UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

Unsupervised Anomaly Detection: investigations on Isolation Forest

Supervisor
Prof. Gian Antonio Susto
Co-supervisor
Tommaso Barbariol

Candidate
Vincenzo Savarino

ACADEMICAL YEAR 2021-2022
DATA 14/04/2022

ABSTRACT (ENG)

In today's world, the increasing amount of available information makes it possible to analyse several factors. One of these factors is anomaly detection. In recent years, this problem has been addressed by machine learning, which makes it possible to recognise instances that do not conform to the expected behaviour of a system, so-called outliers.

One of the sectors that benefits most is the industrial sector, where data is the new wealth of industries, just think of boosting sales or predictive maintenance.

Over the years several classes of methods have been proposed, recently a new class based on isolation has been introduced. The first method of the isolation-based class is Isolation Forest. This method has been very successful both in industrial applications and in academic research, which has made a large number of variants available. The basic intuition is very simple, that is, the anomaly score reflects the propensity of each instance to be separated, based on the average number of random splits required to completely isolate a data instance.

In this thesis, after a preliminary survey of the state of the art and an in-depth study of the Isolation Forest method, several variants of this method are developed, with the aim of improving anomaly detection. These variants were developed thanks to insights into the two main phases, the phase where the feature and its split value are selected and the phase where the anomaly score is calculated for each instance. In conclusion, numerical experiments are provided, using both Artificial and Real World datasets, with the aim of comparing performance in terms of anomaly detection.

These experiments have shown that the Prob Split method appears to be the most promising of all those developed, because it has significant gains in detection and maintains the same computational cost as the Isolation Forest method.

ABSTRACT (ITA)

Nel mondo di oggi, la crescente quantità di informazioni disponibili rende possibile analizzare diversi fattori. Uno di questo fattori è il rilevamento delle anomalie. Negli ultimi anni questo problema viene affrontato grazie al machine learning, il quale permette di riconoscere le istanze che non sono conformi al comportamento atteso di un sistema, i cosiddetti outlier.

Uno dei settori che trae maggiore beneficio è quello industriale, dove i dati sono la nuova ricchezza delle industrie, basta pensare al potenziamento delle vendite o alla manutenzione predittiva.

Negli anni sono stati proposti diverse classi di metodi, recentemente è stata introdotta una nuova classe basata sull'isolamento. Il primo metodo della classe basata sull'isolamento è Isolation Forest. Questo metodo ha riscosso un grande successo sia nelle applicazioni industriali sia nella ricerca accademica rendendo disponibile una notevole quantità di varianti. L'intuizione di base è molto semplice, ovvero, il punteggio di anomalia riflette la propensione di ogni istanza ad essere separata, in base al numero medio di suddivisioni casuali necessarie per isolare completamente un istanza di dati.

In questo lavoro di tesi, dopo un'indagine preliminare dello stato dell'arte e un approfondimento del metodo Isolation Forest, vengono sviluppate diverse varianti di questo metodo, con l'obiettivo di migliorare il rilevamento delle anomalie. Queste varianti sono state sviluppate grazie a delle intuizioni sulle due fasi principali, la fase dove si selezione la caratteristica e il relativo valore di split e la fase dove si calcola il punteggio di anomalia per ogni istanza. In conclusione vengono forniti degli esperimenti numerici, utilizzando sia set di dati Artificiali sia set di dati del mondo Reale, con lo scopo di confrontare le prestazioni con il metodo standard, in termini di rilevamento di anomalie.

Questi esperimenti hanno dimostrato che il metodo Prob Split sembra essere il più promettente tra tutti quelli sviluppati, perché ha incrementi delle prestazioni significativi nel rilevamento e mantiene il costo computazionale invariato.

Acknowledgements

In conclusione di questo percorso, sono tante le persone che sento di dover ringraziare.
Il primo ringraziamento va al Professor Gian Antonio Susto, per avermi dato l'opportunità di collaborare con il suo gruppo di ricerca, per i preziosi consigli, per l'aver creduto in me e la pazienza dimostratami durante la stesura di questa tesi,.
Ringrazio Tommaso, per il suo supporto durante tutta la stesura della tesi, per il continuo confronto con le nostre chiamate settimanali dove discutevamo del lavoro da svolgere e dei progressi fatti, per la voglia e la passione che mi ha trasmetto.
Un sentito ringraziamento va inoltre ai miei genitori, per tutti i sacrifici che fate e che avete fatto, per aver reso possibile questo traguardo. Il loro costante incoraggiamento è stato fondamentale durante tutto il percorso di studi.
Grazie a mia sorella Annamaria per essere sempre presente. Grazie per le tue chiamate dopo gli esami (anche se già sapevi l'esito).
Ringrazio i miei amici Stefano, Jasmina, Mattia e Giulia, con voi ho iniziato e concluso questo percorso.
Ringrazio Monica per essermi stata vicina e sempre pronta ad aiutarmi senza esitare. Grazie per tutto il tempo che mi hai dedicato. Grazie perché ci sei sempre stata.
Grazie a me stesso, ai miei sacrifici e alla mia tenacia che mi hanno permesso di arrivare fin qui.

Contents

1 Introduction **1**

1.1 Structure . . . 2

2 Anomaly Detection **3**

2.1 Anomaly . . . 3

2.1.1 Types of anomalies . . . 3

2.2 Anomaly Detection Methods . . . 5

2.2.1 Statistical-Based methods . . . 6

2.2.2 Distance-Based methods . . . 7

2.2.3 Density-Based methods . . . 8

2.2.4 Model-Based methods . . . 10

2.3 Anomaly Detection Metrics . . . 13

2.3.1 Receiver Operating Characteristic . . . 15

2.3.2 Precision-Recall Curve . . . 15

3 Isolation Forest **17**

3.1 Training Algorithm . . . 18

3.2 Testing Algorithm . . . 20

3.3 A practical example of an isolation tree . . . 21

4 Implementations **24**

4.1 Split Criterion . . . 24

4.1.1 Prob split . . . 24

4.1.2 Neighbours . . . 26

4.1.3 Component Gap . . . 28

4.1.4 Bubble Forest . . . 30

4.2 Anomaly Score Criterion . . . 32

4.2.1 Weighted Path . . . 33

4.2.2 Bubble Forest Weighted Path . . . 34

5 Results **35**

5.1 Datasets . . . 35

5.1.1 Artificial 35
5.1.2 Real 36
5.2 Experimental setup 37
5.3 Results of the split criterion experiments 37
5.3.1 Prob Split 37
5.3.2 Neighbours 40
5.3.3 Component Gap 43
5.3.4 Bubble Forest 49
5.3.5 Discussion on artificial datasets 52
5.3.6 Discussion on real datasets 54
5.4 Results of the anomaly score criterion experiments 56
5.4.1 Weighted Path 56
5.4.2 Bubble Forest Weighted Path 59
5.4.3 Discussion on artificial datasets 62
5.4.4 Discussion on real datasets 64
5.5 Other Results 66

6 Conclusions **68**

Chapter 1

Introduction

Continuous competition in industry is driving industries towards ever shorter product development and improved production efficiency. This is possible thanks to the increased automation and monitoring of production machines. In the context of Industry 4.0, data is the new wealth for industry, just think about the enhancement of sales processes (new customer groups can be defined using predictive analysis) or predictive maintenance (identifying a failure using predictive analysis).

However, data acquisition by companies is a critical process both in terms of correctness (they are not always labelled correctly) and cost. Factories and industrial equipment generate more and more data that are difficult to monitor with traditional approaches, so more reliable anomaly detection systems are needed to manage industrial machines.

Machine learning (ML) and more recently deep learning (DL) enjoy considerable popularity for anomaly detection. The main advantages of these data-driven systems are their ability to capture non-linear phenomena, adapt to many different processes, learn incrementally and improve over time Chalapathy and Chawla (2019a), Mahdavinejad et al. (2018), Görnitz et al. (2014). There are several application cases of industrial machinery monitoring in the literature, such as in power generation systems Cuccu et al. (2017), industrial oil machinery such as engine turbines and pipelines Martí et al. (2015), vehicle and aircraft engines Malhotra et al. (2016).

Nowadays, inadequate maintenance techniques can reduce the overall production capacity of a piece of equipment by 5 to 20%. Machinery downtime that has not been planned due to breakdowns costs industrial manufacturing organisations around $50 billion every year.

Anomaly detection has the important task of analysing data and detecting anomalous data from a dataset. This is an interesting area of data mining research as it involves the discovery of data patterns. Anomalies are considered important because they indicate significant but rare events and can prompt critical actions in a wide range of application domains such as fraud and fake news detection John and Naaz

(2019); Liu et al. (2019), financial surveillance, health and medical risk, healthcare Šabić et al. (2021), astronomy Lochner and Bassett (2021), and cybersecurity.

As mentioned earlier, Anomaly Detection, also known as Outlier Detection, refers to the process of identifying anomalous behaviour that does not conform to expected patterns, in other words, it refers to the process of detecting data instances that deviate significantly from the majority of data instances. Outliers were found as part of the data cleaning process. However, this conception changed in 2000 when researchers discovered that outlier detection can help solve real-world problems. The importance of outlier detection is due to the fact that outliers can be translated into important operational information in various application domains. For example, an unusual traffic pattern in a network could mean that a computer has been hacked and data is being transmitted to unauthorised destinations, abnormal behaviour in credit card transactions could indicate fraudulent activity, and an anomaly in an MRI image could indicate the presence of a malignant tumour.

A well-known algorithm in the field of anomaly detection is the Isolation Forest (IF). It belongs to the class of so-called isolation-based algorithms, which are very simple, efficient and have a very low computational cost compared to other anomaly detection algorithms. Some authors have indicated it as the best performing anomaly detection method in the state-of-the-art Emmott et al. (2015).

One of the main contributions of this thesis will be to provide new variants of the standard IF algorithm. These variants have been divided according to two criteria, modification of the feature selection process and modification of the anomaly score function.

1.1 Structure

In Chapter 2, I provide a broad overview of the state-of-the-art of anomaly detection, basically discussing the most popular approaches.

Chapter 3 is a deep dive into the Isolation Forest Algorithm. In the last part, there is a practical example of building a forest consisting only of an isolation tree with a step-by-step graphical presentation.

In Chapter 4 I present all the modifications made to the Isolation Forest algorithm. These changes are grouped into two groups, the first group concerns changes to the split criterion, while the second group concerns changes to the scoring function.

In Chapter 5, numerical experiments and results are described and discussed, conducted in order to test the modifications and compare their performance against Isolation Forest.

Chapter 2

Anomaly Detection

This chapter has two objectives: The first will be to present the basic terms and concepts, and the second will be to present existing studies and algorithms on anomaly detection.

2.1 Anomaly

Several definitions have been formulated concerning the anomaly and based on that choice the behaviour of anomaly detection algorithms changes. The widely accepted has been formulated by Hawkins (1980): *"Observation which deviates so much from other observations as to arouse suspicion it was generated by a different mechanism"*.

2.1.1 Types of anomalies

To understand the dynamics of an anomaly detection technique, it is essential to establish the nature of the anomalies. In the literature, following one of the most cited surveys Chandola et al. (2009) anomalies are classified into the following three categories: point anomalies, contextual anomalies and collective anomalies.

Point anomaly

A point anomaly occurs when a single instance in a given data set differs from the others in terms of attributes, but clusters of such anomalies can also occur. An example is shown in Figure 2.1 where o_1, o_2 are point anomalies and O_3 is a cluster of point anomalies.

Context anomaly

An anomaly is contextual or conditional when a data instance behaves abnormally in the considered context. The context is defined by the nature of the data. An

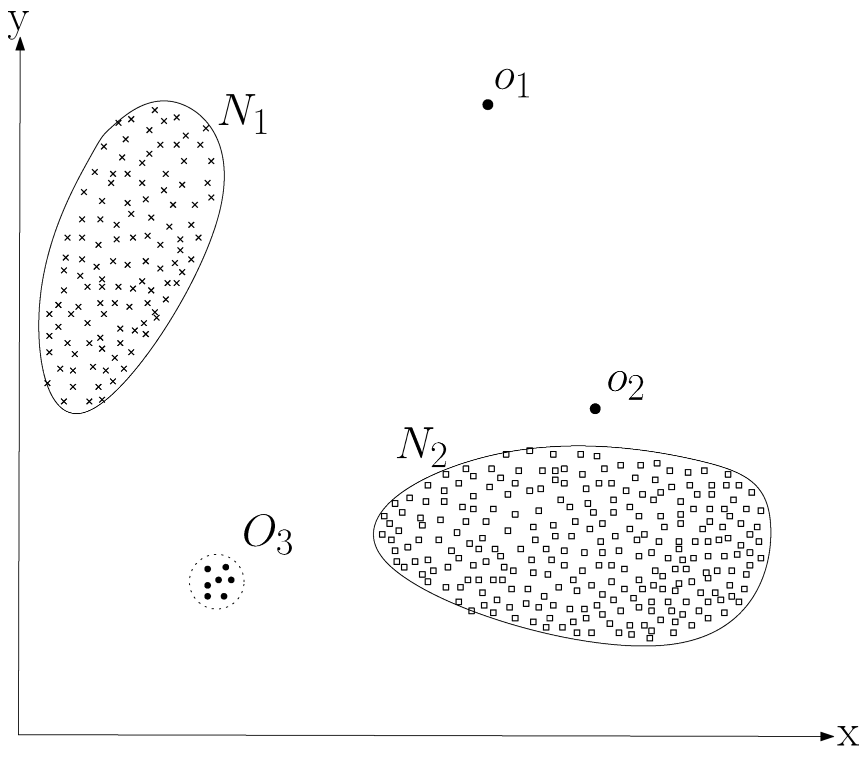


Figure 2.1: Example of point anomalies Chandola et al. (2009)

example is shown in Figure 2.2 where the context is defined by a monthly temperature series at a specific location, The Figure shows that the temperature at time t_1 is equal to time t_2, but in relation to the context, the temperature at time t_2 could be anomalous with respect to the temperature at time t_1.

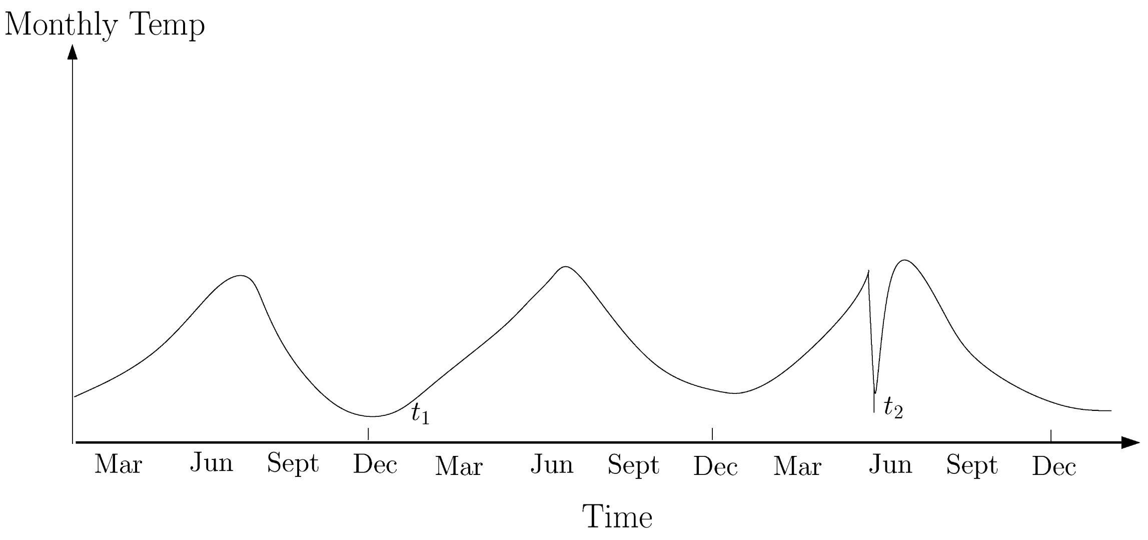


Figure 2.2: Example of contextual anomaly Chandola et al. (2009)

Collective anomaly

An instance may be normal when taken individually, but when a set of similar data instances behave anomalously with respect to the entire data set, the group of data instances is referred to as a collective anomaly, an example is shown in Figure 2.3 which shows a human electrocardiogram, where the low value persists over time and replaces the next waveform. In the literature, anomalies are classified into two

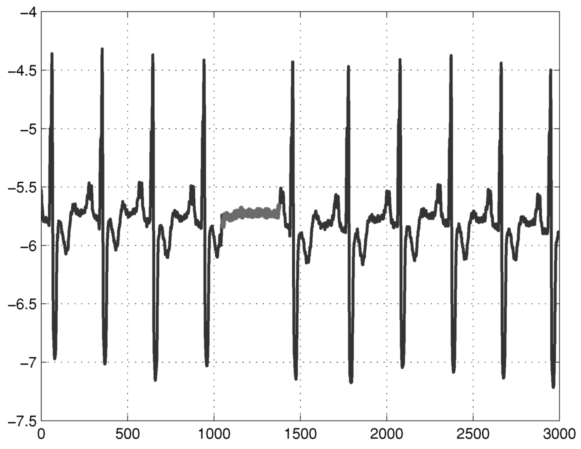


Figure 2.3: Example of collective anomalies Chandola et al. (2009)

groups, the first group distinguishes anomalies based on their proximity from normal instances (global or local), the second group classifies anomalies based on their data distribution (clustered or scattered).

2.2 Anomaly Detection Methods

Several factors must be considered in anomaly detection, such as the input data, the type of anomalies, the availability of data labels and the output of anomaly detection.

Input refers to a collection of instances or observations that are described by a number of features that can take discrete or continuous values.

The existence of data labels is a factor that plays an important role in anomaly detection systems. Depending on the existence or non-existence of these labels, supervised, unsupervised or semi-supervised anomaly detection techniques may be used. Supervised techniques require the existence of all labels, semi-supervised techniques require only a small number of labels, while unsupervised techniques do not require the existence of any labels.

The output of such a system may be a label or score that provides a measure or degree of how abnormal an observation is considered to be.

Several anomaly detection algorithms have been developed over the years and each anomaly detection method uses one of the definitions of anomaly described above (Section 2.1). Depending on the method chosen, a point may have a different anomaly score due to the different behaviour of the algorithm.

The purpose of this section is to introduce some of the most popular techniques, grouped into categories based on their principle of operation, in order to provide the basics of fundamental algorithms

2.2.1 Statistical-Based methods

Outliers can be identified by creating a statistical distribution model and identifying as anomalies data points that occupy the ends of the distribution. An example is shown in Figure 2.4. The normal distribution is the basis of these techniques because it approximates many natural phenomena. The key parameters, mean and standard deviation, for creating the normal distribution curve can be estimated from the dataset.

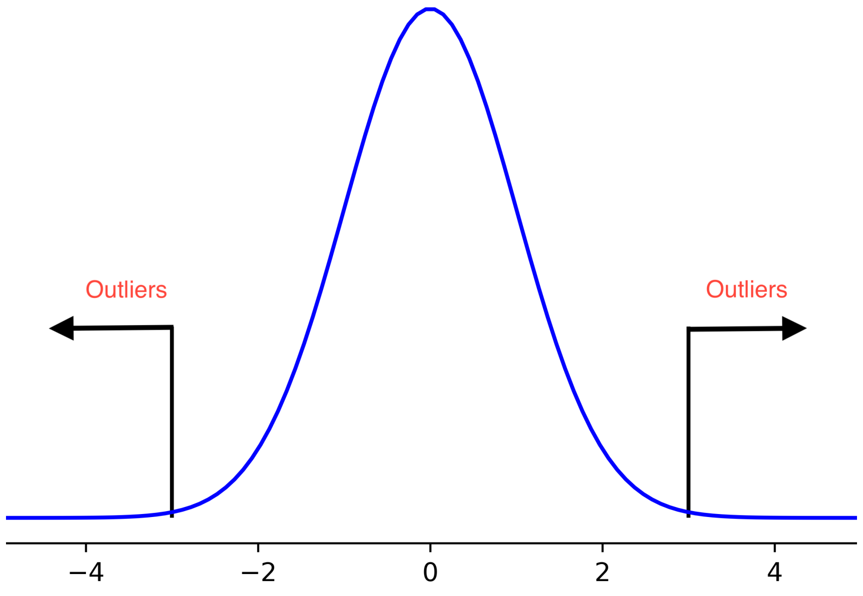


Figure 2.4: Normal distribution and outliers.

It is possible to identify outliers according to their position in the standard distribution curve, usually a threshold is specified to assess whether a point is outlier or not, for example 3σ, and all points that are beyond this threshold are classified as anomalous, since they deviate from normal behaviour represented by the normal distribution Muruti et al. (2018). This method considers only one dimension at a time (univariate statistical analysis), but there are static methods that consider multiple dimensions (multivariate statistical analysis), for example, they calculate the Mahalanobis distance, which was introduced by P. C. Mahalanobis in 1936 Mahalanobis (1936). This metric expresses the measure of the distance between a point P and a distribution D, in other words, it is a multidimensional generalization, it is intended to measure how many standard deviations of P are away from the mean of multivariate D.

These methods are the simplest, in fact, a limitation of using this method to detect outliers is that the distribution of the data set is not known a priori, in this case non-parametric techniques are used, an example of this case is the histogram-based model, but even if it were known a priori, in this case parametric techniques are used, but the actual data does not always fit the model.

2.2.2 Distance-Based methods

Distance or proximity-based approaches are based on the principle that outlier instances are distant far from normal instances (inliers), this is because outlier instances are different from normal instances in the dataset in at least one feature.

Considering the multidimensional Cartesian plane, outlier instances are distant from normal instances, as shown in Figure 2.5 . Distance-based algorithms use a fairly simple property to identify anomalous instances from normal instances, i.e., Calculating the average distance of the k nearest neighbours, normal instances will have a smaller distance than the anomalous instances. The fundamental concept of

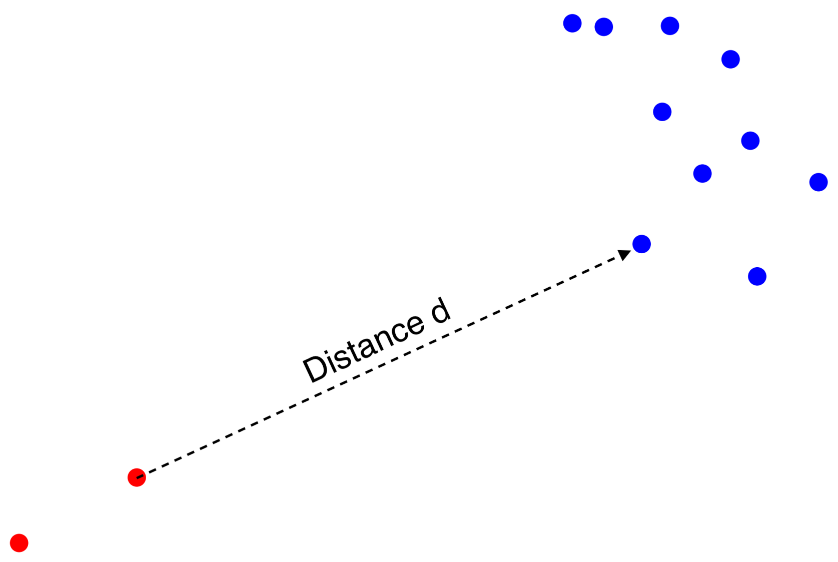


Figure 2.5: Distance-based outlier.

this method is to assign a distance-based score which reflects how far a data point is separated from the rest of the data points.

To use these methods, the following parameters must be chosen:

- Number of neighbours, k, for which the distance is calculated;
- Number of outliers, which is a stop condition;
- A measure of distance, the most widely adopted is the Euclidean distance;
- Score;

It is important to note that the number of neighbours, k, plays a key role. A k value that is too high, with a very dense cluster with fewer points than k, is labelled an outlier. On the other hand, if k is too small, two neighbouring outliers are labelled as normal.

One of the best known algorithms based on this principle is the k-Nearest Neighbors (k-NN) Dixon (2002), Cover and Hart (1967), initially developed for supervised classification, then later adapted to unsupervised classification. It follows the principle that a new instance is correctly classified if neighbouring instances, previously classified in the feature space, are considered.

One of the most common improvements is to speed up the distance computation, which is computationally expensive in high-density environments, e.g., some of them, use an indexing structure and data partitioning to speed up the computation, such as the use of binary search trees, KD-Tree Bentley (1975), Chaudhary et al. (2002).

2.2.3 Density-Based methods

By definition, the outliers, occur less frequently than the inliers, density-based approaches are based on the principle that instances in low-density regions are considered anomalies, since they are separated from most instances.

