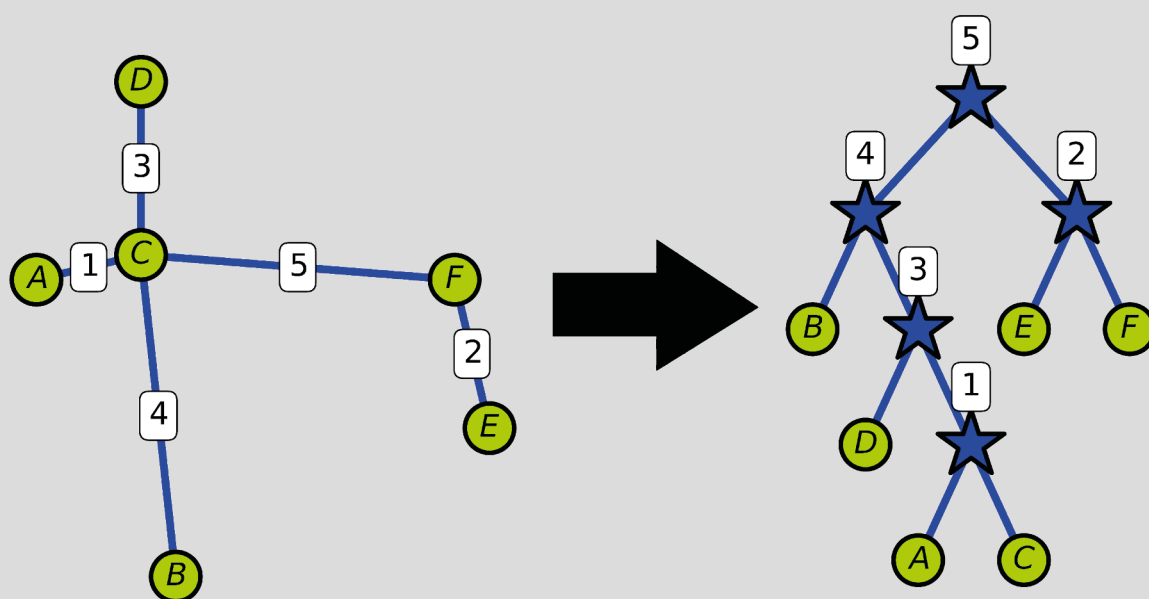


Minimal Spanning Tree Algorithm Applied to the Implementation of an Efficient Isolation Forest for Anomaly Detection

Łukasz Gałka



Minimal Spanning Tree Algorithm Applied to the Implementation of an Efficient Isolation Forest for Anomaly Detection

Łukasz Gałka

**Scientific Council
of the Lublin University of Technology Publishing House**

Head of the Scientific Council:

Agnieszka Rzepka

Director of the Center of Scientific and Technical Information:

Katarzyna Weinper

Representatives of scientific disciplines of Lublin University of Technology:

Marzenna Dudzińska
Małgorzata Franus
Arkadiusz Gola
Beata Kowalska
Grzegorz Komarzyniec
Jarosław Latański
Tomasz Lipecki
Zbigniew Łagodowski
Lucjan Pawłowski
Natalia Przesmycka
Halina Rarot
Magdalena Rzemieniak
Arkadiusz Syta

Lublin University of Technology Publishing House:

Magdalena Chołojczyk
Karolina Famulska-Ciesielska
Katarzyna Pełka-Smętek

Minimal Spanning Tree Algorithm Applied to the Implementation of an Efficient Isolation Forest for Anomaly Detection

Łukasz Gałka



LUBLIN UNIVERSITY
OF TECHNOLOGY
PUBLISHING HOUSE

Lublin 2026

Reviewer:

prof. **Michał Bereta**, DSc (Cracow University of Technology)

Author:

Łukasz Gałka, PhD, ORCID: 0000-0001-7632-5205

Lublin University of Technology – ROR: <https://ror.org/024zjzd49>

Lead editor: **Karolina Famulska-Ciesielska**

Cover design: **Łukasz Maj**

Cover illustration: **Łukasz Gałka**

The illustrations included in the book are the author's own work.

Published with the approval of **Rector of Lublin University of Technology**

ISBN: 978-83-7947-664-0 (print version)

ISBN: 978-83-7947-665-7 (electronic version)

DOI: 10.35784/9788379476657

Publisher: Wydawnictwo Politechniki Lubelskiej
www.wpl.pollub.pl
ul. Nadbystrzycka 36C, 20–618 Lublin
tel. (81) 538–46–59



**LUBLIN UNIVERSITY
OF TECHNOLOGY
PUBLISHING HOUSE**

The book is provided under a Creative Commons Attribution-ShareAlike License 4.0 International (CC BY-SA 4.0)

Print run: 50 copies

Contents

Abstract	8
1 Introduction	10
1.1 The problem of anomaly detection in the literature	12
1.2 Research objectives	15
1.3 Motivation	16
1.4 Overview of the structure of the study	18
2 Basic algorithms	20
2.1 Attribute value standardization	20
2.2 Minimal spanning tree	22
2.3 Minimal spanning tree clustering	24
2.3.1 Full minimal spanning tree clustering	24
2.3.2 Optimized minimal spanning tree clustering in a one-dimensional data space	26
2.4 k -Means-based clustering algorithms	27
2.4.1 k -Means clustering	27
2.4.2 FCM clustering	29
2.5 Algorithms for determining the appropriate number of clusters	32
2.5.1 Elbow rule	33
2.5.2 Fuzzy partition coefficient analysis	33
2.6 Takagi–Sugeno fuzzy inference model	36
2.7 Performance measures of classifiers	38
3 Isolation Forest and its extensions	39
3.1 Isolation Forest	39
3.2 Review of Isolation Forest modifications	43
4 Enhancements of the Isolation Forest algorithm	48
4.1 Dimensionality reduction of data	48

4.1.1	Clustering of elements using the k -Means method	50
4.1.2	Clustering of elements using the FCM method	52
4.2	Selection of split values based on clustering using a minimal spanning tree algorithm	55
4.3	Isolation tree construction based on the minimal spanning tree algorithm	63
4.3.1	Minimal Spanning Tree-Based Isolation Forest	63
4.3.2	Minimal Spanning Tree-Based Isolation Forest with Evaluation Function Built on the Fuzzy Rules in Takagi-Sugeno Model	69
5	Anomaly detection in real-world datasets	76
5.1	Characteristics of the datasets	76
5.2	Threshold-independent evaluation metrics	78
5.2.1	Results	78
5.2.2	Analysis of the results	78
5.2.3	Discussion of the results	89
5.3	Threshold-dependent evaluation measures	91
5.3.1	Determination of the detection threshold value	91
5.3.2	Results	95
5.3.3	Analysis of the results	95
5.3.4	Discussion of the results	126
5.4	Response time	129
5.4.1	Response time for the training phase	130
5.4.2	Response time for the evaluation phase	133
5.5	Hyperparameter Analysis	136
5.5.1	Results	136
5.5.2	Discussion of the results	140
5.6	Separability of the score function results	141
5.6.1	Results	141
5.6.2	Discussion of the results	141
6	Anomaly detection on artificially generated datasets	153
6.1	Dataset characteristics	153
6.2	Heatmaps of score function values	155
6.2.1	Results	155
6.2.2	Discussion of the results	155
6.3	Graphical representation of the anomaly detection process	165
6.3.1	Results	165
6.3.2	Discussion of the results	170

7	Summary	172
7.1	Outcomes of the work	172
7.2	Conclusions	173
7.3	Directions for future research	174
	Bibliography	177
	List of Figures	194
	List of Tables	197
	List of Algorithms	199

Abstract

This study introduces innovative anomaly detection algorithms based on the Isolation Forest (IF) method. It begins by presenting an overview of the anomaly detection issue, encompassing a review of general methods used in this field. Subsequently, the IF method is discussed in detail, along with key concepts necessary for the implementation of the novel techniques. Studies regarding modifications of the IF method are analyzed. Within this review, particular emphasis is placed on those publications focused on the improvements and extensions of the basic IF algorithm.

The study includes the introduction of five new anomaly detection algorithms. Among them are two attribute reduction methods used in the data preprocessing, based on clustering techniques such as k -Means and Fuzzy C -Means. Furthermore, a method is developed to improve the selection of the attribute value at isolation nodes, using an optimized clustering algorithm based on the minimal spanning tree. Within the next two methods, innovative solutions are introduced in the construction of isolation trees, involving the isolation of elements through their merging using the minimal spanning tree algorithm. The newly introduced isolation trees are characterized by two components of anomaly assessment: One associated with the depth of the element in the isolation tree and the other with the distance of the element from the nearest leaf node. The first of the mentioned methods utilizes the assessment function by summing the introduced components of anomaly assessment. The second technique introduces the integration of normalized components of anomaly assessment using the fuzzy rules block in the Takagi-Sugeno inference model.

A comprehensive series of experiments on newly proposed approaches is conducted to evaluate the algorithms and compare them with existing competitive techniques. The studies involved 26 real-world datasets. Measures of classification quality independent of the detection threshold selection are measured, such as the area under the receiver operating characteristic curve and the area under the precision-recall curve. Additionally, optimal detection threshold values for the methods are determined, and values of measures dependent on this threshold are calculated, namely accuracy, precision, recall, specificity, false alarm ratio, and F1 measure. The results of these studies unequivocally confirm that the newly introduced solutions are characterized by high effectiveness. Moreover, other characteristics are also analyzed,

including the response times of the algorithms in the training and evaluation phases, and a thorough analysis of hyperparameters is conducted. Adjusting the hyperparameters of the methods allowed for identifying possibilities for their modification to optimize the performance of the algorithms with specific tasks in mind. To demonstrate the effectiveness of the new approaches in separating anomalous samples from normal ones, a graphical representation of the separability is presented, using normalized values of the assessment function of individual algorithms.

To visualize the characteristics of the newly developed algorithms, four artificially generated, two-dimensional anomaly detection datasets are prepared. Heatmaps reflecting the assessment function values are developed, both for individual isolation trees and for complex forests consisting of one hundred trees. In addition, a graphical representation of the anomaly detection process is presented, using the optimally determined detection threshold. The presented results again confirm the very good detection properties of the newly introduced methods.

In the final part of the study, a synthesis of the results of experiments and analyses is conducted, key conclusions are formulated, and prospects for future research work are outlined. The directions for further development of the techniques proposed are inspired both by the observation of the effects of the experiments and by a deep understanding of the operation mechanisms applied in the discussed algorithms.

Keywords: anomaly detection, Isolation Forest, minimal spanning tree, dimensionality reduction, clustering, Takagi–Sugeno fuzzy inference, outlier detection

Chapter 1

Introduction

Technological progress, especially in the areas of digitization and the implementation of computer systems, significantly contributes to supporting decision-making through the use of artificial intelligence algorithms. For both humans and computer systems, artificial intelligence algorithms play a key role. In recent decades, a dynamic increase in the applications and use of computer systems to support decision-making has been observed. Accordingly, the topic of machine learning is gaining increasing interest among both practitioners and researchers. Two main types of machine learning are distinguished: Unsupervised learning and supervised learning. In the context of unsupervised learning, the algorithm is capable of independently developing solutions to problems without the need for data labels. This is particularly important in systems where the cost or time required to label data is too high, or labeling is impossible. On the other hand, supervised learning relies on training the algorithm using previously labeled data. After the training process, algorithms of both types are capable of identifying unknown samples, i.e., predicting labels. Machine learning algorithms find applications in numerous fields, such as decision support, data classification, pattern recognition, and forecasting of future events.

One of the challenges faced in the development of machine learning is anomaly detection. It should be noted that this term also includes the detection of outliers. Anomaly detection is a more general concept and encompasses the detection of outlying values within its scope. Its primary goal is to classify elements into one of two categories: Normal and anomalous. A characteristic feature is that, during the training phase, the number of anomalous elements is typically much smaller than the number of normal elements. Anomaly detection methods can be either supervised or unsupervised. The detection process begins with a training stage, after which the algorithm is tasked with predicting the class membership of a given test sample. Typically, anomalies differ significantly from normal data — they may deviate from the norm or exhibit a distinctive pattern.

The significance of anomaly detection is manifested through its wide range of applications across various sectors of the economy. Anomaly detection methods play a key role in many areas, significantly contributing to the security and operational efficiency of systems. Particularly important applications of these methods include:

- Ensuring cybersecurity, including the analysis and prevention of hacking attacks on networks and IT systems.
- Data cleaning and exploration, which are essential for maintaining the quality and reliability of information.
- Detection of theft and fraud, which is crucial for protecting the financial sector and other industries vulnerable to such risks.
- Prediction of unexpected behavior in financial markets, enabling early response and loss minimization.
- Monitoring of production systems and detection of defective items on production lines, necessary for maintaining high product quality.
- Prediction of equipment damage and failures, allowing the prevention of costly downtimes and ensuring continuity of manufacturing processes.
- Detection of cancerous changes and disease diagnosis, which is of great importance in the medical sector.
- Detection of computer viruses and malicious software, essential for the protection of data and IT systems.
- Forecasting unexpected weather phenomena, supporting effective planning and risk mitigation for natural disasters.
- Detection and analysis of so-called weak signals, for extracting information from highly noisy signals, such as satellite data.
- Signaling of abuse and extortion, crucial for the rapid identification and response to abnormal or harmful operations.

These and other applications confirm the universality and fundamental importance of anomaly detection, placing it among the essential components of modern analytical and decision-making systems.

This study focuses on the enhancement of one of the most efficient and fast techniques for anomaly detection, namely the Isolation Forest (IF) method. The presented proposals are described and published in a series of articles [52], [54], [55], [77]. The Isolation Forest method is based on the construction of isolation trees, which together form the forest. In particular, this study applies the integration of the minimal spanning tree algorithm with the core mechanism of IF, which constitutes one of the main enhancements of the method.

1.1 The problem of anomaly detection in the literature

Entering the phrase “anomaly detection” into the Google Scholar scientific search engine returns over 600,000 results related to this topic. This confirms that the subject is highly significant and widely studied by researchers. Moreover, the growing volume of data collected by computer systems effectively necessitates the introduction of automated anomaly detection into their analysis. On the one hand, it is essential to verify whether records are correct, and on the other hand, it is valuable to check whether there are records that do not match the rest of the data. It is also important to consider different types of anomalies, as each type exhibits distinct behavior and properties. At this point, the literature identifies the following types of anomalies: point anomalies (also referred to as outliers), contextual anomalies, collective anomalies, conditional anomalies, structural anomalies, sequential anomalies, and multivariate anomalies [5], [101], [104], [166], [168], [182]. The detection and recognition of each of these types of anomalies requires the use of various techniques, and often the simultaneous use of multiple approaches. This subsection presents different anomaly detection approaches found in the literature. It should be emphasized that these techniques can be supervised, unsupervised, or weakly supervised. In this context, weakly supervised methods rely on limited, noisy, or high-level label information rather than fully labeled training data. The focus here is on presenting the algorithms in the context of their operational characteristics, omitting the classification based on the requirement for labeled data.

A review of studies on anomaly detection techniques should begin with works related to statistical methods. In other words, techniques based on statistical computations aimed at identifying elements that statistically deviate from the data pattern. At this point, it is worth mentioning models such as the hidden Markov model [99], the dynamic Markov model [142], and the Gaussian mixture model [33]. Also used are families of probability distributions, such as the Dirichlet distribution and their multivariate generalizations, known as the Dirichlet process [122]. Among statistical methods, it is also relevant to refer to the application of Bayesian inference [196]. Furthermore, entropy-based analysis is introduced for detecting anomalous regions in the data space [19]. Increasing popularity is gained by methods that employ simple statistics, such as the mean or variance, to identify regions and construct

so-called segment matrices [141]. The way of reasoning about data can also be transformed. Therefore, techniques of information granule analysis are employed [86]. In developing anomaly detection solutions, statistical mechanisms are often combined with the concept of fuzzy sets and fuzzy logic [97].

The second group of anomaly detection solutions worth mentioning is the group based on evolutionary algorithms. The importance of evolution is particularly evident in cases where the algorithm is closely related to the changing structure of the data and the corresponding modifications of the algorithm itself. Evolutionary models implemented in dynamic networks are identified in the literature [72]. In addition, evolutionary neural networks, such as evolutionary autoencoders, are introduced to combine unsupervised techniques with evolutionary “intelligence” [64]. These types of approaches are used both in the design phase of the solution and during the training stage of the model [154]. At this point, it is worth referring to methods of adaptive structural fitting and dynamic mapping of the data feature space [113]. Neural networks are receiving increasing attention from researchers. In this context, evolutionary neural networks are employed using efficient weight search methods [60]. Methods aimed at optimizing the assumed objective function should not be overlooked, as these solutions demonstrate high effectiveness with acceptable computational cost [48]. The literature also identifies multi-objective optimization, introducing the transfer of “knowledge” between tasks [169]. Finally, anomaly detection techniques based on genetic algorithms should be mentioned, which implement operations such as selection, crossover, mutation, and reproduction [16].

The third group of methods for identifying anomalous elements is closely related to their density or the density of features associated with those elements. The most well-known method of this type is the analysis of the local outlier factor (LOF), which indicates the anomalousness of a given element [22]. It should be noted that local density can be associated with the embedding of an element within a group of other elements, using clustering algorithms [38]. Hierarchical density levels are also considered, such as density trees, which unify densities from both global and local perspectives [25]. Applications include combinations of the classical kernel density estimator with so-called M -estimation [87]. A major challenge is the analysis of high-dimensional data, where a large number of attributes may lead to the loss of significant information, known as the curse of dimensionality. In such cases, algorithms are adapted through appropriate implementation of density-based techniques, considering reverse nearest neighbors [136]. Approaches using so-called conforming elements in the training sets are also introduced. By comparing the density ratio between test data and the training dataset, an anomaly score for each element is computed [67]. There are also methods in which data are described using support vectors, and anomaly detection is performed through the analysis of relative density levels [95]. For such solutions,

the technique of support vector representation is reconfigured by indicating the local densities of the training dataset [94].

Another segment of anomaly detection techniques is based on distance-based methods. Distance can be interpreted as a measure of similarity or dissimilarity [108]. A commonly used approach for sample separation by a hyperplane is the one-class support vector machine [46], [49]. It is worth noting methods that rely on various definitions of distance [11]. This includes approaches that consider so-called cells for large, high-dimensional data [88]. In such solutions, the feature space is partitioned into hyper-rectangular grid cells and the density of points within a cell and its neighborhood is used to efficiently identify outliers. Distance measures can include distances to cluster centroids or to nearest neighbors, calculated in a way that produces a single value describing the degree of anomaly of a given element [103]. In time series analysis, distances related to the phase difference between test points and a reference point are also applied [71]. The k -nearest neighbors algorithm and its enhancements are introduced [65]. It is also relevant to mention studies on outlier detection in which anomalies are identified as those elements that exhibit the greatest deviations from normal samples [7].

Among the most popular methods for detecting anomalous elements are approaches based on clustering analysis. Considerations should begin with techniques that group elements to obtain a partition of the sample set. Such an operation leads to the separation of elements that are distant from the cluster center [80]. Clustering methods are applied to anomaly detection in time series. Particularly relevant are algorithms that use a sliding window to extract subsequences of samples. This allows the structure of the subsequences to be reconstructed in a way that minimizes the reconstruction error [98]. Additionally, in the case of sample sequences, fuzzy clustering techniques are used to extract the underlying structure of subsequences [68]. Clustering algorithms are applied for cascade processing of elements, such as initial preprocessing and elimination of anomalous elements, followed by clustering analysis of the refined datasets [138]. Among the many clustering approaches, particular attention should be given to those based on element density, which form clusters according to the concentration of elements [27]. Moreover, anomalies can be concentrated within the groups themselves, as addressed by techniques for detecting local anomaly clusters [66]. Finally, it should be noted that clustering analysis solutions can be used to optimize the execution time of the algorithm [85].

Another group of approaches used for anomaly detection is related to broadly understood isolation. Algorithms are applied that separate samples through random partitioning of the dataset, forming so-called isolation trees [106]. It is also worth noting that incomplete isolation trees are constructed based on entropy-related heuristics [150]. Isolation may refer to the separation of events or elements within detection systems [91]. Such algorithms are applied to data streams, that is, rapidly changing data often characterized by high volume [39]. Of particular interest are anomaly detection algorithms that use a dual level of separation, incorporating information

embedded both in space and in time [153]. Finally, isolation algorithms that use distance as a measure of sample separation are also employed [9].

The increase in computing power has enabled the use of artificial neural networks for anomaly detection. For image analysis, deep convolutional neural networks are utilized [109]. However, the literature also identifies techniques for converting tabular information into images. This allows such transformed samples to be analyzed using the aforementioned convolutional networks [74]. For anomaly detection in data streams, spiking neural networks are applied, where the network outputs are not divided into predefined decision classes [114]. In addition, techniques of adversarial prediction of atypical events are introduced [188]. Among various architectures of artificial neural networks, recurrent neural networks should be mentioned [126], particularly those that retain dependencies in data sequences, including both short-term and long-term memory [51]. Furthermore, architectures based on autoencoders are implemented [187], especially their cascade combinations [100]. Graph neural networks designed for data organized in a graph structure also attract attention [50].

The final class of approaches consists of hybrid methods that combine multiple diverse algorithms for the purpose of anomaly detection. This includes, for example, the combination of various neural networks based on different architectures with the application of logistic regression [47]. Fuzzy sets and fuzzy logic are frequently used to improve the aggregation of values, such as similarity or dissimilarity measures [172]. Anomaly scores from independent classifiers are subject to appropriate aggregation, which contributes to improving detection accuracy [76], [78]. In such cases, voting-based techniques are also employed to fuse the results of detection from different approaches [42], [43]. The literature also identifies the synthesis of distance computation operations with rough set analysis [191]. Attention should also be given to solutions that combine clustering techniques with methods inspired by evolutionary algorithms, such as particle swarm optimization or genetic algorithms [190].

The review presented here includes only selected, representative methods. This results from the broad scope of the subject and the diversity of existing solutions in this field. The description provided does not fully exhaust the topic of anomaly detection algorithms. However, it places the approaches introduced in this study within a broader research context.

1.2 Research objectives

The objective of this study is to develop and thoroughly analyze innovative anomaly detection methods based on the concept of the Isolation Forest. The scope of the work includes:

- Applying data preprocessing prior to invoking the core Isolation Forest method, namely developing attribute reduction approaches based on clustering techniques.

- Improving the process of selecting the split point in isolation nodes during the construction of the Isolation Forest by replacing the random split value with a deterministically determined value, using a clustering algorithm based on the minimal spanning tree.
- Introducing a new technique for forming isolation trees, aiming to replace the original node creation process based on space partitioning with node formation using sample merging via the minimal spanning tree algorithm. Additionally, the objective is to develop a new evaluation function based on the properties of the proposed isolation trees using a fuzzy inference model.
- Analyzing and evaluating the proposed solutions with respect to various characteristics, examining the effectiveness of the algorithms using both threshold-independent and threshold-dependent performance metrics. The analysis includes real-world anomaly detection datasets as well as synthetic collections generated specifically for this study.
- Measuring the execution times of individual phases of the approaches, namely the training phase and the evaluation phase.
- Presenting a graphical representation of the detection process of the proposed solutions.

1.3 Motivation

The decision to address anomaly detection based on the Isolation Forest stems from its effectiveness and speed, which make it a method widely used in practice. A thorough analysis of the Isolation Forest mechanism, as well as considerations related to data preparation and the interpretation of the evaluation function results, enable the identification of areas requiring improvement. The aim of the research is not only to increase the method's efficiency but also to expand its potential applications and adapt it to the specific requirements of various domains in which anomaly detection plays a key role.

The initial focus is placed on the Isolation Forest method itself and on the preparation of data for its training and execution. On the one hand, dimensionality reduction can help mitigate the curse of dimensionality. On the other hand, appropriate preparation of the samples passed to the Isolation Forest method can significantly improve detection quality. It is important to emphasize that anomalies typically exhibit different characteristics compared to normal elements and are much rarer than normal samples. Clustering analysis can reveal regions where normal samples are concentrated in clusters, while anomalous samples lie on the outskirts of those clusters. The Isolation Forest method relies on the separation of samples by splitting based on a single attribute. Therefore, data preprocessing should prepare the samples in such

a way that attribute values for normal samples are concentrated, while those for anomalous samples are more dispersed. This corresponds to low attribute value density in anomalous regions. Consequently, elements located at the cluster boundaries, which are in the minority, can be transformed in such a way that the data dimensions reflect this specificity. This study analyzes two clustering algorithms. The first, the k -Means algorithm, assigns samples to clusters in a deterministic manner, where the distance to the cluster center indicates the sample's position. The second algorithm, based on the Fuzzy C-Means (FCM) technique, allows a sample to belong to multiple clusters simultaneously. As a result, sorting the degrees of cluster membership in ascending order enables the identification of samples with atypical behavior. Moreover, this method of analysis allows precise identification of samples located far from cluster centers and characterized by low density, which is particularly important in the context of anomaly detection. Conversely, samples strongly associated with a cluster exhibit dense attribute values. Both algorithms require the number of clusters to be specified prior to execution. Therefore, this study also presents methods for determining the optimal number of clusters for each technique.

