8: $w_r \leftarrow 1 - w_l$
9: return $w_l \cdot$ WeightedPathLength($x$, $T$.left, $e+1$) + $w_r \cdot$ WeightedPathLength($x$, $T$.right, $e+1$)
```