Density can be defined as the number of points in a considered unit of space and is inversely proportional to the distance between the considered points. Since this method relates distance and density, it can be explained by two parameters, the distance (d) and the portion of instances (p).

Given a point X, it is considered an outlier, if at least a portion of points p is located at distance d from point X, as shown in Figure 2.6. Given the definition of density and the definition of outlier, we can say that point X occupies an area of low density, so it is an outlier instance. There are several implementations, some similar to the K-NN seen above, the best known is the Local Outlier Factor (LOF) and its variants, Connectivity-based Outlier Factor (COF), Local Correlation Integral (LOCI).

Each approach has a different density function that the authors design to improve anomaly detection. These approaches are expensive to compute, especially in high-dimensional environments.

Local Outlier Factor (LOF)

Local Outlier Factor (LOF) Breunig et al. (2000a), attempts to overcome a key limitation of density-based methods by detecting variable density outliers. LOF considers both the density of the point and the density of the neighbourhood of the point, combining these two densities yields the relative density (RD) of the point which is used to calculate the score of the point considered, which is the key idea of

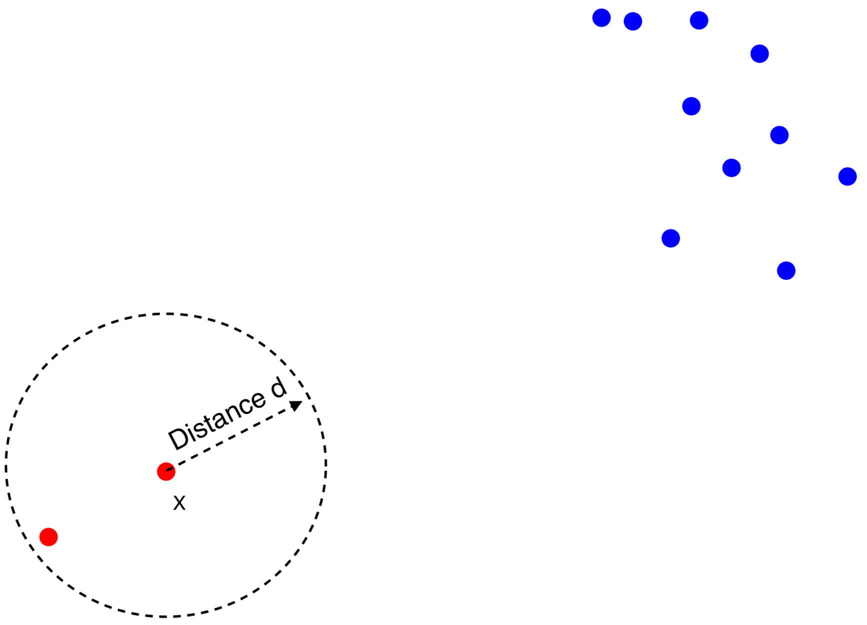


Figure 2.6: Density-based outlier.

this algorithm. The relative density of a data point X with k neighbours is given by the following Equation 2.1:

$$\mathrm{RD(X)}=\frac{\text{Density of X}}{\text{Average density of all data points in the neighbourhood}} \tag{2.1}$$

Where the density of X is the inverse of the average distance of the k nearest neighbours, or the inverse of the maximum distance of the k nearest neighbours. It is important to note the importance of the parameter k, because it forms the size of the neighbourhood. Considering the ratio between the density of the point and the average density of the points belonging to the neighbourhood, if the density of the point is less than the density of the neighbourhood then, we are in the presence of an outlier.

One of the improvements made to this method is the search for neighbours. operation is very similar to that of K-NN, for this we use a fairly efficient variant, which uses binary search trees Munaga and Jarugumalli (2011).

Connectivity-based Outlier Factor (COF)

Connectivity-based Outlier Factor (COF) Tang et al. (2002) is a variant of LOF, the main difference is the way the neighbourhood for an instance is constructed. In COF, the neighbourhood of a considered instance is constructed incrementally, i.e., in the first iteration, the closest instance to the considered instance is added to the set of neighbours, in subsequent iterations, the instance that is added to the set of neighbours is chosen such that its distance from the existing set of neighbours is minimum among all remaining data instances. The distance between an instance and a set of instances is defined as the minimum distance between the considered instance and every apparent instance to the considered set. The set of neighbours is populated until the parameter k is reached and once the neighbourhood construction is finished, we proceed with the calculation of the score which is the same as that seen earlier in the LOF.

LOcal Correlation Integral (LOCI)

Multi-Granularity Deviation Factor (MDEF) is a variation of LOF, this anomaly detection technique has been called LOcal Correlation Integral (LOCI) Breunig et al. (2000b) by the authors. MDEF for a given data instance is equal to the standard deviation of the local densities of the instances present in the neighbourhood, the inverse of the standard deviation is used as the anomaly score for the considered instance, this is the substantial difference from LOF.

This modification allows not only finds anomalous instances but also anomalous micro-clusters.

2.2.4 Model-Based methods

To detect anomalies, a model of the data is built and anomalies are identified as the data points that do not fit the model well. Well-known examples are:

- Classification-based methods;
- Clustering-based methods;
- Isolation-based methods;

Classification-Based methods

Classification-based anomaly detection techniques operate in two phases (training and testing). The first phase consists of training a classifier using labelled data, the second phase, consists of classifying a test instance as normal or abnormal, using the previously trained classifier. These techniques are based on the principle that *a classifier can distinguish between different classes*, and this distinction can be learned in feature space. These approaches can be grouped into two broad categories, multi-class and one-class, this distinction is made taking into account the data labels available in the training phase.

One-class includes training instances that have a single class label, these techniques learn a boundary around normal instances using a one-class classification algorithm and all instances that fall outside the learned boundary is declared as anomalous.

Multi-class includes training data that contains labelled instances belonging to more than one normal class De Stefano et al. (2000). These techniques train a classifier so that it can distinguish between each normal class and the rest of the classes, and a test instance is considered abnormal if it is not classified as normal by any of the classifiers.

One-Class Support Vector Machines

An SVM Cortes (1995) is a supervised learning model that given a set of input-output (instance,label) pairs, where $input \in R^d$ and $output \in \{-1, 1\}$. R^d denotes the d-dimensional feature space. For d = 2, a linear SVM aims to create a linear boundary separating data instances of two classes, this is applicable if the data is linearly separable in the considered space. More generally, the aim is to find the optimal hyperplane in R^d that guarantees the best (read maximum margin) separation between two classes of data. If the data is not linearly separable in the considered space it can be projected into a high-dimensional feature space through a non-linear *kernel function* $\phi()$, once the problem is solved in the new space, the hyperplane is projected into the space of the starting features, the linearity constraint for the separation hyperplane is relaxed and the boundary can take a non-linear form, this procedure is called the kernel trick Hofmann et al. (2006).

As mentioned earlier, SVM does not separate the data with respect to their membership classes but tries to create a boundary to delimit the region where the normal data reside. There are two main approaches, one is to project the data via the kernel trick and separate them from the origin by the maximum margin Schölkopf et al. (1999), or, use spherical boundaries instead of taking a planar approach Liu et al. (2010).

Neural Networks

Neural networks (NNs) have been applied to methods of detecting anomalies in a one-class Hawkins et al. (2002), Williams et al. (2002) or multi-class Hinton (2009) setting. An artificial neural network is a set of connected nodes, called artificial neurons, that resemble the neurons in a biological brain. With these functional units underlying the neural networks, it is possible to build an enormous variety of complex and deep models in all supervised, semi-supervised and unsupervised configurations. The general approach for multi-class anomaly detection is to train an NN on normal multi-class instances, subsequently, the NN determines whether a test instance is normal or anomalous by accepting or rejecting such instances De Stefano et al. (2000). These techniques often outperform machine learning approaches, but with higher computational cost and have little or no explainability (black box models)

Chalapathy and Chawla (2019b).

Clustering-Based Methods

Clustering-based approaches are popular approaches whose goal is to group similar data instances into clusters by maximizing inter-cluster distances and minimizing intra-cluster distances Pecht and Kang (2019). This approach is based on the principle that normal data instances belong to a cluster in the data, while anomalies do not belong to any cluster. Although clustering and anomaly detection appear to have nothing in common, since, in anomaly detection one does not want to cluster normal instances but to quickly find anomalies, some clustering algorithms do not force every data instance to belong to a cluster, this is because an anomalous instance could be assigned to a large cluster being considered as a normal instance. Some examples of algorithms that do not force instances to belong to a cluster are DBSCAN Ester et al. (1996), ROCK Guha (2000), FindOut Yu et al. (2002). Other clustering algorithms force all data to belong to a cluster, they have also been used for purposes of anomaly detection, the best known is the K-Means MacQueen et al. (1967).

The first step in the K-Means algorithm is to define a parameter k that indicates the number of clusters you want to obtain.

The next step is to define the centroid for each cluster obtaining k centroids. The choice of these centroids greatly affects the final result, the easiest method is to choose them at random from the data set, but this is not a recommended choice because we want to put the centroids as far away from each other as possible. Lloyd's algorithm is actually used Chen and Xia (2009) that returns the centroids.

The next step is to calculate the distance between the data and each centroid and a partition is made considering the minimum distance.

The last step is to update the centroids by calculating an average of all the data instances of the same cluster.

These steps are repeated iteratively until a termination condition is reached, for example, if no data instance moves to other clusters. K-Means can only guarantee a local optimum Steinley (2003). In the context of outlier detection, the distance between an instance and the centroid of its cluster reflects the anomaly score.

Isolation-Based Methods

As seen earlier, model-based approaches for anomaly detection build a profile of normal instances, then identify instances that do not conform to the normal profile as anomalies. In other words, a model is built and all instances that do not fit the model well are labelled as anomalies. The other model-based approaches have been adapted to the problem, as they arise to solve other problems such as classification or clustering.

In recent years, there has been a tendency to use an isolation-based approach, which was born precisely to solve the problem of anomaly detection. This method is based on two properties of anomalies, the first is that anomalies outnumber inliners, the second is that anomalies are very different from inliners in terms of features. In other words, they are "few and different", this led to the following insight, it is easier to separate an outlier from the data than an inliner. The first isolation method, Isolation Forest, first described in Liu et al. (2008), is a tree method and is based on a binary tree data structure, called isolation Tree. Actually, the isolation Forest uses a number of trees and the set of these trees form the forest, each isolation tree recursively subdivides the data domain in a hierarchical manner.

The Isolation Forest uses the previous intuition to define the anomaly score, i.e., this score reflects the average amount of random cuts required to completely isolate a data instance. In other words, the easier it is to isolate a data point, the fewer cuts required to achieve isolation, thus, the more isolated the instance. The instances are evaluated from the root to the leaves of the tree and the expected number of cuts to isolate the data point is estimated. There are several modifications made to this algorithm, mainly we modify the splitting criterion or the anomaly score.

Further details about Isolation Forest are discussed in Chapter 3 so that the algorithm is discussed in more detail.

PyOD

Some of the algorithms discussed in the previous sections have been implemented and made available through a Python library, PyOD Zhao et al. (2019), which, makes available more than 30 detection algorithms. This library is of considerable importance both in academia and in the Machine Learning community with several dedicated posts/tutorials, e.g., Analytics Vidhya, Towards Data Science.

2.3 Anomaly Detection Metrics

This section presents the concepts of the most widely used anomaly detection metrics in the literature.

Evaluating the performance of a classifier is of great importance both to attest its validity and to make comparisons with other models, the so-called competitors, in the literature have been proposed several evaluation metrics Sokolova et al. (2006), Powers (2020), Tharwat (2020).

As for binary classification, the classifier makes a prediction of all data instances and divides them into two different classes, positive (P) and negative (N). This procedure produces four types of results, two types of correct classification, true positive (TP) and true negative (TN), and two types of incorrect classification, false positive (FP) and false negative (FN), from these four results one can construct a 2x2 matrix, as shown in the Figure 2.7, called the confusion matrix, from which all the basic evaluation measures used in binary classification are derived.

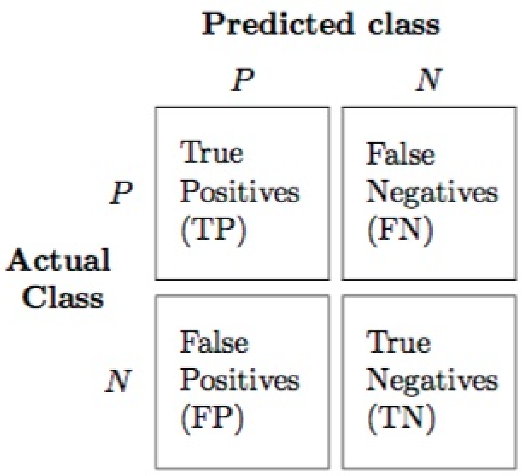


Figure 2.7: Confusion matrix

The basic measures most commonly used in the literature to evaluate the performance of a classifier are accuracy, recall or sensitivity, specificity Altman and Bland (1994), precision, F1-score, these metrics are shown in Table 2.1. These metrics behave differently in balanced and unbalanced data sets Saito and Rehmsmeier (2015), it is important to consider the class distribution of the data so that the appropriate measures can be chosen to evaluate performance. Other popular metrics are the area under the Receiver Operating Characteristics (ROC) curve (AUC ROC) Hanley and McNeil (1982) and the area under the Precision-Recall curve (AUC PRC) He and Garcia (2009).

Measure	Formula
Accuracy (ACC)	(TP + TN) / (TP + TN + FN + FP)
Recall (REC) Sensitivity (SN) True Positive Rate (TPR)	TP / (TP + FN)
Specificity (SP)	TN / (TN + FP) 1- FPR
False Positive Rate (FPR)	FP / (TN + FP)
Precision (PREC), Positive Predictive Value (PPV)	TP / (TP + FP)
F_1	2 * PREC * REC / (PREC + REC)

Table 2.1: Metrics table

All metrics shown in the Table 2.1, are single threshold measures, therefore, they are defined for single anomaly score thresholds, so they cannot give an overview of the range of performance with varying thresholds. To overcome this limitation, the ROC curve and the PRC curve were introduced.

2.3.1 Receiver Operating Characteristic

The ROC curve Fawcett (2006) is a two-dimensional graph where the y-axis represents TPR and the x-axis represents FPR, thus the ROC curve, shows the trade-off between specificity and sensitivity Hanley and McNeil (1982). The ROC curve is plotted by changing the discrimination threshold on the anomaly score, thus, each threshold value generates a point in the ROC space. In addition, the ROC curve has several properties, such as:

- provides a single performance value called the area under the curve;
- the major diagonal $[(0,0),(1,1)]$ represents the performance of a random classifier, which is defined as the baseline;
- The AUC is 0.5 for random classifiers and 1.0 for perfect classifiers.

2.3.2 Precision-Recall Curve

The Precision-Recall curve is another popular metric for comparing classifiers, it plots recall on the x-axis against precision on the y-axis.

It has been shown in the literature that the PR curve is more informative than the ROC curve Saito and Rehmsmeier (2015).

The shape of the PR curve has not some of the properties of the ROC curve, for example, the baseline of the PRC curve is no longer fixed but is determined by the ratio of positives (P) to negatives (N) as y = P / (P + N), thus it represents a horizontal line whose position depends on the distribution of classes.

The left-hand side of the Figure 2.8 shows the ROC curve with non-dominated points (red circles) and convex hull (dashed red line). The right-hand side shows the corresponding Precision-Recall curve with non-dominated points (red circles).

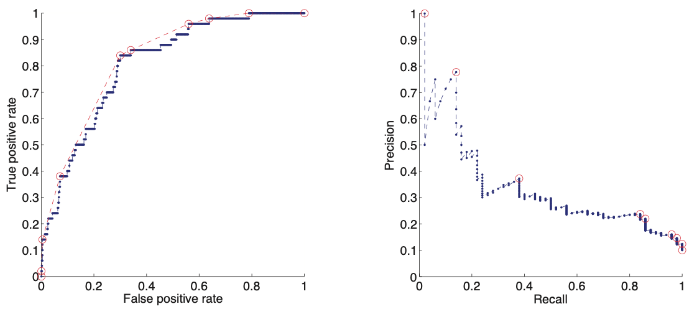


Figure 2.8: Illustrative graphs of Receiver Operating Characteristic, Precision-Recall curves (from left to right). Flach and Kull (2015)

Chapter 3

Isolation Forest

This chapter introduces the Isolation Forest algorithm. Part of this chapter is adapted from "A Review of Tree-Based Approaches for Anomaly Detection" Barbariol et al. (2022). written in collaboration with PhD students Tommaso Barbariol, Davide Marcato and Professor Gian Antonio Susto.

In 2008, a real revolution took place in the field of anomaly detection, that is, a new algorithm was introduced, the Isolation Forest Liu et al. (2008), which is based on the principle of isolation, in a field dominated by algorithms based on density and distance. Only in 2016 this algorithm was officially implemented in the Python Scikit-learn library (since v0.18) Pedregosa et al. (2011), after this implementation, as illustrated in Figures 3.1, the interest from the academic community literally exploded and this resulted in a significant amount of new research papers published in the last few years. This success is not only due to its performance but also due to its simplicity, the authors relied on the fact that anomalies are rare and different in features, therefore, an anomalous instance is more likely to be isolated than a normal instance. IF is an ensemble algorithm that resembles in some respects the popular Random Forest algorithm revised in the unsupervised anomaly detection settings, in fact, IF uses a set of binary trees, called isolation trees, that aim to isolate a region of space where only a data point or a predefined number is found, while in Random Forest decision trees are used. Each Isolation Tree uses only a small portion of the data (generally 256 subsamples), moreover, the authors have shown that it is possible to train the model using only normal instances while still obtaining good anomaly detection performance Liu et al. (2008).

The algorithm is divided into two phases. The first is the training phase followed by the testing phase, which will be discussed in the next section.

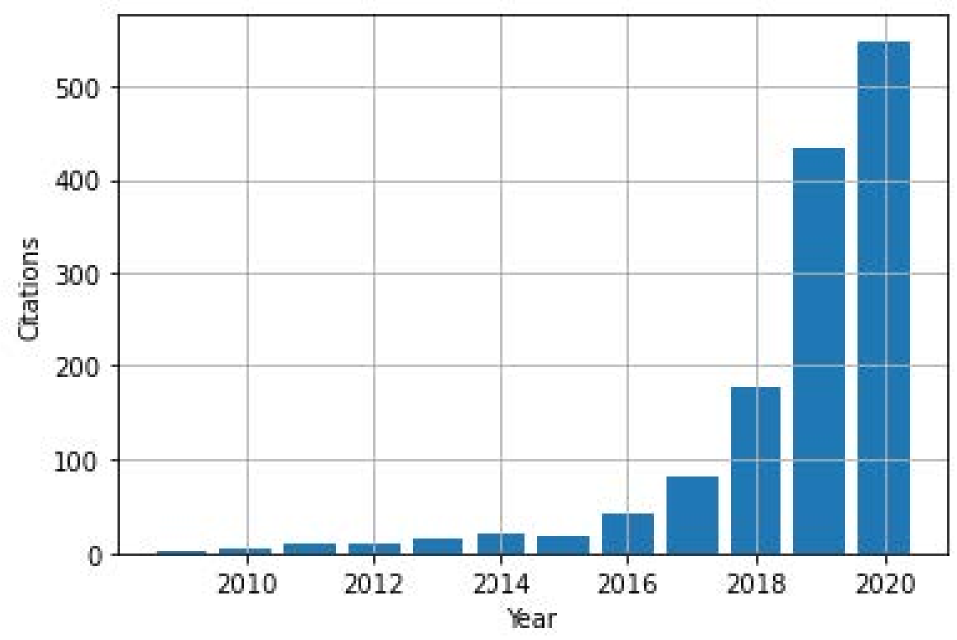


Figure 3.1: Combined citations of IF original paper Liu et al. (2008) and its extended version Liu et al. (2012) by the same authors. Source: Scopus. Retrieved on the 30th of March 2021.

3.1 Training Algorithm

In the training phase, the forest is defined, that is, all isolation trees are constructed by recursively partitioning the data into random partitions of the domain. In this phase, the entire dataset is not used, but a subset of the dataset.

For the sake of clarity, the pseudo-codes for training, testing and path length are given (Algorithm 1, 2 and 3)

Algorithm 1: IsolationForest(X, n, ψ)

```
Input: X - finite set of data in $\mathbb{R}^d$, n – number of trees, $\psi$ – sample size
Output: list of Isolation Trees
1 forest ← empty list of size n;
2 $h_{max} \leftarrow \lceil \log_2(|X|) \rceil$;
3 for $i = 1$ to $n$ do
4     $\hat{X} \leftarrow sample(X, \psi)$;
5     $forest[i] \leftarrow IsolationTree(\hat{X}, 0, h_{max})$;
6 end
7 return forest
```

Algorithm 1 has the task of creating the entire forest, and this algorithm needs three input parameters:

- X: the entire dataset, each instance belongs to $\mathbb{R}^d$.
- n: number of trees to be created, this parameter also defines the number of random subsets of X

Algorithm 2: IsolationTree(X, h, h_{max})

```
Input: $\hat{X}$ - finite set of data in $\mathbb{R}^d$, h - current depth of the tree, $h_{max}$ – depth limit
Output: Isolation Tree (root node)
if $h \geq h_{max}$ or $|X| \leq 1$ then
    return Leaf {
        $size \leftarrow |X|;$
    };
else
    $q \leftarrow$ randomly select a dimension from {1, 2, . . . , d};
    $p \leftarrow$ randomly select a threshold from [min $X^{(q)}$, max $X^{(q)}$];
    $X_L \leftarrow filter(X, X^{(q)} \leq p)$;
    $X_R \leftarrow filter(X, X^{(q)} > p)$;
    return Node {
        $left \leftarrow IsolationTree(X_L, h+1, h_{max});$
        $right \leftarrow IsolationTree(X_R, h+1, h_{max});$
        $split_{dim} \leftarrow q$ $split_{thresh} \leftarrow p$
    };
end
```

- ψ: sample size, i.e., defines the number of instances used to create each tree

The first step is to calculate the parameter h_{max} that defines the maximum height of the tree, this is because it is not necessary to grow the tree completely, since the interest is in identifying the anomalies that reside in the upper part of the tree and therefore have a shorter than average path length.

The second step is to call n times the Algorithm 2, through the function *IsolationTree()*, passing as input not the entire dataset but a subsampling of size ψ, this parameter not only defines the size, but also controls the processing time and memory size.

Algorithm 2 has the task of creating the isolation tree, and needs three input parameters:

- $\hat{X}$: the ψ size subset dataset;
- h: the current tree depth
- h_{max}: tree limit depth

The first step is to check if one of two termination conditions is satisfied:

- $h > h_{max}$
- $|X| \leq 1$

If these conditions are not satisfied, a feature with a uniformly distributed probability on all features, denoted by q, is randomly chosen at each node in the tree. The split point is uniformly sampled between the minimum and maximum values of the data along the selected feature, denoted with p.

The split criterion consists of comparing all values of q with p, this procedure produces two partitions, left and right.

This procedure is repeated recursively on the two newly created partitions until one of the two stop conditions is reached.

3.2 Testing Algorithm

In the second phase (Test), all instances are examined, in particular, once an instance is taken, the isolation tree is traversed (Algorithm 3) to obtain the path length, which reflects the anomaly score for each instance in the isolation tree considered, using the assumption that an anomalous instance is easier to separate than a normal instance, we obtain that the inliers live in a leaf in the deepest part of the tree, while the outliers live in a leaf near the root, in other words, the anomaly score associated with each instance is proportional to the average depth of the leaf $E(h(x))$.

Algorithm 3: PathLength(x, T, h)

```
Input: x - instance in $\mathbb{R}^d$, T – node of IsolationTree, l – current length (to
       be initialized to 0 when first called on the root node
Output: path length of x
1 if T is a leaf node then
2     return return $h + c(T.size)$;
3 end
4 $q \leftarrow T.split_{\mathrm{dim}}$;
5 if $x^{(q)} < T.split_{thresh}$ then
6     return PathLength(x, T.left, l + 1);
7 else
8     return PathLength(x, T.right, l + 1);
9 end
```

Algorithm 3 has the task of calculating the path length of an instance in the isolation tree, i.e. the nodes traversed to reach the leaf where the instance resides starting from the root, needs three input parameters:

- x: generic instance contained in the subset $\hat{X} \in \mathbb{R}^d$
- T: an isolation tree
- h: current height, initialised to 0 (root node)

The first step of the algorithm is to check if the node considered is a leaf, in case of positive result, the algorithm ends by returning the length of the path plus a correction factor (c(n)) that takes into account a subtree not built beyond the height limit of the tree. Otherwise, the projection of the instance is compared with the previously chosen split value, so as to choose the next node to be examined. The procedure continues recursively through the nodes of the considered tree until the termination condition is reached, i.e. the considered node is a leaf.