The motivation for developing the second proposed improvement stems from the possibility of modifying the Isolation Forest method itself. The IF method is based on isolating elements through random splits of the sample space, where each node in an isolation tree divides the samples into two subsets. However, a question arises as to whether random selection of the split point is optimal. It appears justified to consider selecting the split point in a location that enables the fastest possible isolation of anomalous samples. Random selection of the dimension along which the samples are split may not be the most effective. This leads to the concept of performing the sample split based on clustering using the minimal spanning tree (MST) algorithm. The main advantage of this approach lies in the high sensitivity of the MST construction algorithm to outliers, which allows them to be identified as separate clusters. In such cases, it becomes possible to isolate a given cluster by selecting a split point that effectively separates it from the remaining clusters. Therefore, when analyzing clusters using MST, the goal is to identify at least two distinct clusters and to determine the split point in such a way that the extracted clusters are effectively separated.

The final improvement considered involves rethinking the very concept of creating isolation nodes through sample space partitioning. Instead of the original idea of dividing the dataset into disjoint subsets, it becomes possible to separate samples through a merging process. Such separation can be performed using an algorithm that constructs a minimal spanning tree. As a result, this opens a new perspective on the formation of isolation trees. During this process of node creation, a minimal spanning tree is built and subsequently transformed into an isolation tree. As a result, two trees are obtained: One minimal spanning tree and one isolation tree. Consequently, it becomes evident that the evaluation function in such a tree structure should incorporate two components. To derive a single anomaly score, an inference model

can be employed to aggregate these components in a precise manner. However, it is crucial to maintain the speed of the method. Therefore, this study applies a Takagi–Sugeno fuzzy inference model, which enables an effective combination of computational efficiency and accurate anomaly assessment.

1.4 Overview of the structure of the study

This study is divided into seven chapters. The introductory chapter provides an overview of the topic of anomaly detection, which constitutes one of the key areas of machine learning. Within this chapter, a review of relevant publications and studies related to anomaly detection is presented, outlining the current state of knowledge in this field. The research objectives are defined, and the motivation behind undertaking the topic of this study is also discussed.

The second chapter introduces key concepts that form the foundation for the remainder of this study. The process of attribute standardization is discussed in detail, the concept of the minimal spanning tree is defined, and its application in clustering is presented, including an optimized version of the algorithm for one-dimensional space. A significant part of the chapter is devoted to clustering analysis using two popular algorithms: k -Means and Fuzzy C-Means. Methods for determining the optimal number of clusters for these techniques are also described, namely the Elbow method and the fuzzy partition coefficient analysis. In addition, the chapter includes a review of the definition of the Takagi–Sugeno fuzzy inference model and recalls the definitions of detection performance measures, enabling a full understanding of the methods used and the results obtained in this study.

The third chapter is devoted to the baseline Isolation Forest method. This chapter discusses the two main phases of the algorithm's operation: The construction phase of isolation trees and the evaluation phase of individual elements. Each stage is explained in detail, starting from the tree construction process to the methodology for determining the anomaly score of individual samples. The chapter concludes with a literature review concerning previous enhancements and modifications of the IF method. This provides a clear positioning of the newly developed solutions within the existing body of knowledge and highlights the unique contribution of this study.

The fourth chapter is dedicated to a detailed description and analysis of the proposed solutions. The first part presents methods based on dimensionality reduction applied prior to the main execution phase of the IF method. Emphasis is placed on two techniques rooted in clustering analysis: One based on the k -Means algorithm and the other using the fuzzy clustering algorithm Fuzzy C-Means. The next part of the chapter focuses on a technique that replaces the random selection of attribute values in isolation nodes with a deterministic selection derived from clustering using the minimal spanning tree. In the final section,

an innovative technique for constructing isolation trees based on the minimal spanning tree algorithm is introduced, including the implementation of an evaluation function built upon fuzzy rules in the Takagi–Sugeno model. For each proposed method, the operational mechanism is discussed along with both algorithmic phases, as described analogously in Chapter 3.

The fifth chapter presents the results and their discussion, based on a series of experiments conducted on real-world datasets intended for algorithm analysis. Specifically, the newly proposed methods are compared with existing competitive solutions. The datasets are thoroughly characterized, providing a foundation for the presentation of results. First, the results obtained using threshold-independent performance measures are discussed. This allows for the assessment of the overall effectiveness of the methods. Next, for each method, the optimal detection threshold is determined, and the results of threshold-dependent metrics are presented. Finally, the execution times of both algorithmic phases, namely the training phase and the evaluation phase, are characterized. The latter part of the chapter focuses on the potential for modifying the hyperparameters of the newly developed methods, placing them in the context of the baseline IF method. The discussion concludes with an analysis of the separability of the output values of the evaluation function, providing deeper insight into the method’s effectiveness in distinguishing normal from anomalous samples.

The sixth chapter, dedicated to the presentation of results for artificially generated datasets, provides graphical analyses. First, the individual data collections and their characteristics are described. Then, heatmap analyses are presented, illustrating the results obtained using the evaluation functions of the respective methods. These maps allow for a deeper understanding of how each solution operates and help to identify their strengths and weaknesses. Additionally, the outcomes of the detection process are illustrated graphically, enabling direct observation of algorithm behavior and visual comparison of their effectiveness.

The final, seventh chapter summarizes the conducted experiments and studies. The outcomes of the work and analyses, as well as the conclusions drawn from them, are discussed. The discussion concludes with potential directions for future research.

Chapter 2

Basic algorithms

This chapter presents the basic concepts and algorithms used in the subsequent parts of this study.

2.1 Attribute value standardization

In data analysis, attribute values often have different scales and value ranges. Such characteristics of attributes can affect their interpretation as well as the effectiveness of applied methods. In the following chapters, attribute standardization is used, defined by the formula:

$$y_i = \frac{x_i - \mu}{\sigma} \quad (2.1)$$

where x_i denotes the value of the attribute for the i -th sample, μ is the mean of the attribute in the dataset, and σ is the standard deviation of the attribute values in the dataset.

Standardization for a single attribute proceeds in two steps. First, the values of μ and σ are calculated. Then, the values of the considered attribute are standardized. Due to the frequent references in the later parts of this study, we explicitly present two standardization algorithms: Algorithm 2.1 and Algorithm 2.2. Algorithm 2.1 performs standardization of attribute values. Algorithm 2.2 handles standardization of a new sample based on existing records. Standardizing new elements does not change the previously determined standardization parameters.

Algorithm 2.1 Standardization of attribute values.

Require: X – input data

Ensure: Y – data with standardized attributes, AVG – list of attribute means, STD – list of attribute standard deviations

```
1: let  $A$  be the list of attributes in  $X$ 
2: let  $AVG$  be an empty list of attribute means
3: let  $STD$  be an empty list of attribute standard deviations
4:  $Y \leftarrow X$ 
5: for  $a \leftarrow 1$  to  $|A|$  do
6:    $\mu \leftarrow 0$ 
7:   for each  $x \in X$  do
8:      $\mu \leftarrow \mu + x[a]$ 
9:   end for
10:   $\mu \leftarrow \mu/|X|$ 
11:   $\sigma \leftarrow 0$ 
12:  for each  $x \in X$  do
13:     $\sigma \leftarrow \sigma + (x[a] - \mu) \cdot (x[a] - \mu)$ 
14:  end for
15:   $\sigma \leftarrow \sqrt{\sigma/|X|}$  ▷  $\sqrt{\cdot}$  – square root
16:  append  $\mu$  to  $AVG$ 
17:  append  $\sigma$  to  $STD$ 
18:  for each  $y \in Y$  do
19:     $y[a] \leftarrow (y[a] - \mu)/\sigma$  ▷ Formula (2.1)
20:  end for
21: end for
22: return  $Y, AVG, STD$ 
```

Algorithm 2.2 Standardization of attribute values for a new record

Require: x – input element, AVG – list of attribute means, STD – list of attribute standard deviations

Ensure: y – element with standardized attributes

```
1: let  $A$  be the list of attributes in  $x$ 
2:  $y \leftarrow x$ 
3: for  $i \leftarrow 1$  to  $|A|$  do
4:    $y[i] \leftarrow (y[i] - AVG[i])/STD[i]$  ▷ Formula (2.1)
5: end for
6: return  $y$ 
```

2.2 Minimal spanning tree

This subsection discusses the algorithm for finding a minimal spanning tree (MST) [37], [57]. To this end, we first introduce the concepts of an undirected graph and a minimal spanning tree.

Definition 2.1 [37] *An undirected graph is defined as a structure described by a pair of sets (V, E) . The elements of the set V are called the vertices (nodes) of the graph, while the elements of the set E are referred to as edges. Each edge in the set E is defined by an unordered pair of two distinct vertices. This is the so-called incident edge. Weights can be associated with the edges.*

Definition 2.2 [37] *A minimal spanning tree on the set V is a tree T that is a connected (i.e., linking all vertices in V) and acyclic (i.e., containing no cycles) undirected graph. The set of edges of this tree $E_T \subseteq E$ satisfies the condition that the sum of the edge weights W_T is minimal.*

Let us consider a set of graph vertices:

$$V = v_1, v_2, \dots, v_k \quad \text{where } v_i \in \mathbb{R}^l \text{ and } k, l \in \mathbb{N}^+ \quad (2.2)$$

and a set of undirected edges connecting the graph vertices:

$$E = e_{ij} : e(v_i, v_j) \quad \text{for } i, j \in 1, \dots, k \quad (2.3)$$

Each edge is associated with a weight:

$$w_{ij} = d(v_i, v_j) \quad \text{for } i, j \in 1, \dots, k \quad (2.4)$$

where $d(\cdot)$ is a dissimilarity measure. Then, the condition imposed on the edges of the minimal spanning tree for the pair (V, E) can be written as:

$$W_T = \min_T \sum_{e_{ij} \in T} w_{ij} \quad (2.5)$$

In this study, the Euclidean distance is used as the similarity/dissimilarity measure. An example of a graph and its corresponding minimal spanning tree is shown in Figure 2.1.

To determine the minimal spanning tree T , the most commonly used algorithms are those of Kruskal and Prim [12], [92], [133]. Due to the intended use of the algorithm for constructing isolation trees, this study presents Kruskal's method. In this method, the graph edges are first sorted by weight in non-decreasing order, and a forest T is initialized, where each tree contains a single vertex from the original graph. Following the initialization phase, the procedure is repeated iteratively until the forest T becomes a minimal spanning tree, i.e., until the forest is reduced to a single tree. Each iteration consists of two steps. First, the edge

with the lowest weight is selected from the sorted list of edges. If the selected edge connects two different trees from the forest T , these trees are merged using the selected edge, i.e., the edge is added to T . The computational complexity of constructing a minimal spanning tree using Kruskal's method is $O(|E| \log |E|)$, which, in terms of the number of vertices, can be expressed as $O(|V|^2 \log |V|)$. The implementation scheme is presented in Algorithm 2.4. It is worth noting that in the considered implementation, only the distances extracted from the edges are stored in memory, not the edges themselves. This approach is motivated by the intended later use of the algorithm and plays an important role in its optimization. Additionally, auxiliary functions are provided in Algorithms 2.5 and 2.3.

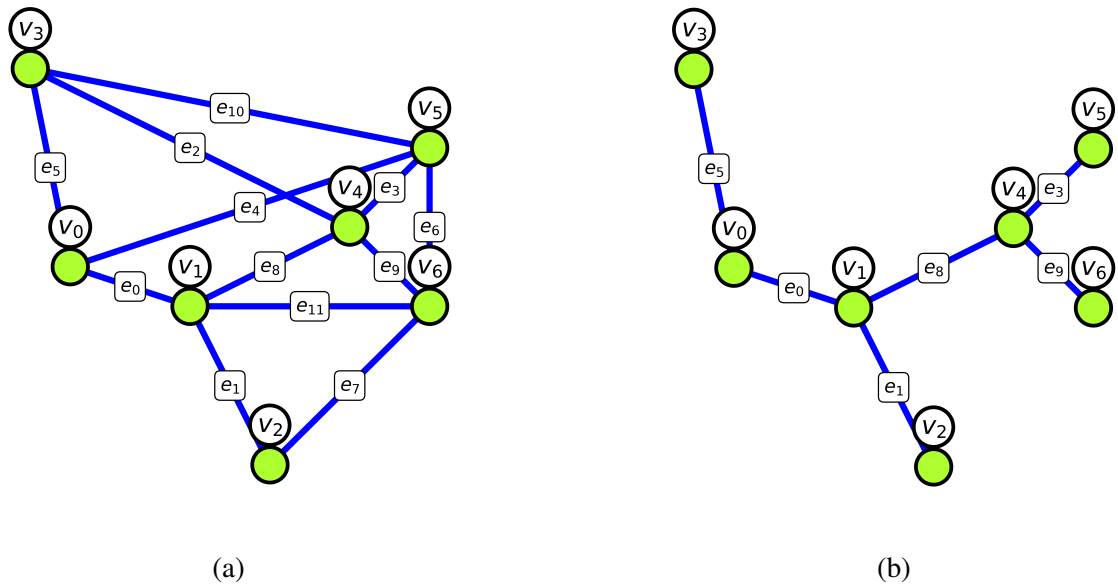


Figure 2.1: (a) Example of a graph with labeled vertices and edges. The edge weights are defined as the distances between the connected vertices. (b) Minimal spanning tree constructed based on the graph shown in (a).

Algorithm 2.3 Tree merging algorithm

Require: y – root node, z – root node, $dist$ – distance between the merged trees

- 1: create a node $node$
 - 2: $node.left \leftarrow y$
 - 3: $node.right \leftarrow z$
 - 4: $node.parent \leftarrow null$
 - 5: $node.distance \leftarrow dist$
 - 6: $y.parent \leftarrow node$
 - 7: $z.parent \leftarrow node$
-

Algorithm 2.4 Kruskal's algorithm for finding the minimal spanning tree

Require: V – list of graph vertices, E – list of graph edges**Ensure:** minimal spanning tree

```

1: sort  $E$  by weight in non-decreasing order
2: let  $T$  be a list containing individual vertices from  $V$ 
3:  $counter \leftarrow 0, index \leftarrow 1$ 
4: while  $counter < |V|$  do
5:    $edge \leftarrow E[index]$ 
6:    $index \leftarrow index + 1$ 
7:    $y \leftarrow \text{find root node of } edge.source$        $\triangleright$  Algorithm 2.5:  $i \leftarrow edge.source, T \leftarrow T$ 
8:    $z \leftarrow \text{find root node of } edge.destination$    $\triangleright$  Algorithm 2.5:  $i \leftarrow edge.destination,$ 
    $T \leftarrow T$ 
9:   if  $y \neq z$  then
10:      $counter \leftarrow counter + 1$ 
11:     merge nodes  $y$  and  $z$        $\triangleright$  Algorithm 2.3:  $y \leftarrow y, z \leftarrow z, dist \leftarrow edge.distance$ 
12:   end if
13: end while
14: return  $T$ 

```

Algorithm 2.5 Root node finding algorithm

Require: i – index of the base node, T – list of nodes**Ensure:** root node

```

1:  $node \leftarrow T[i]$ 
2: while  $node.parent \neq null$  do
3:    $node \leftarrow node.parent$ 
4: end while
5: return  $node$ 

```

2.3 Minimal spanning tree clustering

The previous subsection introduced the concept of a minimal spanning tree and the algorithms used to construct such a tree. This subsection describes clustering based on the minimal spanning tree (MST) [58], [127].

2.3.1 Full minimal spanning tree clustering

MST-based clustering involves forming clusters by removing the $c - 1$ edges with the highest weights from the minimal spanning tree [58], [127], where c is the desired number of disjoint clusters, i.e.,

$$V_i \subset V \quad \text{for } i \in 1, \dots, c, \quad c \leq |V| \quad (2.6)$$

and

$$V_i \cap V_j = \emptyset \quad \text{for } i \neq j, \quad V_1 \cup V_2 \cup \dots \cup V_c = V \quad (2.7)$$

where V_i represents the set of elements in the i -th cluster.

The operation of removing a single edge is defined as $E_T \setminus \{e_{ij} : w_{ij} = \max_{E_T} w\}$, where E_T is the set of edges forming the minimal spanning tree. An example of MST-based clustering is shown in Figure 2.2. Figure 2.2(a) illustrates the minimal spanning tree. The subsequent parts show the clustering results, with clusters indicated in different colors, for two, three, and four target groups. Particular attention should be paid to the behavior of the algorithm in the presence of outliers. In Figure 2.2(d), such values can be seen as being isolated from the others, forming a separate group. This property enables effective anomaly detection in datasets represented in graph form.

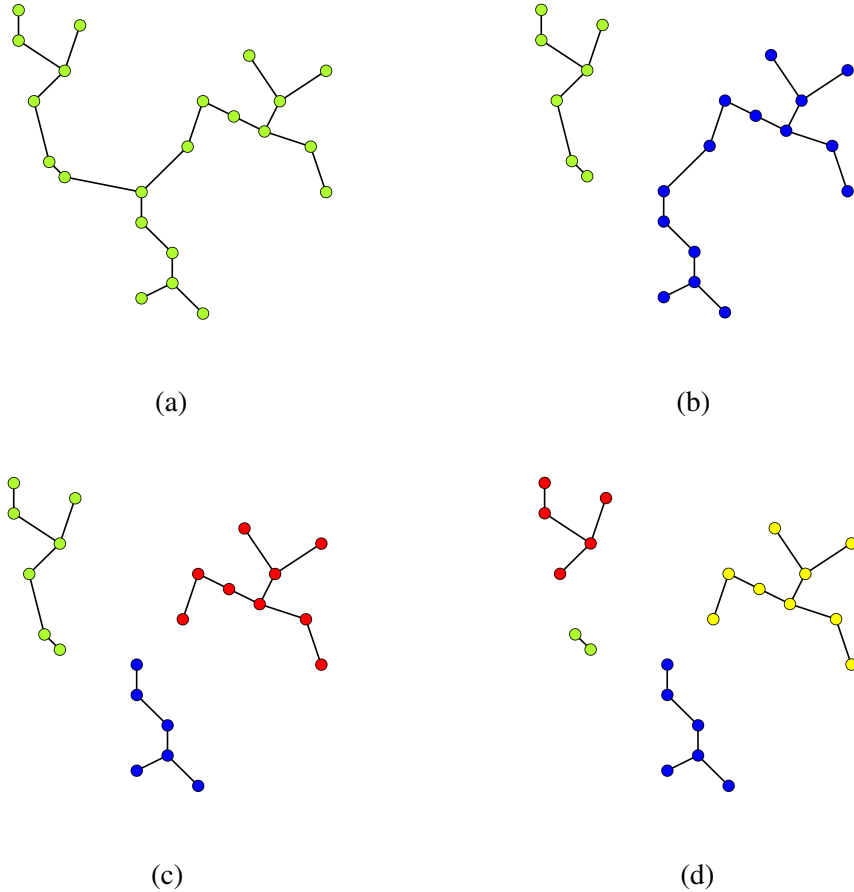


Figure 2.2: Example of clustering using a minimal spanning tree. In each subfigure, elements belonging to the same cluster are marked with the same color. (a) Minimal spanning tree forming a single cluster. (b) Two clusters formed by removing the edge with the highest weight. (c) Three clusters formed by removing two edges with the highest weights. (d) Four clusters formed by removing three edges with the highest weights.

Additionally, when Kruskal's algorithm is used to construct the MST, the clustering technique can be optimized. Specifically, it is possible to identify the final clusters already during the MST construction phase. This is achievable because edges are added to the tree in sorted order. Therefore, removing the $c - 1$ edges with the largest weights is equivalent to halting the MST construction at the point where exactly $c - 1$ edges remain to be added. The computational complexity of MST-based clustering is of the same order as that of constructing the minimal spanning tree.

2.3.2 Optimized minimal spanning tree clustering in a one-dimensional data space

Let us consider the case where the data are characterized by a single attribute. The minimal spanning tree clustering algorithm can be optimized for such a scenario [55]. To this end, we analyze a set of vertices sorted by attribute value:

$$V = \{v_1, v_2, \dots, v_k\}, \quad v_i \in \mathbb{R}, \quad v_i \leq v_{i+1}, \quad k \in \mathbb{N}_+ \quad (2.8)$$

where k is the number of vertices in the set V .

For the problem thus defined, a significant number of edges used in the full version of the algorithm can be excluded. This is due to the fact that the edges with the smallest weights connect neighboring vertices. In other words, to construct a minimal spanning tree on the set V , it is sufficient to use edges defined as follows:

$$E = \{e_{i,i+1} : e(v_i, v_{i+1})\}, \quad i \in \{1, \dots, k - 1\} \quad (2.9)$$

Each edge is associated with a weight:

$$w_{i,i+1} = d(v_i, v_{i+1}), \quad i \in \{1, \dots, k - 1\} \quad (2.10)$$

In the one-dimensional space, the weight function (2.10) can be expressed as:

$$d(v_i, v_{i+1}) = v_{i+1} - v_i, \quad i \in \{1, \dots, k - 1\} \quad (2.11)$$

For the set of vertices V , the edge set E forms a minimal spanning tree. Therefore, in the optimized version of the algorithm, the phase of constructing the minimal spanning tree is replaced by sorting the vertices with respect to the attribute and creating edges connecting adjacent vertices. The time complexity of the sorting operation is $O(|V| \log |V|)$, while the preparation of edges takes $O(|V|)$. Hence, sorting remains the dominant procedure. Ultimately, the overall complexity of creating the minimal spanning tree is $O(|V| \log |V|)$,

which reduces the complexity compared to full clustering. Importantly, the number of required edges is reduced from $\frac{|V|(|V|-1)}{2}$ (in the full clustering case) to $|V| - 1$ (in the optimized case). As in full clustering, $c - 1$ edges with the highest weights must be removed.

This solution is universal and allows the creation of up to $|V| - 1$ clusters. In this study, a division into two groups is considered, i.e., in the final stage of the algorithm, a single edge with the highest weight is removed. If multiple such edges exist, one of them is selected at random. This operation is performed in $O(|V|)$ time. The split value at which the two groups are separated is determined as the midpoint of the removed edge. The implementation scheme for determining this split value is presented in Algorithm 2.6. In general, the input data may be represented by multiple attributes. In such a case, the attribute to be used for clustering must be specified.

2.4 *k*-Means-based clustering algorithms

This subsection presents clustering algorithms based on the mean value with a predefined number of clusters. First, the basic *k*-Means algorithm [63] is described. Next, the Fuzzy *C*-Means (FCM) clustering algorithm [21] is introduced.

2.4.1 *k*-Means clustering

The *k*-Means algorithm is based on the concept of centroids, i.e., points that represent the centers of clusters. The method operates iteratively, optimizing the positions of the centroids at each step. Before the first iteration, elements are assigned to random groups. Then, the centroids are calculated based on the mean values of the elements assigned to each cluster:

$$C_i = \frac{1}{|S_i|} \sum_{x \in S_i} x, \quad i \in \{1, \dots, k\} \quad (2.12)$$

where S_i is the set of elements belonging to the i -th cluster, C_i is the centroid of that cluster, and k denotes the number of clusters.

Algorithm 2.6 Selection of the split value for two clusters by optimized clustering using the minimal spanning tree

Require: X – input data, a – considered attribute

Ensure: split value for attribute a

```

1: sort  $X$  by attribute  $a$  in ascending order
2:  $temp1 \leftarrow X[1][a]$ 
3:  $temp2 \leftarrow X[|X|][a]$ 
4: if  $temp1 = temp2$  then
5:   return  $temp1$ 
6: end if
7:  $maxDistance \leftarrow -1$ ,  $index \leftarrow 0$ ,  $sameDistanceCounter \leftarrow 0$ 
8: for  $x \leftarrow 1$  to  $|X|$  do
9:    $temp1 \leftarrow X[x][a]$ 
10:   $temp2 \leftarrow X[x+1][a]$ 
11:   $distance \leftarrow temp2 - temp1$ 
12:  if  $distance > maxDistance$  then
13:     $maxDistance \leftarrow distance$ 
14:     $index \leftarrow x$ 
15:     $sameDistanceCounter \leftarrow 1$ 
16:  else if  $distance = maxDistance$  then
17:     $sameDistanceCounter \leftarrow sameDistanceCounter + 1$ 
18:     $random \leftarrow$  retrieve a random value from the range 1 to  $sameDistanceCounter$ 
19:    if  $random = sameDistanceCounter$  then
20:       $index \leftarrow x$ 
21:    end if
22:  end if
23: end for
24: return  $\frac{X[index][a] + X[index+1][a]}{2}$ 

```

In the next step, the cluster assignments are updated based on the proximity of elements to the current centroids. The algorithm iteratively repeats centroid computation and cluster assignment updates. The optimization terminates either when no changes in cluster assignments occur or when the maximum number of allowed iterations is reached. Formally, the algorithm aims to minimize the following objective function:

$$J = \sum_{i=1}^k \sum_{x \in S_i} \|x - C_i\|^2 \quad (2.13)$$

where $\|\cdot\|$ denotes the selected distance metric (in this case, the Euclidean distance).

This means that elements within a cluster should be as similar as possible, whereas elements in different clusters should be as dissimilar as possible. The implementation scheme of the k -Means algorithm is presented in Algorithm 2.7.

Algorithm 2.7 *k*-Means

Require: X – input data, k – target number of clusters, $maxIterations$ – maximum number of algorithm steps

Ensure: C – list of cluster centers, M – list of memberships of elements to clusters

```

1: let  $M$  be an empty list of memberships of points from  $X$  to clusters
2: for  $i \leftarrow 1$  to  $|X|$  do
3:    $m \leftarrow$  randomly select a cluster number from the range 1 to  $k$ 
4:   add  $m$  to the list  $M$ 
5: end for
6:  $counter \leftarrow 0$ ,  $isChanged \leftarrow true$ 
7: while  $isChanged = true$  and  $counter < maxIterations$  do
8:   let  $C$  be an empty list containing the cluster centers
9:   for  $i \leftarrow 1$  to  $k$  do
10:     $c \leftarrow$  calculate the center of cluster  $i$  based on the mean value of the attributes of  $X$ 
    belonging to cluster  $i$  in the membership list  $M$  ▷ Formula (2.12)
11:    add  $c$  to the list  $C$ 
12:   end for
13:    $isChanged \leftarrow false$ 
14:   for  $i \leftarrow 1$  to  $|X|$  do
15:     $old \leftarrow M[i]$ 
16:     $new \leftarrow$  retrieve the number of the cluster to which the point  $X[i]$  has the smallest
    distance
17:     $M[i] \leftarrow new$ 
18:    if  $old \neq M[i]$  then  $isChanged \leftarrow true$ 
19:   end for
20:    $counter \leftarrow counter + 1$ 
21: end while

```

2.4.2 FCM clustering

The FCM algorithm (Fuzzy C-Means) is an extension of the *k*-Means method, based on fuzzy set theory. Specifically, it introduces a fuzzy membership value of an element to a cluster. In contrast to *k*-Means, where an element belongs to only one cluster, FCM allows each element to belong to multiple clusters with varying degrees of membership. Similar to *k*-Means, the algorithm operates iteratively. Each iteration updates both the centroids and the membership values. Before the procedure begins, the algorithm initializes and assigns random memberships of elements to clusters. Then, the following steps are repeated. The cluster centers are computed based on fuzzy memberships according to the formula:

$$C_j = \frac{\sum_{i=1}^{|X|} w_{ij}^m x_i}{\sum_{i=1}^{|X|} w_{ij}^m}, \quad j \in \{1, \dots, c\} \quad (2.14)$$

where c is the number of clusters, X is the set of elements, x_i is the i -th element in X , w_{ij} is the fuzzy membership of element i to cluster j , and m is a parameter defining the fuzziness level of the clusters, with $m > 1$. In practice, the value of m is often chosen from the interval $[1.5, 2.5]$ [128], [174].

Based on the updated centroids, the distances between elements and clusters are calculated, and the membership values are updated:

$$w_{ij} = \frac{1}{\sum_{k=1}^c \left(\frac{\|x_i - C_j\|^2}{\|x_i - C_k\|^2} \right)^{\frac{1}{m-1}}}, \quad j \in \{1, \dots, c\}, \quad i \in \{1, \dots, |X|\} \quad (2.15)$$

where x_i is the i -th element of set X , and C_j, C_k are the centroids of the j -th and k -th clusters, respectively. The norm $\|\cdot\|$ denotes the selected distance metric. Additionally, the following condition is satisfied:

$$\forall i \in \{1, \dots, |X|\} : \sum_{j=1}^c w_{ij} = 1 \quad (2.16)$$

It is important to emphasize that the algorithm aims to minimize the following objective function:

$$J_m = \sum_{j=1}^c \sum_{i=1}^{|X|} w_{ij}^m \|x_i - C_j\|^2 \quad (2.17)$$

The algorithm terminates when the minimum change in the objective function is reached or when the maximum number of iterations is exceeded. The implementation scheme is presented in Algorithms 2.9, 2.12, 2.11, and 2.10. Additionally, Algorithm 2.8 describes the calculation of the objective function value.

Algorithm 2.8 FCM – calculation of the objective function value

Require: M – fuzzy membership matrix, C – list of cluster centers, X – input data, m – fuzziness parameter

Ensure: value of the objective function ▷ Formula (2.17)

```

1:  $value \leftarrow 0$ 
2: for  $j \leftarrow 1$  to  $|M[1]|$  do
3:   for  $i \leftarrow 1$  to  $|M|$  do
4:      $temp \leftarrow dist_E(X[i], C[j]) \cdot dist_E(X[i], C[j])$    ▷  $dist_E(\cdot)$  – Euclidean distance
5:      $value \leftarrow value + temp \cdot pow(M[i][j], m)$            ▷  $pow(\cdot)$  – exponentiation
6:   end for
7: end for
8: return  $value$ 
```

Algorithm 2.9 FCM – general scheme

Require: X – input data, c – target number of clusters, $maxIterations$ – maximum number of algorithm steps, m – fuzziness parameter, $maxDifference$ – maximum difference between objective function values

Ensure: C – list of cluster centers, M – fuzzy membership matrix of elements to clusters

```

1: let  $M$  be the fuzzy membership matrix of points from  $X$  to clusters
2: for  $i \leftarrow 1$  to  $|X|$  do
3:   for  $j \leftarrow 1$  to  $c$  do
4:      $M[i][j] \leftarrow 0$ 
5:   end for
6:    $j \leftarrow$  randomly select a cluster number from the range 1 to  $c$ 
7:    $M[i][j] \leftarrow 1$ 
8: end for
9:  $counter \leftarrow 0$ 
10:  $value \leftarrow \infty$ 
11:  $difference \leftarrow \infty$ 
12: while  $difference > maxDifference$  and  $counter < maxIterations$  do
13:   calculate the cluster centers  $C$   $\triangleright$  Algorithm 2.12:  $X \leftarrow X, M \leftarrow M, m \leftarrow m$ 
14:   calculate the distances from the cluster centers  $D$   $\triangleright$  Algorithm 2.11:  $X \leftarrow X, M \leftarrow M,$ 
      $C \leftarrow C$ 
15:   update the membership matrix  $M$   $\triangleright$  Algorithm 2.10:  $M \leftarrow M, D \leftarrow D, m \leftarrow m$ 
16:    $newValue \leftarrow$  retrieve the value of the objective function  $\triangleright$  Algorithm 2.8:  $M \leftarrow M,$ 
      $C \leftarrow C, X \leftarrow X, m \leftarrow m$ 
17:    $difference \leftarrow value - newValue$ 
18:    $value \leftarrow newValue$ 
19:    $counter \leftarrow counter + 1$ 
20: end while

```

Algorithm 2.10 FCM – updating the fuzzy membership matrix

Require: *ref* M – fuzzy membership matrix, D – distance matrix between elements and clusters, m – fuzziness parameter

```

1: for  $j \leftarrow 1$  to  $|M[1]|$  do
2:   for  $i \leftarrow 1$  to  $|M|$  do
3:      $value \leftarrow 0$ 
4:     for  $k \leftarrow 1$  to  $|M[1]|$  do
5:        $temp \leftarrow D[j][i]/D[k][i]$ 
6:        $value \leftarrow value + pow(temp, 2/(m - 1))$   $\triangleright pow(\cdot)$  – exponentiation
7:     end for
8:      $M[i][j] \leftarrow 1/value$   $\triangleright$  Formula (2.15)
9:   end for
10: end for

```

Algorithm 2.11 FCM – calculation of distances between elements and cluster centers**Require:** X – input data, M – fuzzy membership matrix, C – list of cluster centers**Ensure:** D – distance matrix between elements and clusters

```

1: let  $D$  be an empty distance matrix between elements and cluster centers
2: for  $j \leftarrow 1$  to  $|M[1]|$  do
3:   let  $temp$  be an empty list of distances between cluster  $j$  and individual elements from  $X$ 
4:   for  $i \leftarrow 1$  to  $|M|$  do
5:      $distance \leftarrow dist_E(X[i], C[j])$  ▷  $dist_E(\cdot)$  – Euclidean distance
6:     add  $distance$  to  $temp$ 
7:   end for
8:   add  $temp$  to  $D$ 
9: end for
10: return  $D$ 

```

Algorithm 2.12 FCM – calculation of cluster centers**Require:** X – input data, M – fuzzy membership matrix, m – fuzziness parameter**Ensure:** C – list of cluster centers

```

1: let  $C$  be an empty list of cluster centers
2: for  $j \leftarrow 1$  to  $|M[1]|$  do
3:   let  $temp$  be a point with  $|X[1]|$  coordinates
4:   for  $k \leftarrow 1$  to  $|X[1]|$  do
5:      $n \leftarrow 0$ 
6:      $d \leftarrow 0$ 
7:     for  $i \leftarrow 1$  to  $|X|$  do
8:        $n \leftarrow n + pow(M[i][j], m) \cdot X[i][k]$  ▷  $pow(\cdot)$  – exponentiation
9:        $d \leftarrow d + pow(M[i][j], m)$  ▷  $pow(\cdot)$  – exponentiation
10:    end for
11:     $temp[k] \leftarrow n/d$  ▷ Formula (2.14)
12:  end for
13:  add  $temp$  to  $C$ 
14: end for
15: return  $C$ 

```

2.5 Algorithms for determining the appropriate number of clusters

Unsupervised clustering algorithms require the number of groups to be specified in advance. Typically, this number is not known a priori and must be appropriately determined. For the algorithms presented in Subsection 2.4, two methods for determining the number of clusters are described: The Elbow Rule for the k -Means algorithm [84], [161] and the analysis

of the fuzzy partition coefficient for the Fuzzy c -Means algorithm [128], [175]. Determining the optimal number of clusters allows for a reduction in classification error (better fitting of dataset elements to groups) and minimizes the risk of overfitting the model.

2.5.1 Elbow rule

A technique used to determine the optimal number of clusters for the k -Means algorithm is the Elbow rule [84], [161]. This method involves identifying the inflection point on a plot of classification error drawn for different numbers of clusters used in the method. This point is referred to as the “elbow.” The classification error is defined as the sum of squared distances between elements and the centroids of the clusters to which they were assigned:

$$error_{kmeans} = \sum_{i=1}^{|X|} \|x_i - C_{M[i]}\|^2 \quad (2.18)$$

where x_i is the i -th element in X , M is the vector of cluster membership values, $C_{M[i]}$ is the centroid of the cluster to which x_i belongs, and $\|\cdot\|$ denotes the chosen distance metric.

To identify the inflection point, the second-order finite difference ratio of the classification error is computed. The elbow is typically located near the maximum of this ratio. Consequently, the number of clusters following the maximum value of the second-order difference ratio is considered. The implementation scheme is presented in Algorithm 2.13.

Additionally, Figure 2.3 provides a visualization of the Elbow rule for a sample dataset. The data is prepared to contain three distinct clusters. In Figure 2.3 (a), clustering is performed using the k -Means algorithm with $k = 3$. Cluster membership is indicated by different colors. Figure 2.3 (b) presents the classification error plot with the elbow point marked.

2.5.2 Fuzzy partition coefficient analysis

Similarly to the k -Means algorithm, it is possible to determine the optimal number of clusters for the Fuzzy C-Means (FCM) algorithm. A fuzzy partition coefficient is introduced to evaluate the properties of the generated clusters. Several techniques for computing this coefficient can be found in the literature [175]. However, for the purpose of this study, an intuitive and computationally simple solution is adopted [20], [165], defined as:

$$fpc = \frac{1}{|X|} \sum_{j=1}^c \sum_{i=1}^{|X|} w_{ij}^2 \quad (2.19)$$

where X is the dataset, c is the number of clusters, and w_{ij} denotes the fuzzy membership of the i -th element to the j -th cluster.

Algorithm 2.13 Elbow rule for determining the number of clusters in the k -Means algorithm

Require: X – input data, $start$ – minimum number of clusters analyzed, $stop$ – maximum number of clusters analyzed, $maxIterations$ – maximum number of steps of the k -Means algorithm

Ensure: optimal number of clusters

```

1: let  $E$  be an empty list of classification errors
2: for  $i \leftarrow start$  to  $stop$  do
3:   invoke the  $k$ -Means algorithm, retrieve the list of cluster centers  $C$  and the list of element
     memberships  $M$        $\triangleright$  Algorithm 2.7:  $X \leftarrow X, k \leftarrow i, maxIterations \leftarrow maxIterations$ 
4:    $error \leftarrow 0$ 
5:   for  $j \leftarrow 1$  to  $|X|$  do
6:      $error \leftarrow error + dist_E(X[j], C[M[j]]) \cdot dist_E(X[j], C[M[j]])$        $\triangleright$  Formula
       (2.18),  $dist_E(\cdot)$  – Euclidean distance
7:   end for
8:   add  $error$  to the list  $E$ 
9: end for
10:  $max \leftarrow -\infty$ 
11:  $index \leftarrow -1$ 
12: for  $i \leftarrow 2$  to  $|E| - 1$  do
13:    $diffQuotient2 \leftarrow E[i - 1] - 2 \cdot E[i] + E[i + 1]$ 
14:   if  $diffQuotient2 > max$  then
15:      $max \leftarrow diffQuotient2$ 
16:      $index \leftarrow i$ 
17:   end if
18: end for
19: return  $start + index - 1$ 

```

To determine the optimal number of clusters, the value of c for which the coefficient reaches its maximum should be selected. The implementation scheme is presented in Algorithm 2.14.

Furthermore, Figure 2.4 shows a visualization of the fuzzy partition coefficient analysis conducted for several values of c . The data is prepared such that three distinct clusters are visible. In this case, the highest coefficient value is observed for exactly three clusters.

Algorithm 2.14 Fuzzy partition coefficient analysis for determining the number of clusters in the FCM algorithm

Require: X – input data, $start$ – minimum number of clusters analyzed, $stop$ – maximum number of clusters analyzed, $maxIterations$ – maximum number of steps of the Fuzzy C -Means algorithm, m – fuzziness parameter, $maxDifference$ – maximum difference between objective function values

Ensure: optimal number of clusters

```

1:  $max \leftarrow -1$ 
2:  $index \leftarrow -1$ 
3: for  $i \leftarrow start$  to  $stop$  do
4:   invoke the FCM algorithm and retrieve the matrix of element memberships  $M$   $\triangleright$  Algorithm 2.9:  $X \leftarrow X, c \leftarrow i, maxIterations \leftarrow maxIterations, m \leftarrow m, maxDifference \leftarrow maxDifference$ 
5:    $fpc \leftarrow 0$ 
6:   for  $j \leftarrow 1$  to  $i$  do
7:     for  $k \leftarrow 1$  to  $|X|$  do
8:        $fpc \leftarrow fpc + M[k][j] \cdot M[k][j]$ 
9:     end for
10:  end for
11:   $fpc \leftarrow fpc / |X|$   $\triangleright$  Formula (2.19)
12:  if  $fpc > max$  then
13:     $max \leftarrow fpc$ 
14:     $index \leftarrow i$ 
15:  end if
16: end for
17: return  $index$ 

```

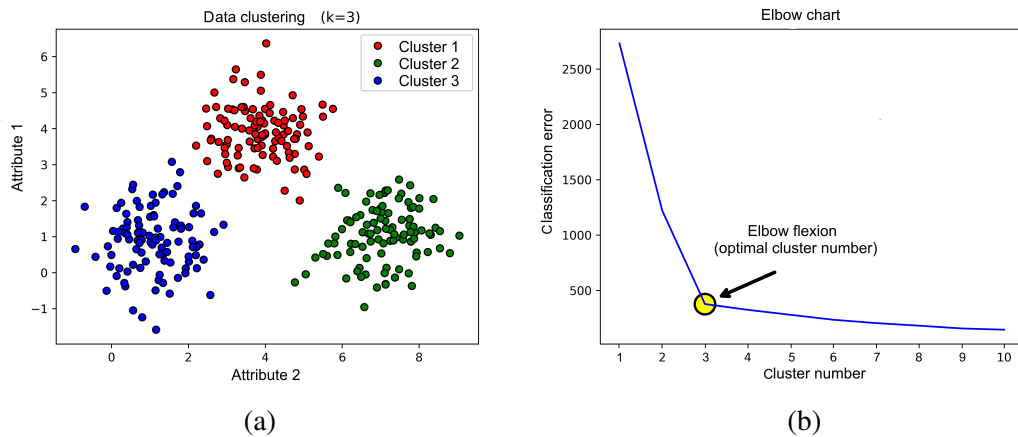


Figure 2.3: Visualization of the Elbow method for the k -Means algorithm. (a) Elements grouped by the k -Means algorithm for $k = 3$, with cluster membership indicated by color. (b) Classification error plot with the Elbow point (optimal number of clusters) indicated for the data shown in (a).

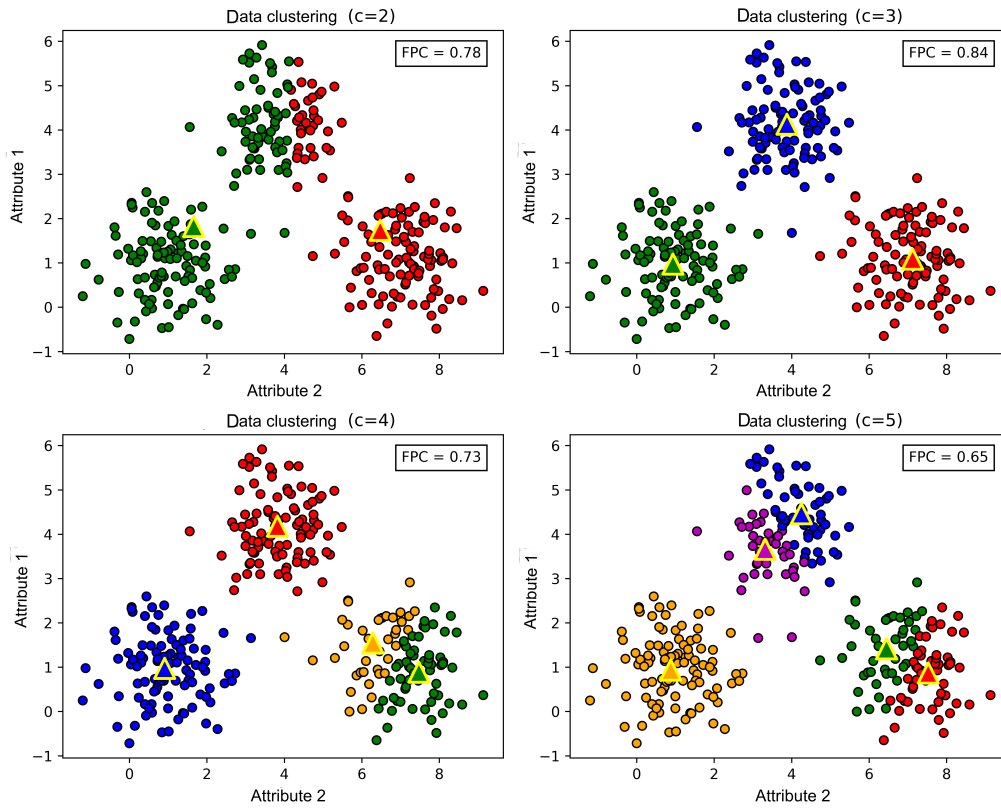


Figure 2.4: Visualization of fuzzy partition coefficient analysis for the FCM algorithm. The figure shows element memberships to individual clusters marked with colors, along with the coefficient values. The analysis was performed for $c \in \{2, 3, 4, 5\}$.

2.6 Takagi–Sugeno fuzzy inference model

The primary objective of introducing fuzzy inference models is to improve decision-making accuracy and precision. Among the most important and widely used fuzzy inference models are the Mamdani model [116], the Takagi–Sugeno model [157], [158], fuzzy relational models [129], fuzzy multi-models [130], and fuzzy neural models [70]. This section provides a detailed description of the Takagi–Sugeno (TS) model.

The TS model belongs to the class of MISO (Multiple Input, Single Output) models, meaning that the model may have multiple inputs, but the output is a single value. Several key components can be distinguished in the TS model. First, variables are fuzzified in the fuzzification block. Fuzzification assigns degrees of membership of the variables to fuzzy sets. The second important element is the inference block, which is directly related to the fuzzy rule base. The input to this block consists of fuzzy membership values obtained from the fuzzification block. Based on fuzzy rules, the model calculates the rule conclusions and the degrees of activation. The final stage is the defuzzification block, where the outputs of the inference block are aggregated. The structure of the model is illustrated in Figure 2.5.

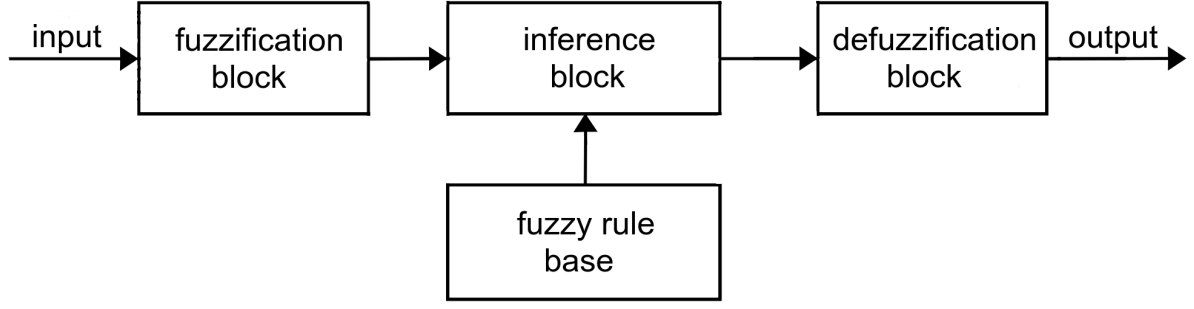


Figure 2.5: Diagram of the Takagi–Sugeno fuzzy inference model.

In the TS model, the rules take a characteristic form in which the consequents are functional expressions, typically linear functions. The general form of the rules is presented below:

$$\begin{aligned}
 R_1 : & \textbf{If } (x_1 \text{ is } A_1^1 \text{ and } x_2 \text{ is } A_2^1 \text{ and } \dots \text{ and } x_n \text{ is } A_n^1) \textbf{ Then } (y_1 = f_1(x_1, x_2, \dots, x_n)) \\
 & \vdots \\
 R_m : & \textbf{If } (x_1 \text{ is } A_1^m \text{ and } x_2 \text{ is } A_2^m \text{ and } \dots \text{ and } x_n \text{ is } A_n^m) \textbf{ Then } (y_m = f_m(x_1, x_2, \dots, x_n))
 \end{aligned} \tag{2.20}$$

where x_n is the n -th linguistic variable, A_n^m is the n -th fuzzy set in the m -th rule, f_m is the function associated with the m -th rule, m is the number of rules, and n is the number of inputs to the model.

In the inference block, the functions $f_1, \dots, f_m$ are evaluated for the given input values of the linguistic variables $x_1, \dots, x_n$. The computed values are referred to as the rule conclusions. In addition, the activation levels of individual rules are calculated using the following formula:

$$w_i(x_1, x_2, \dots, x_n) = \min(\mu_{A_1^i}(x_1), \mu_{A_2^i}(x_2), \dots, \mu_{A_n^i}(x_n)) \tag{2.21}$$

where x_n is the n -th linguistic variable, and $\mu_{A_n^i}$ is the membership function of the linguistic variable to the fuzzy set A_n^i for the i -th rule.