For example, the anomaly score for an instance, s (x, n), is calculated using Equation 3.1

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{3.1}$$

Where the parameter E(h(x)) represents the average path length, in more detail, an instance is fixed and the Algorithm 3 is applied to all isolation trees, so that all path lengths (from root to leaf) are obtained, finally, the average is calculated.

The parameter c(n) is a normalization factor, it estimates the average depth of the outer nodes in case of a fully branched tree and takes into account the path length of the failed searches in a binary search tree, more in detail it is defined as 3.2, where, the parameter H(i) is the harmonic number and can be estimated by $\ln(i) + 0.5772156649$ (Euler's constant).

$$c(n) = \begin{cases} 2H(n-1) - 2(n-1)/n & \text{for } n > 2 \\ 1 & \text{for } n = 2 \\ 0 & \text{for } n < 1 \end{cases} \tag{3.2}$$

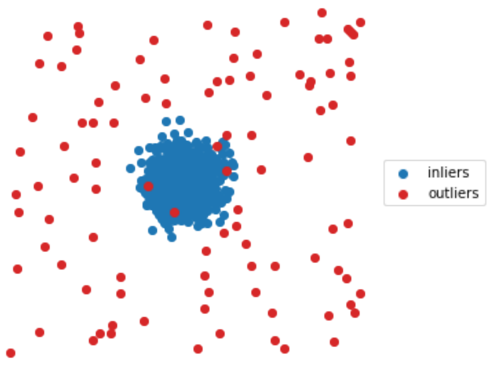
3.3 A practical example of an isolation tree

This section shows a practical example of the construction of an isolation tree using the Algorithms 1 and 2.

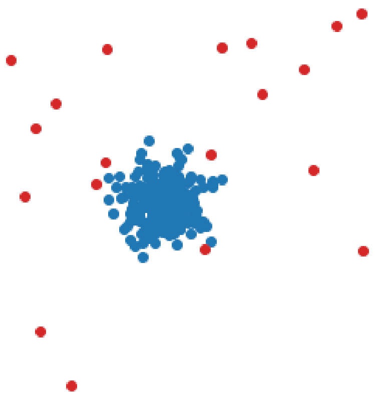
The dataset used is illustrated in Figure 3.2a, has been artificially created and consists of a central cluster of normal data instances and a few anomalies scattered throughout the space considered, more in detail, the whole dataset consists of 1000 normal data instances and 100 anomalous data instances, each instance $x \in \mathbb{R}^d$ with $d = 2$, that is, it has only two features.

Based on Algorithm 1, the entire dataset is subsampled ($\psi = 256$), as shown in Figure 3.2b, then the maximum height is calculated, in this case $\mathrm{h_{max}}= 8$ but let's consider $\mathrm{h_{max}}= 6$ for simplicity, finally Algorithm 2 is invoked n times to create all individual isolation trees, in this case n=1.

Based on Algorithm 2, at each iteration, a feature is randomly chosen, which determines the orientation of the cut, as illustrated in Figure 3.3, which is a step-by-step representation of the growth of a tree. This procedure is repeated until the depth $\mathrm{h_{max}}$ is reached or the number of points in the subspace generated by the cut is less than or equal to two. Once this phase is completed, the tree is formed and



(a) Artificial Dataset X



(b) Artificial Dataset Subsampled $\hat{X}$

added to the forest, and then the anomaly scores are calculated using Equation 3.1, where h(x) is calculated using Algorithm 3.

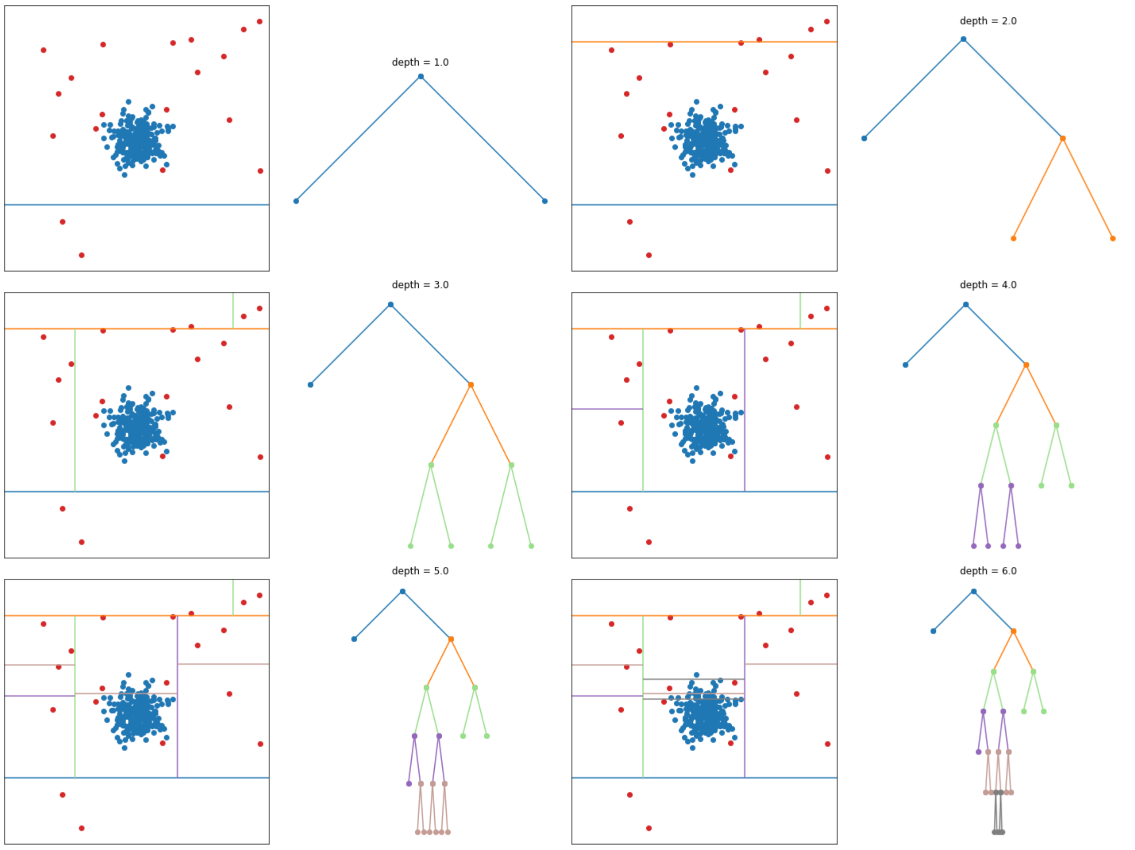


Figure 3.3: Step-by-step tree growth

Regarding the results obtained, in Figure 3.4, the Score Map is illustrated and in Figure 3.5 the metrics obtained AUC ROC and AUC PRC are reported.

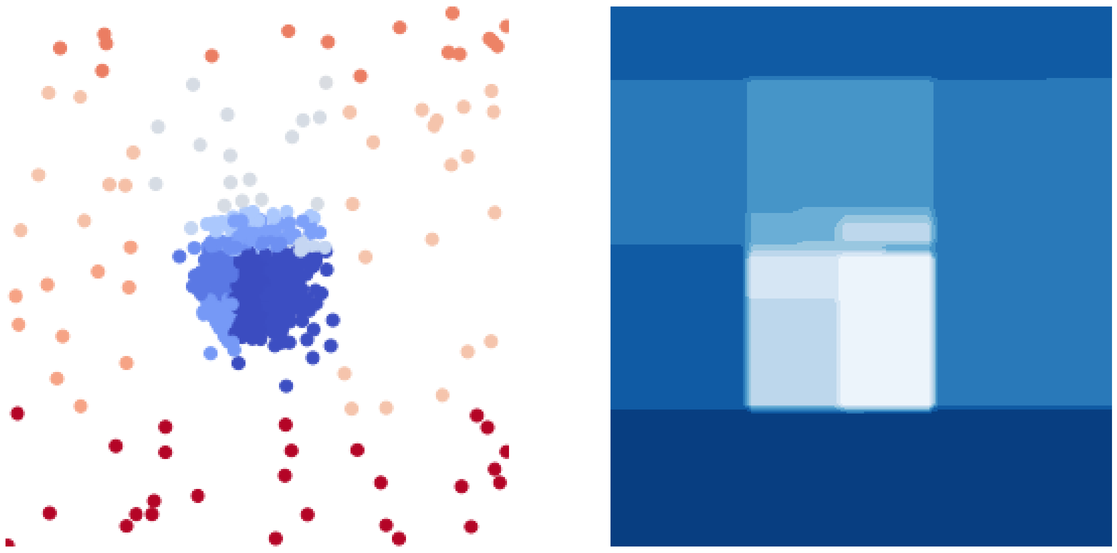


Figure 3.4: Score map over the whole dataset X

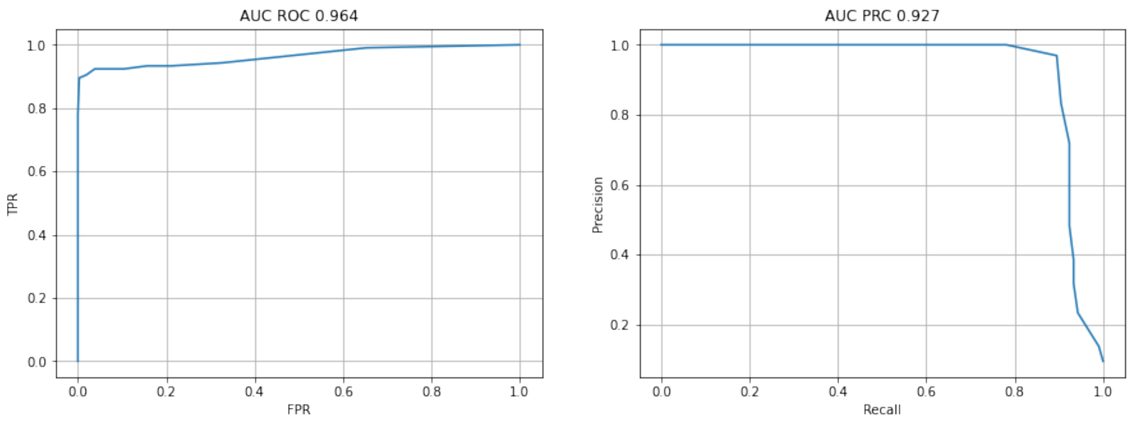


Figure 3.5: AUC ROC and AUC PRC metrics

Chapter 4

Implementations

In this chapter, we will present and discuss all the changes made to the Isolation Tree algorithm, i.e. the part of the method responsible for creating each individual tree in the forest (Algorithm 2), while the Isolation Forest algorithm (1) is not changed.

The modifications made have been grouped into two families, according to the modified criterion, split criterion or scoring function.

4.1 Split Criterion

In this section, we propose several changes to the process of selecting a feature and calculating the split value.

4.1.1 Prob split

This method weighs the choice of split value, so the probability of choosing the split value to the left or right of a randomly chosen projection (q) on a randomly chosen feature is calculated. The general idea of this method is to choose one feature at random from all those considered, then the distance between the projection (q) and the two extremes (maximum and minimum projection) is calculated through the equations 4.2 and 4.1, called left gap and right gap respectively. The choice of the side with respect to q depends on the gap, so a random choice is made considering the weight of the gap.

$$\text{left gap} = \frac{\text{q} - \text{min}}{\text{max} - \text{min}} \tag{4.1}$$

$$\text{right gap} = \frac{\text{max} - q}{\text{max} - \text{min}} \tag{4.2}$$

Once the side is chosen, the split value is calculated and two variants are implemented to do this:

First variant

In this case the mean value between q and the extreme is chosen. The extreme can be the maximum or the minimum of the projections, as mentioned above, and this depends on the random choice of the side which is affected by the gap

Second variant

In this case, the split point is uniformly sampled between q and one of the two extremes.

Graphical intuition

Assuming that both the feature and the random value (excluding the maximum and minimum projections) have already been chosen as shown in Figure 4.1, the next step is to calculate the split value based on one of the two variants as shown in Figure 4.2.

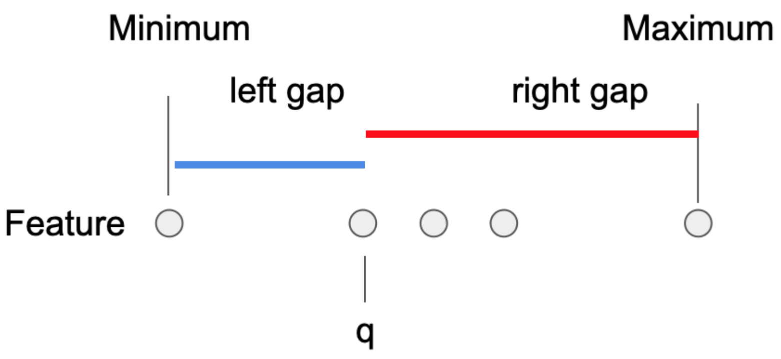


Figure 4.1: Representation of the left and right gap

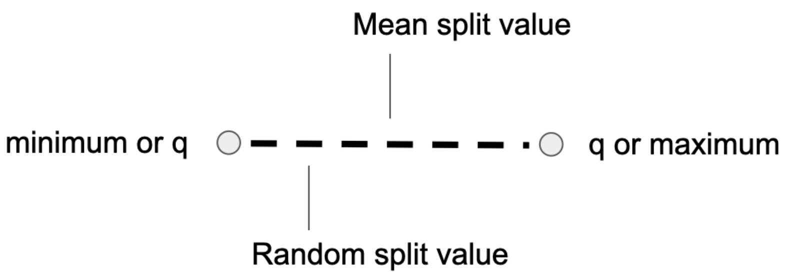


Figure 4.2: Choice of split value

General steps

1. A dimension is chosen at random, i.e. a feature is chosen at random;
2. The maximum and minimum projections for the previously selected feature are found;
3. A random value is chosen from the projections, in other words, a random value is chosen from the feature considered;
4. Equations 4.1 and 4.2 are used to calculate the respective gaps, which influence the choice of side where the split value will be calculated;
5. A random choice is made to choose the weighted side on the calculated gap;
6. The split value is calculated using one of two variants
7. The split is performed, which creates two nodes in the tree;
8. The procedure from step 1 is repeated for each node created in the tree until the depth of the tree is less than the maximum depth or the number of instances is greater than two.

4.1.2 Neighbours

This method takes inspiration from Tokovarov and Karczmarek (2022), where the feature to calculate the split value is chosen randomly, then the data projections are sorted, and finally the split value is chosen according to the Cumulative Distribution Function (CDF). Our method proposes a new criterion for selecting the split value.

The general idea is to exploit the sorting by a randomly chosen feature and not use the CDF to choose the split value. Our method instead calculates the distance between neighbours, called the gap, using the equation 4.3, where q is the projection of an instance with respect to the chosen feature.

$$\mathrm{gap(i) = diff(q_{i+1}, q_i)\ with\ i \in [0, \ldots, n-1]\ where\ n{=}|X|} \tag{4.3}$$

Once all gaps have been calculated, the maximum gap is chosen to calculate the split value, we have implemented two variants for this calculation:

First variant

This variant chooses as the split value the mean value between the two neighbours with maximum gap from the randomly chosen feature.

In this case, the aim is to eliminate the part of randomness in the selection of the split value.

Second variant

This variant, chooses as split a value uniformly sampled between the two neighbours with the maximum gap.

Graphical intuition

Assuming that the feature has already been randomly selected and that the data have already been sorted with respect to the chosen feature. Figure 4.3 shows the gap between neighbours, once the maximum gap has been selected, the split value is calculated using one of the two proposed variants as a criterion (Figure 4.4).

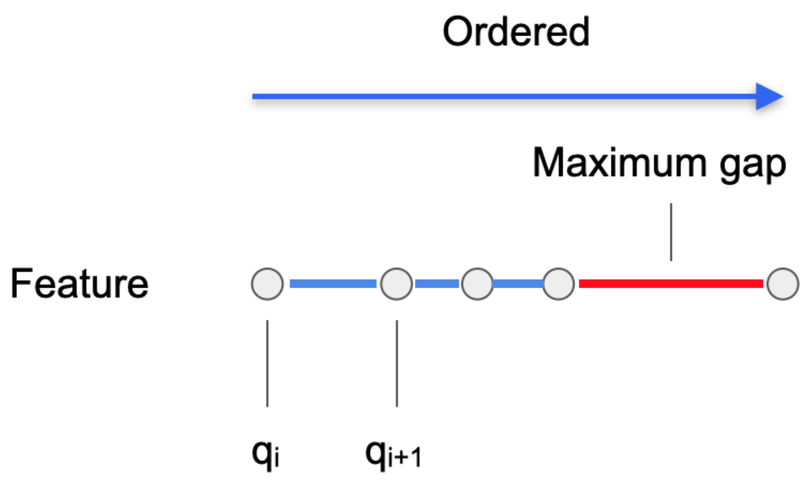


Figure 4.3: Representation of the gap between two neighbours

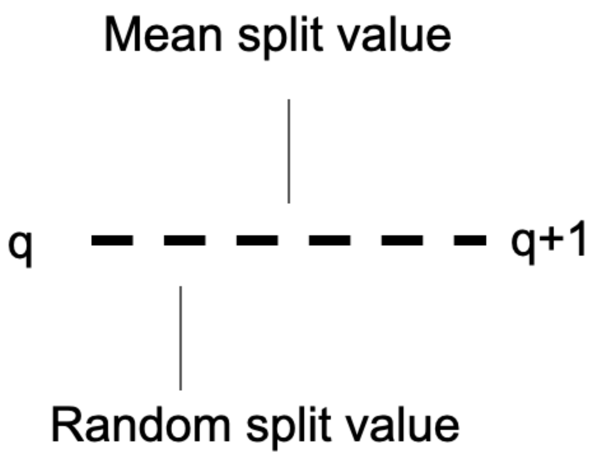


Figure 4.4: Choice of split value

General steps

1. A feature is chosen randomly;
2. The data is sorted according to the chosen feature;
3. The gap between the projections $\mathrm{q}_{\mathrm{i+1}}$ and q_{i} is calculated;
4. The two neighbours with maximum gaps are selected, i.e. the two projections with the maximum distance;
5. The split value is calculated using one of the two variants described above;
6. The split is performed, which creates two nodes in the tree;
7. The procedure from step 1 is repeated for each node created in the tree until the depth of the tree is less than the maximum depth or the number of instances is greater than two.

4.1.3 Component Gap

This method tries to weight the choice of split value against all features or fixed number (chosen randomly).

The general idea is that for each feature, the data are sorted and the distance between the neighbours is calculated by choosing the maximum, called the maximum gap, thus obtaining a list of possible candidate neighbours on which to calculate the split value. The calculation of the split value has been implemented in several variations.

The first and second variants are reminiscent of the Neighbours method seen above but with the difference that the gap is calculated on all the features considered.

The third and fourth variants introduce a randomness in the selection of the split value.

First variant

This variant, calculates the mean value between the two neighbours, of all possible candidates, obtaining a list of mean values.

The maximum value of the list of mean values is chosen as the split value.

Second variant

This variant, calculates a uniformly sampled value between the two neighbours, of all possible candidates, obtaining a list of random values.

The maximum value of the list of random values is chosen as the split value.

Third variant

This variant, calculates the mean value between the two neighbours in the list of possible candidates.

To calculate the split value, a weighted random choice is made from the calculated mean values (the higher the value, the higher the probability of being selected as the split value).

Fourth variant

This variant, calculates a uniformly sampled value between the two neighbours of the list of possible candidates.

Graphical intuition

Assume that all gaps have been calculated for each feature. Figure 4.5 shows the process of selecting the maximum gaps for each feature. Applying one of the variants described above, the split value is calculated as shown in Figure 4.6.

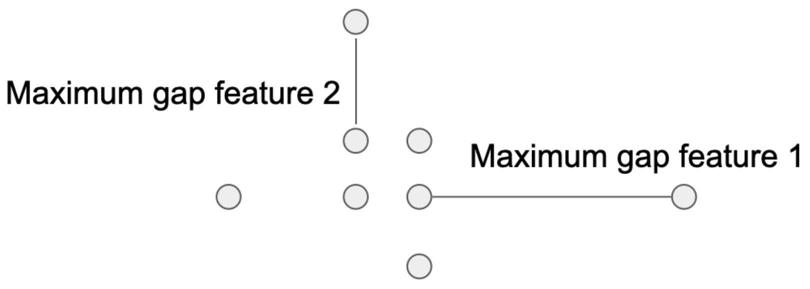


Figure 4.5: Choice of split value

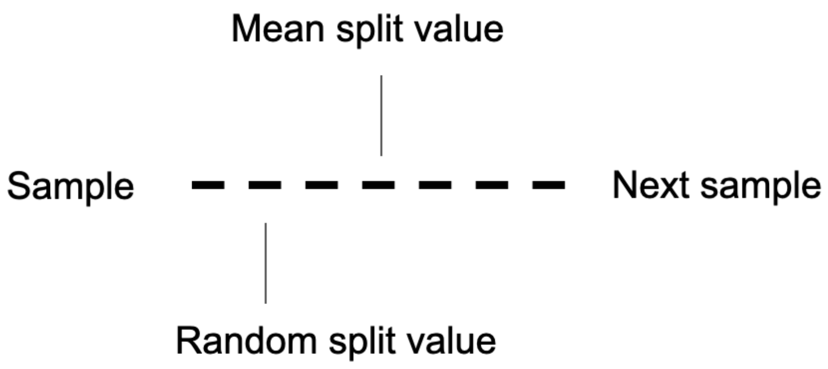


Figure 4.6: Choice of split value

General steps

1. For each feature considered;
    (a) A sorting of the data is carried out;
    (b) The gap between neighbours is calculated
    (c) The maximum gap is selected
2. One of the variants described above is applied to calculate the split value;
3. The split is performed, which creates two nodes in the tree;
4. The procedure from step 1 is repeated for each node created in the tree until the depth of the tree is less than the maximum depth or the number of instances is greater than two.

4.1.4 Bubble Forest

This method was developed by introducing a new definition of split, it is no longer considered left or right with respect to the split but outside or inside with respect to a geometric figure. In this case, we used three different types of norms ($\| \cdot \|_1$,$\| \cdot \|_2$ and $\| \cdot \|_\infty$) to calculate the subspace generated by the split. All the points that reside in the generated subspace will always occupy the left-hand side of the tree, whereas those outside will occupy the right-hand side.

The nodes created are numbered according to a global index. The global index induces a new reading of the tree with respect to the standard method, both from left to right and from top to bottom.

To choose the radius of the subspace to be created, we used two criteria, data-dependent or data-independent.

Data-dependent

In this case, we want to choose the radius based on the available data.

There are two approaches to this:

- A subsampling of k samples is performed, then the norm between the centre and the k samples is calculated and the mean (V1), minimum (V2) or maximum (V3) value of these distances is chosen;
- The condition presented in Choudhury et al. (2021) is used, more in detail, the radius is selected randomly and must satisfy the following condition $R_{\min} < R < R_{\max}$, where $\mathrm{R_{min}}$ and $\mathrm{R_{max}}$ are defined in Equations 4.4 and 4.5 respectively, $i \in \{1, \dots, N\}$ and $\mathrm{x_j}$ is a data point in the branch. This strategy was called V6.

$$\mathrm{R_{min}} = \min\{\min((\max(\mathrm{x_{j,i}}) - \mathrm{c_i}), (\mathrm{c_i} - \min(x_{\mathrm{j,i}})))\} \tag{4.4}$$

$$\mathrm{R_{max}} = \max\{\max((\max(\mathrm{x_{j,i}}) - \mathrm{c_i}), (\mathrm{c_i} - \min(\mathrm{x_{j,i}})))\} \tag{4.5}$$

Data-independent

In this case we want to scale the radius by depth using one of the Equations 4.6 (V5) or 4.7 (V6).

$$\text{radius} = \frac{1}{1 + \text{depth}} \tag{4.6}$$

$$\text{radius} = \frac{1}{2^{\text{depth}}} \tag{4.7}$$

Graphical intuition

Assuming a centre has already been chosen and a subsample of size three has been performed as shown in Figure 4.7. Subsequently, the distance between the centre and the subsamples is calculated. Finally, one of the variants described above is applied to choose the radius.

Figure 4.8 shows the subspace generated for the different types of norms.