In the final block, i.e. the defuzzification block, the results from the inference block are aggregated. The output value is computed using the weighted average (center of gravity) method as follows:

$$y(x_1, x_2, \dots, x_n) = \frac{\sum_{i=1}^m w_i(x_1, x_2, \dots, x_n) f_i(x_1, x_2, \dots, x_n)}{\sum_{i=1}^m w_i(x_1, x_2, \dots, x_n)} \tag{2.22}$$

where w_i is the activation level of the i -th rule, f_i is the output function of the i -th rule, x_n is the n -th linguistic variable, m is the number of rules, and n is the number of model inputs.

2.7 Performance measures of classifiers

The final part of this chapter presents an overview of the definitions of anomaly detection performance measures. These measures can be divided into two categories: those that depend on the detection threshold and those that are independent of it.

We begin with the definitions of threshold-dependent measures, which are as follows:

- **accuracy**

$$accuracy = \frac{TN + TP}{TN + FP + TP + FN} \quad (2.23)$$

- **precision**

$$precision = \frac{TP}{TP + FP} \quad (2.24)$$

- **recall**

$$recall = \frac{TP}{TP + FN} \quad (2.25)$$

- **specificity**

$$specificity = \frac{TN}{TN + FP} \quad (2.26)$$

- **false alarm ratio**

$$falsealarmratio = \frac{FP}{FP + TN} \quad (2.27)$$

- **F1 score**

$$F1\ score = \frac{2 \cdot precision \cdot recall}{precision + recall} \quad (2.28)$$

where TP , TN , FP , and FN denote the number of true positives (correctly identified anomalies), true negatives (correctly identified normal samples), false positives (normal samples misclassified as anomalies), and false negatives (anomalies misclassified as normal), respectively, for a given detection threshold.

The second group of measures includes threshold-independent metrics. These include the area under the receiver operating characteristic curve (ROC AUC), where the axes represent the recall and false alarm ratio for each considered detection threshold. The second measure is the area under the precision-recall curve (PR AUC), where the axes correspond to precision and recall values for each threshold.

Chapter 3

Isolation Forest and its extensions

The Isolation Forest (IF) algorithm was proposed in 2008 in the work [106], and later described in more detail in the article [107]. This technique gained significant popularity and has established itself as one of the key algorithms for anomaly detection. This is due to its two most important characteristics: execution speed and classification accuracy. In the literature, IF is recognized as one of the fastest unsupervised learning methods for detecting anomalous elements (outliers) [29], [44], [124], [145]. The simplicity of the Isolation Forest structure and the intuitiveness of its processing technique enable its application across a wide range of problems. This has motivated many researchers to propose numerous specialized and improved versions of IF. This chapter presents the main concept of the base IF method, as well as related algorithms that extend its mechanisms with new ideas.

3.1 Isolation Forest

The operation of the IF algorithm can be divided into two main phases. The first phase involves the construction of a forest composed of isolation trees. It is important to emphasize that the forest is built only once, at the start of the algorithm. The second phase concerns the evaluation of the anomaly degree of elements using a specific scoring function. This evaluation phase is executed each time an element is assessed.

The construction of the Isolation Forest proceeds as follows. First, the number of isolation trees t and the number of samples n used to build a single tree are specified. Each isolation tree is built independently, based on randomly sampled data. All constructed trees form the forest. By convention, the algorithm is typically run with $t = 100$ and $n = 256$. These are hyperparameters that can be adjusted to obtain the optimal model. It is recommended that n be a power of 2. After sampling n elements from the dataset, they are subjected to isolation through recursive partitioning. The split point is determined by randomly selecting a data attribute and a random value between the minimum and maximum value of that attribute

among the current node's samples. The sample set is then divided into two subsets. One contains samples where the attribute value is less than the split point and is passed to the left child node. The other subset is passed to the right child node. The split attribute and its value are stored in the current node so that this information can be used later during element evaluation. The partitioning procedure is called recursively for the left and right child nodes until a node contains only a single sample or the current depth reaches the tree height limit:

$$l = \lceil \log_2 n \rceil \quad (3.1)$$

where n is the number of samples used to build the isolation tree.

Complete implementation schemes for constructing a single isolation tree and the entire forest are presented in Algorithms 3.1 and 3.2 [106].

Note that the computational complexity of the first phase is $O(n \log n)$ [106]. The second phase of the algorithm computes the final anomaly score of a given element. Evaluation is performed by traversing each tree from the root node. At each node, a condition is checked: If the value of the relevant attribute in the evaluated element is less than the stored threshold, the left child node is selected; otherwise, the right one is chosen. Traversal ends when a leaf node (a node without children) is reached. The depth of the reached node is recorded. The result of the evaluation for a single tree depends on the number of samples stored in the final node. If the node contains only one sample, the result is the depth. Otherwise, the result is the depth plus an estimated average path length in a binary search tree:

$$h(x) = \begin{cases} h & \text{for } q \leq 1 \\ h + c(q) & \text{for } q > 1 \end{cases} \quad (3.2)$$

where x is the evaluated element, h is the depth reached in the isolation tree, q is the number of samples in the terminal node, and

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n} \quad (3.3)$$

[132], where

$$H(n) = \ln n + 0.5772156649 \quad (3.4)$$

is the harmonic number approximation.

Algorithm 3.1 Isolation Forest – single tree formation phase

Require: X – input data, e – current node depth, l – maximum tree height

Ensure: isolation tree

```

1: if  $e \geq l$  or  $|X| \leq 1$  then
2:   create a node  $node$ 
3:    $node.size \leftarrow |X|$ 
4:   return  $node$ 
5: else
6:   let  $A$  be the list of attributes in  $X$ 
7:   randomly select an attribute  $a$  from the set  $A$ 
8:   select a random split point  $p$  between the minimum and maximum value of attribute  $a$  in  $X$ 
9:    $X_{left} \leftarrow filter(X, a < p)$ 
10:   $X_{right} \leftarrow filter(X, a \geq p)$ 
11:  create a node  $node$ 
12:   $node.left \leftarrow$  create an isolation tree from the set  $X_{left}$   $\triangleright$  Algorithm 3.1:  $X \leftarrow X_{left}$ ,
     $e \leftarrow e + 1, l \leftarrow l$ 
13:   $node.right \leftarrow$  create an isolation tree from the set  $X_{right}$   $\triangleright$  Algorithm 3.1:  $X \leftarrow X_{right}$ ,
     $e \leftarrow e + 1, l \leftarrow l$ 
14:   $node.size \leftarrow null$ 
15:   $node.splitAttribute \leftarrow a$ 
16:   $node.splitValue \leftarrow p$ 
17:  return  $node$ 
18: end if
  
```

Algorithm 3.2 Isolation Forest – forest formation phase

Require: X – input data, t – number of isolation trees, n – number of samples

```

1: let  $iForest$  be an empty list of isolation trees
2:  $l \leftarrow \lceil \log_2 n \rceil$ 
3: for  $i \leftarrow 1$  to  $t$  do
4:    $X' \leftarrow$  retrieve  $n$  samples from the set  $X$ 
5:    $iTree \leftarrow$  create an isolation tree from the set  $X'$   $\triangleright$  Algorithm 3.1:  $X \leftarrow X', e \leftarrow 0$ ,
     $l \leftarrow l$ 
6:   add the isolation tree  $iTree$  to  $iForest$ 
7: end for
  
```

The implementation scheme is presented in Algorithm 3.3 [106]. It is important to emphasize the role of Equation (3.3), which describes the estimated average path length for unsuccessful searches in a binary search tree [132]. This formula is applied to reduce evaluation time. If a node is reached where partitioning has been terminated due to the height limit, the height is approximated accordingly.

Algorithm 3.3 Isolation Forest – calculation of the path length of an element in the isolation tree**Require:** x – evaluated element, $iTree$ – isolation tree, e – current depth**Ensure:** path length of the element x

```

1: if  $iTree.size > 1$  then
2:   return  $e + c(iTree.size)$  ▷ Formula (3.3)
3: else if  $iTree.size \geq 0$  then
4:   return  $e$ 
5: end if
6:  $a \leftarrow iTree.splitAttribute$ 
7:  $p \leftarrow iTree.splitValue$ 
8:  $x_a \leftarrow$  retrieve the value of attribute  $a$  for the element  $x$ 
9: if  $x_a < p$  then
10:  return calculate the path length in the isolation tree for the left child ▷ Algorithm 3.3:
     $x \leftarrow x, iTree \leftarrow iTree.left, e \leftarrow e + 1$ 
11: else
12:  return calculate the path length in the isolation tree for the right child ▷ Algorithm 3.3:
     $x \leftarrow x, iTree \leftarrow iTree.right, e \leftarrow e + 1$ 
13: end if

```

The values of $h(x)$ are computed for all trees in the forest. The final anomaly score is given by:

$$score_h(x, n) = 2^{-\frac{1}{c(n)} \frac{1}{t} \sum_{i=1}^t h_i(x)} \quad (3.5)$$

where x is the evaluated element, n is the number of samples used to build a single tree, t is the number of trees in the forest, and h_i is the path length in the i -th tree.

Anomaly scores are within the range $[0, 1]$. The higher the value, the more likely the element is an anomaly. A value of 0.5 indicates that the average node depth equals the theoretical expected average path length in a binary tree. The overall complexity of the second phase is $O(t \log n)$ [106].

To better illustrate the process of isolation forest construction and evaluation, an example of a single tree is shown in Figure 3.1. In Figure 3.1 (a), 16 two-dimensional samples are presented, along with the partitioning lines. Note that not all samples are fully isolated due to the height constraint. The resulting tree structure is shown in Figure 3.1 (b). The maximum height limit means the tree construction halts at that depth, which may result in a leaf node containing multiple samples. Evaluation of a new element is shown in Figure 3.1 (c), where the number of traversed nodes corresponds to the evaluation depth.

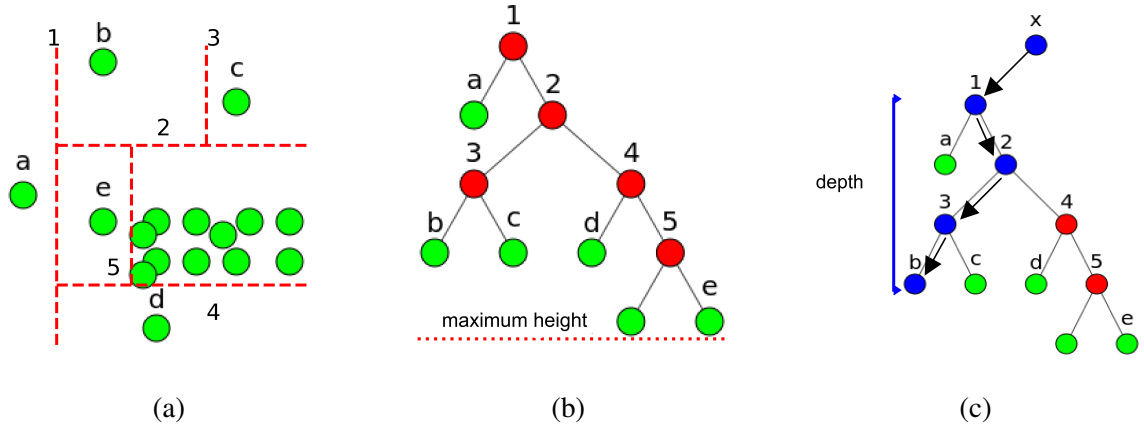


Figure 3.1: Example of the Isolation Forest algorithm operation. (a) Randomly selected samples with marked split lines. (b) Isolation tree constructed based on the samples. Leaf nodes are marked in green; internal nodes are marked in red. (c) Tree traversal for the evaluated element x . Visited nodes are marked in blue.

3.2 Review of Isolation Forest modifications

The Isolation Forest algorithm is still not perfect. Many studies have been conducted to improve its weaknesses and enhance its efficiency. Readers interested in anomaly detection methods based on tree structures may refer to the survey papers [5], [17]. IF is applied to the analysis of various types of anomalies. This subsection presents the most significant modifications.

Let us begin our discussion with a description of a technique that changes the isolation of elements in isolation nodes. The basic method isolates samples by randomly determining a threshold value of an attribute. This threshold separates the samples into two parts so that elements with attribute values smaller than the threshold are separated from the remaining ones. In [62], the authors introduce splits based on a hyperplane embedded in the multidimensional attribute space of the elements. In other words, elements are isolated based on more than one attribute. The hyperplane is selected randomly, which may result in no real division into two sets. This issue is addressed in [96]. The proposed randomness of hyperplane selection is limited to the region where an actual division into two sets is obtained. In [35], [36], isolation based on a hypersphere is proposed. Not only the entire data space is considered, but the shape of the boundary itself is also changed. A similar solution [15] uses hyperspheres to evaluate elements. During the training phase of the algorithm, hyperspheres are generated with sizes dependent on the distance to the nearest record.

The second important improvement is the appropriate selection of the attribute split value. The original method maintains randomness in this respect. In the case of Isolation Forest, anomalous elements should be isolated as quickly as possible. Therefore, attention should be paid to the fact that selecting a split point by random sampling may not be the most effective

solution. The works [117], [164], [184] introduce an increased probability of selecting an optimal split point. This idea increases the chance of avoiding splits in dense regions, which in turn leads to faster isolation of elements in low-density areas. On the other hand, the attribute itself may be selected more effectively [53]. The quality of separation provided by an attribute may vary. Therefore, it is worth choosing an attribute that enables optimal splitting and isolation of outlier elements. In [183], a dimension entropy measure is proposed to find the optimal attribute. Attributes with higher entropy values have greater potential to isolate elements. Therefore, instead of selecting an attribute at random, the one with the highest entropy is chosen. In [102], various scenarios of selecting the split value depending on the obtained entropy value are also introduced.

Another group of noteworthy techniques for improving the effectiveness of Isolation Forest involves modifications of both algorithm phases, i.e., the forest construction phase and the element evaluation phase. Here, trees are used in which the number of child nodes can be greater than two [23], [75]. This leads to a division into at least two subsets and allows for a reduction in tree depth, thus speeding up the evaluation of elements in the second phase of the algorithm. Additionally, the evaluation method itself is modified. This is related to the introduction of additional elements into the forest construction mechanism, as well as the incorporation of new ideas for evaluation. The most important works focus on varying the computation of the depth of the evaluated element in the isolation tree. The authors of [119] use weights during transitions between nodes. In contrast, in [10], the depth is based on the so-called relative node mass. Finally, intelligent weight selection is used [170]. The weights are a combination of simple tree traversal and local heights. Another common approach is to change the distance metric to something other than Euclidean distance. This is particularly relevant in the analysis of high-dimensional datasets, for which Euclidean distance becomes ineffective. In such cases, the Mahalanobis distance is used [186]. Also, [178] proposes approaches for datasets with arbitrary distance functions.

The algorithms listed above focus on individual trees. An isolation forest contains many trees in its structure. Therefore, introducing appropriate aggregation of results from individual trees is crucial. In [28], the authors consider using only a subset of trees forming the forest. It turns out that in many cases, it is not necessary to evaluate an element across the entire forest. The evaluation obtained from a selected part of the forest can accurately indicate the anomaly or normality of a sample without losing efficiency. Reducing the number of trees can be achieved through the technique known as simulated annealing [179]. In this approach, the full forest is constructed, and only those trees that perform best in anomaly detection (based on a precision criterion) are retained. Another interesting and noteworthy approach is the aggregation of trees through the introduction of weighted trees [167]. Specifically, the results obtained from trees are weighted using an attention mechanism. It can be said that stronger results contribute more to the final outcome.

Anomaly detection based on isolation, as proposed in IF, is often supported by clustering algorithms. In such cases, the anomalousness of an element may be associated with its membership in a group of elements. The authors of [80] proposed trees whose nodes may have an arbitrary number of leaves. These depend on the optimal number of clusters for the randomly selected attribute, estimated using the so-called elbow rule. The subsequent work [81] employs the more robust against outliers k -medoids algorithm. Additionally, clustering techniques based on fuzzy sets are also used [13], [82]. Nonetheless, clustering analysis is also used prior to the forest construction phase [32], [90], [105]. In other words, data are first clustered, and then an Isolation Forest is created based on the resulting clusters. To determine the degree of membership of an evaluated element to a specific group, a selection matrix is implemented [149].

The diversity of approaches enhancing IF is significant. The aforementioned clustering algorithms support the isolation of elements for data grouped into clusters. Nevertheless, mechanisms that incorporate other techniques into IF have also emerged. One of the main challenges in achieving high efficiency is the high dimensionality of data. To address this issue, preprocessing of raw data is applied. The results are then passed to the input of the IF method. The most commonly used approach for dimensionality reduction is principal component analysis [69], [192]. Linear transformation of data into a lower-dimensional space preserves information about data variability, resulting in improved method performance. A deeper analysis of algorithm properties also allows the introduction of completely different preprocessing strategies. The authors of [160] use data transformation via a sparse random projection matrix. The method of reasoning about the anomalousness of a record can also be changed using techniques based on fuzzy logic and fuzzy sets. On the one hand, the data themselves can be identified by a membership function to a fuzzy set [30], leading to fuzzy data partitioning. On the other hand, fuzzy logic is used in the evaluation function itself [195].

The IF method is an unsupervised learning approach. This means that data used in the training phase do not require labels assigning them to specific classes, i.e., classes of anomalous or normal elements. It should be noted here that the introduction of supervision can yield better results in terms of execution time as well as classification accuracy [118]. Moreover, in low-resource scenarios, weakly supervised learning has significant deployment potential [18]. An interesting issue is addressed in [59], where a cascade combination of IF as an unsupervised method and other supervised methods is applied. Such a cascade allows for obtaining labels for data using IF. Subsequently, the labeled data are fed into methods that require labels.

The previously presented extensions of the IF algorithm are mainly aimed at improving general classification accuracy or execution speed. In addition to this, the IF algorithm also finds applications in specialized problems. Due to its execution speed, it allows for the analysis of streaming data transmitted through telecommunication networks. In this case, it is combined

with various hashing functions and also incorporates sliding time windows that extract fragments of the analyzed data [135], [185]. The authors of [112] propose incorporating the Mondrian process [144] into the IF mechanism. On the other hand, [177] highlights the use of hashing functions in a two-stage learning scheme. Streaming data may originate from sources emitting weak or noisy signals. Enhancements of the IF algorithm are being implemented for the analysis of streaming data from weak magnetic and geochemical signals [34], [193].

The second group of specialized IF extensions are models related to cloud technologies and application containerization. The authors of [24] propose anomaly detection in data derived from monitoring the infrastructure of cloud data centers. For this purpose, the isolation algorithm is enhanced by introducing attributes whose values come from encoding information from time series generated by the system. In the context of application containerization, we find modifications involving the introduction of weighted feature selection (attributes) into IF [198]. Furthermore, anomaly detection based on an IF extension is used for container management via the Kubernetes platform [61]. The IF technique is combined with other techniques for deployment in microservice-based systems [73].

The low computational complexity of the IF algorithm, and consequently low resource consumption and minimal operational cost, lead to its adoption in anomaly detection for Internet of Things (IoT) devices. In [31], forest training is performed based on data filtering through box plot analysis. Then, those trees that exhibit the highest classification accuracy are selected. In [2], the authors perform modulation, i.e., combining two detection subsystems. The first subsystem is trained on a dataset containing only attack-related data, while the second subsystem is trained solely on normal data, without attacks. Each subsystem is based on an Isolation Forest with a dynamically changing parameter representing the proportion of anomalous elements in the training dataset.

A key role is played by IF developments within the broadly understood concept of cybersecurity. On the one hand, solutions are implemented to protect against cyberattacks. The authors of [151] propose dynamically changing detection thresholds to adapt the algorithm to changing operating conditions of energy distribution systems. In [110], attribute grouping is introduced during the Isolation Forest formation phase. Furthermore, the height parameter of an element in the isolation tree is replaced by a parameter associated with the so-called data mass for analyzing communication information from transport systems [45]. On the other hand, solutions verifying vulnerability to cyberattacks are also implemented. In particular, IF can support the analysis of application source code [40]. Software testing is supplemented by feature pruning (selection of preferred attributes) during the forest training phase [41]. This process involves interpreting histograms of individual attributes. Anomaly detection using a modified IF is also employed for assessing applications or detecting fraud. The authors of [194] evaluate loan applications using balanced datasets with injected noise to train

the isolation trees. In [156], insurance fraud is detected through anomaly scoring with a conditional evaluation function. In such detection scenarios, the IF algorithm is combined with local density techniques, such as deviations from nearest neighbors [125].

The Isolation Forest and its variants are applied in image analysis, e.g., in anomaly detection in hyperspectral images. In [155], the authors extract spatial features from the image and then apply an improved Isolation Forest. Other researchers extract two profiles from the spatial features [171]. Detection is conducted in two independent directions, resulting in improved classification performance. A similar approach of creating a double Isolation Forest is employed in [123]. Additionally, sub-classes are extracted from hyperspectral image streams, allowing verification of whether the transmitted data are anomalous within the prepared sub-classes [189].

The final group of IF developments worth discussing are hybrid solutions. These combine various algorithms in such a way as to strengthen advantages and mitigate drawbacks. To the element isolation technique, solutions based on element density, distribution, or their grouping are added [111], [115]. However, hybrids of Isolation Forest and artificial neural networks (ANNs) [134] are gaining increasing attention. In particular, one should mention embedding networks into the structure of the IF algorithm itself. One concept involves transforming the data space using neural networks [180]. Architectures of the Isolation Forest algorithm resembling deep network structures are also used [176]. Finally, an increasingly visible trend in research is the desire to understand the reason behind the anomaly score value. In other words, there is a need to explain what caused a given element to be assigned a particular score. The authors of [26] propose reasoning about the importance of features that lead to the interpretation of the result by the Isolation Forest algorithm. In contrast, [6] presents a system combining IF with an additional decision support system based on a model-independent interpretation module. Additionally, fuzzy techniques [79] are employed, which use aggregated fractional membership values to compute the depth for element positions in the dataset.

Chapter 4

Enhancements of the Isolation Forest algorithm

This chapter presents innovative anomaly detection algorithms based on the Isolation Forest concept. The proposed methods are introduced in a series of publications [52], [54], [55], [77] and represent a significant contribution to the field. The chapter begins with a discussion of dimensionality reduction techniques applied to input data prior to the anomaly detection process. Subsequent sections describe approaches that incorporate the mechanism of minimal spanning tree construction into the core IF algorithm.

4.1 Dimensionality reduction of data

The first of the considered techniques for improving the performance of the IF method is the reduction of data dimensionality at the preprocessing stage [77]. On the one hand, the IF algorithm handles various types of data effectively due to the randomness of the isolation tree construction and the use of ensemble learning. On the other hand, the imposed height limit on isolation trees (see formula (3.1)) results in trees with shallow depth. Consequently, evaluated elements traverse significantly fewer split nodes compared to a fully grown tree. Each isolation node is capable of separating data based on a single attribute (dimension) of the analyzed dataset. In high-dimensional data, the number of attributes used for tree construction may be incomplete. This phenomenon becomes particularly evident when the number of trees or the number of samples used for training is small. The issue of high data dimensionality, referred to as the “curse of dimensionality,” can lead to a decrease in detection accuracy. To counteract this behavior, one could increase the number of trees or the number of training samples.

However, this results in increased computational time. Another important aspect is that some attributes are correlated, while others may have no influence on the degree of anomaly of a given record. Therefore, preprocessing of the data, prior to anomaly detection, is crucial for the performance of the IF algorithm.

In the literature, we observe techniques that reduce the dimensionality of input data for the IF method by employing principal component analysis [69], [192]. The result of such transformation is a reduced number of dimensions, while the data still retain the most relevant information. However, it is worth noting that some information may be lost in the process. An alternative approach for dimensionality reduction is the use of random sparse projection matrices [160]. This method introduces a partially random structure to the data. Nevertheless, particular attention should be paid to the placement of anomalous elements within the dataset. Intuitively, normal elements tend to form dense clusters, while anomalies are expected to be located farther away from those clusters. Thus, anomaly detection algorithms can benefit from being supplemented with clustering analysis techniques. The following subsections present two selected algorithms for dimensionality reduction [77]. Both solutions share a common execution scheme, illustrated in Figure 4.1. Two main phases can be distinguished in this process. The first phase involves reducing the dimensionality of the input data. The second phase consists of applying the unmodified Isolation Forest method to the data resulting from the first phase. Each data record after dimensionality reduction corresponds to a record in the original input space. Therefore, the anomaly score assigned to a reduced-dimension element corresponds to the score of the original input element.