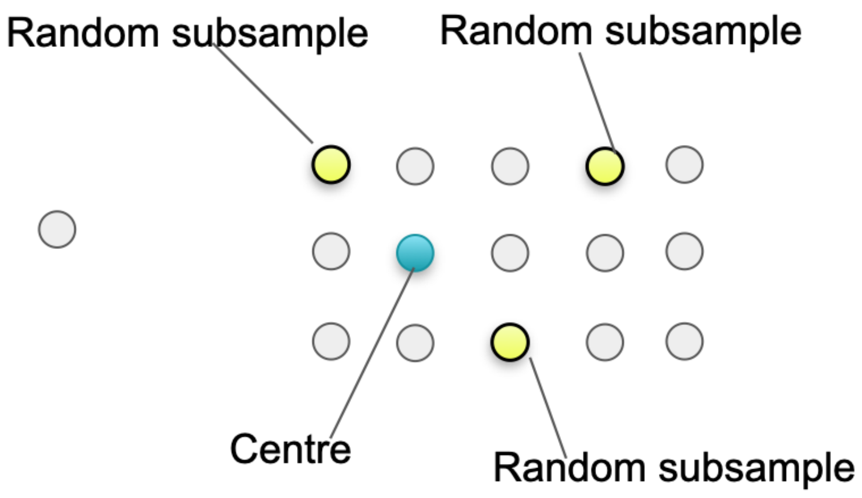
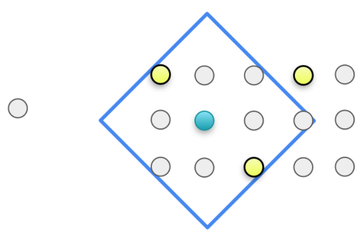
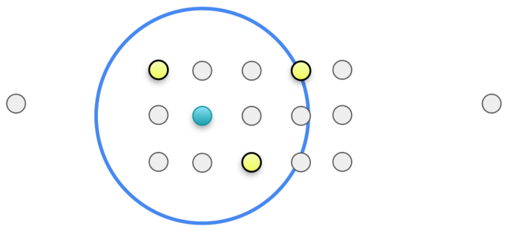


Figure 4.7: Chosen subsamples


(a) $\|\cdot\|_1$


(b) $\|\cdot\|_2$

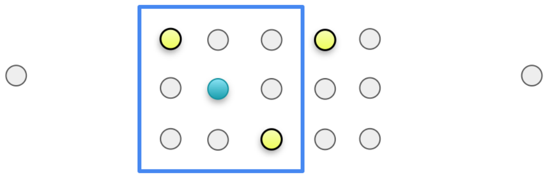

(c) $\|\cdot\|_\infty$

Figure 4.8: Subspace generated using the different norms

General steps

1. A point is chosen at random, which is called the centre;
2. The radius must be chosen to create the subspace according to one of the criteria described above
3. A split is made by checking if the distance between the centre and all instances considered is less than or equal to the radius, then a node is created on the left-hand side of the tree, otherwise, on the right-hand side;
4. The procedure from step 1 is repeated for each node created in the tree until the depth of the tree is less than the maximum depth or the number of instances is greater than two;

4.2 Anomaly Score Criterion

In this section, two anomaly scoring functions are proposed to replace the standard function, for which we have modified the Algorithm 3.

The proposed anomaly scoring functions do not consider only the depth of the tree but consider additional information to calculate the anomaly score of each instance in the dataset.

The general idea is to weigh the path from root to leaf. Adding the various path weights and the normalisation value c(n) defined in Equation 3.2 gives the new anomaly score.

4.2.1 Weighted Path

The general idea is to weight the path according to the result of the split. In particular, once the split has been made, two nodes are created in the tree. Since the split is a random process, the number of instances on the two nodes is not always balanced. Therefore, one node may contain more instances than the other.

This is the key idea that allowed us to develop three variants to weight the path.

First variant

$$\text{Weighted left-hand path} = \frac{\text{left}_{\text{size}}}{\text{node}_{\text{size}}} \tag{4.8}$$

$$\text{Weighted right-hand path} = \frac{\text{right}_{\text{size}}}{\text{node}_{\text{size}}} \tag{4.9}$$

Second variant

$$\text{Weighted left-hand path} = \frac{\text{left}_{\text{size}}}{\text{node}_{\text{size}}} * \text{gap} \tag{4.10}$$

$$\text{Weighted right-hand path} = \frac{\text{right}_{\text{size}}}{\text{node}_{\text{size}}} * \text{gap} \tag{4.11}$$

Third variant

$$\text{Weighted left-hand path} = \frac{\text{left}_{\text{size}}}{\text{node}_{\text{size}}} * \text{maxGap} \tag{4.12}$$

$$\text{Weighted right-hand path} = \frac{\text{right}_{\text{size}}}{\text{node}_{\text{size}}} * \text{maxGap} \tag{4.13}$$

Where:

- $\text{node}_{\text{size}}$ is the number of instances belonging to the considered node
- $\text{left}_{\text{size}}$ is the number of instances that will occupy the left-hand side of the tree
- $\text{right}_{\text{size}}$ is the number of instances that will occupy the right-hand side of the tree
- gap is the distance between the two projections where the random choice of the split value falls.
- maxGap is the distance between the minimum value of the projections and the maximum value of the projections.

Once the variant has been chosen, for each instance, all the weights of the path (from root to leaf) must be added together, which must then be summed with the normalisation value c(n) defined by Equation 3.2 to obtain the final weight which is used as the anomaly score.

4.2.2 Bubble Forest Weighted Path

This anomaly score function can only be used in the Bubble Forest method described in Section 4.1.4, because it has been designed based on the global index assigned to each node.

The general idea is to weight the path on the basis of the global index of each node. In particular, after splitting, the two nodes are created (left node and right node), then the weight of each node is calculated, to do this, we have designed two variants.

First variant

The weight assigned to each node is equal to the sum of the weights of the nodes on the path, where the weight of each individual node is defined as the global index (ordered from left to right and top to bottom), as defined in Equation 4.14

$$\text{Weighted path} = \text{node}_{\text{index}} \tag{4.14}$$

Second variant

The weight assigned to each node is equal to the sum of the weights of the nodes on the path, where the weight of each individual node is defined by Equations 4.15 and 4.16. in other words, the number of nodes occupying the left and right sides as a result of the split is calculated and multiplied by the global index of the node.

$$\text{Weighted left-hand path} = \frac{\text{left}_{\text{size}}}{\text{node}_{\text{size}}} * \text{node}_{\text{index}} \tag{4.15}$$

$$\text{Weighted right-hand path} = \frac{\text{right}_{\text{size}}}{\text{node}_{\text{size}}} * \text{node}_{\text{index}} \tag{4.16}$$

Chapter 5

Results

In this Chapter, the results obtained from the various experiments carried out with the methods introduced in Chapter 4 will be presented and discussed. The discussion focuses on the comparison of the AUC PRC metric with the standard method.

The results obtained are divided by the modification criteria (split and anomaly score). This choice allows us to select the best method from all the variants proposed. For the sake of clarity, for each method, we present the variants with the best results.

It is important to note that in order to obtain the most truthful results, the results obtained are an average of 10 repetitions of the Algorithms.

For each method, is provided:

- A Table showing the values of the AUC PRC metric and their standard deviation. From this Table a chart is created that allows a visual comparison of the results obtained;
- A chart to evaluate the performance of the proposed methods compared to the standard method (IF);

In addition, this chapter will also present the data sets used to carry out the experiments.

5.1 Datasets

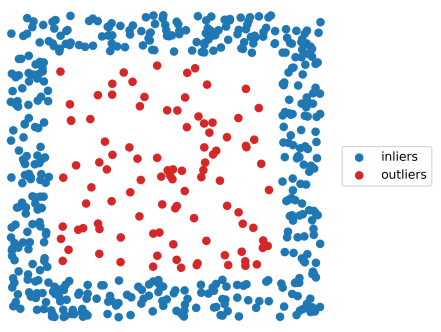
5.1.1 Artificial

The data sets in Table 5.1 have been created specifically for conducting the experiments.

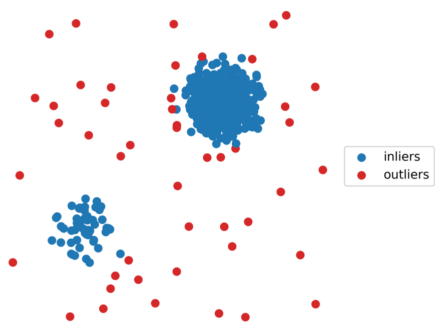
The data sets are represented graphically in Figure 5.1

Dataset name	Number of instances	Number of outliers	Number of inliers
Square Toroid	561	100	461
Anomalous Cluster	600	48	552
Double Clusters	1050	56	994
Central Cluster	1100	107	993
Four Clusters	2100	109	1991

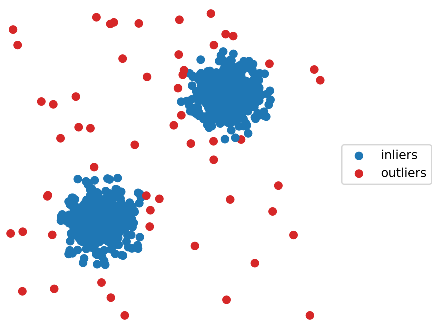
Table 5.1: Description of artificial data sets



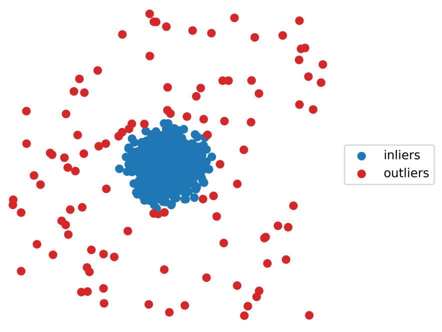
(a) Square Toroid



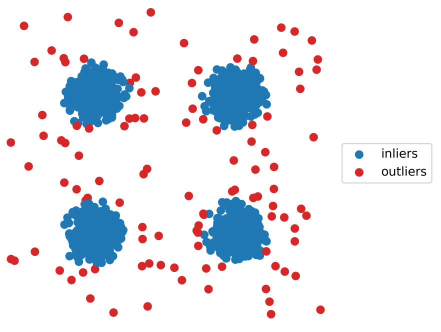
(b) Anomalous Cluster



(c) Double Clusters



(d) Central Cluster



(e) Four Clusters

Figure 5.1: Graphical representation of artificial data sets

5.1.2 Real

All data sets used for the experiments were made available to the UCI machine learning repository Dua and Graff (2017).

Table 5.2 lists them and their properties.

Dataset name	Number of instances	Number of features	Number di outliers
Annthyroid	7200	6	534
Arrhythmia	452	274	66
Breastw	683	9	239
Cardio	1831	21	176
Glass	214	9	9
Ionosphere	351	33	126
Mammography	11183	6	260
Pendigits	6870	16	156
Pima	768	8	268
Satellite	6435	36	2036
Thyroid	3772	6	93
Wine	129	13	10

Table 5.2: Real Datasets

5.2 Experimental setup

For the standard method, we used the *Scikit-learn* implementation.
The parameters used to conduct both experiments are as follows:

- n=100, number of isolation trees in the forest;
- $\psi = 256$, number of samples used for each isolation tree extracted randomly from the whole dataset;
- $h_{max} = 8$, maximum depth of each isolation tree
- The minimum number of instances per partition is two

5.3 Results of the split criterion experiments

5.3.1 Prob Split

The AUC PRC values and standard deviation for the Artificial and Real experiments are shown in Tables 5.3 and 5.4 respectively.

In the experiment with Artificial data sets, the first variant of the proposed method performs better than the second. This is because when the split value is uniformly sampled in a large gap, it does not ensure a good cut, especially when the chosen split value is very close to q. On the contrary, when the split value is chosen as the mean value of the gap, a better cut is obtained, since all instances closer to q are considered normal.

In the experiment with Real data sets, the second variant of the proposed method performs better than the first. This is because by using real-world data sets and not specially created data sets, the random strategy from the results obtained ensures a better cut.

Artificial

Dataset names	Isolation Forest	Prob Split	
	Standard	V1	V2
Anomalous Cluster	0.9024 ± 0.0170	**0.9123 ± 0.0080**	0.9064 ± 0.0134
Central Cluster	**0.9992 ± 0.0005**	0.9964 ± 0.0018	0.9989 ± 0.0016
Double Clusters	0.9546 ± 0.0132	**0.9700 ± 0.0078**	0.9416 ± 0.0124
Four Clusters	0.9287 ± 0.0132	**0.9742 ± 0.0054**	0.9270 ± 0.0086
Square Toroid	0.2385 ± 0.0258	**0.8930 ± 0.0275**	0.2349 ± 0.0221

Table 5.3: Precision-Recall AUC

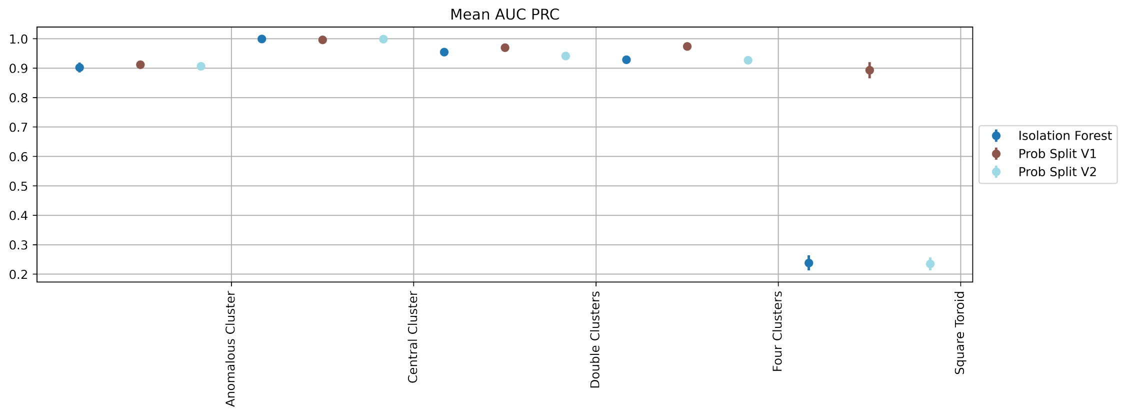


Figure 5.2: Comparison of AUC PRC Metrics with Standard Deviation

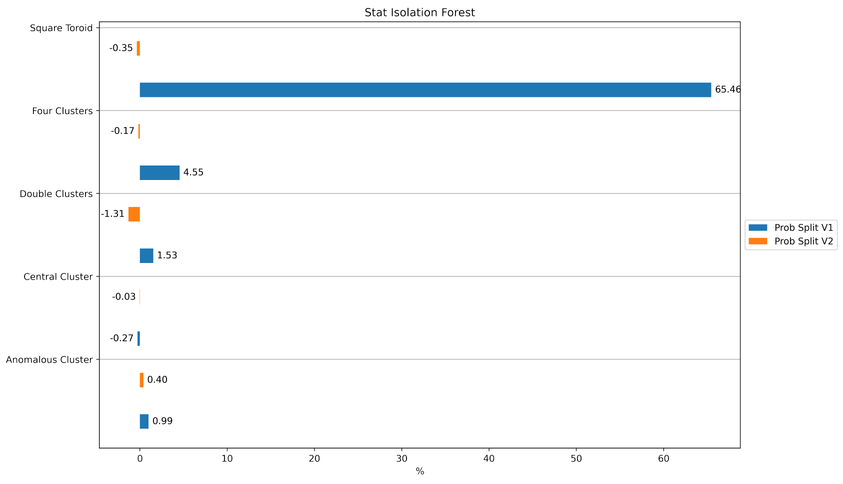


Figure 5.3: Graphical view of performance against the standard method

Real

Dataset names	Isolation Forest	Prob Split	
	Standard	V1	V2
AnnThyroid	0.3042 ± 0.0323	0.2757 ± 0.0237	**0.3227 ± 0.0195**
Arrhythmia	**0.4729 ± 0.0140**	0.4655 ± 0.0333	0.4344 ± 0.0371
Breastw	0.9707 ± 0.0043	0.9610 ± 0.0050	**0.9715 ± 0.0032**
Cardio	**0.5776 ± 0.0343**	0.5534 ± 0.0299	0.5490 ± 0.0369
Glass	**0.1033 ± 0.0106**	0.0905 ± 0.0073	0.0931 ± 0.0112
Ionosphere	**0.8095 ± 0.0072**	0.7912 ± 0.0076	0.8021 ± 0.0085
Mammography	0.2211 ± 0.0357	0.2211 ± 0.0439	**0.2286 ± 0.0476**
Pendigits	0.2631 ± 0.0639	0.2579 ± 0.0269	**0.2843 ± 0.0574**
Pima	0.5005 ± 0.0089	**0.5293 ± 0.0104**	0.5024 ± 0.0135
Satellite	0.6583 ± 0.0237	**0.7081 ± 0.0203**	0.6651 ± 0.0221
Thyroid	0.5257 ± 0.0908	0.4788 ± 0.0670	**0.5461 ± 0.0380**
Wine	**0.2133 ± 0.0257**	0.2080 ± 0.0307	0.1877 ± 0.0472

Table 5.4: Precision-Recall AUC

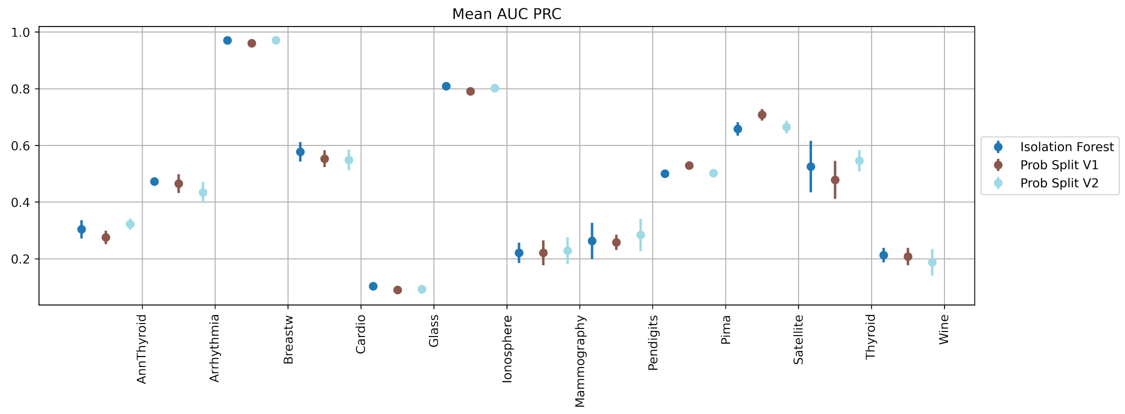


Figure 5.4: Comparison of AUC PRC Metrics with Standard Deviation

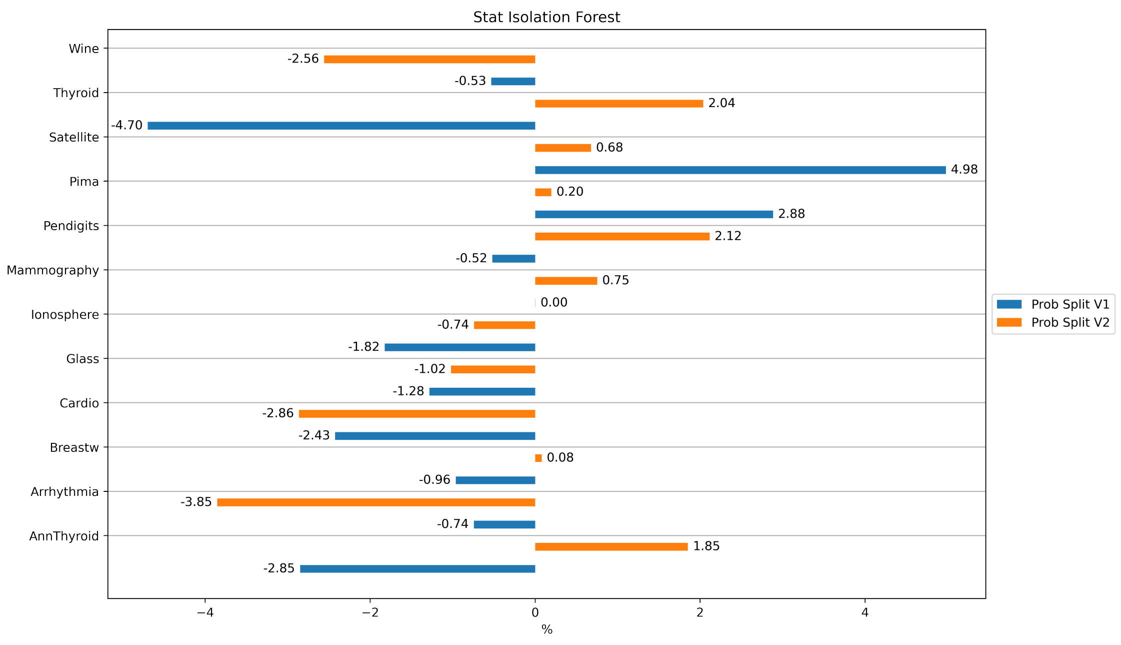


Figure 5.5: Graphical view of performance against the standard method

5.3.2 Neighbours

The AUC PRC values and standard deviation for the Artificial and Real experiments are shown in Tables 5.5 and 5.6 respectively.

In the experiment with artificial data sets, the two variants are almost identical, although the second performs slightly better. This is because by exploiting the ordering of the data and the concept of neighbours, a gap is obtained that better separates the data.

In the experiment with real datasets, the difference between the two variants can be seen. In this case, the first variant performs better than the second. This is because very close cuts to the neighbours are avoided.

Artificial

Dataset names	Isolation Forest	Neighbors	
	Standard	V1	V2
Anomalous Cluster	0.9024 ± 0.0170	0.9392 ± 0.0091	**0.9438 ± 0.0045**
Central Cluster	**0.9992 ± 0.0005**	0.9971 ± 0.0005	0.9969 ± 0.0008
Double Clusters	0.9546 ± 0.0132	0.9928 ± 0.0033	**0.9936 ± 0.0038**
Four Clusters	0.9287 ± 0.0132	0.9834 ± 0.0049	**0.9850 ± 0.0045**
Square Toroid	0.2385 ± 0.0258	**0.7172 ± 0.0469**	0.6928 ± 0.0468

Table 5.5: Precision-Recall AUC

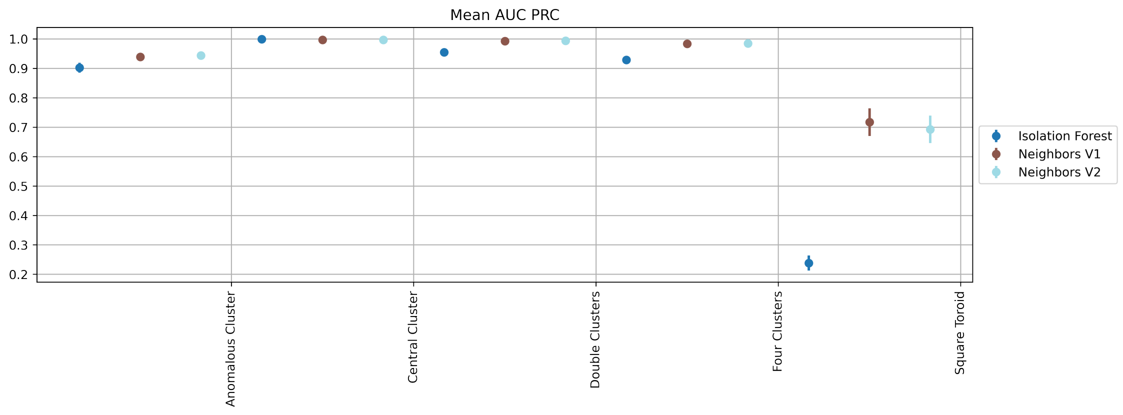


Figure 5.6: Comparison of AUC PRC Metrics with Standard Deviation

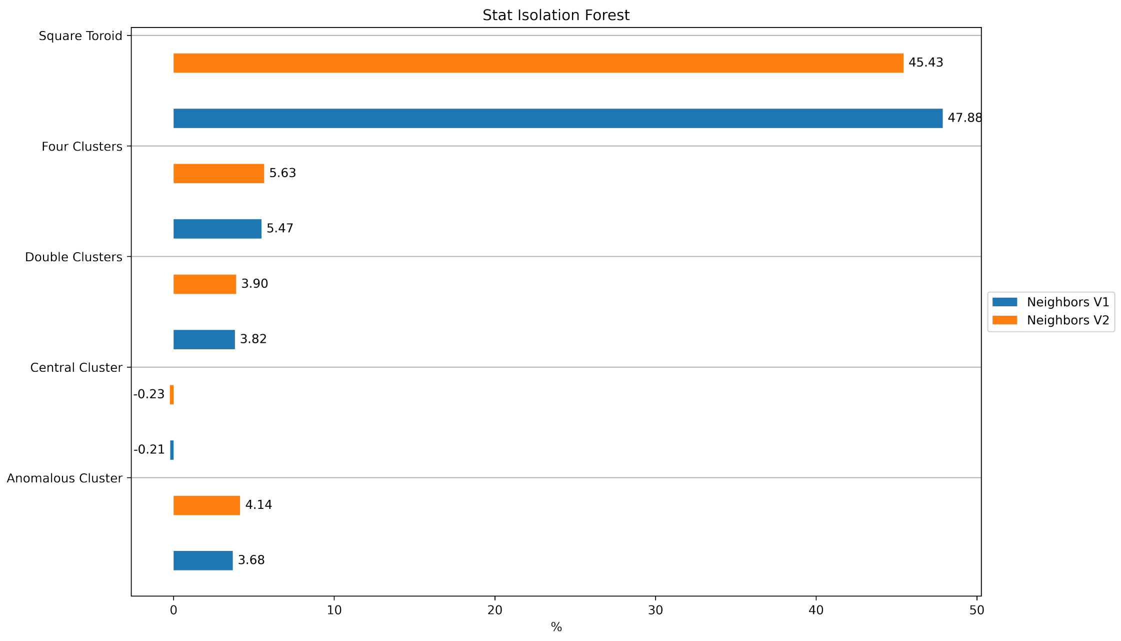


Figure 5.7: Graphical view of performance against the standard method

Real

Dataset names	Isolation Forest	Neighbors	
	Standard	V1	V2
AnnThyroid	**0.3042 ± 0.0323**	0.2006 ± 0.0215	0.1991 ± 0.0124
Arrhythmia	**0.4729 ± 0.0140**	0.4547 ± 0.0233	0.4335 ± 0.0187
Breastw	**0.9707 ± 0.0043**	0.6671 ± 0.0277	0.6715 ± 0.0251
Cardio	0.5776 ± 0.0343	0.5983 ± 0.0278	**0.6060 ± 0.0226**
Glass	0.1033 ± 0.0106	**0.1222 ± 0.0363**	0.1149 ± 0.0364
Ionosphere	0.8095 ± 0.0072	**0.8485 ± 0.0055**	0.8469 ± 0.0067
Mammography	**0.2211 ± 0.0357**	0.1217 ± 0.0155	0.1370 ± 0.0220
Pendigits	0.2631 ± 0.0639	**0.3239 ± 0.0347**	0.3196 ± 0.0161
Pima	**0.5005 ± 0.0089**	0.4115 ± 0.0060	0.4114 ± 0.0045
Satellite	0.6583 ± 0.0237	**0.7139 ± 0.0050**	0.7125 ± 0.0081
Thyroid	**0.5257 ± 0.0908**	0.3076 ± 0.0272	0.2971 ± 0.0243
Wine	0.2133 ± 0.0257	**0.2541 ± 0.0205**	0.2451 ± 0.0231

Table 5.6: Precision-Recall AUC

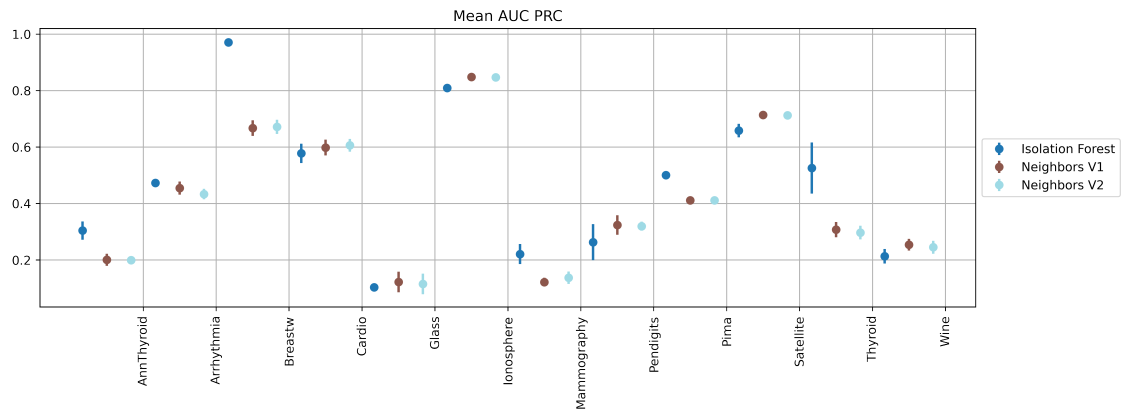


Figure 5.8: Comparison of AUC PRC Metrics with Standard Deviation

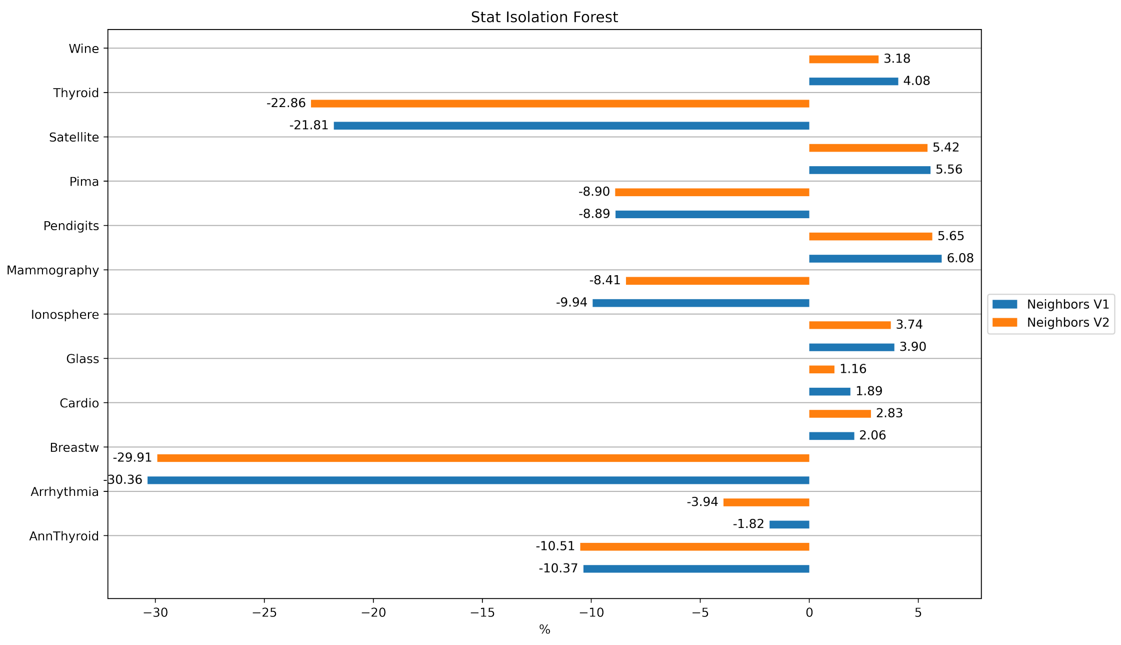


Figure 5.9: Graphical view of performance against the standard method

5.3.3 Component Gap

The AUC PRC values and standard deviation for the Artificial and Real experiments are shown in Tables 5.7 and 5.9 respectively.

Additional tests were performed in which a random number of features or half of the available features were considered to select the split value. The results obtained, for both the Artificial and Real datasets, from these additional tests in terms of AUC PRC are shown in Tables 5.8 and 5.10 respectively.

In the experiment with Artificial datasets, when all features are considered, the best version is undoubtedly the fourth (Table 5.7). This is because once all split values have been calculated (uniformly sampled), the split value is chosen randomly (influenced by the gap). This means that the maximum value is not always chosen. It is important to note that when using a random or fixed number of features (Table 5.8), the results are better. This is because uninformative features are avoided.

In the experiment with Real datasets, when all features are considered, the best version is undoubtedly the fourth (Table 5.9), as in the previous case. When using a random or fixed number of features, the results improve considerably (Table 5.10).

Artificial

Dataset names	Isolation Forest	Component Gap			
	Standard	V1	V2	V3	V4
Anomalous Cluster	0.9024 ± 0.0170	0.7988 ± 0.0221	0.8684 ± 0.0142	0.9068 ± 0.0096	**0.9160 ± 0.0077**
Central Cluster	**0.9992 ± 0.0005**	0.9934 ± 0.0028	0.9940 ± 0.0021	0.9963 ± 0.0006	0.9960 ± 0.0004
Double Clusters	0.9546 ± 0.0132	0.9279 ± 0.0139	0.9304 ± 0.0134	0.9795 ± 0.0131	**0.9847 ± 0.0084**
Four Clusters	0.9287 ± 0.0132	0.9246 ± 0.0186	0.9375 ± 0.0123	0.9718 ± 0.0105	**0.9805 ± 0.0076**
Square Toroid	0.2385 ± 0.0258	0.4941 ± 0.0345	0.5248 ± 0.0300	0.5571 ± 0.0553	**0.6142 ± 0.0440**

Table 5.7: Precision-Recall AUC

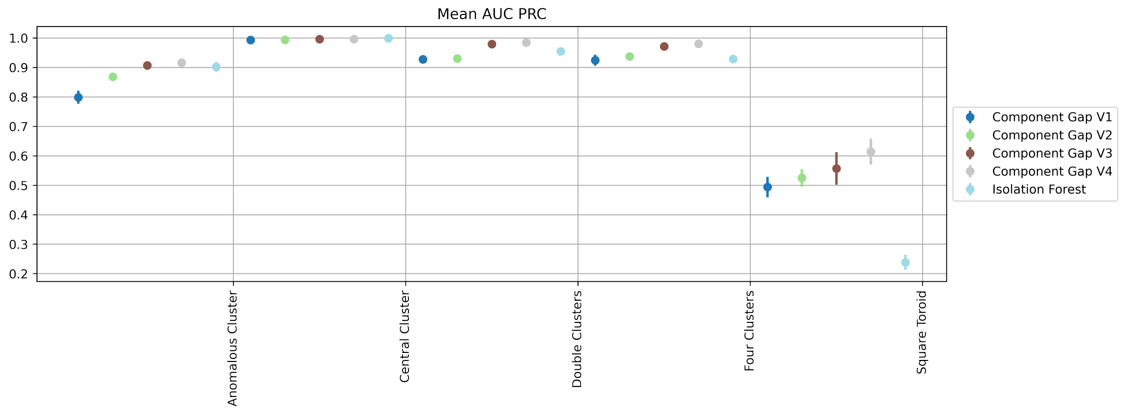


Figure 5.10: Comparison of AUC PRC Metrics with Standard Deviation

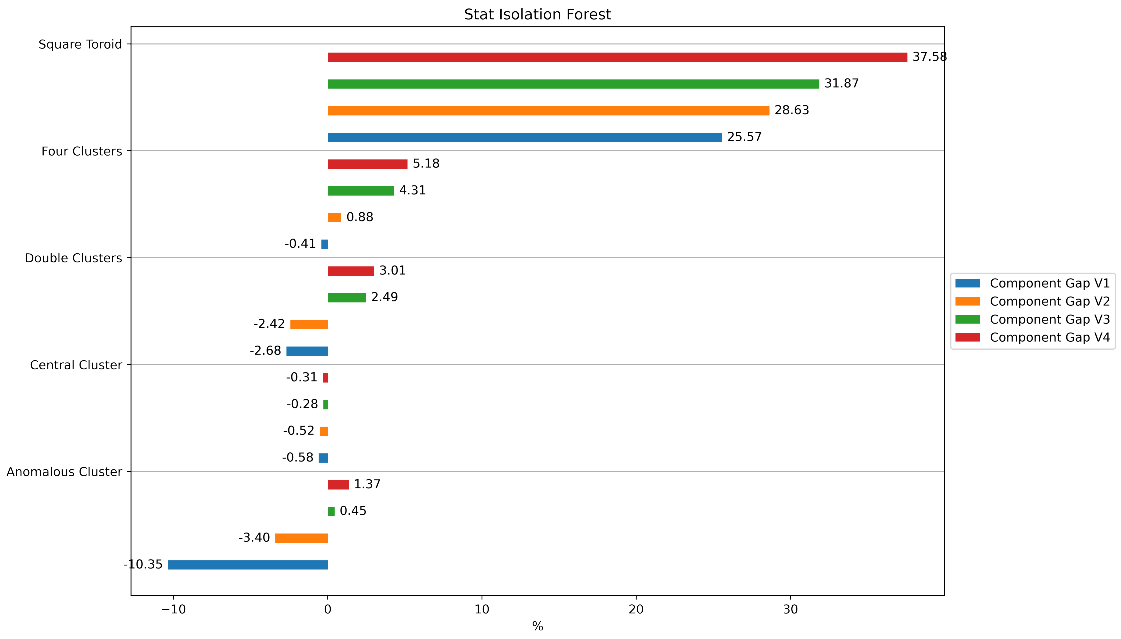


Figure 5.11: Graphical view of performance against the standard method

Dataset names	Isolation Forest	Component Gap	Component Gap	Component Gap
	Standard	V4	V4 n/2 features	V4 random number of features
Anomalous Cluster	0.9024 ± 0.0170	0.9160 ± 0.0077	**0.9432 ± 0.0066**	0.9398 ± 0.0062
Central Cluster	**0.9992 ± 0.0005**	0.9960 ± 0.0004	0.9967 ± 0.0005	0.9969 ± 0.0005
Double Clusters	0.9546 ± 0.0132	0.9847 ± 0.0084	0.9928 ± 0.0037	**0.9939 ± 0.0045**
Four Clusters	0.9287 ± 0.0132	0.9805 ± 0.0076	0.9876 ± 0.0028	**0.9889 ± 0.0031**
Square Toroid	0.2385 ± 0.0258	0.6142 ± 0.0440	**0.7099 ± 0.0488**	0.6936 ± 0.0727

Table 5.8: Precision-Recall AUC

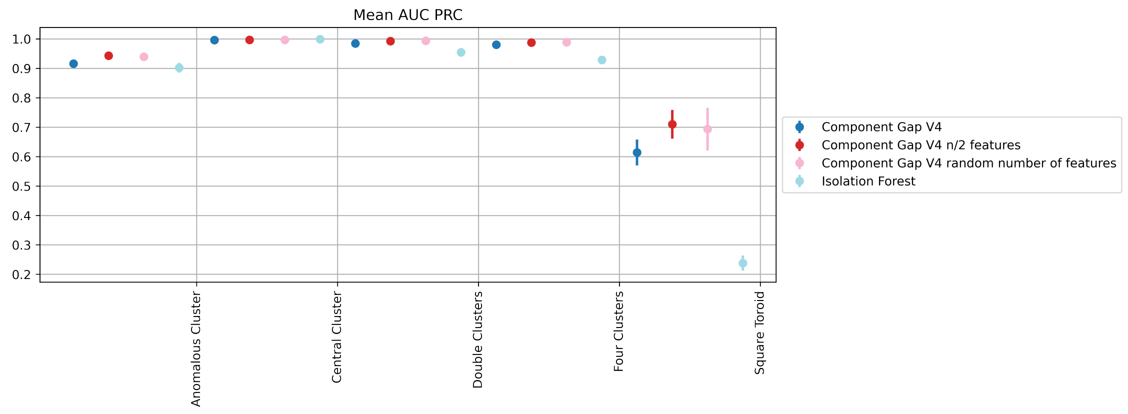


Figure 5.12: Comparison of AUC PRC Metrics with Standard Deviation

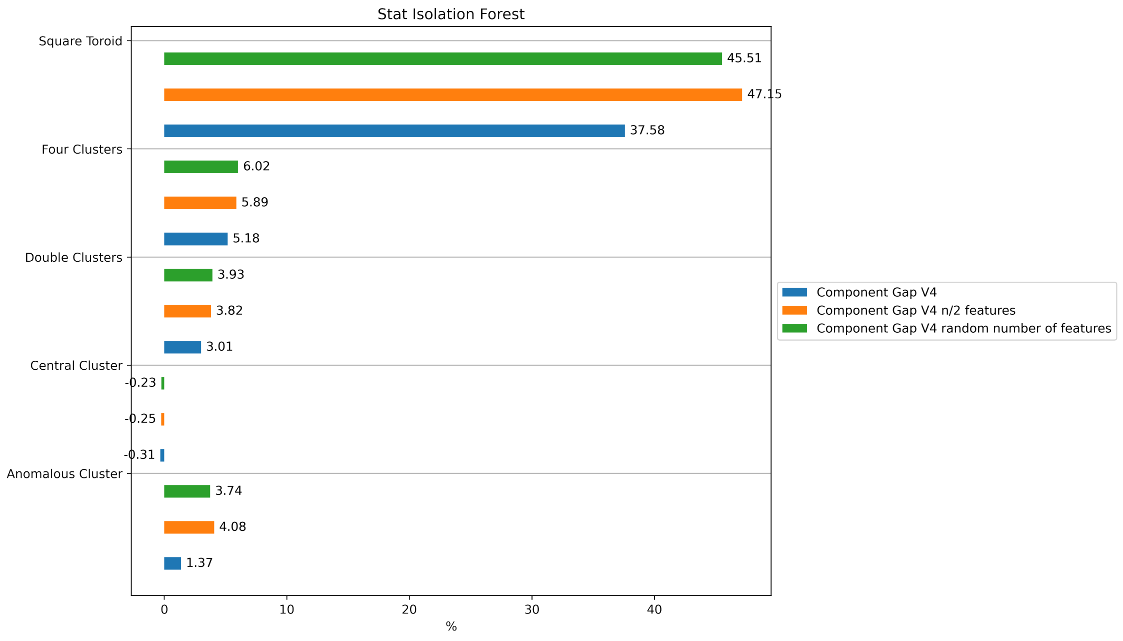


Figure 5.13: Graphical view of performance against the standard method

Real

Dataset names	Isolation Forest	Component Gap			
	Standard	V1	V2	V3	V4
AnnThyroid	**0.3042 ± 0.0323**	0.0812 ± 0.0014	0.0795 ± 0.0022	0.1207 ± 0.0058	0.1246 ± 0.0085
Arrhythmia	0.4729 ± 0.0140	0.3334 ± 0.0045	0.3394 ± 0.0072	**0.4796 ± 0.0149**	0.4750 ± 0.0123
Breastw	**0.9707 ± 0.0043**	0.7689 ± 0.0343	0.9476 ± 0.0040	0.7341 ± 0.0361	0.7032 ± 0.0485
Cardio	**0.5776 ± 0.0343**	0.4671 ± 0.0176	0.4495 ± 0.0229	0.5598 ± 0.0180	0.5727 ± 0.0283
Glass	0.1033 ± 0.0106	0.1460 ± 0.0000	0.1460 ± 0.0000	0.2023 ± 0.0076	**0.2044 ± 0.0062**
Ionosphere	0.8095 ± 0.0072	0.7360 ± 0.0056	0.8030 ± 0.0047	**0.8514 ± 0.0039**	0.8469 ± 0.0076
Mammography	0.2211 ± 0.0357	**0.2285 ± 0.0336**	0.2282 ± 0.0226	0.1756 ± 0.0146	0.1740 ± 0.0195
Pendigits	**0.2631 ± 0.0639**	0.0823 ± 0.0074	0.0881 ± 0.0102	0.2471 ± 0.0182	0.2606 ± 0.0268
Pima	**0.5005 ± 0.0089**	0.4860 ± 0.0057	0.4499 ± 0.0060	0.4647 ± 0.0055	0.4605 ± 0.0080
Satellite	0.6583 ± 0.0237	0.4873 ± 0.0078	0.4923 ± 0.0054	0.6758 ± 0.0113	**0.6827 ± 0.0060**
Thyroid	**0.5257 ± 0.0908**	0.0486 ± 0.0033	0.0474 ± 0.0068	0.1295 ± 0.0076	0.1242 ± 0.0091
Wine	0.2133 ± 0.0257	**0.9587 ± 0.0000**	**0.9587 ± 0.0000**	0.9375 ± 0.0221	0.9384 ± 0.0178

Table 5.9: Precision-Recall AUC

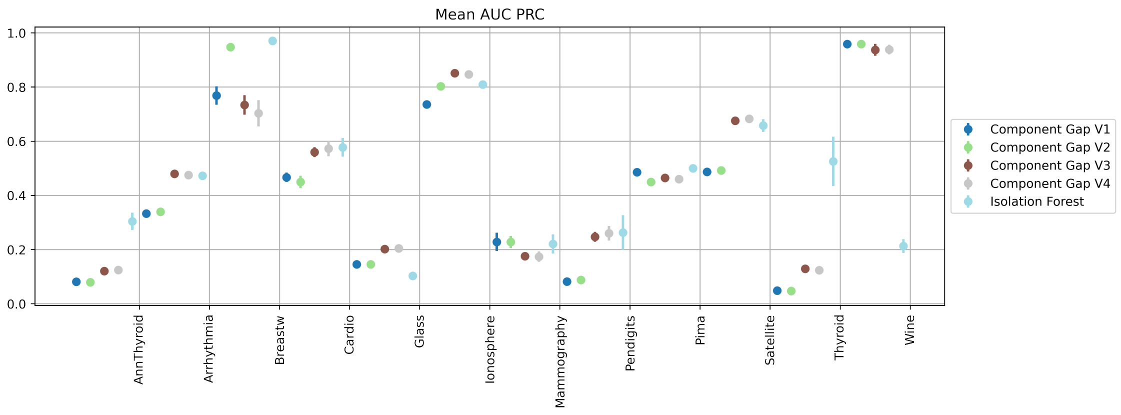


Figure 5.14: Comparison of AUC PRC Metrics with Standard Deviation

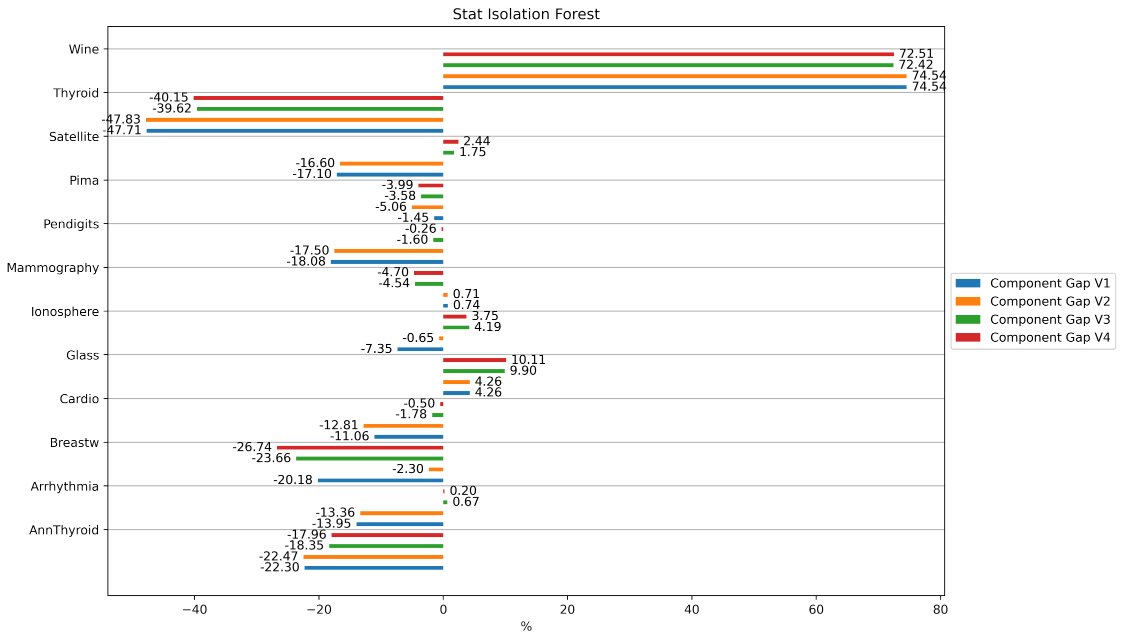


Figure 5.15: Graphical view of performance against the standard method

Dataset names	Isolation Forest	Component Gap	
	Standard	V4 n/2 features	V4 random number of features
AnnThyroid	**0.3042 ± 0.0323**	0.1471 ± 0.0078	0.1533 ± 0.0090
Arrhythmia	0.4729 ± 0.0140	0.4749 ± 0.0170	**0.4798 ± 0.0123**
Breastw	**0.9707 ± 0.0043**	0.6123 ± 0.0399	0.5839 ± 0.0256
Cardio	0.5776 ± 0.0343	0.5585 ± 0.0152	**0.5843 ± 0.0260**
Glass	0.1033 ± 0.0106	0.1714 ± 0.0031	**0.1750 ± 0.0042**
Ionosphere	0.8095 ± 0.0072	0.8456 ± 0.0050	**0.8472 ± 0.0052**
Mammography	**0.2211 ± 0.0357**	0.1545 ± 0.0353	0.1672 ± 0.0249
Pendigits	0.2631 ± 0.0639	**0.2785 ± 0.0198**	0.2620 ± 0.0185
Pima	**0.5005 ± 0.0089**	0.4457 ± 0.0071	0.4461 ± 0.0086
Satellite	0.6583 ± 0.0237	0.6882 ± 0.0110	**0.6893 ± 0.0050**
Thyroid	**0.5257 ± 0.0908**	0.1647 ± 0.0087	0.1776 ± 0.0127
Wine	0.2133 ± 0.0257	0.6898 ± 0.0366	**0.7117 ± 0.0430**