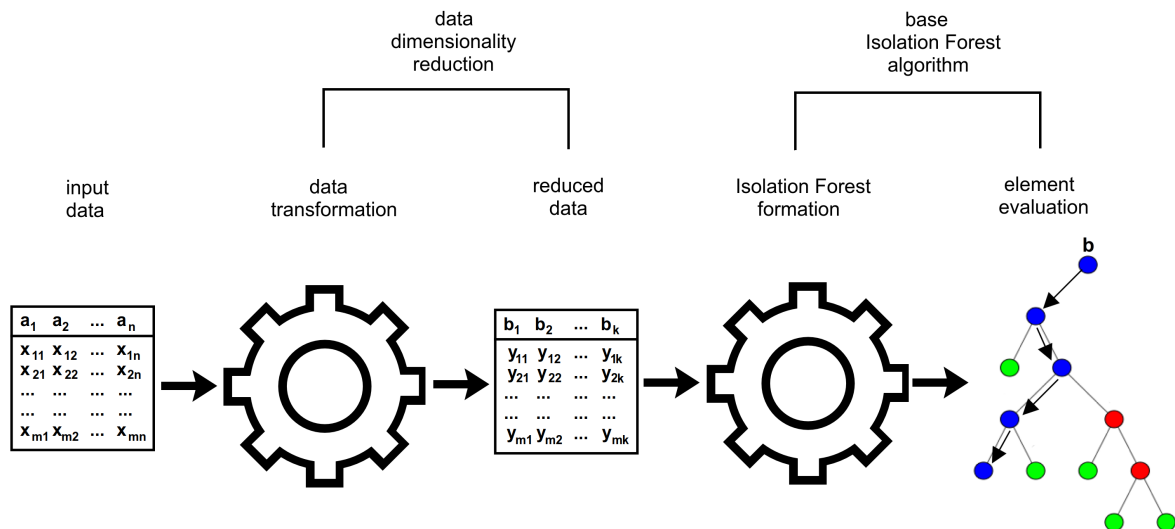


Figure 4.1: General scheme of the dimensionality reduction algorithm.

4.1.1 Clustering of elements using the k -Means method

The first clustering algorithm applied to reduce the dimensionality of data (Dimensionality Reduction-Based Isolation Forest, DRIF) is the k -Means algorithm, which is described in subsection 2.4.1. The set of elements is divided into k clusters (groups). Then, the centroids of the clusters are determined along with the distances of the elements from their respective cluster centroids. Figure 4.2 presents a visual example, where different colors are used to denote membership in each cluster.

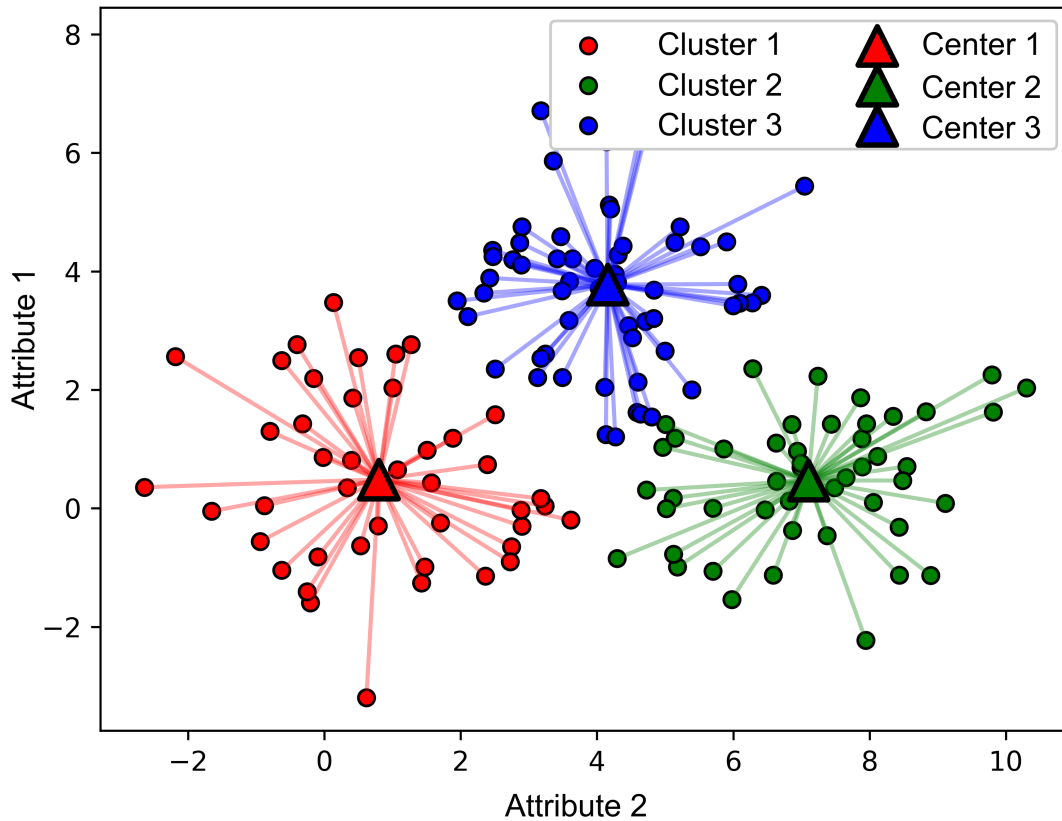


Figure 4.2: Visualization of clustering analysis using the k -Means algorithm. The figure shows the elements connected to the centroids of the clusters to which they belong.

In this case, after the transformation, each record has $k + 1$ attributes. The first k attributes represent the membership of the element in each of the k clusters. A value of 0 indicates no membership in the corresponding cluster, while a value of 1 indicates full membership. The last attribute stores the distance of the element from the centroid of the cluster it belongs to. In summary, the dimensionality is reduced from n to $k + 1$ attributes.

For the k -Means clustering algorithm, the value of the parameter k must be specified before running the algorithm. In the proposed solution, this parameter is determined using the so-called Elbow rule [84], [161], described in subsection 2.5.1. Another method used to determine

the optimal number of clusters for k -Means is the Silhouette method [143], [148]. However, this method is more computationally demanding, which is why the Elbow rule is used in the proposed solution. Since attributes can have different value scales, the input data is standardized using Formula (2.1).

The dimensionality reduction process is divided into two main phases. The first phase is related to training the solution. It is executed only once, before the actual attribute reduction stage, to obtain the cluster centroids. The implementation scheme for the first phase is presented in Algorithm 4.1. Several key parts of the algorithm can be identified here. First, the mean and standard deviation values for each attribute of the training data are prepared. Then, the attribute values are transformed using formula (2.1). The transformed data is passed to the method that determines the optimal number of clusters (in this case, using the Elbow rule). Finally, the clusters for the optimal number of k are selected, as determined by the Elbow rule. It should be noted that the lists of cluster centroids, mean values, and standard deviations are stored for later use in the second phase of the attribute reduction process. The standardization of attribute values is described in more detail in subsection 2.1.

After preparing the solution for dimensionality reduction, the actual data transformation is performed. Each element is transformed independently. The implementation scheme for the second phase of attribute reduction is presented in Algorithm 4.2. Similar to the first phase, the element is standardized, as shown in Formula (2.1). The next step is to find the nearest cluster for the given record. Finally, the attributes of the returned element are filled with values of 0 or 1, depending on the membership in a given cluster. Additionally, the last attribute takes the value of the distance from the nearest cluster.

In summary, for dimensionality reduction based on k -Means clustering, the computational complexity of this technique is analyzed. Let n be the number of training samples, a be the number of input attributes, k be the number of clusters, and i be the number of iterations of the algorithm. The first phase of execution is relatively complex. Preparing and standardizing the elements is of order $O(an)$. In the next step, the optimal number of clusters is determined through multiple calls to the k -Means algorithm. It is important to note that a single call to the algorithm has a complexity of $O(knia)$ [56]. The second phase, i.e., transforming the elements, is relatively simple and quick to compute, with a complexity of order $O(ak)$. In practice, to reduce execution time, the number of clusters is limited to values between 2 and 10.

Algorithm 4.1 Dimensionality reduction using k -Means algorithm – training phase

Require: X – input data

Ensure: C – list of cluster centers, AVG – list of attribute means, STD – list of attribute standard deviations

- 1: perform standardization of X and retrieve: standardized data Y , list of attribute means AVG , list of attribute standard deviations STD ▷ Algorithm 2.1: $X \leftarrow X$
 - 2: $k \leftarrow$ retrieve the optimal number of clusters based on the Elbow rule ▷ Algorithm 2.13:
 $X \leftarrow Y, start \leftarrow 2, stop \leftarrow 10, maxIterations \leftarrow 100$
 - 3: invoke the k -Means algorithm and retrieve the list of cluster centers C ▷ Algorithm 2.7:
 $X \leftarrow Y, k \leftarrow k, maxIterations \leftarrow 100$
-

Algorithm 4.2 Dimensionality reduction using k -Means algorithm – data transformation phase

Require: x – input element, C – list of cluster centers, AVG – list of attribute means, STD – list of attribute standard deviations

Ensure: element after attribute reduction

- 1: $x \leftarrow$ retrieve the standardized element ▷ Algorithm 2.2: $x \leftarrow x, AVG \leftarrow AVG, STD \leftarrow STD$
 - 2: $min \leftarrow \infty$
 - 3: $index \leftarrow -1$
 - 4: **for** $i \leftarrow 1$ to $|C|$ **do**
 - 5: $d \leftarrow dist_E(x, C[i])$ ▷ $dist_E(\cdot)$ – Euclidean distance
 - 6: **if** $d < min$ **then**
 - 7: $min \leftarrow d$
 - 8: $index \leftarrow i$
 - 9: **end if**
 - 10: **end for**
 - 11: let y be an element with $|C| + 1$ attributes
 - 12: **for** $i \leftarrow 1$ to $|C|$ **do**
 - 13: **if** $i = index$ **then**
 - 14: $y[i] \leftarrow 1$
 - 15: **else**
 - 16: $y[i] \leftarrow 0$
 - 17: **end if**
 - 18: **end for**
 - 19: $y[|C| + 1] \leftarrow min$
 - 20: **return** y
-

4.1.2 Clustering of elements using the FCM method

The next clustering algorithm considered in the context of dimensionality reduction (Fuzzy Dimensionality Reduction-Based Isolation Forest, FDRIF) is the Fuzzy C-Means (FCM) clustering algorithm, as described in subsection 2.4.2. The technique presented in the previous

subsection identifies the membership of an element in a single cluster along with the distance to its centroid. However, it is also worth considering the membership of an element in multiple clusters using the FCM technique. This approach provides greater differentiation in the description of the resulting records during the training phase. Figure 4.3 illustrates an example of the FCM method in action. The membership of elements in clusters is denoted using different colors, and the values of these memberships (distances to the centroids) are indicated by the lengths of the lines. Additionally, the centroids of all clusters are highlighted.

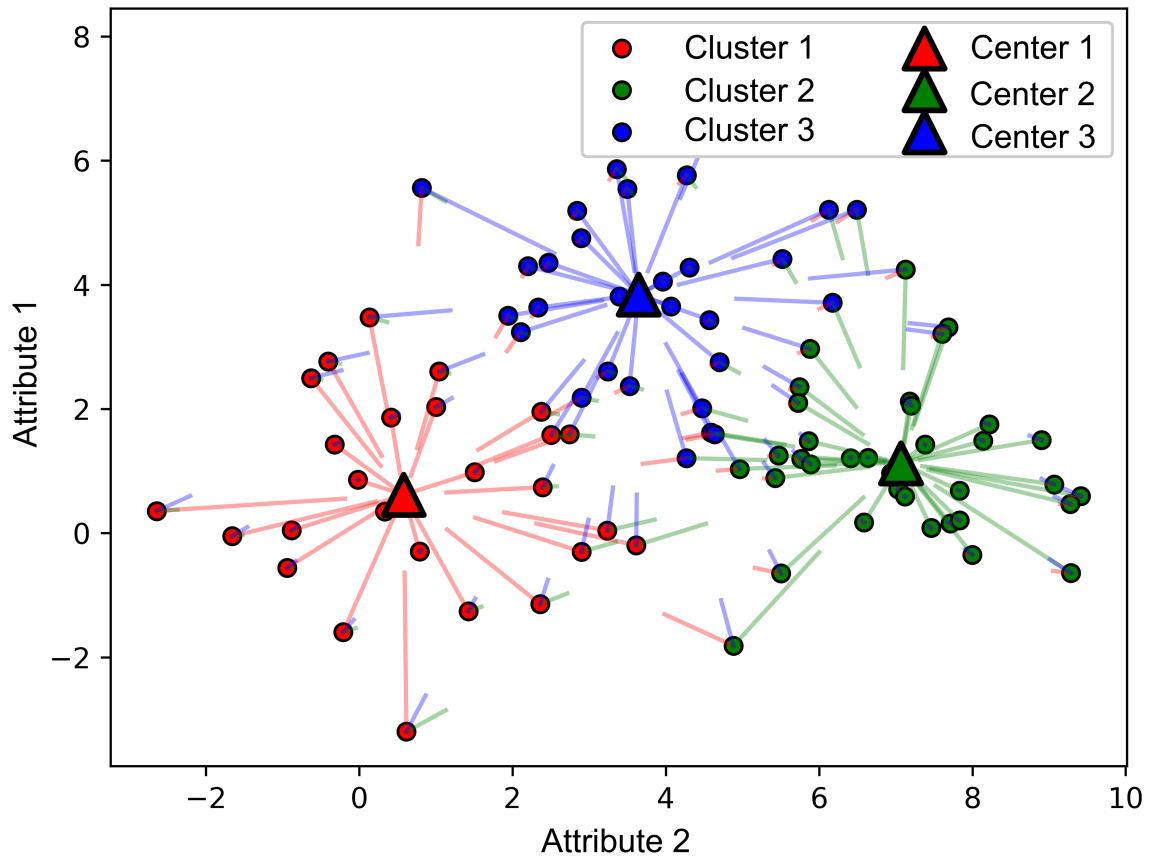


Figure 4.3: Visualization of clustering analysis using the FCM algorithm. The figure shows cluster centers and the membership values of elements to individual clusters, represented by the lengths of the line segments.

For this technique, we propose transforming the data by creating c clusters and calculating the distances and membership values for each of the clusters. After the transformation, each data record contains $2c$ columns. The values of these columns are determined by sorted fuzzy membership values and sorted distances from the cluster centroids. The sorting is applied so that the first attribute corresponds to the closest cluster, the second to the second-closest cluster, and so on, which prepares a consistent feature ordering for the subsequent phase.

The value of the parameter c must be determined such that the number of clusters is optimal. To achieve this, we use the analysis of the so-called fuzzy partition coefficient (FPC) [128], [175]. This method allows the determination of the optimal number of clusters for the FCM algorithm. Similar to the k -Means algorithm, the attributes are standardized to ensure comparable scales of the attribute values. The same standardization process is applied using formula (2.1).

In this case, the dimensionality reduction process is performed similarly to the one discussed in subsection 4.1.1. Again, two phases are distinguished: The training phase and the element transformation phase. The implementation scheme for the first phase is presented in Algorithm 4.3. In this phase, first, the elements are standardized, and then the mean and standard deviation values are recorded. After preparing the data, the optimal number of clusters is determined. Finally, the FCM algorithm is executed with the value of c set to the optimal number of clusters. The cluster centroids obtained in this step are stored in memory for use in the second phase.

In the second phase, the actual dimensionality reduction is performed. In this process, each element is transformed independently, as outlined in the implementation scheme in Algorithm 4.4. The first step is to standardize the element. Then, the distances from the element to the cluster centroids are calculated. Finally, based on these distances, the degree of membership of the element to each cluster is determined.

Let us now consider the computational complexity of the dimensionality reduction method based on FCM. Let n be the number of training samples, c be the number of clusters, a be the number of input attributes, and i be the number of iterations of the algorithm. The training phase is executed only once. The first step, which is the standardization of the attributes, is performed in $O(an)$. However, the most computationally demanding part is determining the optimal number of clusters. Therefore, the FCM algorithm must be invoked multiple times, with the complexity of a single invocation being $O(c^2nia)$ [56]. The second phase has a low computational complexity, i.e., $O(ac + c^2)$. It is worth noting that it is sensible to introduce a limitation for the maximum number of clusters c .

Algorithm 4.3 Dimensionality reduction using the FCM algorithm – training phase

Require: X – input data

Ensure: C – list of cluster centers, AVG – list of attribute means, STD – list of attribute standard deviations

- 1: perform standardization of X and retrieve: standardized data Y , list of attribute means AVG , list of attribute standard deviations STD ▷ Algorithm 2.1: $X \leftarrow X$
 - 2: $c \leftarrow$ retrieve the optimal number of clusters based on fuzzy partition coefficient analysis
▷ Algorithm 2.14: $X \leftarrow Y$, $start \leftarrow 2$, $stop \leftarrow 10$, $maxIterations \leftarrow 100$, $m \leftarrow 1.5$, $maxDifference \leftarrow 0.00001$
 - 3: invoke the FCM algorithm and retrieve the list of cluster centers C ▷ Algorithm 2.9: $X \leftarrow Y$, $c \leftarrow c$, $maxIterations \leftarrow 100$, $m \leftarrow 1.5$, $maxDifference \leftarrow 0.00001$
-

Algorithm 4.4 Dimensionality reduction using the FCM algorithm – data transformation phase

Require: x – input element, C – list of cluster centers, AVG – list of attribute means, STD – list of attribute standard deviations

Ensure: element after attribute reduction

```

1:  $m \leftarrow 1.5$ 
2:  $x \leftarrow$  retrieve the standardized element            $\triangleright$  Algorithm 2.2:  $x \leftarrow x$ ,  $AVG \leftarrow AVG$ ,
    $STD \leftarrow STD$ 
3: let  $D$  be the list of distances between the element and the individual clusters
4: for  $i \leftarrow 1$  to  $|C|$  do
5:    $D[i] \leftarrow dist_E(x, C[i])$                       $\triangleright dist_E(\cdot)$  – Euclidean distance
6: end for
7: let  $M$  be the list of fuzzy memberships of the element to the clusters
8: for  $i \leftarrow 1$  to  $|C|$  do
9:    $value \leftarrow 0$ 
10:  for  $j \leftarrow 1$  to  $|C|$  do
11:     $temp \leftarrow D[i]/D[j]$ 
12:     $value \leftarrow value + pow(temp, 2/(m - 1))$         $\triangleright pow(\cdot)$  – exponentiation
13:  end for
14:   $M[i] \leftarrow 1/value$ 
15: end for
16: sort  $D$  in descending order
17: sort  $M$  in descending order
18: append subsequent elements of the list  $D$  to the end of the list  $M$ 
19: return  $M$ 

```

4.2 Selection of split values based on clustering using a minimal spanning tree algorithm

Another solution presented here is the introduction of an improved method for selecting split values in isolation trees within the enhanced Isolation Forest algorithm. The basic method works by randomly selecting both the attribute and the split value. It is easy to observe that the quality of the split depends on the selected value. Therefore, a random split value is not always optimal. In [164], the authors modified the probability distribution to strengthen the likelihood of selecting values in regions with lower densities. At this point, the idea is introduced to precisely select the split value. In other words, for a given attribute, we aim to find the value that most quickly isolate anomalies. This task can be performed using clustering. The problem of separation is addressed in [80], [81] by using centroid-based and medoid-based clustering algorithms. In this subsection, a new method is introduced, which involves selecting the split value based on clustering using a minimal spanning tree [55] (Minimal Spanning Tree Clustering for Isolation Forest, MSTCIF). A key feature of this algorithm is its high sensitivity to outliers. Therefore,

the selection of split values is based on separating the created clusters. It is worth emphasizing that the attribute selection remains random, which allows the method to retain its non-deterministic nature. Additionally, the element evaluation function is modified. The subsequent subsections describe in detail the phases of tree formation and element evaluation for the proposed algorithm.

4.2.1 Tree formation phase

The main concept of constructing an Isolation Forest, as presented in subsection 3.1, remains unchanged. Specifically, individual isolation trees are constructed from samples and then added to the isolation forest. The new proposal introduces a modification in the process of determining the split value. Instead of random selection, the split point is determined at the center of the segment that separates two clusters formed for the randomly selected attribute, using minimal spanning tree clustering (see Algorithm 2.6). This is a case of clustering performed for a single attribute. Therefore, an optimized algorithm, discussed in subsection 2.3.2, is used. The implementation scheme for the phase of constructing a single tree is shown in Algorithm 4.5. To ensure the best possible accuracy of the results, the tree height limit condition in equation (3.1) has been removed. The isolation nodes are created until all points are exhausted, resulting in a complete isolation tree. Additionally, a loop has been introduced to check whether the selected attribute actually separates the samples. If it does not, i.e., the samples have identical attribute values, another attribute is selected.

To illustrate the idea of determining the split point, Figure 4.4 shows example data, with the created clusters for the randomly selected attribute marked, and the split line indicated. For the prepared data, it can be seen that a random split point is not the best solution. Separation by determining the center between clusters more effectively isolates the groups, which, when repeatedly applied, allows for the isolation of anomalous elements in fewer steps.

Additionally, Figure 4.5 shows an example of isolating a single element using the MSTCIF method. The element is placed in the middle of the figure, which is a very difficult position for the IF method. In the proposed split determination method, the element is isolated in the third step of the algorithm. Thus, the element is placed at a low height in the isolation tree, making it highly likely to be classified as an anomalous element.

Now, let us consider the computational complexity of the first phase of the MSTCIF method. Let n denote the number of samples forming a single isolation tree, and t denote the number of trees in the forest. As with the base IF method, the first phase is executed only once. It can be shown that constructing a complete isolation tree involves creating $n - 1$ isolation nodes. Each node requires determining the split point through optimized minimal spanning tree clustering. These operations are repeated for each isolation tree in the forest. Therefore, the computational complexity of the algorithm for the tree formation phase is of the order of $O(tn^2 \log n)$. The algorithm has safeguards against situations where an attribute

does not divide the samples into two subsets. However, this situation is rare and unlikely, so it is not included in the final complexity of the first phase of the method. In the worst-case scenario, the result should be multiplied by the number of attributes.

Algorithm 4.5 Minimal Spanning Tree Clustering for Isolation Forest – single tree formation phase

Require: X – input data, **ref** $node$ – considered node

```

1: if  $|X| \leq 1$  then
2:   return
3: else
4:   create a node  $node$ 
5:   let  $A$  be the list of attributes in  $X$ 
6:    $counter \leftarrow 0$ 
7:    $i \leftarrow$  retrieve a random value from the range 1 to  $|A|$ 
8:   repeat
9:      $counter \leftarrow counter + 1$ 
10:     $i \leftarrow (i + 1) \bmod (|A| + 1)$ 
11:     $a \leftarrow A[i]$ 
12:     $d \leftarrow$  retrieve the split value for attribute  $a$  from the set  $X$   $\triangleright$  Algorithm 2.6:  $X \leftarrow X$ ,
     $a \leftarrow a$ 
13:     $node.splitAttribute \leftarrow a$ 
14:     $node.splitValue \leftarrow d$ 
15:    until  $d = X[1][a]$  and  $counter \neq |A|$ 
16:    let  $X_{left}$  be an empty set
17:    let  $X_{right}$  be an empty set
18:     $i \leftarrow 0$ 
19:    for each  $x \in X$  do
20:      if  $x[a] < d$  then
21:         $X_{left} \leftarrow X_{left} \cup x$ 
22:      else if  $x[a] > d$  then
23:         $X_{right} \leftarrow X_{right} \cup x$ 
24:      else if  $i \bmod 2 = 0$  then
25:         $X_{left} \leftarrow X_{left} \cup x$ 
26:      else
27:         $X_{right} \leftarrow X_{right} \cup x$ 
28:      end if
29:       $i \leftarrow i + 1$ 
30:    end for
31:    invoke the formation of the left subtree  $\triangleright$  Algorithm 4.5:  $X \leftarrow X_{left}$ ,  $node \leftarrow node.left$ 
32:    invoke the formation of the right subtree  $\triangleright$  Algorithm 4.5:  $X \leftarrow X_{right}$ ,
     $node \leftarrow node.right$ 
33: end if

```

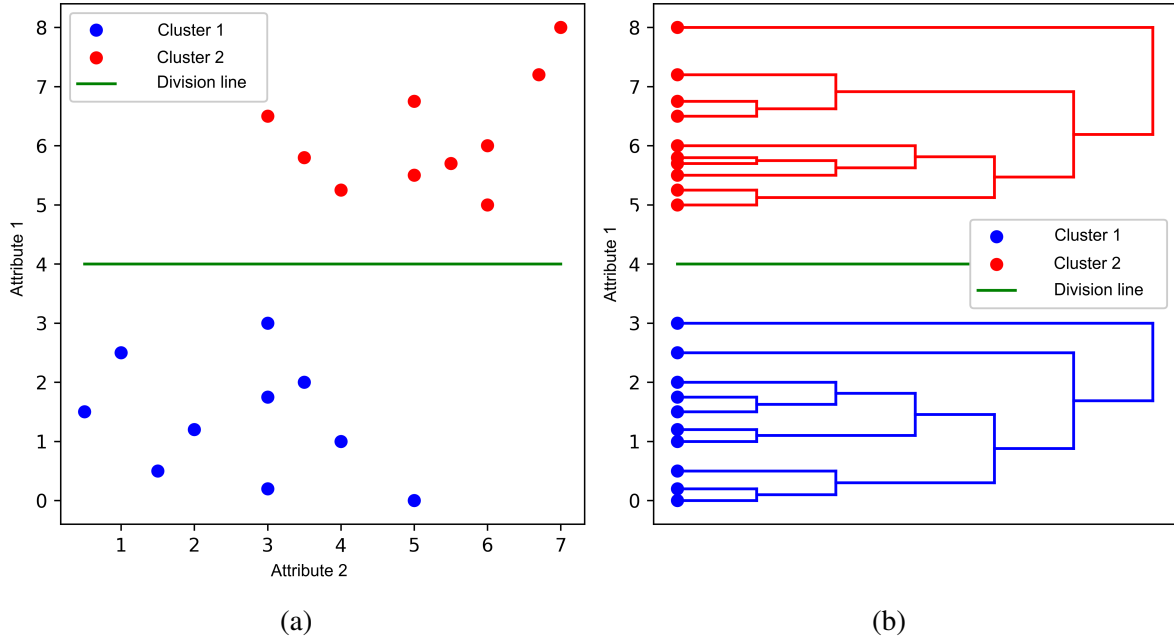


Figure 4.4: Example of split value determination using MST-based clustering. (a) Sample data with a marked split line. Colors indicate the cluster membership of each sample for a randomly selected attribute. (b) Projection of the sample data onto the selected attribute space with highlighted clusters and the determined split line. The dendrograms indicate successive merges in the base minimal spanning tree algorithm.