Table 5.10: Precision-Recall AUC

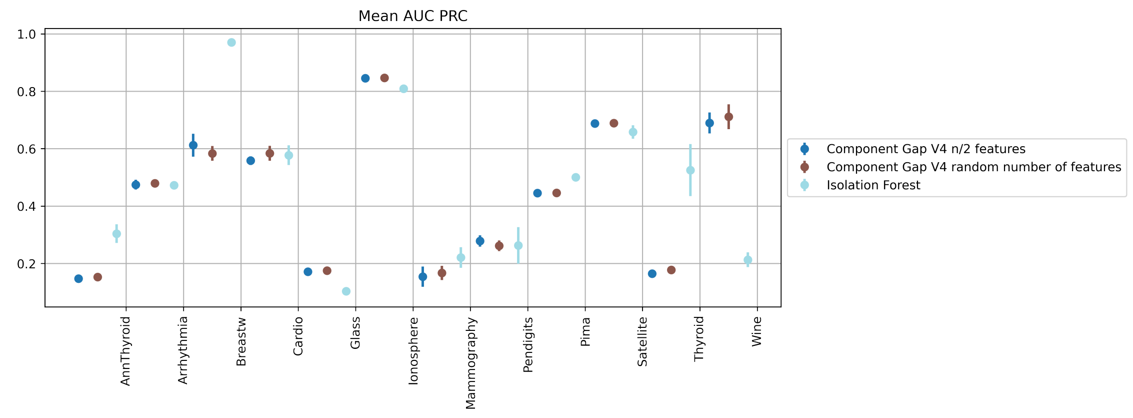


Figure 5.16: Comparison of AUC PRC Metrics with Standard Deviation

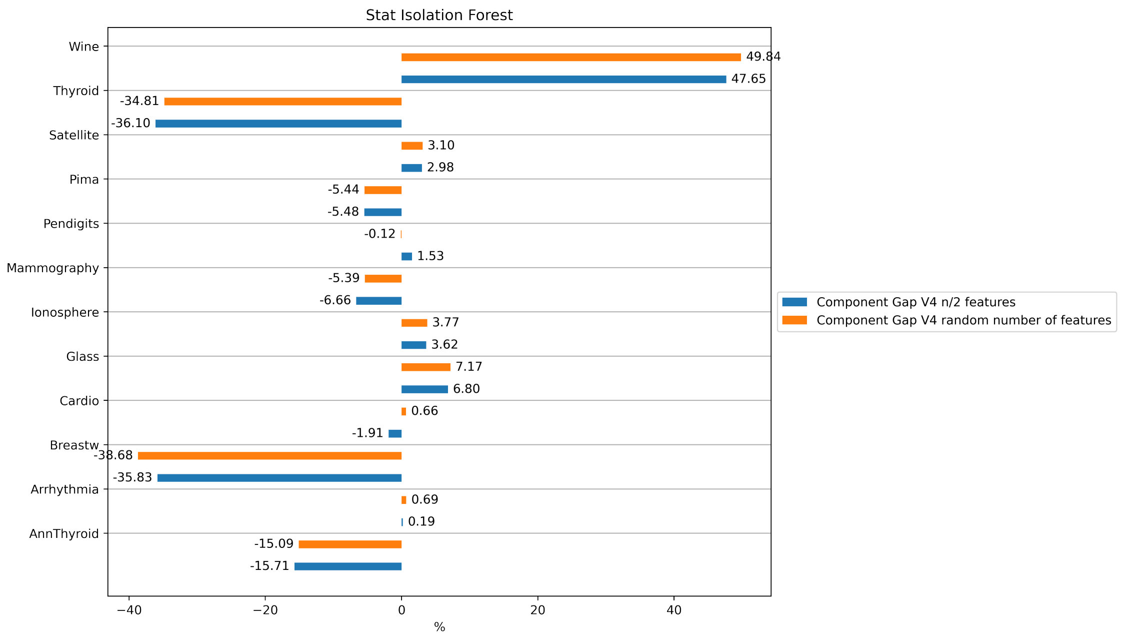


Figure 5.17: Graphical view of performance against the standard method

5.3.4 Bubble Forest

The AUC PRC values and standard deviation for the Artificial and Real experiments are shown in Tables 5.11 and 5.12 respectively.

To our great surprise, for the experiment with the Artificial dataset, the variant that performed well was the data-independent variant (V5).

On the other hand, for the experiment with the real dataset, the variant that performed well was the data-dependent one (V4). This happens because the features are correlated.

Artificial

Dataset names	Isolation Forest	Bubble Forest L1	Bubble Forest L2	Bubble Forest LINF
	Standard	V4 random component	V5 $\frac{1}{1+depth}$	V5 $\frac{1}{1+depth}$
Anomalous Cluster	0.9024 ± 0.0170	0.9579 ± 0.0155	**0.9641 ± 0.0062**	0.9376 ± 0.0053
Central Cluster	**0.9992 ± 0.0005**	0.9991 ± 0.0005	0.9696 ± 0.0090	0.9790 ± 0.0080
Double Clusters	0.9546 ± 0.0132	0.9737 ± 0.0077	**0.9959 ± 0.0024**	0.9928 ± 0.0036
Four Clusters	0.9287 ± 0.0132	0.9900 ± 0.0050	**0.9987 ± 0.0008**	0.9974 ± 0.0012
Square Toroid	0.2385 ± 0.0258	**0.8082 ± 0.0570**	0.3394 ± 0.0529	0.5907 ± 0.0549

Table 5.11: Precision-Recall AUC

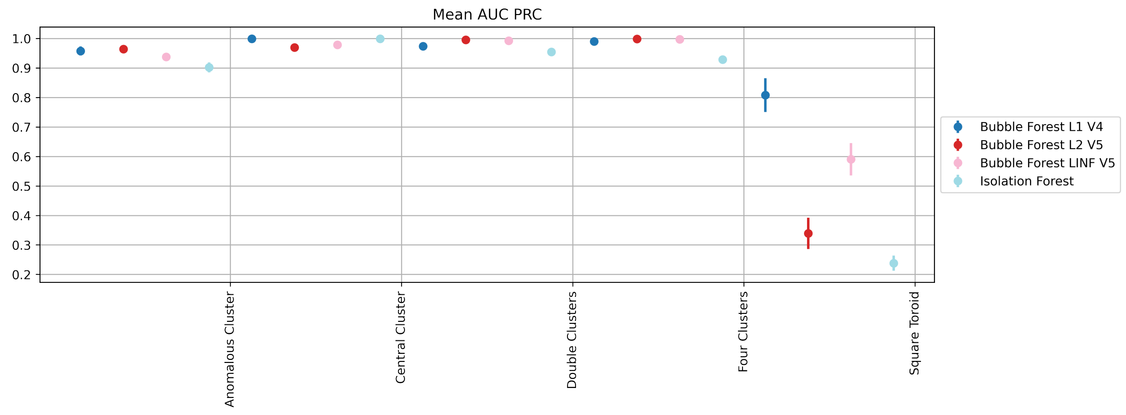


Figure 5.18: Comparison of AUC PRC Metrics with Standard Deviation

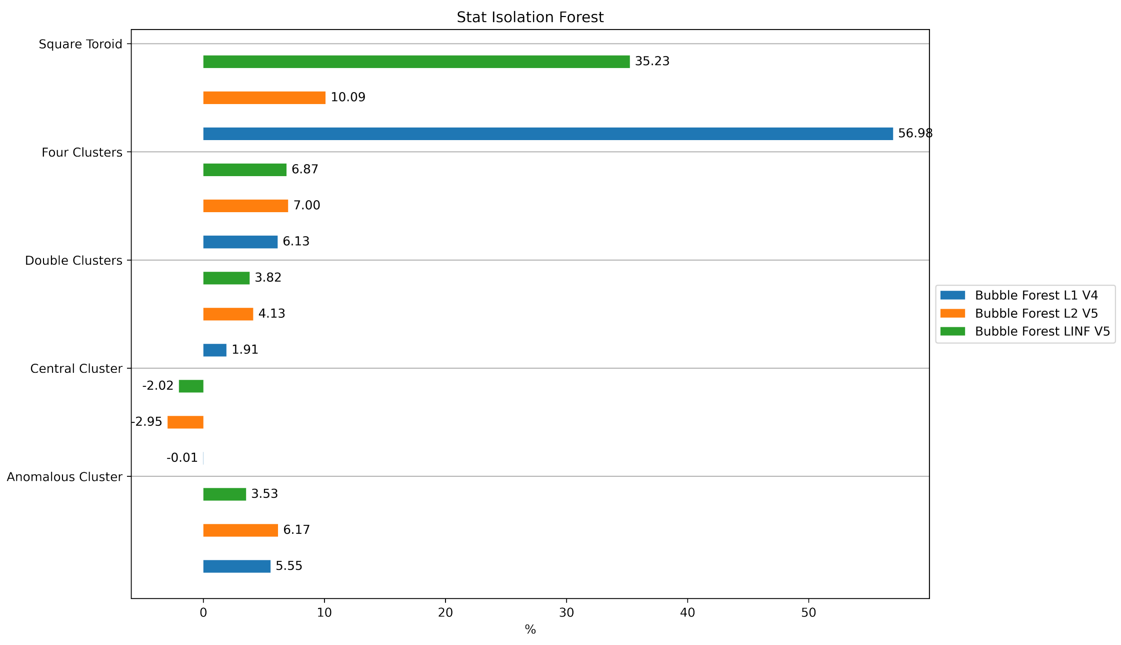


Figure 5.19: Graphical view of performance against the standard method

Real

Dataset names	Isolation Forest	Bubble Forest L1	Bubble Forest L2	Bubble Forest LINF
	Standard	V1 mean	V4 random component	V2 max
AnnThyroid	**0.3042 ± 0.0323**	0.2119 ± 0.0102	0.1257 ± 0.0042	0.1331 ± 0.0051
Arrhythmia	0.4729 ± 0.0140	**0.4974 ± 0.0175**	0.4269 ± 0.0243	0.3515 ± 0.0088
Breastw	**0.9707 ± 0.0043**	0.7894 ± 0.0455	0.7497 ± 0.0770	0.9209 ± 0.0282
Cardio	0.5776 ± 0.0343	**0.6253 ± 0.0462**	0.6041 ± 0.0218	0.4557 ± 0.0096
Glass	0.1033 ± 0.0106	**0.1629 ± 0.0258**	0.1190 ± 0.0126	0.1033 ± 0.0145
Ionosphere	0.8095 ± 0.0072	0.4978 ± 0.0299	**0.8831 ± 0.0298**	0.6638 ± 0.0396
Mammography	**0.2211 ± 0.0357**	0.1634 ± 0.0175	0.1746 ± 0.0114	0.1852 ± 0.0067
Pendigits	0.2631 ± 0.0639	**0.3710 ± 0.0897**	0.1178 ± 0.0816	0.1589 ± 0.0406
Pima	**0.5005 ± 0.0089**	0.4815 ± 0.0146	0.4793 ± 0.0109	0.4863 ± 0.0058
Satellite	**0.6583 ± 0.0237**	0.3369 ± 0.0138	0.3904 ± 0.0174	0.5349 ± 0.0909
Thyroid	**0.5257 ± 0.0908**	0.3545 ± 0.0515	0.1849 ± 0.0191	0.1655 ± 0.0109
Wine	0.2133 ± 0.0257	0.8923 ± 0.0664	**0.9294 ± 0.0174**	0.7044 ± 0.0476

Table 5.12: Precision-Recall AUC

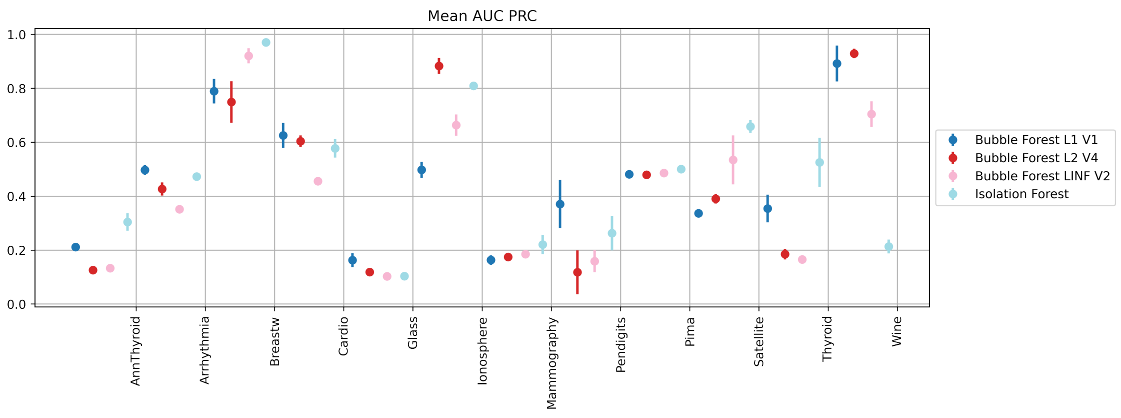


Figure 5.20: Comparison of AUC PRC Metrics with Standard Deviation

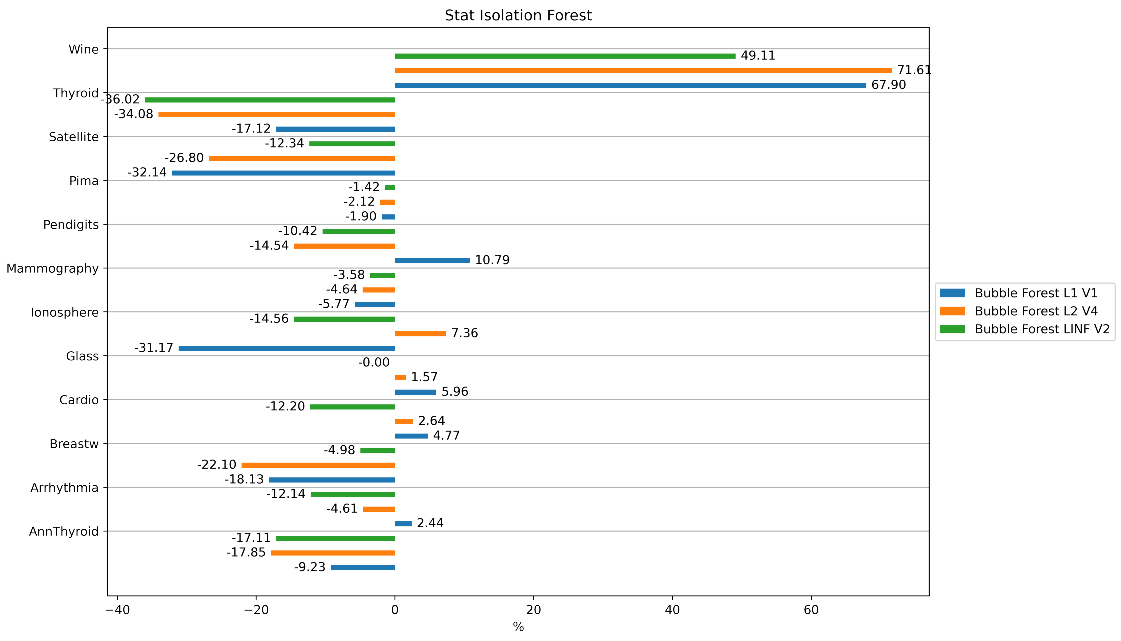


Figure 5.21: Graphical view of performance against the standard method

5.3.5 Discussion on artificial datasets

Table 5.13 shows the AUC PRC values of the best implementations regarding the split criterion on artificial datasets.

Dataset names	Isolation Forest	Bubble Forest L2	BFWP V2	Prob Split
	Standard	V5	Bubble Forest V4	V1
Anomalous Cluster	0.9024 ± 0.0170	**0.9641 ± 0.0062**	0.9572 ± 0.0067	0.9123 ± 0.0080
Central Cluster	0.9992 ± 0.0005	0.9696 ± 0.0090	**0.9998 ± 0.0002**	0.9964 ± 0.0018
Double Clusters	0.9546 ± 0.0132	**0.9959 ± 0.0024**	0.9886 ± 0.0071	0.9700 ± 0.0078
Four Clusters	0.9287 ± 0.0132	**0.9987 ± 0.0008**	0.9910 ± 0.0069	0.9742 ± 0.0054
Square Toroid	0.2385 ± 0.0258	0.3394 ± 0.0529	0.2135 ± 0.0302	**0.8930 ± 0.0275**

Table 5.13: Precision-Recall AUC

Bubble Forest

In this case, norm 2 is used and V5 is used to choose the radius, i.e., the radius is independent of the data and is scaled by depth, as described above.

As can be seen in Table 5.13 and Graph 5.23 the AUC PRC values are always greater than the standard method, except for the Central Cluster dataset.

Bubble Forest Weighted Path (BFWP)

In this case, the Bubble Forest V4 approach is used, where the choice of radius is data-dependent. As for the anomaly score assigned to each instance, V2 is used.

As we can see from Table 5.13 and graph 5.23 the AUC PRC values are always greater than the standard method except for the Square Toroid dataset. This is the only method that manages to outperform the standard method in the Central Cluster dataset, this is due to the definition of the new scoring function.

Prob split

In this case, V1 of the proposed method is used, where the split value is chosen as the midpoint between a point q and the chosen extreme (influenced by the gap). As we can see from Table 5.13 and Graph 5.23 the AUC PRC values are always greater than the standard method except for the Central Cluster dataset. Furthermore we can observe a significant increase in performance in the Square Toroid dataset of about 65% compared to the standard method.

It is important to note that this method is the simplest and least expensive in terms of computational cost among all the proposed methods.

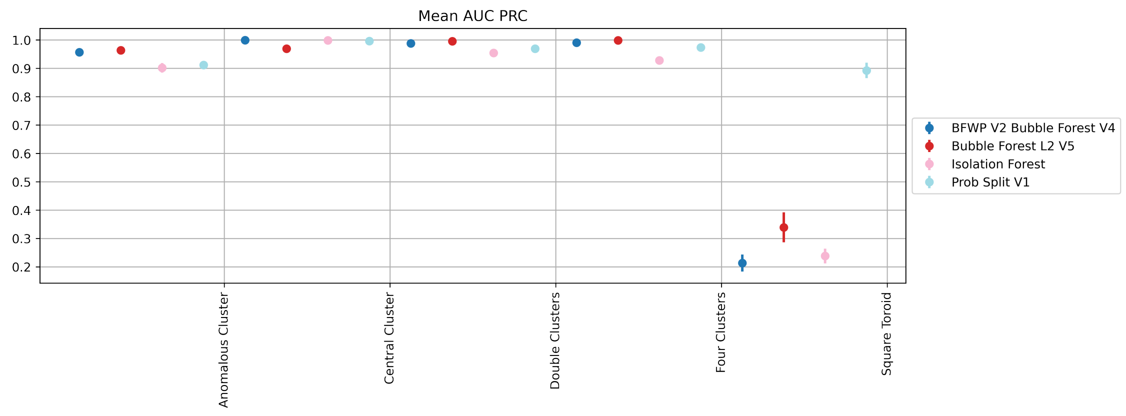


Figure 5.22: Comparison of AUC PRC Metrics with Standard Deviation

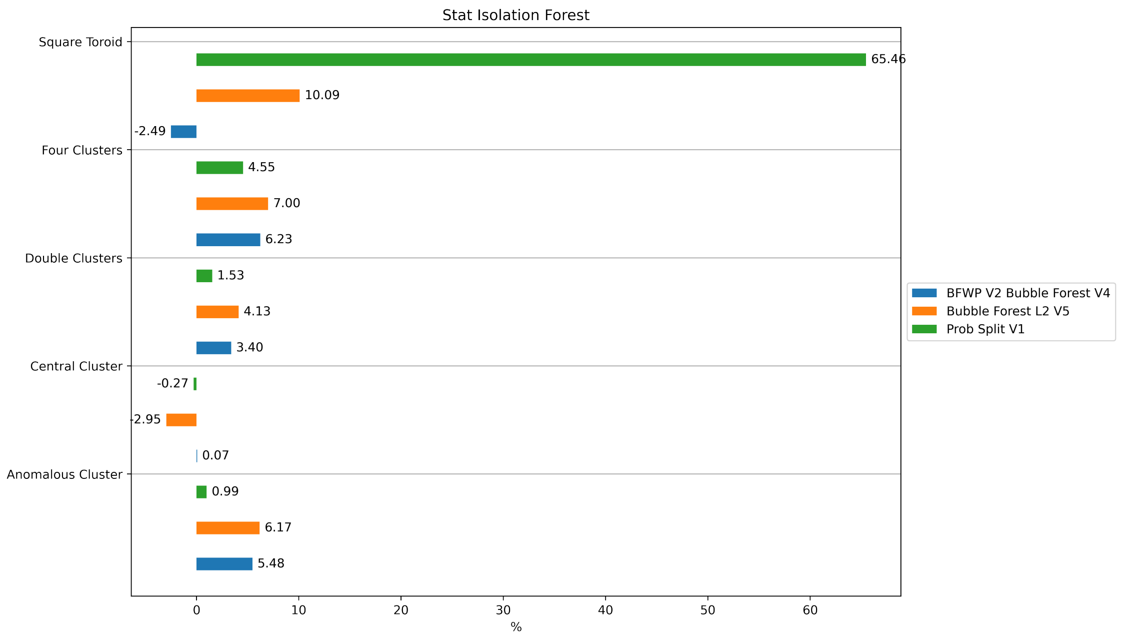


Figure 5.23: Graphical view of performance against the standard method

5.3.6 Discussion on real datasets

Table 5.14 shows the AUC PRC values of the best implementations concerning the split criterion on real datasets.

The method Bubble Forest has a noticeable performance increase, but in a few datasets.

The Prob split method, although it does not have any noticeable increase in performance compared to the standard method, works on average well on datasets.