4.2.2 Element evaluation phase

The second phase, i.e., the element evaluation phase, is performed for each element being evaluated based on the isolation forest created in the first phase. This part implements improvements compared to the IF algorithm. Specifically, the evaluation function is expanded to include an additional component. This component is related to the precise selection of the split point. When the split point is not randomly chosen, it can be inferred that the selected position is the most suitable for separating elements. Therefore, if the evaluated point is far from the value of the split, it indicates a high probability that the element is anomalous. For this reason, the second component of the evaluation function is based on the distance of the element from the value of the last split. As for the first component related to the depth of the evaluated element in the isolation tree, it remains unchanged and is the same as in the original IF method. Since both components of the evaluation function need to have the same scale, they are normalized to allow for their combination via aggregation using the sum function.

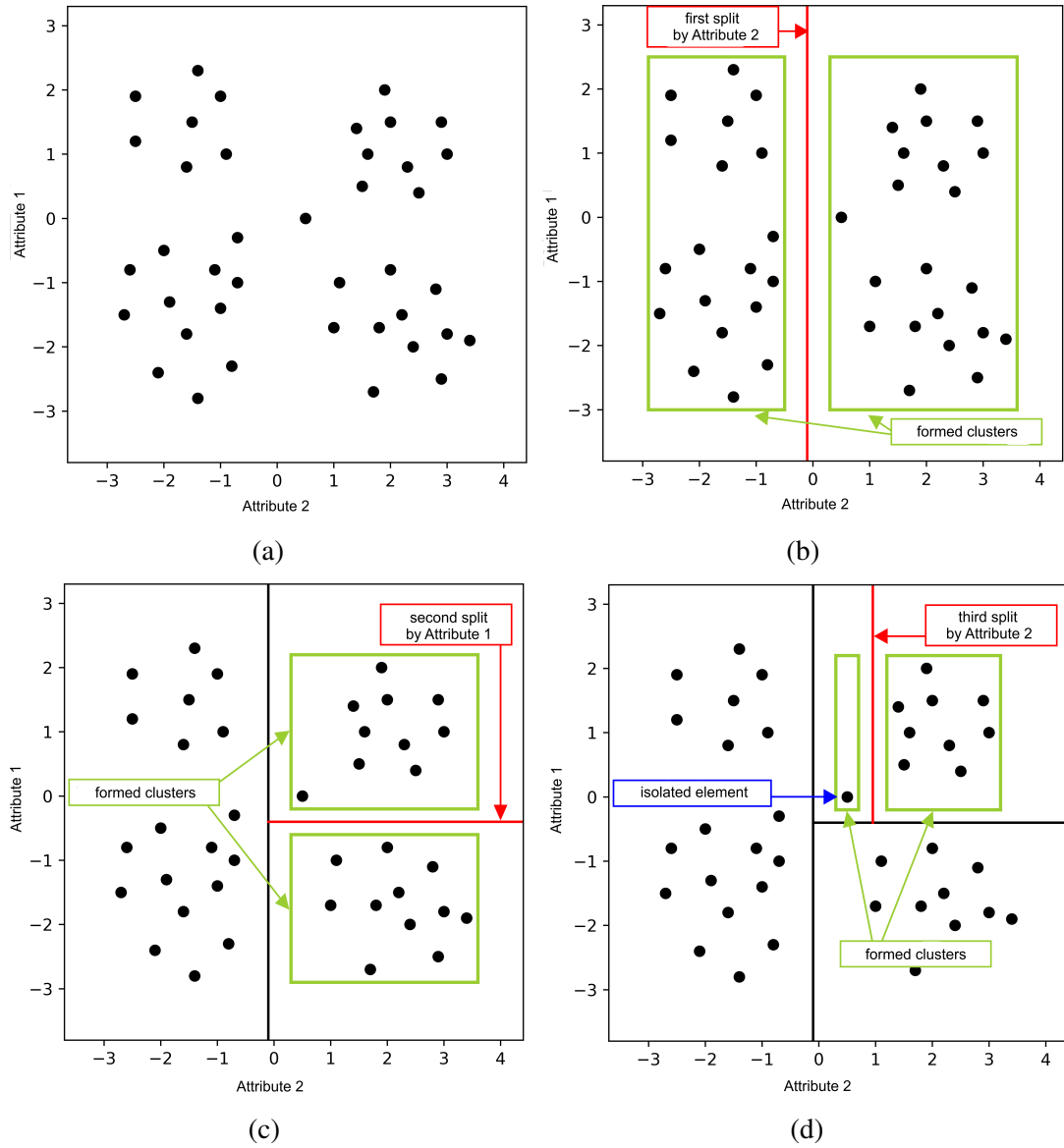


Figure 4.5: Example of element isolation using the MSTCIF method. (a) Sample data. (b) First split. (c) Second split. (d) Third split with the isolated element highlighted.

First, let us discuss the component related to the depth of the element in a single isolation tree. The forest prepared in the formation phase is structurally identical to the forest in the base method (see subsection 3.1). Additionally, the traversal scheme for individual trees by the evaluated element remains unchanged. Thus, the evaluated element traverses the tree until it reaches a leaf node. The depth of this node in the isolation tree is normalized. A new normalization technique for this factor is introduced here. In the base IF method, this factor is normalized by dividing by the estimated average path length of an unsuccessful search in a binary search tree, as shown in Equation (3.3). In the proposed method, the trees are fully formed, thanks to the removal of the tree height limit. This allows for the determination of the actual average depth of leaf nodes in the tree. This is particularly important for highly unbalanced trees, where the estimated

average length can differ significantly from the actual value. Therefore, a new normalization for the depth factor of the evaluated element is used, as described in equation:

$$h_{norm}(x) = \frac{h(x)}{h_{avg}} \quad (4.1)$$

where x is the evaluated element, h is the function returning the depth of element x in the isolation tree, and h_{avg} is the actual average depth of leaf nodes in the considered isolation tree.

The actual average depth of leaf nodes is calculated only once, when the isolation tree is created. The leaf nodes hold individual samples, so the actual average depth is given by the formula:

$$h_{avg} = \frac{1}{n} \sum_{i=1}^n h(x_i) \quad (4.2)$$

where n is the number of samples forming the isolation tree, x_i is the i -th sample, and h is the function returning the depth of element x_i in the isolation tree.

The second newly introduced component of the evaluation is the factor related to the distance between the element's attribute and the split value, which are analyzed in the last split node. This distance is normalized according to the formula:

$$d_{norm}(x) = \frac{d(x)}{\sigma} \quad (4.3)$$

where x is the evaluated element, d is the function returning the distance between the last checked attribute of the element and the split value in the last split node, and σ is the standard deviation of the attribute value in the samples forming the isolation tree. The parameter σ is also determined once the isolation tree is created. The final evaluation value is the sum of the two components:

$$score(x) = score_h(x) + score_d(x) \quad (4.4)$$

where the evaluation value for the component related to the depth is given by the formula:

$$score_h(x) = 2^{-\frac{1}{t} \sum_{i=1}^t h_{norm}^i(x)} \quad (4.5)$$

4.2 Selection of split values based on clustering using a minimal spanning tree algorithm

and the evaluation value for the component related to the distance is given by the formula:

$$score_d(x) = 1 - 2^{-\frac{1}{t} \sum_{i=1}^t d_{norm}^i(x)} \quad (4.6)$$

where t is the number of isolation trees forming the forest, h_{norm}^i is the normalization function (4.1) for the depth factor, invoked for the i -th tree of the forest, and d_{norm}^i is the normalization function (4.3) for the distance factor, invoked for the i -th tree of the forest.

Algorithm 4.7 shows the implementation scheme for calculating the normalized depth and distance factors for a single isolation tree. Additionally, the implementation scheme for the element evaluation phase for the entire isolation forest is shown in Algorithm 4.6.

Now, let us consider the computational complexity of the element evaluation phase. During the evaluation of an element, the isolation nodes are traversed starting from the root node. This operation is repeated for each tree in the isolation forest. The additional operation of calculating the distance between the element's attribute value and the split value stored in the last split node is performed only once. Therefore, it does not affect the final computational complexity. Consequently, the computational complexity of the element evaluation phase is the same as for the base IF method, i.e., $O(t \log n)$, where n is the number of samples forming a single isolation tree, and t is the number of trees in the forest.

Algorithm 4.6 Minimal Spanning Tree Clustering for Isolation Forest – evaluation phase

Require: x – evaluated element, $forest$ – list of isolation trees

Ensure: value of the evaluation function

```

1:  $sum_h \leftarrow 0$ 
2:  $sum_d \leftarrow 0$ 
3: for  $i \leftarrow 1$  to  $|forest|$  do
4:    $h \leftarrow 0$ 
5:    $d \leftarrow 0$ 
6:   calculate the path length and distance for  $forest[i]$             $\triangleright$  Algorithm 4.7:  $x \leftarrow x$ ,
      $iTree \leftarrow forest[i]$ ,  $h_{norm} \leftarrow h$ ,  $d_{norm} \leftarrow d$ 
7:    $sum_h \leftarrow sum_h + h$ 
8:    $sum_d \leftarrow sum_d + d$ 
9: end for
10:  $avg_h \leftarrow sum_h / |forest|$ 
11:  $avg_d \leftarrow sum_d / |forest|$ 
12:  $score_h \leftarrow 2^{-avg_h}$                                       $\triangleright$  Formula (4.5)
13:  $score_d \leftarrow 1 - 2^{-avg_d}$                                   $\triangleright$  Formula (4.6)
14: return  $score_h + score_d$                                         $\triangleright$  Formula (4.4)
```

Algorithm 4.7 Minimal Spanning Tree Clustering for Isolation Forest – calculation of the path length in the tree and distance from the split point

Require: x – evaluated element, $iTree$ – isolation tree

Ensure: **ref** h_{norm} – normalized path length of element x in the isolation tree, **ref** d_{norm} – normalized distance between element x and the split point

```

1: let  $node$  be the root of the isolation tree  $iTree$ 
2:  $h \leftarrow 0$ 
3: while  $node \neq null$  do
4:    $a \leftarrow node.splitAttribute$ ,  $d \leftarrow node.splitValue$ 
5:    $x_a \leftarrow x[a]$ 
6:    $temp \leftarrow |x_a - d|$ 
7:   let  $\sigma$  be the standard deviation of attribute  $a$  for the elements forming the isolation tree  $iTree$ 
8:   if  $x_a < d$  then
9:     if  $node.left = null$  then
10:      if  $\sigma = 0$  and  $temp = 0$  then
11:         $d_{norm} \leftarrow 0$ 
12:      else if  $\sigma = 0$  and  $temp \neq 0$  then
13:         $d_{norm} \leftarrow 1$ 
14:      else
15:         $d_{norm} \leftarrow \frac{temp}{\sigma}$  ▷ Formula (4.3)
16:      end if
17:    end if
18:     $node \leftarrow node.left$ 
19:  else
20:    if  $node.right = null$  then
21:      if  $\sigma = 0$  and  $temp = 0$  then
22:         $d_{norm} \leftarrow 0$ 
23:      else if  $\sigma = 0$  and  $temp \neq 0$  then
24:         $d_{norm} \leftarrow 1$ 
25:      else
26:         $d_{norm} \leftarrow \frac{temp}{\sigma}$  ▷ Formula (4.3)
27:      end if
28:    end if
29:     $node \leftarrow node.right$ 
30:  end if
31:   $h \leftarrow h + 1$ 
32: end while
33: let  $h_{avg}$  be the average leaf depth of the isolation tree  $iTree$ 
34:  $h_{norm} \leftarrow \frac{h}{h_{avg}}$  ▷ Formula (4.1)

```

4.3 Isolation tree construction based on the minimal spanning tree algorithm

The last technique presented as an enhancement to the IF algorithm is the introduction of a new construction method for isolation trees. In the research field, there is a lack of solutions that boldly change the concept of forest construction. There are concepts that focus on rotating the division space based on hyperplanes [62], [96] or hyperspheres [35], [36], divisions into more than two subsets [23], [75], or the use of structures resembling deep neural networks [176]. However, there is a lack of definitive solutions that would fundamentally change the way trees are constructed. Therefore, the research is directed toward finding an alternative way to construct isolation trees. Specifically, a new scheme is introduced in which the isolation tree construction is not based on divisions but on the merging of elements. In the original IF method, nodes are created based on a division starting from the root node. The new method starts merging elements from the leaves of the tree. Specifically, the node merging operation is carried out using a graph-based minimal spanning tree algorithm, described in more detail in subsection 2.2. This innovative isolation tree construction results in a tree structure very similar to the ones introduced in the base IF method. As a result, the depth of an element in such a constructed tree determines its anomaly. The following subsections describe in detail the innovative isolation tree construction technique and its enhancement through modifying the element evaluation function based on fuzzy inference modeling.

4.3.1 Minimal Spanning Tree-Based Isolation Forest

This subsection provides a detailed description of the solution that introduces a new isolation tree construction scheme based on the minimal spanning tree algorithm (Minimal Spanning Tree-Based Isolation Forest, MSTIF) [54]. The algorithm is executed in two phases.

4.3.1.1 Tree formation phase

The first phase is executed in the same manner as in the base IF method. Specifically, a forest containing t trees is created. Each tree is formed independently based on n samples drawn from the considered dataset. The improvements aimed at preparing the data for the new construction scheme begin at this point. First, to ensure comparable distances between attribute values of the samples, the attributes of the samples are standardized according to equation (2.1). The transformed elements obtained in this way are used in further calculations. Additionally, the parameters of the attribute standardization (mean values, standard deviations) are stored in memory. A detailed description of the standardization process is provided in subsection 2.1.

The innovative tree construction based on merging using the minimal spanning tree algorithm starts at this point. The process of creating the minimal spanning tree has been thoroughly discussed in subsection 2.2. For the MSTIF method, each standardized sample creates a single vertex in the graph. Additionally, a set of edges connecting the individual vertices is prepared. The edges are created for every combination of two vertices, where the weight of the edge is set as the Euclidean distance between those vertices. Then, the Kruskal algorithm is applied to the prepared vertices and edges (see Algorithm 2.4). This algorithm allows the creation of a minimal spanning tree that serves as the basis for building the isolation tree. To explain the process step by step, Figure 4.6 shows sample elements and the successive stages of the algorithm, illustrating the creation of the minimal spanning tree and the isolation tree. Specifically, each merging of two separate subtrees in Kruskal's algorithm leads to the creation of a node that connects these subtrees. This node is then added to the isolation tree. The merging process is repeated until the minimal spanning tree is formed, that is, until all the initial vertices representing the samples are connected into one tree. The implementation scheme for the formation phase of a single tree is presented in Algorithm 4.8.

To summarize the operation of the first phase of the MSTIF method, we analyze its computational complexity. Let n denote the number of elements, t the number of trees in the forest, and a the number of attributes of the elements. It is important to note that this phase is performed only once. In the case of forming a single tree, three stages can be distinguished: Preparing the elements, preparing the structure of vertices and edges, and building the minimal spanning tree. The preparation stage of the elements is executed in time $O(an)$. The second stage has a complexity of $O(an^2)$. The complexity of the last stage has been analyzed in subsection 2.2 and is $O(n^2 \log n)$. The above stages must be repeated for each tree in the forest. Finally, the overall computational complexity of the first phase can be described as the sum of the complexities of the individual stages, multiplied by the number of trees:

$$O(t(an + an^2 + n^2 \log n)) = O(tn^2(a + \log n)) \quad (4.7)$$

4.3 Isolation tree construction based on the minimal spanning tree algorithm

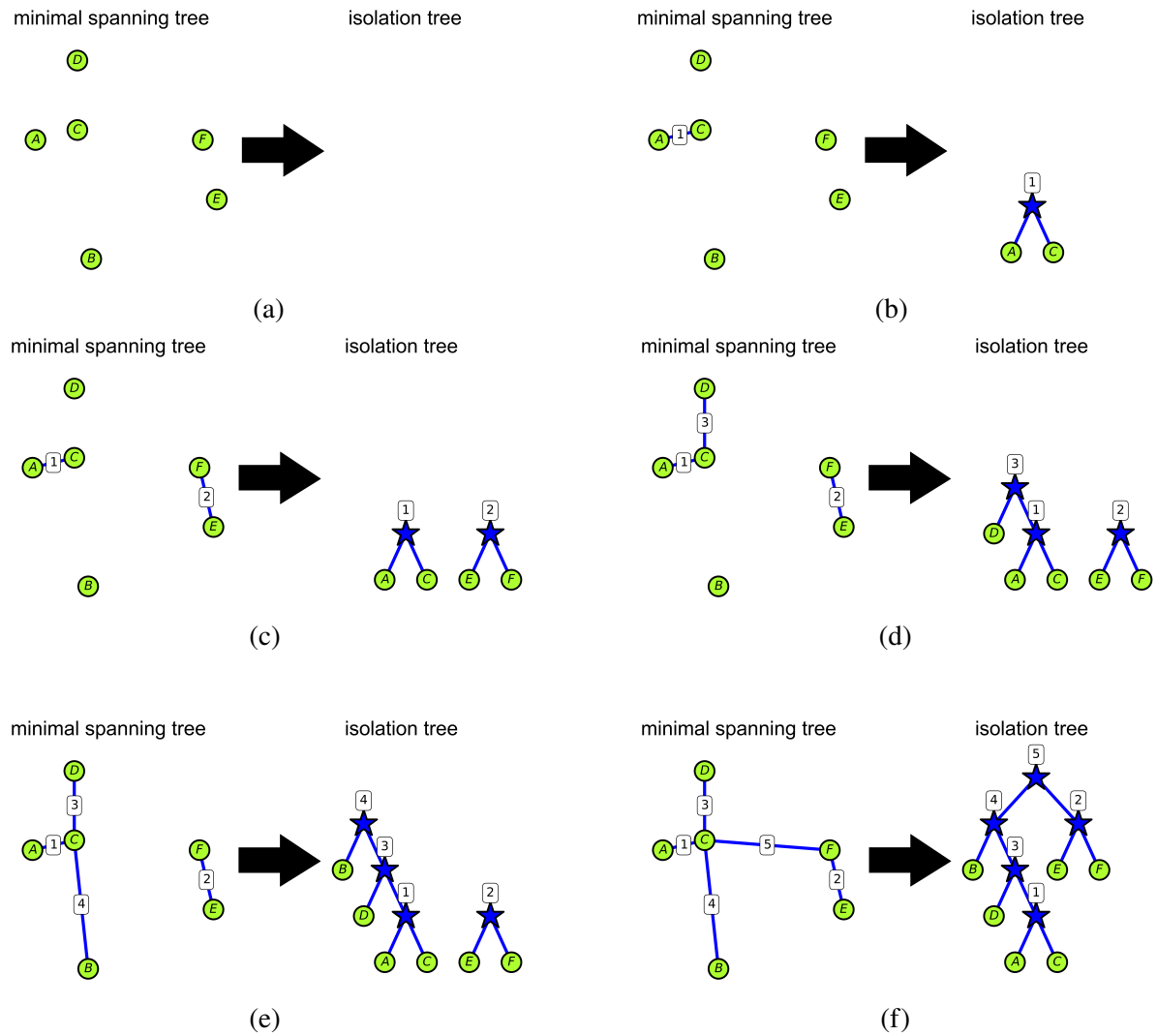


Figure 4.6: Example of the formation of an isolation tree using the MSTIF algorithm. (a)-(f) Subsequent steps of the algorithm. For each step, a diagram of the minimal spanning tree construction and the corresponding formation of the isolation tree based on it are presented.

Algorithm 4.8 Minimal Spanning Tree-Based Isolation Forest – single tree formation phase**Require:** X – input data

```

1: perform standardization of  $X$  and retrieve: standardized data  $Y$ , list of attribute means  $AVG$ ,
   and list of attribute standard deviations  $STD$  ▷ Algorithm 2.1:  $X \leftarrow X$ 
2:  $X \leftarrow Y$ 
3: let  $E$  be an empty list of edges
4: for  $i \leftarrow 1$  to  $|X|$  do
5:   for  $j \leftarrow i + 1$  to  $|X|$  do
6:     create an edge  $edge$ 
7:      $edge.source \leftarrow i$ 
8:      $edge.destination \leftarrow j$ 
9:      $edge.distance \leftarrow$  calculate the Euclidean distance between points  $X[i]$  and  $X[j]$ 
10:    add the edge  $edge$  to  $E$ 
11:   end for
12: end for
13: let  $V$  be an empty list of vertices
14: for  $i \leftarrow 1$  to  $|X|$  do
15:   create a node  $node$ 
16:    $node.left \leftarrow null$ 
17:    $node.right \leftarrow null$ 
18:    $node.parent \leftarrow null$ 
19:   add the node  $node$  to  $V$ 
20: end for
21:  $T \leftarrow$  create the minimum spanning tree on  $V$  with edges from  $E$  ▷ Algorithm 2.4:  $V \leftarrow V$ ,
    $E \leftarrow E$ 

```

4.3.1.2 Element evaluation phase

The second phase of the algorithm is conducted for each evaluated element. The evaluation is based on the forest prepared in the first phase. Given the significant change in tree construction, a modification is introduced to the element evaluation function. The proposed isolation trees are structurally very similar to the trees from the original IF method. Therefore, the anomaly of an element is determined by a factor related to the depth of the evaluated element in the isolation tree. Specifically, the nearest leaf node to the evaluated element is found. The depth of this leaf node is returned as the depth factor. The improvements do not end here. A second factor related to the distance of the evaluated element from the nearest leaf node in the isolation tree is also introduced. To provide intuition for these factors, Figure 4.7 shows a sample isolation tree and highlights the mentioned parameters.

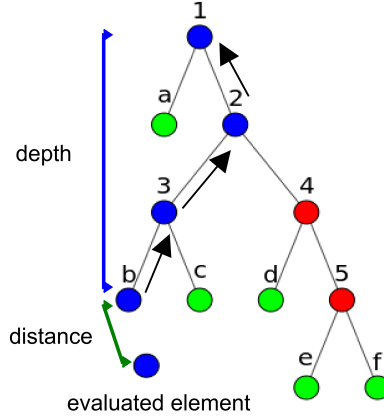


Figure 4.7: Example of element evaluation parameter determination for a single isolation tree using the MSTIF method. Leaf nodes are marked in green and labeled with letters. Internal nodes are marked in red and labeled with numbers. The traversal path of the evaluated element for determining the depth parameter is marked in blue. The depth parameter and distance parameter are indicated in the figure.