Dataset names	Isolation Forest	Prob Split	Bubble Forest L1
	Standard	V2	V1
AnnThyroid	0.3042 ± 0.0323	**0.3227 ± 0.0195**	0.2119 ± 0.0102
Arrhythmia	0.4729 ± 0.0140	0.4344 ± 0.0371	**0.4974 ± 0.0175**
Breastw	0.9707 ± 0.0043	**0.9715 ± 0.0032**	0.7894 ± 0.0455
Cardio	0.5776 ± 0.0343	0.5490 ± 0.0369	**0.6253 ± 0.0462**
Glass	0.1033 ± 0.0106	0.0931 ± 0.0112	**0.1629 ± 0.0258**
Ionosphere	**0.8095 ± 0.0072**	0.8021 ± 0.0085	0.4978 ± 0.0299
Mammography	0.2211 ± 0.0357	**0.2286 ± 0.0476**	0.1634 ± 0.0175
Pendigits	0.2631 ± 0.0639	0.2843 ± 0.0574	**0.3710 ± 0.0897**
Pima	0.5005 ± 0.0089	**0.5024 ± 0.0135**	0.4815 ± 0.0146
Satellite	0.6583 ± 0.0237	**0.6651 ± 0.0221**	0.3369 ± 0.0138
Thyroid	0.5257 ± 0.0908	**0.5461 ± 0.0380**	0.3545 ± 0.0515
Wine	0.2133 ± 0.0257	0.1877 ± 0.0472	**0.8923 ± 0.0664**

Table 5.14: Precision-Recall AUC

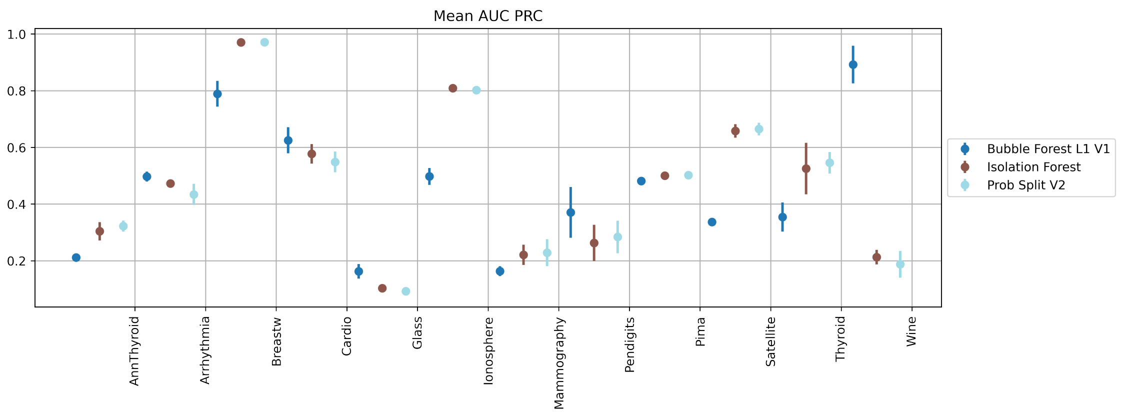


Figure 5.24: Comparison of AUC PRC Metrics with Standard Deviation

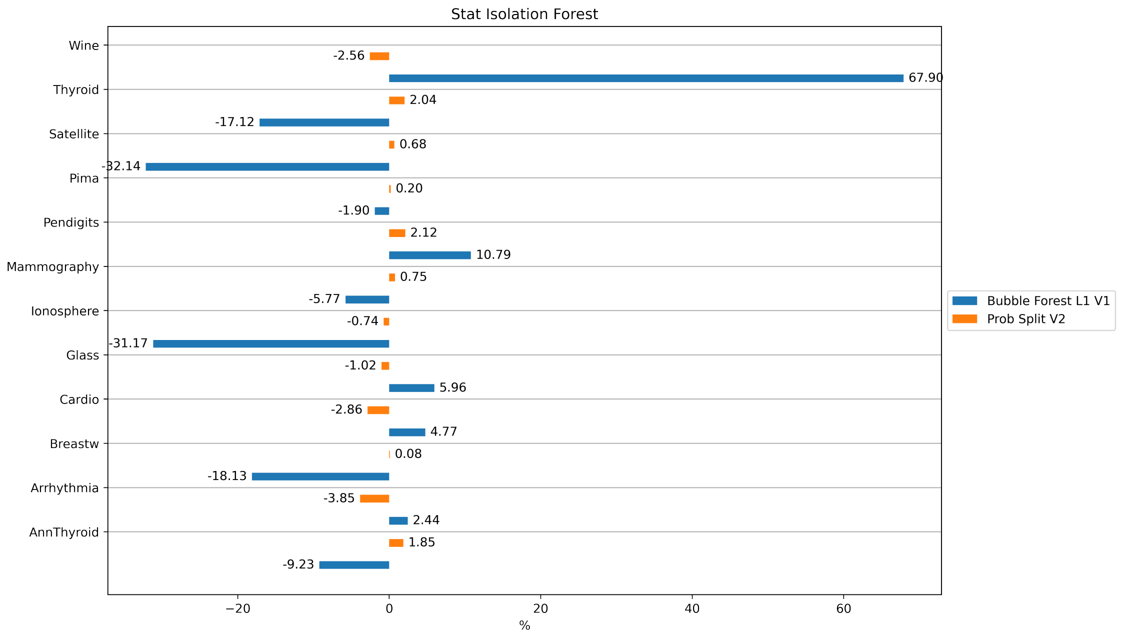


Figure 5.25: Comparison of AUC PRC Metrics with Standard Deviation

5.4 Results of the anomaly score criterion experiments

5.4.1 Weighted Path

The AUC PRC values and standard deviation for the Artificial and Real experiments are shown in Tables 5.15 and 5.16 respectively.

In the experiment with Artificial datasets all variants have good but similar performance, the one with the best performance is V2 with the Component Gap V4 method.

In the experiment with the datasets, there are performance gains, but in a few datasets.

Artificial

Dataset names	Isolation Forest	Weighted Path V1	Weighted Path V2	Weighted Path V3
	Standard	Component Gap V4	Component Gap V4	Component Gap V4
Anomalous Cluster	0.9024 ± 0.0170	0.9180 ± 0.0079	**0.9184 ± 0.0105**	0.8811 ± 0.0096
Central Cluster	**0.9992 ± 0.0005**	0.9968 ± 0.0005	0.9970 ± 0.0007	0.9968 ± 0.0005
Double Clusters	0.9546 ± 0.0132	0.9915 ± 0.0052	0.9918 ± 0.0046	**0.9935 ± 0.0050**
Four Clusters	0.9287 ± 0.0132	**0.9858 ± 0.0040**	0.9780 ± 0.0051	0.9843 ± 0.0038
Square Toroid	0.2385 ± 0.0258	0.6978 ± 0.0447	**0.7267 ± 0.0549**	0.4560 ± 0.0773

Table 5.15: Precision-Recall AUC

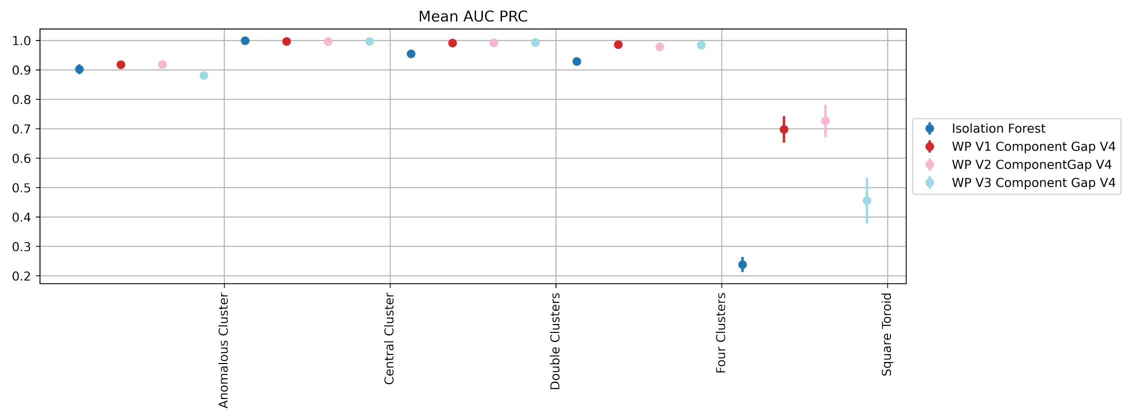


Figure 5.26: Comparison of AUC PRC Metrics with Standard Deviation

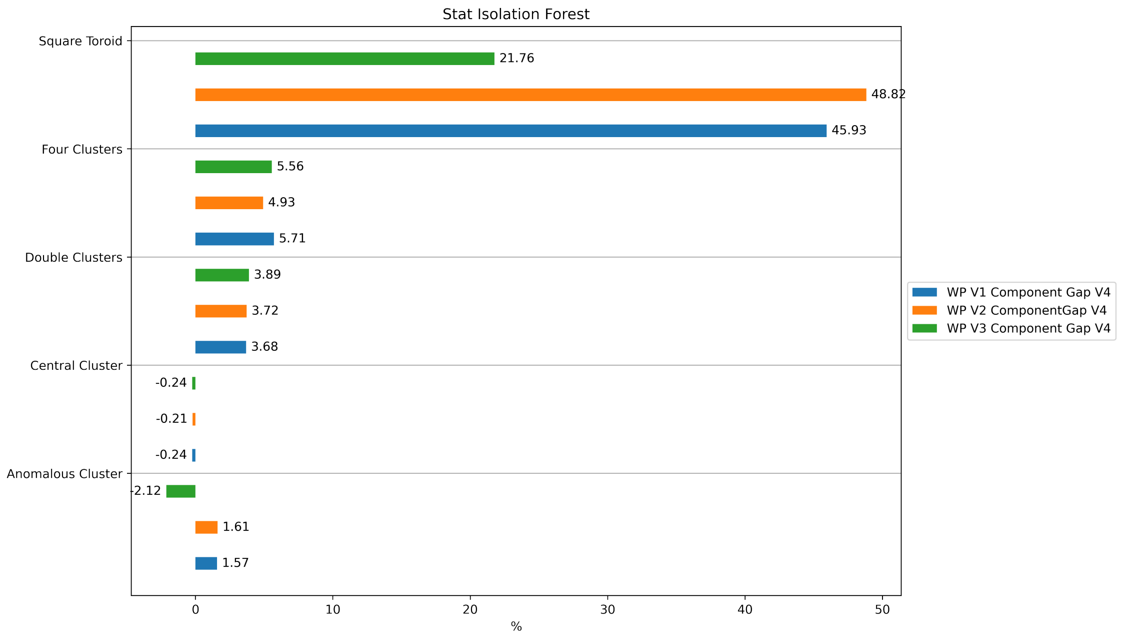


Figure 5.27: Graphical view of performance against the standard method

Real

Dataset names	Isolation Forest	Weighted Path V1	Weighted Path V2	Weighted Path V3
	Standard	Component Gap V4 random number of features	Component Gap V4 random number of features	Component Gap V4 random number of features
AnnThyroid	**0.3042 ± 0.0323**	0.1472 ± 0.0054	0.1476 ± 0.0110	0.1467 ± 0.0105
Arrhythmia	**0.4729 ± 0.0140**	0.4722 ± 0.0120	0.4316 ± 0.0502	0.4435 ± 0.0240
Breastw	**0.9707 ± 0.0043**	0.5594 ± 0.0491	0.8761 ± 0.0176	0.8351 ± 0.0259
Cardio	0.5776 ± 0.0343	**0.5839 ± 0.0314**	0.5610 ± 0.0242	0.5720 ± 0.0286
Glass	0.1033 ± 0.0106	0.1734 ± 0.0041	0.1735 ± 0.0044	**0.1806 ± 0.0048**
Ionosphere	0.8095 ± 0.0072	0.8424 ± 0.0038	**0.8459 ± 0.0079**	0.8423 ± 0.0056
Mammography	**0.2211 ± 0.0357**	0.1679 ± 0.0336	0.1863 ± 0.0380	0.1195 ± 0.0319
Pendigits	0.2631 ± 0.0639	0.2808 ± 0.0323	**0.2982 ± 0.0285**	0.2890 ± 0.0288
Pima	**0.5005 ± 0.0089**	0.4476 ± 0.0051	0.4219 ± 0.0181	0.4251 ± 0.0077
Satellite	0.6583 ± 0.0237	**0.6988 ± 0.0139**	0.6609 ± 0.0271	0.6861 ± 0.0084
Thyroid	**0.5257 ± 0.0908**	0.1698 ± 0.0151	0.1822 ± 0.0069	0.1700 ± 0.0152
Wine	0.2133 ± 0.0257	0.7152 ± 0.0023	**0.7594 ± 0.1033**	0.7403 ± 0.0944

Table 5.16: Precision-Recall AUC

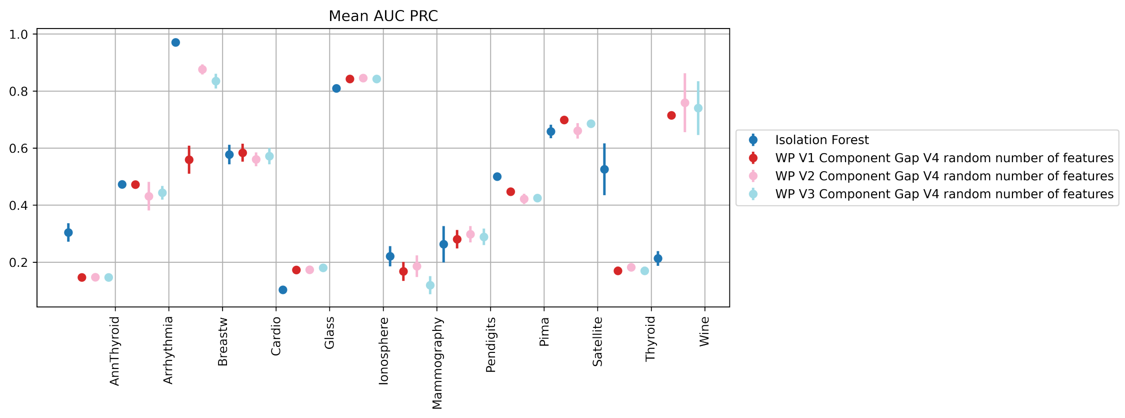


Figure 5.28: Comparison of AUC PRC Metrics with Standard Deviation

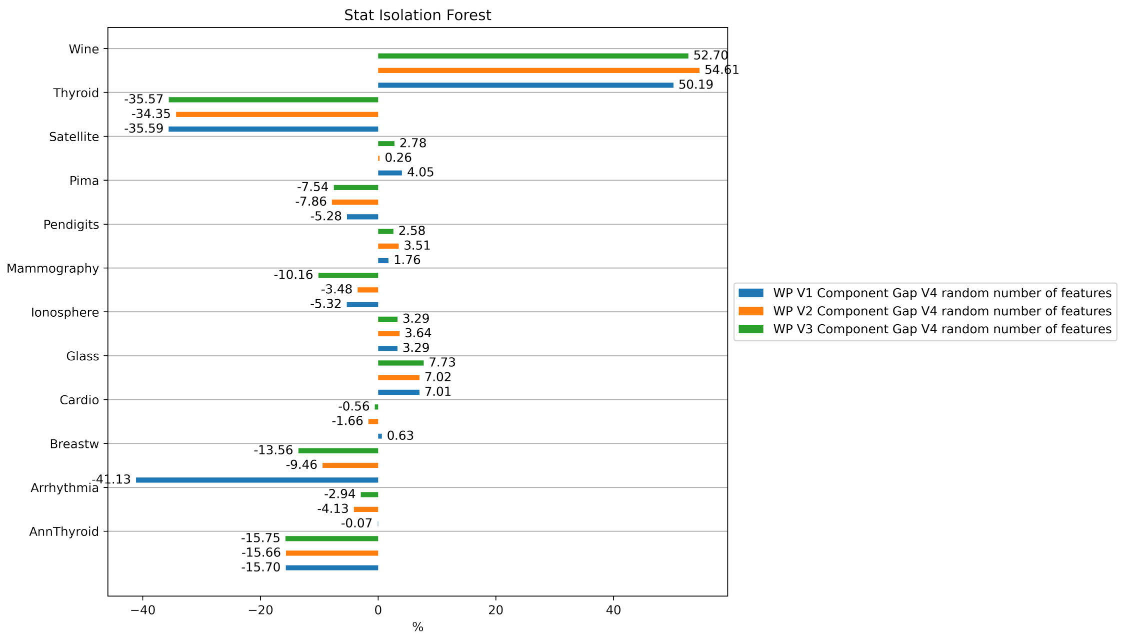


Figure 5.29: Graphical view of performance against the standard method

5.4.2 Bubble Forest Weighted Path

The AUC PRC values and standard deviation for the Artificial and Real experiments are shown in Tables 5.17 and 5.18 respectively.

In the experiment with Artificial datasets, there is a noticeable increase in performance except for the Square Toroid dataset.

In the experiment with the datasets, there are performance gains, but in a few datasets.

Artificial

Dataset names	Isolation Forest	BFWP V1	BFWP V2
	Standard	Bubble Forest V4	Bubble Forest V4
Anomalous Cluster	0.9024 ± 0.0170	0.9361 ± 0.0071	**0.9572 ± 0.0067**
Central Cluster	0.9992 ± 0.0005	0.9998 ± 0.0002	**0.9998 ± 0.0002**
Double Clusters	0.9546 ± 0.0132	**0.9888 ± 0.0087**	0.9886 ± 0.0071
Four Clusters	0.9287 ± 0.0132	0.9903 ± 0.0040	**0.9910 ± 0.0069**
Square Toroid	**0.2385 ± 0.0258**	0.2220 ± 0.0437	0.2135 ± 0.0302

Table 5.17: Precision-Recall AUC

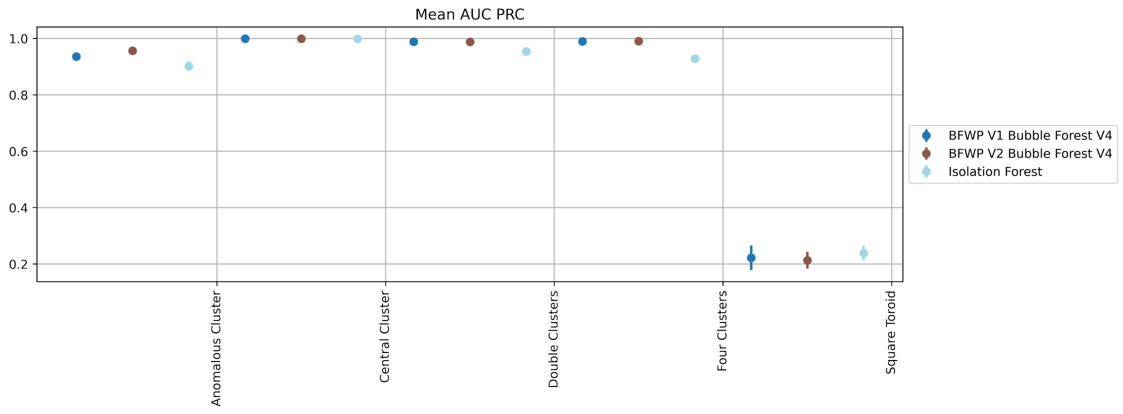


Figure 5.30: Comparison of AUC PRC Metrics with Standard Deviation

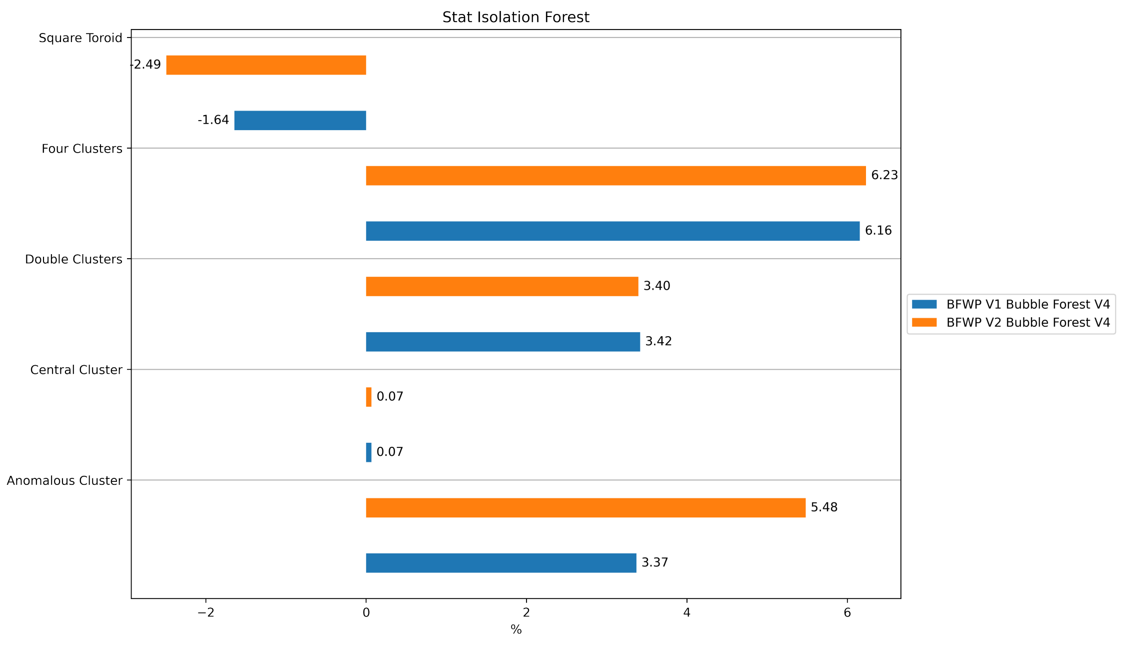


Figure 5.31: Graphical view of performance against the standard method

Real

Dataset names	Isolation Forest	BFWP V1	BFWP V2
	Standard	Bubble Forest L1 V1	
AnnThyroid	**0.3042 ± 0.0323**	0.2159 ± 0.0145	0.2135 ± 0.0119
Arrhythmia	0.4729 ± 0.0140	0.5242 ± 0.0110	**0.5282 ± 0.0195**
Breastw	**0.9707 ± 0.0043**	0.7034 ± 0.0628	0.4756 ± 0.0444
Cardio	0.5776 ± 0.0343	**0.6106 ± 0.0245**	0.5968 ± 0.0166
Glass	0.1033 ± 0.0106	**0.1408 ± 0.0289**	0.1370 ± 0.0098
Ionosphere	**0.8095 ± 0.0072**	0.6599 ± 0.0117	0.6203 ± 0.0241
Mammography	**0.2211 ± 0.0357**	0.1630 ± 0.0204	0.1540 ± 0.0247
Pendigits	0.2631 ± 0.0639	0.3291 ± 0.1006	**0.3736 ± 0.0508**
Pima	**0.5005 ± 0.0089**	0.4885 ± 0.0104	0.4803 ± 0.0136
Satellite	**0.6583 ± 0.0237**	0.4782 ± 0.0170	0.4747 ± 0.0202
Thyroid	**0.5257 ± 0.0908**	0.3596 ± 0.0499	0.3491 ± 0.0585
Wine	0.2133 ± 0.0257	**0.9596 ± 0.0151**	0.9026 ± 0.0327

Table 5.18: Precision-Recall AUC

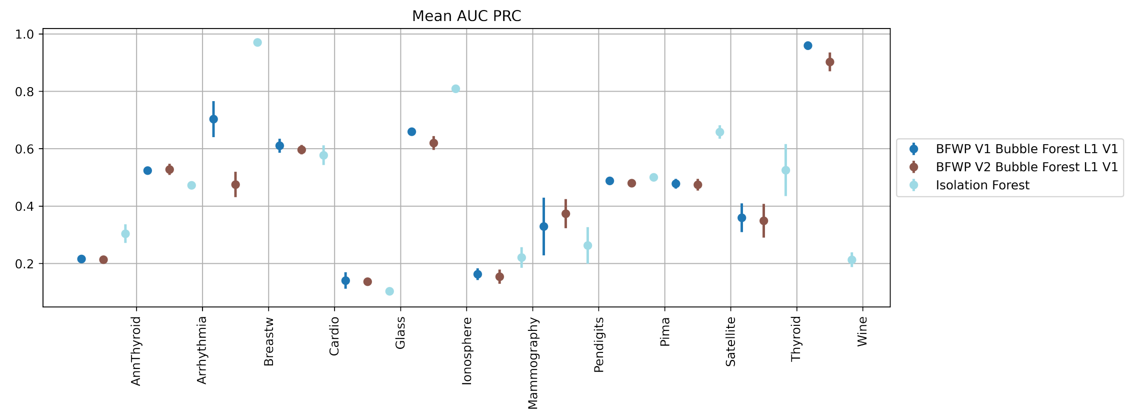


Figure 5.32: Comparison of AUC PRC Metrics with Standard Deviation

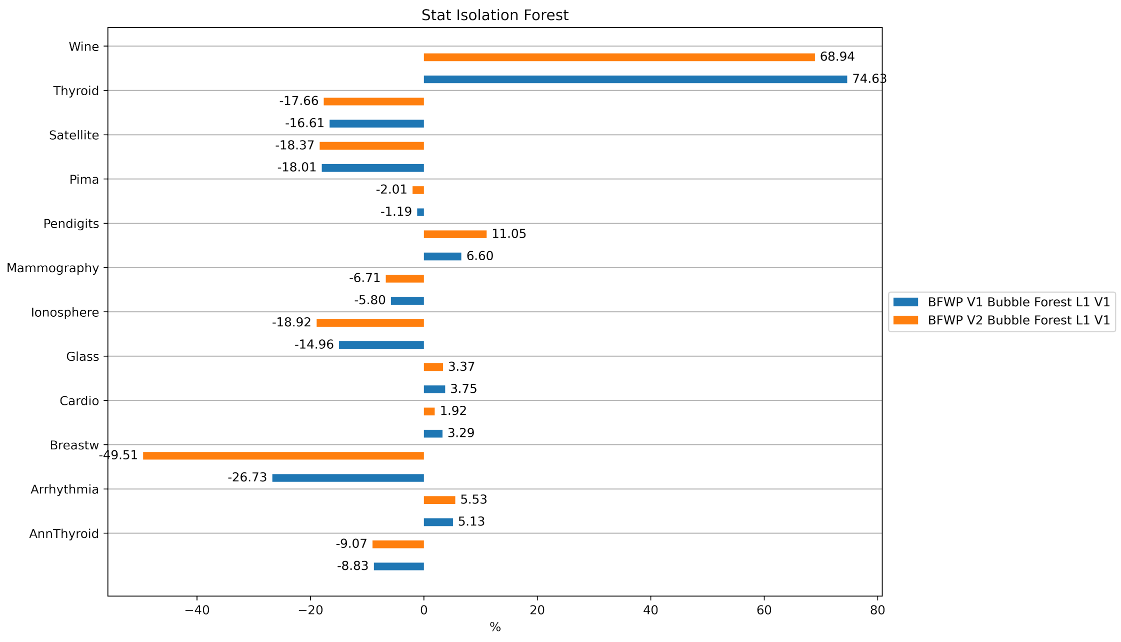


Figure 5.33: Graphical view of performance against the standard method

5.4.3 Discussion on artificial datasets

This new anomaly score function improves the performance of the method compared to the same method with the standard anomaly score function, as can be seen in Table 5.19.

It is important to note that this method is the only one that manages to outperform the standard method in the Central Cluster dataset.

Dataset names	Isolation Forest	BFWP V2
	Standard	Bubble Forest V4
Anomalous Cluster	0.9024 ± 0.0170	**0.9572 ± 0.0067**
Central Cluster	0.9992 ± 0.0005	**0.9998 ± 0.0002**
Double Clusters	0.9546 ± 0.0132	**0.9886 ± 0.0071**
Four Clusters	0.9287 ± 0.0132	**0.9910 ± 0.0069**
Square Toroid	**0.2385 ± 0.0258**	0.2135 ± 0.0302

Table 5.19: Precision-Recall AUC

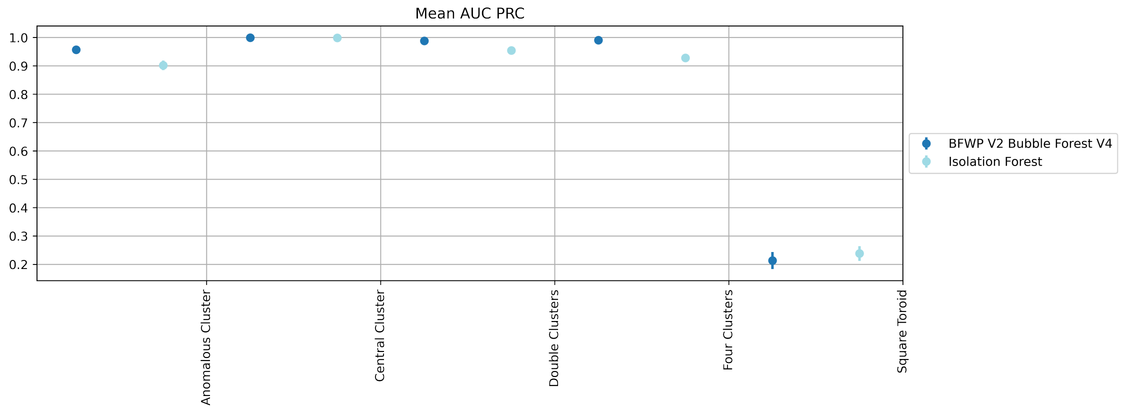


Figure 5.34: Comparison of AUC PRC Metrics with Standard Deviation

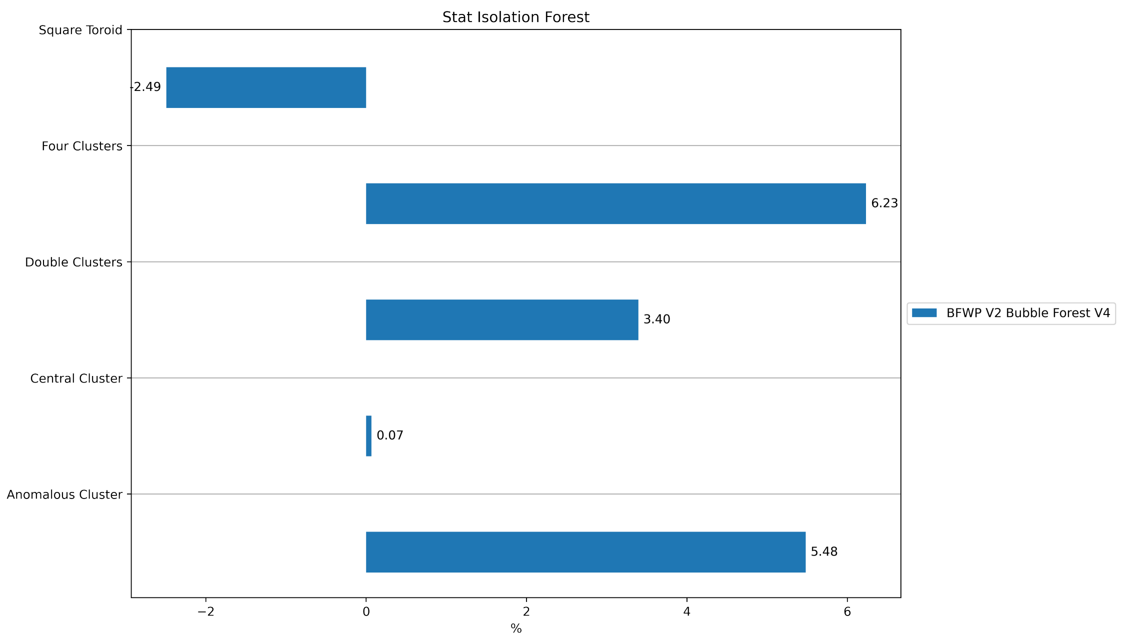


Figure 5.35: Graphical view of performance against the standard method

5.4.4 Discussion on real datasets

In Table 5.20, the best performance is shown against all the proposed implementations of the anomaly score criterion. There is a significant increase in performance compared to the standard method, but only in a few datasets.

Dataset names	Isolation Forest	BFWP V1	WP V2
		Bubble Forest L1 V1	Component Gap V4 random number of features
AnnThyroid	**0.3042 ± 0.0323**	0.2159 ± 0.0145	0.1476 ± 0.0110
Arrhythmia	0.4729 ± 0.0140	**0.5242 ± 0.0110**	0.4316 ± 0.0502
Breastw	**0.9707 ± 0.0043**	0.7034 ± 0.0628	0.8761 ± 0.0176
Cardio	0.5776 ± 0.0343	**0.6106 ± 0.0245**	0.5610 ± 0.0242
Glass	0.1033 ± 0.0106	0.1408 ± 0.0289	**0.1735 ± 0.0044**
Ionosphere	0.8095 ± 0.0072	0.6599 ± 0.0117	**0.8459 ± 0.0079**
Mammography	**0.2211 ± 0.0357**	0.1630 ± 0.0204	0.1863 ± 0.0380
Pendigits	0.2631 ± 0.0639	**0.3291 ± 0.1006**	0.2982 ± 0.0285
Pima	**0.5005 ± 0.0089**	0.4885 ± 0.0104	0.4219 ± 0.0181
Satellite	0.6583 ± 0.0237	0.4782 ± 0.0170	**0.6609 ± 0.0271**
Thyroid	**0.5257 ± 0.0908**	0.3596 ± 0.0499	0.1822 ± 0.0069
Wine	0.2133 ± 0.0257	**0.9596 ± 0.0151**	0.7594 ± 0.1033

Table 5.20: Precision-Recall AUC

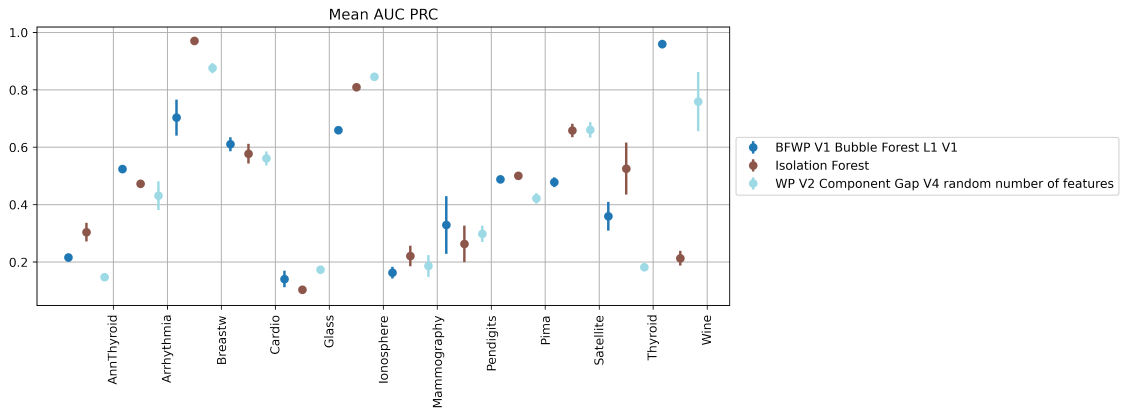


Figure 5.36: Comparison of AUC PRC Metrics with Standard Deviation

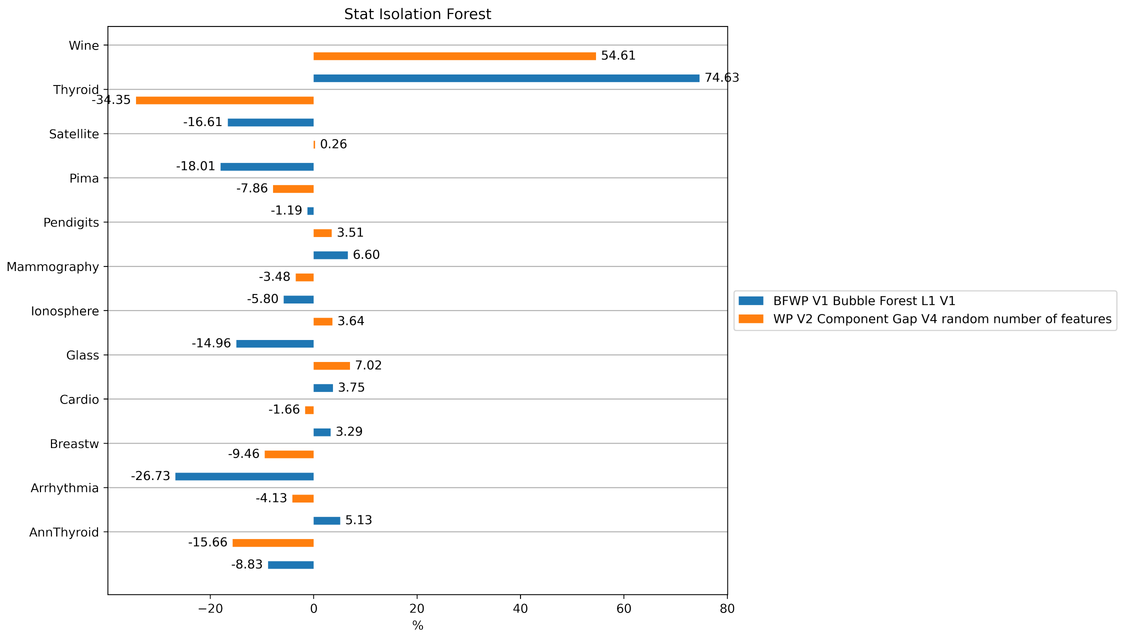


Figure 5.37: Graphical view of performance against the standard method

5.5 Other Results

In this section we show two methods that, while not achieving significant gains, perform on average well, which prompts careful consideration of the proposed new split function (Weighted path).

Table 5.21 shows the AUC PRC values, from which the chart 5.39 is constructed and as we can see the WP V1 Standard IF method where only the anomaly scoring function is replaced, in some cases, performance is improved.

Table 5.21 shows the AUC PRC values, from which the chart 5.39 is constructed, and, as we can see, the WP V1 Standard IF method where only the anomaly scoring function is replaced, in some cases performance is improved.

It is important to note that with the introduction of this new anomaly scoring function, the performance of the Prob Split method improves.

Dataset names	Isolation Forest	WP V1	
	Standard	Prob Split V2	Standard IF
AnnThyroid	0.3042 ± 0.0323	0.3008 ± 0.0282	**0.3107 ± 0.0184**
Arrhythmia	**0.4729 ± 0.0140**	0.4681 ± 0.0304	0.4623 ± 0.0241
Breastw	0.9707 ± 0.0043	**0.9759 ± 0.0034**	0.9753 ± 0.0037
Cardio	0.5776 ± 0.0343	**0.5776 ± 0.0256**	0.5569 ± 0.0299
Glass	**0.1033 ± 0.0106**	0.0903 ± 0.0043	0.0897 ± 0.0050
Ionosphere	**0.8095 ± 0.0072**	0.8028 ± 0.0059	0.8080 ± 0.0054
Mammography	0.2211 ± 0.0357	**0.2375 ± 0.0515**	0.2114 ± 0.0425
Pendigits	0.2631 ± 0.0639	**0.2936 ± 0.0433**	0.2708 ± 0.0545
Pima	0.5005 ± 0.0089	**0.5114 ± 0.0195**	0.5067 ± 0.0122
Satellite	0.6583 ± 0.0237	**0.6628 ± 0.0166**	0.6754 ± 0.0189
Thyroid	**0.5257 ± 0.0908**	0.5048 ± 0.0679	0.5185 ± 0.1037
Wine	0.2133 ± 0.0257	**0.2179 ± 0.0245**	0.2109 ± 0.0339

Table 5.21: Precision-Recall AUC

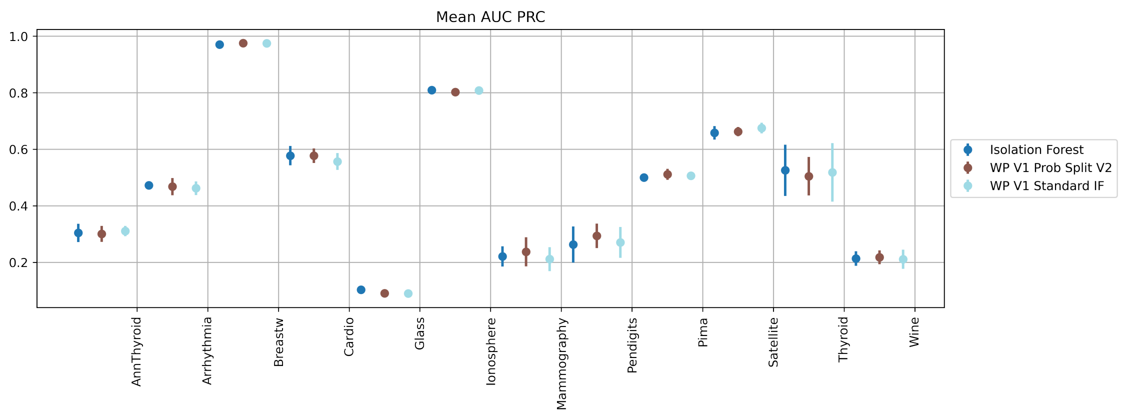


Figure 5.38: Comparison of AUC PRC Metrics with Standard Deviation

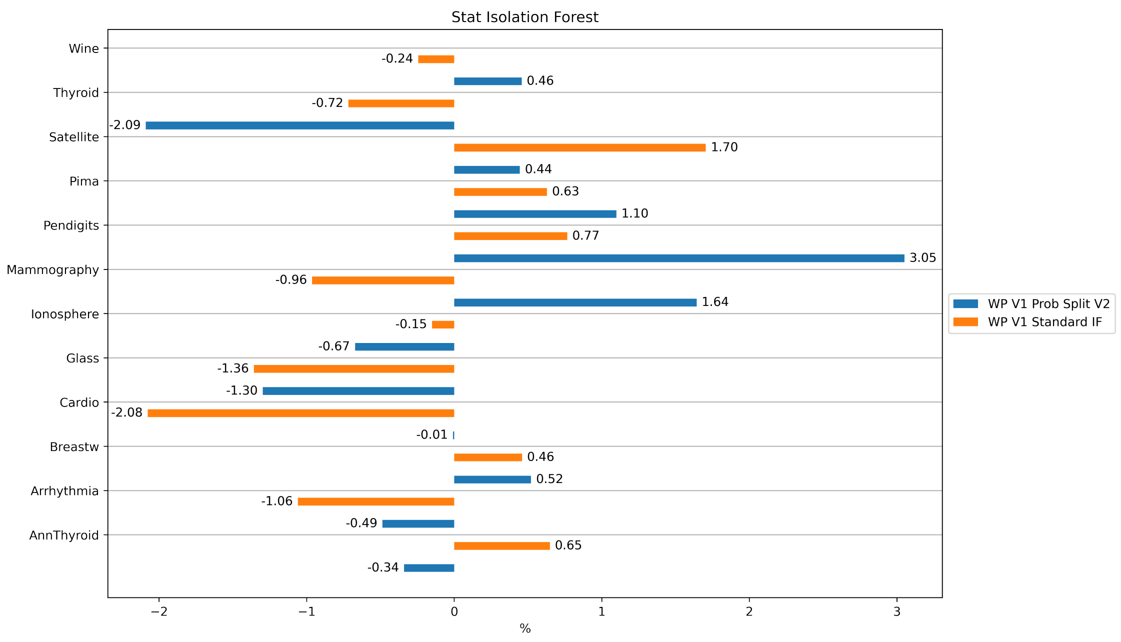


Figure 5.39: Graphical view of performance against the standard method

Chapter 6

Conclusions

In this work we focused on improving anomaly detection, in particular, we tried to improve the Isolation Forest algorithm.

The first part of the thesis consists of exposing the problem, introducing anomaly detection, the various methods used to address this problem and the metrics most commonly used to compare the proposed methods, choosing the AUC PRC metric. We have seen how the Isolation Forest algorithm defines a new concept to explicitly isolate anomalies instead of creating a profile of normal instances like the other model-based methods. This algorithm is very advantageous in terms of computational cost compared to other classical methods based on distance or density. Indeed, the research community is investing time and resources on this algorithm in the hope of further improving its performance in terms of detection.

Subsequently, we concentrated on modifying the algorithm in its two main aspects: First in the training phase, different methods were proposed to select the feature and split value, and then in the Evaluation phase, where two new anomaly scores were defined that consider additional information in addition to the simple depth.

In the last part of the thesis, we tested all proposed variants both with specially created Artificial datasets and with Real datasets in order to compare the results obtained with the standard method. In experiments with Artificial datasets all the proposed methods show remarkable performance. In the experiments with real datasets the best performing method is definitely Prob Split.

In conclusion, the aim of this work was to try to overcome some limitations of the standard method, such as central cuts. The hope is that this work will be useful to the research community.

Bibliography

Altman, D. G. and Bland, J. M. (1994). Diagnostic tests. 1: Sensitivity and specificity. *BMJ: British Medical Journal*, 308(6943):1552.

Barbariol, T., Chiara, F. D., Marcato, D., and Susto, G. A. (2022). A review of tree-based approaches for anomaly detection. *Control Charts and Machine Learning for Anomaly Detection in Manufacturing*, pages 149–185.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. (2000a). Lof: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, page 93–104, New York, NY, USA. Association for Computing Machinery.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. (2000b). Lof: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, page 93–104, New York, NY, USA. Association for Computing Machinery.

Chalapathy, R. and Chawla, S. (2019a). Deep learning for anomaly detection: A survey. *CoRR*, abs/1901.03407.

Chalapathy, R. and Chawla, S. (2019b). Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3).

Chaudhary, A., Szalay, A. S., and Moore, A. W. (2002). Very fast outlier detection in large multidimensional data sets. In *DMKD*.

Chen, Z. and Xia, S. (2009). K-means clustering algorithm with improved initial center. In *2009 Second International Workshop on Knowledge Discovery and Data Mining*, pages 790–792.

Choudhury, J., Ky, P., Ren, Y., and Shi, C. (2021). Hypersphere for branching node for the family of isolation forest algorithms. In *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 418–423.

Cortes, C. (1995). Wsupport-vector network. *Machine learning*, 20:1–25.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27.

Cuccu, G., Danafar, S., Cudré-Mauroux, P., Gassner, M., Bernero, S., and Kryszczuk, K. (2017). A data-driven approach to predict nox-emissions of gas turbines. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1283–1288.

De Stefano, C., Sansone, C., and Vento, M. (2000). To reject or not to reject: that is the question-an answer in case of neural classifiers. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(1):84–94.

Dixon, P. M. (2002). Nearest neighbor methods. *Encyclopedia of environmetrics*, 3:1370–1383.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Emmott, A., Das, S., Dietterich, T., Fern, A., and Wong, W.-K. (2015). A meta-analysis of the anomaly detection problem. *arXiv preprint arXiv:1503.01158*.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231.

Fawcett, T. (2006). An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874.

Flach, P. and Kull, M. (2015). Precision-recall-gain curves: Pr analysis done right. *Advances in neural information processing systems*, 28.

Görnitz, N., Kloft, M., Rieck, K., and Brefeld, U. (2014). Toward supervised anomaly detection. *CoRR*, abs/1401.6424.

Guha, S. (2000). Rastogi, rajeev. shim, kyuseok. *CURE: An Efficient Clustering Algorithm for Large Databases*.

Hanley, J. A. and McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36.

Hawkins, D. M. (1980). *Identification of outliers*, volume 11. Springer.

Hawkins, S., He, H., Williams, G., and Baxter, R. (2002). Outlier detection using replicator neural networks. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 170–180. Springer.

He, H. and Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284.

Hinton, G. E. (2009). Deep belief networks. *Scholarpedia*, 4(5):5947.

Hofmann, T., Schölkopf, B., and Smola, A. J. (2006). A review of kernel methods in machine learning. *Mac-Planck-Institute Technical Report*, 156.

John, H. and Naaz, S. (2019). Credit card fraud detection using local outlier factor and isolation forest. *International Journal of Computer Sciences and Engineering*, 7(4):1060–1064.

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. (2012). Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(1):1–39.

Liu, W., He, J., Han, S., Cai, F., Yang, Z., and Zhu, N. (2019). A method for the detection of fake reviews based on temporal features of reviews and comments. *IEEE Engineering Management Review*, 47(4):67–79.

Liu, X., Li, M., Sun, Y., Deng, X., et al. (2010). Support vector data description for weed/corn image recognition. *Journal of Food, Agriculture and Environment*, 8(1):214–219.

Lochner, M. and Bassett, B. (2021). Astronomaly: Personalised active anomaly detection in astronomical data. *Astronomy and Computing*, 36:100481.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Mahalanobis, P. C. (1936). On the generalized distance in statistics. National Institute of Science of India.

Mahdavinejad, M. S., Rezvan, M., Barekatain, M., Adibi, P., Barnaghi, P., and Sheth, A. P. (2018). Machine learning for internet of things data analysis: a survey. *Digital Communications and Networks*, 4(3):161–175.

Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., and Shroff, G. (2016). Lstm-based encoder-decoder for multi-sensor anomaly detection. *CoRR*, abs/1607.00148.

Martí, L., Sanchez-Pi, N., Molina, J. M., and Garcia, A. C. B. (2015). Anomaly detection based on sensor data in petroleum industry applications. *Sensors*, 15(2):2774–2797.

Munaga, H. and Jarugumalli, V. (2011). Performance evaluation: Ball-treeand kd-tree in the context of mst. In *International Joint Conference on Advances in Signal Processing and Information Technology*, pages 225–228. Springer.

Muruti, G., Rahim, F. A., and bin Ibrahim, Z.-A. (2018). A survey on anomalies detection techniques and measurement methods. In *2018 IEEE Conference on Application, Information and Network Security (AINS)*, pages 81–86.

Pecht, M. G. and Kang, M. (2019). Machine learning: Anomaly detection.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Powers, D. M. (2020). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*.

Šabić, E., Keeley, D., Henderson, B., and Nannemann, S. (2021). Healthcare and anomaly detection: using machine learning to predict anomalies in heart rate data. *AI & SOCIETY*, 36(1):149–158.

Saito, T. and Rehmsmeier, M. (2015). The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432.

Schölkopf, B., Williamson, R. C., Smola, A., Shawe-Taylor, J., and Platt, J. (1999). Support vector method for novelty detection. *Advances in neural information processing systems*, 12.

Sokolova, M., Japkowicz, N., and Szpakowicz, S. (2006). Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *Australasian joint conference on artificial intelligence*, pages 1015–1021. Springer.

Steinley, D. (2003). Local optima in k-means clustering: what you don't know may hurt you. *Psychological methods*, 8(3):294.

Tang, J., Chen, Z., Fu, A. W.-C., and Cheung, D. W. (2002). Enhancing effectiveness of outlier detections for low density patterns. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 535–548. Springer.

Tharwat, A. (2020). Classification assessment methods. *Applied Computing and Informatics*.

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

Williams, G., Baxter, R., He, H., Hawkins, S., and Gu, L. (2002). A comparative study of rnn for outlier detection in data mining. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 709–712. IEEE.

Yu, D., Sheikholeslami, G., and Zhang, A. (2002). Findout: Finding outliers in very large datasets. *Knowledge and information Systems*, 4(4):387–412.

Zhao, Y., Nasrullah, Z., and Li, Z. (2019). Pyod: A python toolbox for scalable outlier detection. *arXiv preprint arXiv:1901.01588*.