The evaluation of an element proceeds in several steps. The first step is to standardize the evaluated element according to the parameters stored in memory, such as the mean values and standard deviations of the attributes, as shown in Algorithm 2.2. Then, the nearest leaf node to the evaluated element is searched for. The depth of the found leaf node and the distance from the evaluated element to this node are computed.

The introduction of an additional factor related to the distance leads to a modification of the final evaluation function. The basic evaluation function for the depth factor is retained, while a new evaluation function is added for the distance factor. Ultimately, the anomaly score of the element is the sum of both evaluations, as described by the formula:

$$score(x, n) = score_h(x, n) + score_d(x) \quad (4.8)$$

where $score_h$ is the function from the base IF method defined by equation (3.5), with h_i for this function being the depth of the nearest leaf node for the i -th isolation tree, and

$$score_d(x) = 1 - 2^{-\frac{1}{t} \sum_{i=1}^t d_i(x)} \quad (4.9)$$

where x is the evaluated element, t is the number of trees in the isolation forest, and d_i is the distance of the element from the nearest leaf node for the i -th isolation tree.

The implementation scheme for the element evaluation phase for the entire isolation forest is presented in Algorithm 4.9. Additionally, the implementation scheme for calculating the depth and distance to the nearest element for a single isolation tree is presented in Algorithm 4.10.

Algorithm 4.9 Minimal Spanning Tree-Based Isolation Forest – evaluation phase

Require: x – evaluated element, $forest$ – list of isolation trees, n – number of samples
Ensure: value of the evaluation function

```

1:  $sum_h \leftarrow 0$ 
2:  $sum_d \leftarrow 0$ 
3: for  $i \leftarrow 1$  to  $|forest|$  do
4:    $h \leftarrow 0$ 
5:    $d \leftarrow 0$ 
6:   calculate the path length and distance for  $forest[i]$             $\triangleright$  Algorithm 4.10:  $x \leftarrow x$ ,
      $iTree \leftarrow forest[i]$ ,  $h \leftarrow h$ ,  $d \leftarrow d$ 
7:    $sum_h \leftarrow sum_h + h$ 
8:    $sum_d \leftarrow sum_d + d$ 
9: end for
10:  $avg_h \leftarrow sum_h / |forest|$ 
11:  $avg_d \leftarrow sum_d / |forest|$ 
12:  $score_h \leftarrow 2^{-\frac{avg_h}{c(n)}}$                                 $\triangleright$  Formula (3.5)
13:  $score_d \leftarrow 1 - 2^{-avg_d}$                                 $\triangleright$  Formula (4.9)
14: return  $score_h + score_d$                                       $\triangleright$  Formula (4.8)
```

Algorithm 4.10 Minimal Spanning Tree-Based Isolation Forest – calculation of the path length and distance to the nearest element in the isolation tree

Require: x – evaluated element, $iTree$ – isolation tree
Ensure: **ref** h – path length of element x in the isolation tree, **ref** d – distance between element x and the nearest element in the isolation tree

```

1:  $T \leftarrow$  retrieve the minimal spanning tree from  $iTree$ 
2:  $X \leftarrow$  retrieve the list of samples forming  $iTree$ 
3:  $AVG \leftarrow$  retrieve the list of attribute means from  $iTree$ 
4:  $STD \leftarrow$  retrieve the list of attribute standard deviations from  $iTree$ 
5:  $y \leftarrow$  retrieve the standardized element  $x$             $\triangleright$  Algorithm 2.2:  $x \leftarrow x$ ,  $AVG \leftarrow AVG$ ,
    $STD \leftarrow STD$ 
6:  $index \leftarrow 1$ 
7:  $d \leftarrow$  calculate the Euclidean distance between points  $x$  and  $X[index]$ 
8: for  $i \leftarrow 2$  to  $|X|$  do
9:    $temp \leftarrow$  calculate the Euclidean distance between points  $x$  and  $X[i]$ 
10:  if  $d > temp$  then
11:     $d \leftarrow temp$ ,  $index \leftarrow i$ 
12:  end if
13: end for
14:  $h \leftarrow 0$ 
15:  $node \leftarrow T[index]$ 
16: while  $node.parent \neq null$  do
17:    $node \leftarrow node.parent$ ,  $h \leftarrow h + 1$ 
18: end while
```

To summarize, the computational complexity of the element evaluation phase is analyzed. The calculation of the depth and distance parameters for a single isolation tree takes $O(an)$ time, where a is the number of attributes and n is the number of samples. These calculations need to be repeated for all t trees in the forest. Therefore, the computational complexity of the element evaluation phase is $O(tan)$.

4.3.2 Minimal Spanning Tree-Based Isolation Forest with Evaluation Function Built on the Fuzzy Rules in Takagi-Sugeno Model

The new technique for constructing isolation trees specified in the previous subsection introduces two components to the evaluation function. The first is related to the depth of the element in the tree, and the second is associated with the distance. The calculation of the evaluation function value is a simple aggregation of these two components using the sum function. This subsection presents a new method for calculating the final evaluation function value based on fuzzy rules in the Takagi-Sugeno model (Minimal Spanning Tree-Based Isolation Forest with Evaluation Function Built on the Fuzzy Rules in Takagi-Sugeno Model, MSTIF-TS) [52]. The Takagi-Sugeno fuzzy inference model has been discussed in more detail in subsection 2.6.

The tree formation phase and the structure of the trees remain unchanged and are identical to those described in subsection 4.3.1.1. Modifications are introduced in the second phase of the algorithm, i.e., the element evaluation phase. The calculation of the depth and the distance to the nearest leaf element in the isolation tree remains unchanged and is performed according to Algorithm 4.10. At this point, improvements are introduced by properly normalizing the depth and distance factors. Additionally, two normalization parameters for each tree in the forest must be computed. The first parameter is the actual average depth of the leaf nodes in the tree, calculated according to the formula:

$$h_{avg} = \frac{1}{n} \sum_{i=1}^n h(x_i) \quad (4.10)$$

where n is the number of samples forming the isolation tree, x_i is the i -th sample, and h is a function that returns the depth of element x_i in the isolation tree.

This parameter replaces the estimated average path length of an unsuccessful search in a binary search tree, as described in Equation (3.3). The second normalization parameter is related to the actual average edge length forming the minimal spanning tree, calculated according to the formula:

$$d_{avg} = \frac{1}{|E_T|} \sum_{e \in E_T} w \quad (4.11)$$

where E_T is the set of edges forming the minimal spanning tree, and w is the weight of edge e .

These parameters are computed in the tree formation phase to avoid extending the evaluation phase's computation time. The depth factor is then normalized according to the formula:

$$h_{norm}(x) = \frac{h(x)}{h_{avg}} \quad (4.12)$$

where x is the evaluated element, h is a function that returns the depth of element x in the isolation tree, and h_{avg} is the actual average depth of the leaf nodes in the considered isolation tree.

The rationale behind introducing this new normalization of the depth factor, based on the actual average depth value, is to improve accuracy. This is particularly important when isolation trees are highly unbalanced. In such cases, there may be a significant difference between the actual computed value and its estimated value. Additionally, the distance factor is normalized according to the formula:

$$d_{norm}(x) = \frac{d(x)}{d_{avg}} \quad (4.13)$$

where x is the evaluated element, d is a function returning the distance of element x from the nearest leaf node in the isolation tree, and d_{avg} is the actual average length of the edges forming the considered minimal spanning tree.

This factor is normalized similarly to the depth factor, using real values as the reference point. The average edge length of the minimal spanning tree is computed to better understand the structure of the tree and predict the length of edges that can be expected when adding a normal or anomalous element.

It is important to note that after normalization, both factors take the value of 1 when their values before normalization are equal to the average values of the corresponding parameters in the considered tree. In the final step, the evaluation value for both factors, based on all trees in the forest, is calculated. For the depth factor, it takes the form:

$$score_h(x) = 2^{-\frac{1}{t} \sum_{i=1}^t h_{norm}^i(x)} \quad (4.14)$$

where x is the evaluated element, t is the number of trees in the isolation forest, and h_{norm}^i is the normalization function for the depth factor, as shown in Equation (4.12), called for the i -th tree in the forest. For the distance factor, the evaluation value is described by the formula:

$$score_d(x) = 1 - 2^{-\frac{1}{t} \sum_{i=1}^t d_{norm}^i(x)} \quad (4.15)$$

where x is the evaluated element, t is the number of trees in the isolation forest, and d_{norm}^i is the normalization function for the distance factor, as shown in Equation (4.13), called for the i -th tree in the forest.

The MSTIF method originally aggregated these two evaluations using the sum function. However, it is easy to see that this approach may not be precise enough. Even if one factor indicates strong anomaly, it is necessary to amplify this evaluation or weaken it if the element exhibits strong normality. Therefore, a Takagi-Sugeno fuzzy inference model (TS) is introduced to combine both evaluations more accurately. A detailed description of the TS model is presented in subsection 2.6. The main factor for choosing the TS model is its fast response time. Additionally, the TS model has a continuous output, typically defined by a linear combination of linguistic variables, making the output easily interpretable for anomaly detection (i.e., binary classification).

The new model has two inputs and, according to the MISO (Multiple Input, Single Output) principle, one output. Two linguistic variables are introduced for each of these inputs. The linguistic variable corresponding to the depth evaluation function, as shown in Equation (4.14), is denoted as x_1 , and the variable corresponding to the distance evaluation function, as shown in Equation (4.15), is denoted as x_2 .

The next step in preparing the model structure is to define the fuzzy sets for the linguistic variables. For both variables, x_1 and x_2 , two fuzzy sets are introduced: One for “normal” and one for “anomaly.” To better understand the process of determining the boundaries and membership values for both sets, a brief characterization of key points is provided. The evaluation functions take values between 0 and 1, where 0 means no anomaly, and 1 indicates the presence of an anomaly. Moreover, several important points are considered. The first point has the value 0.5. If this value is achieved, it means the element is likely not an anomaly. If the value is below this threshold, the element is presumed to be normal. The second key point is 0.8. If the evaluation function exceeds this threshold, the element is likely an anomaly with high probability. Based on these considerations, the boundaries of the sets are determined as shown in Figure 4.8, using trapezoidal membership functions.

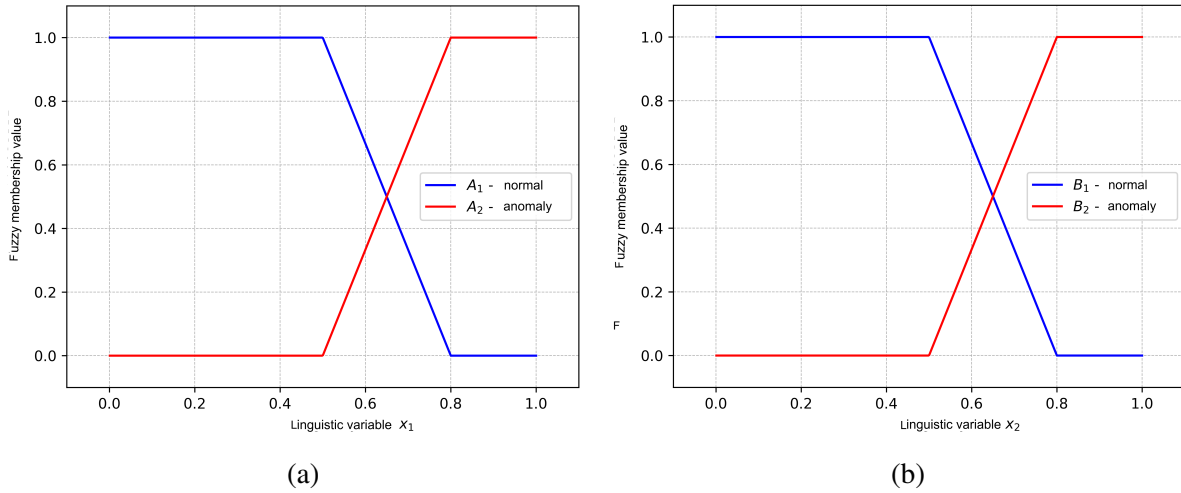


Figure 4.8: Fuzzy set membership functions for linguistic variables. (a) Linguistic variable x_1 . (b) Linguistic variable x_2 .

The last step in preparing the model is defining the rules used in the inference block. The rules are formulated according to the general form of the TS model rules presented in Equation (2.20). Four fuzzy rules are proposed, described as follows:

$$\begin{aligned}
 R_1 : & \text{ If } (x_1 \text{ is } A_1 \text{ and } x_2 \text{ is } B_1) \text{ Then } (y_1 = 1x_1 + 1x_2) \\
 R_2 : & \text{ If } (x_1 \text{ is } A_1 \text{ and } x_2 \text{ is } B_2) \text{ Then } (y_2 = 2x_1 + 3x_2) \\
 R_3 : & \text{ If } (x_1 \text{ is } A_2 \text{ and } x_2 \text{ is } B_1) \text{ Then } (y_3 = 3x_1 + 2x_2) \\
 R_4 : & \text{ If } (x_1 \text{ is } A_2 \text{ and } x_2 \text{ is } B_2) \text{ Then } (y_4 = 4x_1 + 4x_2)
 \end{aligned} \tag{4.16}$$

The rule functions are prepared as linear combinations of the linguistic variables. In the first case, both variables belong to the “normal” set. Therefore, the equation coefficients are set to 1. The second and third equations describe the situation in which one variable belongs to the “anomaly” set and the other to the “normal” set. As a result, the coefficients are increased to 2 (for “normal”) and 3 (for “anomaly”). The last rule describes the situation where both variables belong to the “anomaly” set. There is a high probability that the element is anomalous. Therefore, the equation coefficients are set to 4.

The degrees of applicability of the rules are given by the following formulas, see the general Formula (2.21),

$$\begin{aligned}
 w_1(x_1, x_2) &= \min(\mu_{A_1}(x_1), \mu_{B_1}(x_2)) \\
 w_2(x_1, x_2) &= \min(\mu_{A_1}(x_1), \mu_{B_2}(x_2)) \\
 w_3(x_1, x_2) &= \min(\mu_{A_2}(x_1), \mu_{B_1}(x_2)) \\
 w_4(x_1, x_2) &= \min(\mu_{A_2}(x_1), \mu_{B_2}(x_2))
 \end{aligned} \tag{4.17}$$

where x_1 and x_2 are the linguistic variables, and μ are the membership functions of the linguistic variables to the fuzzy sets, as characterized in Figure 4.8.

The final evaluation result of the model is the average of the conclusions of the rules, weighted by their degree of applicability. Specifically:

$$y(x_1, x_2) = \frac{\sum_{i=1}^4 w_i(x_1, x_2) y_i(x_1, x_2)}{\sum_{i=1}^4 w_i(x_1, x_2)} \tag{4.18}$$

The plot of the evaluation function value based on the linguistic variables x_1 and x_2 is shown in Figure 4.9. It is clearly visible that the value of the function increases as the values of the variables rise, with transitions between the fuzzy sets “normal” and “anomaly.”

Additionally, Algorithms 4.11 and 4.12 present the implementation schemes for calculating the value returned by the proposed model and the entire evaluation process for the isolation forest.

To conclude the introduction of the MSTIF-TS method, it should be noted that the forest formation phase has not been modified. Therefore, its computational complexity is of the order of $O(tn^2(a + \log n))$, see Equation (4.7). Moreover, the element evaluation phase has been extended with the Takagi-Sugeno fuzzy inference model and normalization. These operations have a computational complexity of $O(1)$. As a result, the complexity of the second phase of the algorithm is equal to the complexity of the second phase of the MSTIF solution, i.e., $O(tan)$.

Algorithm 4.11 Minimal Spanning Tree-Based Isolation Forest with Evaluation Function Built on the Fuzzy Rules in Takagi-Sugeno Model – calculation of the evaluation value for the fuzzy inference model

Require: x_1 – linguistic variable, x_2 – linguistic variable

Ensure: resulting model evaluation

- 1: let μ_{A_1} be the membership of x_1 to set A_1
 - 2: let μ_{A_2} be the membership of x_1 to set A_2
 - 3: let μ_{B_1} be the membership of x_2 to set B_1
 - 4: let μ_{B_2} be the membership of x_2 to set B_2
 - 5: $y_1 \leftarrow 1 \cdot x_1 + 1 \cdot x_2$
 - 6: $y_2 \leftarrow 2 \cdot x_1 + 3 \cdot x_2$
 - 7: $y_3 \leftarrow 3 \cdot x_1 + 2 \cdot x_2$
 - 8: $y_4 \leftarrow 4 \cdot x_1 + 4 \cdot x_2$
 - 9: $w_1 \leftarrow \min(\mu_{A_1}, \mu_{B_1})$
 - 10: $w_2 \leftarrow \min(\mu_{A_1}, \mu_{B_2})$
 - 11: $w_3 \leftarrow \min(\mu_{A_2}, \mu_{B_1})$
 - 12: $w_4 \leftarrow \min(\mu_{A_2}, \mu_{B_2})$
 - 13: **return** $(w_1 \cdot y_1 + w_2 \cdot y_2 + w_3 \cdot y_3 + w_4 \cdot y_4) / (w_1 + w_2 + w_3 + w_4)$ ▷ Formula (4.18)
-

Algorithm 4.12 Minimal Spanning Tree-Based Isolation Forest with Evaluation Function Built on the Fuzzy Rules in Takagi-Sugeno Model – evaluation phase

Require: x – evaluated element, *forest* – list of isolation trees

Ensure: value of the evaluation function

- 1: $sum_h \leftarrow 0$
 - 2: $sum_d \leftarrow 0$
 - 3: **for** $i \leftarrow 1$ to $|forest|$ **do**
 - 4: $h \leftarrow 0$
 - 5: $d \leftarrow 0$
 - 6: $h_{avg} \leftarrow$ retrieve the average leaf depth from $forest[i]$
 - 7: $d_{avg} \leftarrow$ retrieve the average edge length from $forest[i]$
 - 8: calculate the path length and distance for $forest[i]$ ▷ Algorithm 4.10: $x \leftarrow x$,
 - $iTree \leftarrow forest[i], h \leftarrow h, d \leftarrow d$
 - 9: $h_{norm} \leftarrow h / h_{avg}$ ▷ Formula (4.12)
 - 10: $sum_h \leftarrow sum_h + h_{norm}$
 - 11: $d_{norm} \leftarrow d / d_{avg}$ ▷ Formula (4.13)
 - 12: $sum_d \leftarrow sum_d + d_{norm}$
 - 13: **end for**
 - 14: $avg_h \leftarrow sum_h / |forest|$
 - 15: $avg_d \leftarrow sum_d / |forest|$
 - 16: $score_h \leftarrow 2^{-avg_h}$ ▷ Formula (4.14)
 - 17: $score_d \leftarrow 1 - 2^{-avg_d}$ ▷ Formula (4.15)
 - 18: **return** calculate the resulting value of the fuzzy inference model ▷ Algorithm 4.11:
 - $x_1 \leftarrow score_h, x_2 \leftarrow score_d$
-

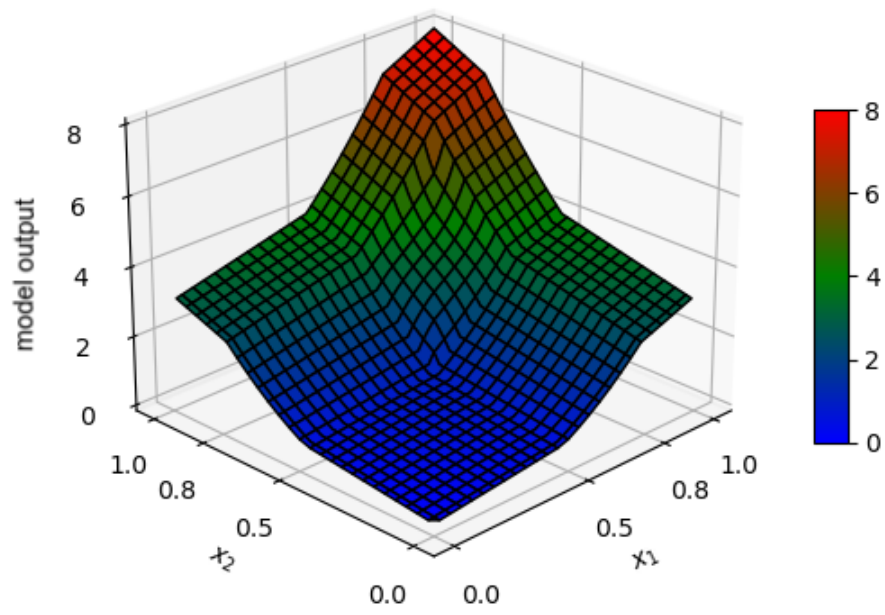


Figure 4.9: Output value plot of the proposed TS model depending on the values of linguistic variables.

Chapter 5

Anomaly detection in real-world datasets

One of the key aspects of this dissertation is the analysis of potential applications of the methods introduced in Chapter 4. Therefore, detailed experiments are conducted on real datasets. The following subsections discuss the datasets used and the results of testing the properties of individual solutions. The complete analysis is complemented by comparisons with competitive anomaly detection solutions, such as: Isolation Forest (IF) [106], [107], Local Outlier Factor (LOF) [22], One-Class Support Vector Machine Detector (OCSVM) [147], k -Nearest Neighbors Detector (KNN) [8], [137], and Principal Component Analysis Outlier Detector (PCA) [3], [152].

5.1 Characteristics of the datasets

The analysis begins with the presentation and characterization of the datasets under study. To enable comparison of the results, data from the publicly available “ODDS” (Outlier Detection DataSets) repository [139] are used. The individual observation sets come from medical, industrial, environmental, computer science, speech recognition, handwriting recognition, and chemical analysis data. Here are references to the original sources of the datasets used: [1], [4], [14], [83], [93], [106], [120], [140], [146], [159], [162], [163], [173], [181], [197].

Information about the studied anomaly detection datasets is presented in Table 5.1. All observations have labels indicating whether the observation is normal or anomalous. In the studies, unsupervised anomaly detection methods are analyzed. Therefore, the labels are not used in any phase of the algorithm’s operation and are only used to assess the quality of the prepared solutions.

Table 5.1: Characteristics of the anomaly detection datasets.

Dataset name	Number of observations	Number of attributes	Number of anomalies
Annthyroid	7200	6	534 (7.42%)
Arrhythmia	452	274	66 (14.60%)
BreastW	683	9	239 (34.99%)
Cardio	1831	21	176 (9.61%)
ForestCover	286048	10	2747 (0.96%)
Glass	214	9	9 (4.21%)
Http	567498	3	2211 (0.39%)
Ionosphere	351	33	126 (35.90%)
Letter	1600	32	100 (6.25%)
Lympho	148	18	6 (4.05%)
Mammography	11183	6	260 (2.32%)
Mnist	7603	100	700 (9.21%)
Musk	3062	166	97 (3.17%)
Optdigits	5216	64	150 (2.88%)
Pendigits	6870	16	156 (2.27%)
Pima	768	8	268 (34.90%)
Satellite	6435	36	2036 (31.64%)
Satimage-2	5803	36	71 (1.22%)
Shuttle	49097	9	3511 (7.15%)
Smtpt	95156	3	30 (0.03%)
Speech	3686	400	61 (1.65%)
Thyroid	3772	6	93 (2.47%)
Vertebral	240	6	30 (12.50%)
Vowels	1456	12	50 (3.43%)
WBC	378	30	21 (5.56%)
Wine	129	13	10 (7.75%)

5.2 Threshold-independent evaluation metrics

The first part of the evaluation presents results based on metrics that are independent of the threshold selection used by the detection method. Specifically, the assessment is carried out using the area under the receiver operating characteristic curve (ROC AUC) and the area under the precision-recall curve (PR AUC).

5.2.1 Results

The ROC AUC measurement results are presented in Tables 5.2 and 5.3, while the results for the PR AUC metric are shown in Tables 5.4 and 5.5. For non-deterministic methods, such as those based on the Isolation Forest, all experiments are repeated 100 times. The presented outcomes represent the average results along with standard deviation values. For each tested algorithm, the mean and median scores across all datasets are also included.

5.2.2 Analysis of the results

To facilitate comparison and simplify the analysis of the raw statistics presented in the previous subsection, the results have been visualized using heatmaps and plots. In these visualizations, better results are highlighted in green, while worse outcomes are marked in red. The analysis includes the following aspects: mean values (Figures 5.1, 5.2), median values (Figures 5.3, 5.4), differences between the number of better and worse results (Figures 5.5, 5.6), and the number of best results among all methods considered (Figures 5.7, 5.8). This approach allows for a rapid evaluation and comparison of the effectiveness of the various methods. Furthermore, a more precise comparison of the obtained values is performed using statistical tests. The p-values resulting from pairwise Wilcoxon tests between algorithms are provided in Tables 5.6, 5.7, 5.8, and 5.9. The alternative hypothesis assumed in the tests is that the distribution of differences between samples is statistically greater than a symmetric distribution. A significance level of $\alpha = 0.05$ is adopted.

Table 5.2: ROC AUC values for the new methods, along with standard deviations given in parentheses.

Dataset name	DRIF [%]	FDRIF [%]	MSTCIF [%]	MSTIF [%]	MSTIF-TS [%]
Annthroid	55.4 (0.6)	58.9 (0.6)	82.6 (0.9)	80.4 (0.6)	77.6 (0.6)
Arrhythmia	80.1 (1.0)	65.9 (2.0)	79.2 (0.7)	77.6 (0.2)	77.8 (0.2)
BreastW	96.5 (0.4)	95.0 (0.4)	97.6 (0.1)	98.0 (0.0)	98.0 (0.0)
Cardio	93.6 (0.6)	75.9 (2.8)	91.1 (0.7)	77.5 (0.5)	84.0 (0.6)
ForestCover	92.9 (0.8)	83.2 (1.3)	93.9 (0.8)	89.3 (0.4)	86.6 (0.5)
Glass	78.4 (0.5)	85.7 (1.0)	76.0 (1.1)	77.6 (0.3)	78.0 (0.4)
Http	99.5 (0.0)	99.4 (0.0)	100.0 (0.0)	100.0 (0.0)	100.0 (0.0)
Ionosphere	67.7 (0.8)	74.4 (1.6)	92.9 (0.2)	93.5 (0.1)	93.8 (0.1)
Letter	58.4 (1.6)	64.3 (1.1)	77.9 (1.2)	82.1 (0.2)	83.6 (0.2)
Lympho	93.4 (2.1)	97.6 (0.7)	95.8 (1.5)	94.2 (0.2)	94.5 (0.3)
Mammography	78.8 (0.9)	84.8 (0.4)	80.8 (0.6)	81.5 (0.6)	81.2 (0.8)
Mnist	71.9 (3.3)	68.4 (4.6)	85.9 (0.8)	88.0 (0.2)	87.8 (0.3)
Musk	100.0 (0.0)	100.0 (0.0)	99.7 (0.2)	100.0 (0.0)	99.8 (0.0)
Optdigits	32.3 (3.8)	33.0 (4.6)	57.3 (3.3)	76.7 (2.8)	83.3 (0.9)
Pendigits	89.1 (1.5)	82.0 (1.5)	94.5 (0.8)	96.4 (0.5)	95.7 (0.4)
Pima	69.2 (1.0)	55.9 (1.4)	70.3 (0.4)	71.1 (0.1)	73.3 (0.1)
Satellite	69.0 (1.8)	75.4 (0.7)	76.9 (0.5)	77.7 (0.2)	77.9 (0.2)
Satimage-2	99.4 (0.2)	99.1 (0.1)	99.8 (0.0)	99.9 (0.0)	99.8 (0.0)
Shuttle	98.6 (0.1)	84.9 (0.7)	97.9 (0.2)	99.1 (0.1)	97.3 (0.1)
Smtpt	80.1 (1.9)	88.8 (1.1)	93.5 (0.3)	94.4 (0.1)	93.3 (0.1)
Speech	47.7 (2.4)	52.1 (2.1)	47.2 (0.8)	36.5 (1.2)	47.7 (0.3)
Thyroid	94.1 (0.6)	92.6 (0.5)	96.9 (0.2)	97.2 (0.1)	97.3 (0.1)
Vertebral	33.4 (2.0)	36.8 (1.1)	29.5 (0.9)	32.9 (0.4)	33.3 (0.4)
Vowels	74.6 (1.2)	80.5 (1.1)	86.2 (0.8)	97.2 (0.1)	96.8 (0.1)
WBC	93.0 (0.8)	80.3 (1.6)	94.4 (0.3)	93.5 (0.2)	93.1 (0.2)
Wine	92.1 (0.6)	75.0 (1.5)	87.6 (1.2)	93.5 (0.5)	93.4 (0.5)
Average:	78.4 (1.2)	76.5 (1.3)	84.1 (0.7)	84.8 (0.4)	85.6 (0.3)
Median:	80.1 (0.9)	80.4 (1.1)	89.4 (0.7)	91.4 (0.2)	90.5 (0.2)

Table 5.3: ROC AUC values for the reference methods, along with standard deviations given in parentheses.

Dataset name	IF [%]	LOF [%]	OCSVM [%]	KNN [%]	PCA [%]
Annthroid	82.0 (1.5)	73.9	54.7	75.6	67.6
Arrhythmia	80.0 (1.4)	78.9	46.8	80.0	77.5
BreastW	98.9 (0.1)	38.0	97.9	97.5	95.7
Cardio	92.1 (1.2)	54.6	93.5	68.6	95.0
ForestCover	87.6 (2.8)	55.5	20.4	80.1	93.4
Glass	69.0 (2.0)	83.3	59.3	86.4	60.4
Http	99.7 (0.1)	38.3	99.4	25.3	99.6
Ionosphere	84.5 (0.6)	87.2	76.2	93.0	79.5
Letter	63.0 (1.8)	90.0	89.0	91.1	52.9
Lympho	99.4 (0.5)	98.6	97.8	97.4	98.5
Mammography	87.3 (0.6)	71.9	87.2	83.8	88.2
Mnist	80.9 (2.3)	64.5	50.0	79.4	85.0
Musk	99.9 (0.3)	41.2	57.3	30.4	100.0
Optdigits	69.2 (5.3)	58.8	50.4	43.6	51.4
Pendigits	95.0 (0.9)	47.9	93.5	70.9	93.8
Pima	67.4 (1.3)	53.8	53.9	60.8	64.8
Satellite	70.4 (1.7)	53.9	48.7	67.0	60.1
Satimage-2	99.3 (0.1)	53.3	42.1	93.0	97.7
Shuttle	99.7 (0.1)	55.5	72.5	70.4	99.0
Smtpt	89.4 (0.8)	90.3	84.1	89.9	85.8
Speech	47.5 (1.9)	50.9	46.7	49.3	46.9
Thyroid	97.8 (0.3)	80.2	84.4	94.9	95.6
Vertebral	36.4 (2.5)	49.5	62.6	33.1	37.7
Vowels	75.7 (2.7)	94.3	77.8	97.7	60.4
WBC	93.9 (0.8)	93.0	93.8	95.0	93.5
Wine	79.9 (3.9)	99.7	47.5	99.6	81.6
Average:	82.5 (1.4)	67.6	68.8	75.1	79.3
Median:	85.9 (1.3)	61.7	67.5	80.1	85.4

Table 5.4: PR AUC values for the new methods, along with standard deviations given in parentheses.

Dataset name	DRIF [%]	FDRIF [%]	MSTCIF [%]	MSTIF [%]	MSTIF-TS [%]
Annthroid	20.4 (0.7)	17.1 (0.3)	27.7 (1.2)	29.5 (0.9)	27.7 (1.0)
Arrhythmia	50.1 (1.7)	27.2 (1.7)	42.5 (1.9)	41.2 (1.0)	39.8 (0.8)
BreastW	87.7 (1.5)	85.4 (0.8)	92.6 (0.5)	95.0 (0.2)	95.0 (0.2)
Cardio	60.1 (2.2)	31.1 (1.9)	45.6 (2.1)	45.4 (0.6)	45.3 (1.0)
ForestCover	7.1 (0.7)	3.8 (0.3)	9.4 (1.3)	4.6 (0.2)	4.2 (0.2)
Glass	12.7 (1.3)	18.7 (1.7)	10.9 (3.1)	9.7 (0.2)	9.7 (0.3)
Http	22.0 (0.7)	19.9 (0.7)	95.9 (3.4)	97.5 (6.6)	92.3 (10.4)
Ionosphere	45.7 (0.8)	59.9 (2.3)	90.8 (0.3)	93.8 (0.0)	93.9 (0.1)
Letter	11.9 (1.0)	10.5 (0.6)	16.1 (0.9)	16.8 (0.3)	19.9 (0.4)
Lympho	56.7 (16.5)	81.0 (4.7)	71.7 (4.6)	72.8 (0.2)	73.2 (0.3)
Mammography	10.6 (1.1)	13.5 (0.7)	13.9 (0.7)	14.4 (0.6)	14.0 (0.6)
Mnist	26.2 (1.8)	24.7 (2.9)	36.5 (2.3)	36.0 (0.8)	37.2 (1.0)
Musk	100.0 (0.1)	99.5 (0.7)	92.9 (4.1)	100.0 (0.0)	94.1 (1.4)
Optdigits	1.9 (0.1)	1.9 (0.1)	3.1 (0.2)	5.8 (0.7)	7.6 (0.4)
Pendigits	14.1 (1.9)	11.5 (1.0)	22.7 (2.6)	28.5 (2.9)	24.4 (1.9)
Pima	49.7 (1.0)	40.0 (1.0)	49.7 (0.4)	52.6 (0.3)	54.6 (0.3)
Satellite	63.8 (1.1)	57.0 (0.9)	69.8 (0.6)	71.3 (0.3)	70.6 (0.2)
Satimage-2	91.1 (3.9)	95.6 (0.3)	93.5 (1.8)	97.3 (0.2)	89.9 (2.8)
Shuttle	86.7 (3.6)	35.5 (1.4)	64.2 (2.0)	83.5 (1.1)	62.8 (1.2)
Smtpt	1.0 (0.9)	21.1 (7.6)	36.1 (7.5)	7.5 (2.9)	0.7 (0.1)
Speech	2.1 (0.3)	1.9 (0.2)	1.6 (0.1)	1.2 (0.0)	1.7 (0.0)
Thyroid	54.5 (2.2)	34.2 (1.2)	44.6 (2.7)	52.3 (1.8)	49.8 (2.0)
Vertebral	8.7 (0.2)	9.2 (0.1)	8.2 (0.1)	8.6 (0.0)	8.6 (0.0)
Vowels	15.1 (1.8)	8.5 (0.5)	21.1 (2.5)	60.7 (2.3)	61.6 (2.0)
WBC	62.3 (2.7)	43.3 (2.6)	48.0 (2.5)	53.2 (2.7)	45.1 (0.9)
Wine	31.4 (1.5)	14.0 (1.3)	23.5 (1.7)	35.5 (1.6)	35.4 (1.4)
Average:	38.2 (2.0)	33.3 (1.4)	43.6 (2.0)	46.7 (1.1)	44.6 (1.2)
Median:	28.8 (1.2)	22.9 (0.9)	39.5 (1.9)	43.3 (0.6)	42.5 (0.7)

Table 5.5: PR AUC values for the reference methods, along with standard deviations given in parentheses.

Dataset name	IF [%]	LOF [%]	OCSVM [%]	KNN [%]	PCA [%]
Annth thyroid	31.0 (2.4)	20.4	10.4	21.8	19.1
Arrhythmia	45.7 (2.6)	40.9	10.6	46.5	38.8
BreastW	97.5 (0.4)	27.3	94.0	93.1	95.8
Cardio	55.8 (3.8)	15.2	52.9	28.1	60.8
ForestCover	5.4 (1.8)	1.7	0.6	6.3	7.3
Glass	6.9 (0.6)	12.9	6.8	13.3	5.6
Http	33.6 (10.0)	0.4	21.6	0.5	24.8
Ionosphere	79.3 (0.8)	82.5	75.2	93.3	74.6
Letter	8.7 (0.5)	48.4	38.1	38.5	7.3
Lympho	91.5 (5.0)	85.8	46.9	69.4	82.5
Mammography	26.6 (3.4)	8.8	18.1	15.5	19.3
Mnist	28.6 (3.7)	18.4	54.6	33.7	38.2
Musk	97.8 (6.6)	2.6	4.1	2.2	100.0
Optdigits	4.6 (0.9)	6.0	2.8	2.8	2.6
Pendigits	26.8 (4.0)	3.4	20.3	6.1	22.3
Pima	50.9 (1.3)	36.7	37.0	45.6	47.6
Satellite	66.3 (1.6)	37.1	30.8	53.1	60.6
Satimage-2	92.0 (1.0)	3.0	1.0	33.9	87.2
Shuttle	97.9 (0.5)	10.2	19.5	18.1	91.5
Smt p	0.4 (0.0)	0.5	38.3	27.8	32.8
Speech	1.8 (0.4)	2.2	1.7	1.8	1.7
Thyroid	54.2 (6.5)	10.7	14.6	24.4	35.4
Vertebral	9.1 (0.3)	11.5	18.0	8.6	9.2
Vowels	13.6 (3.8)	30.0	18.3	53.6	6.1
WBC	59.8 (3.4)	39.8	54.5	47.4	54.1
Wine	20.2 (3.8)	96.8	7.9	95.6	22.2
Average:	42.5 (2.7)	25.1	26.9	33.9	40.3
Median:	32.3 (2.1)	14.0	18.9	27.9	34.1

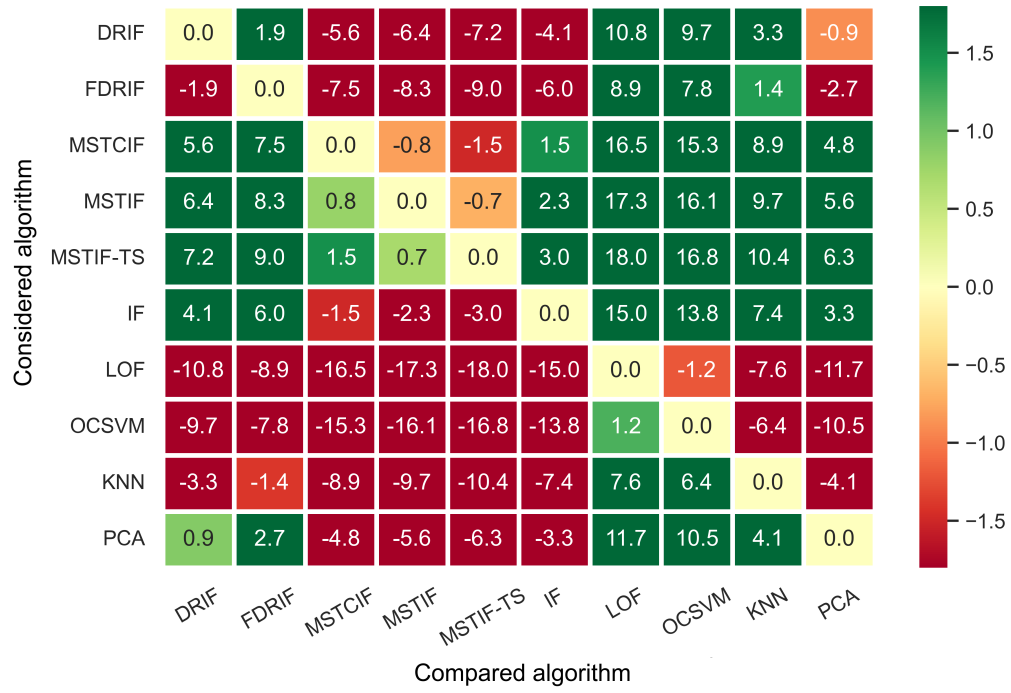


Figure 5.1: Heatmap presenting the difference in the average ROC AUC scores (in percentage) for all datasets, determined between the compared algorithms.

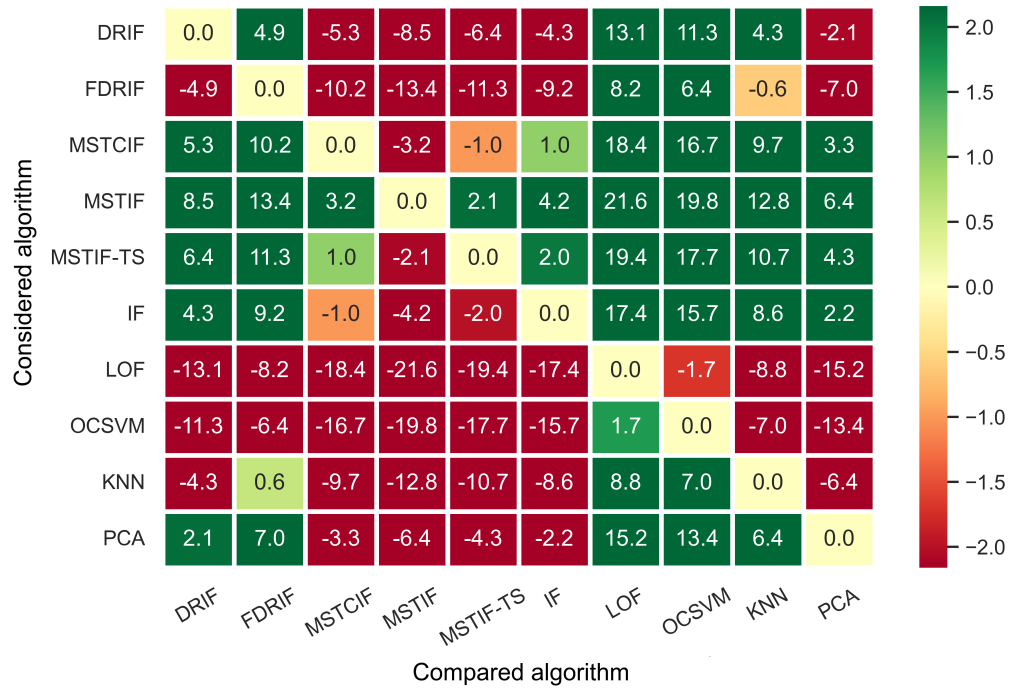


Figure 5.2: Heatmap presenting the difference in average PR AUC scores (in percentage) for all datasets, determined between the compared algorithms.

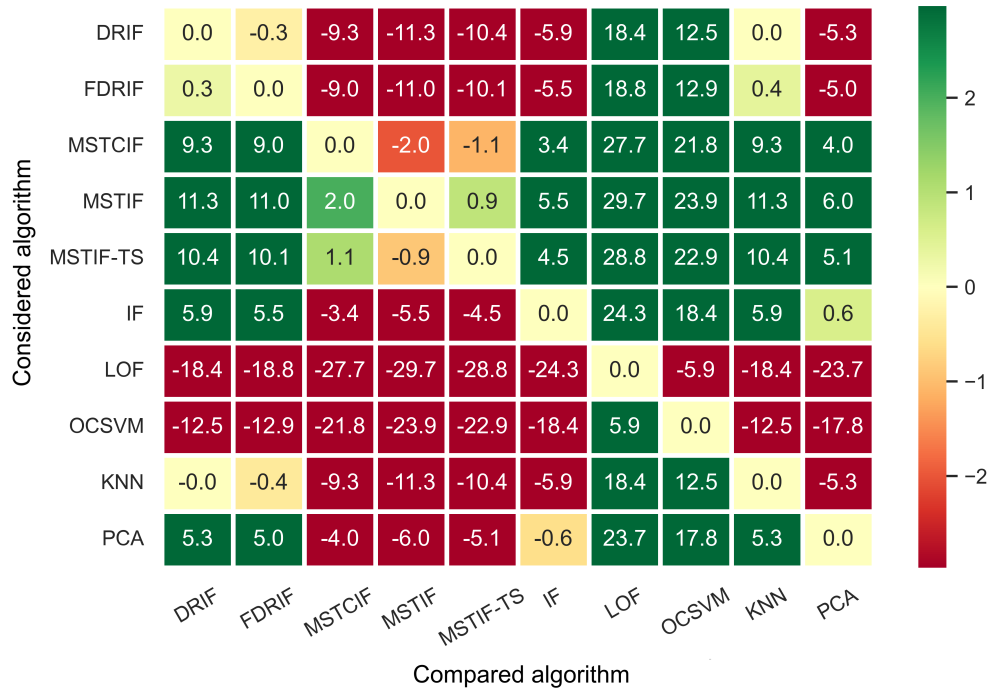


Figure 5.3: Heatmap presenting the difference in the median ROC AUC scores (in percentage) across all datasets, determined between the compared algorithms.

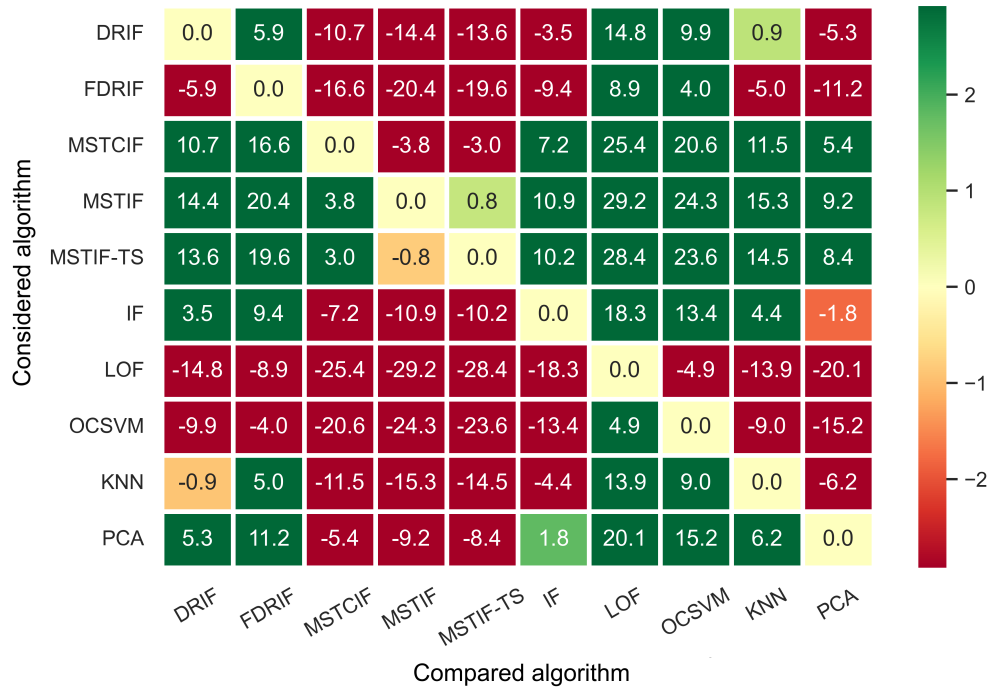


Figure 5.4: Heatmap presenting the difference in median PR AUC scores (in percentage) for all datasets, determined between the compared algorithms.

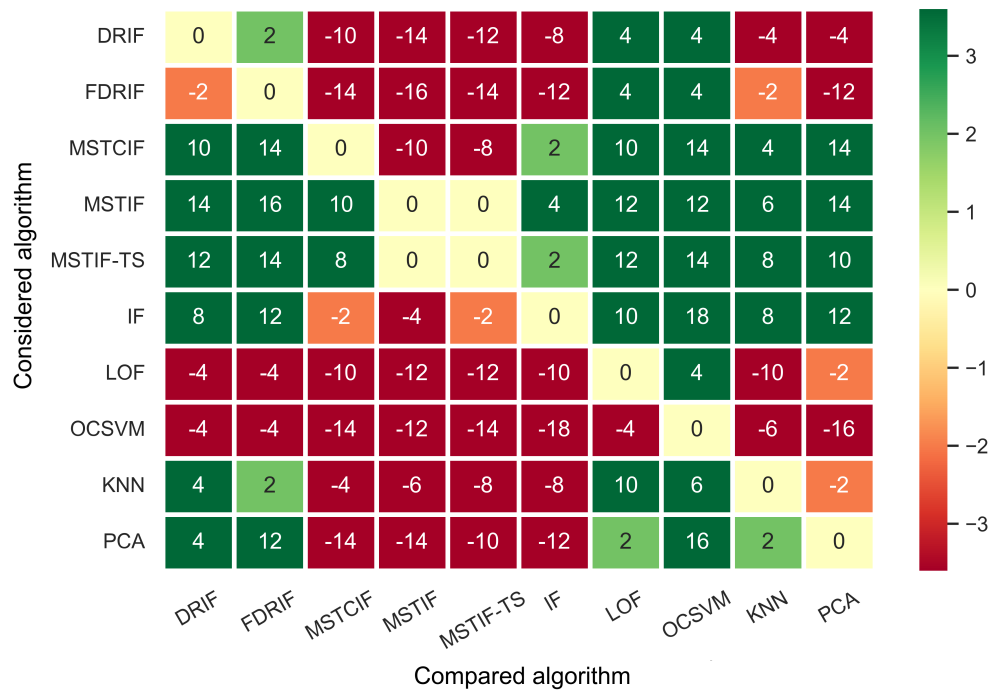


Figure 5.5: Heatmap presenting the differences in the number of better and worse ROC AUC scores determined between the compared algorithms.

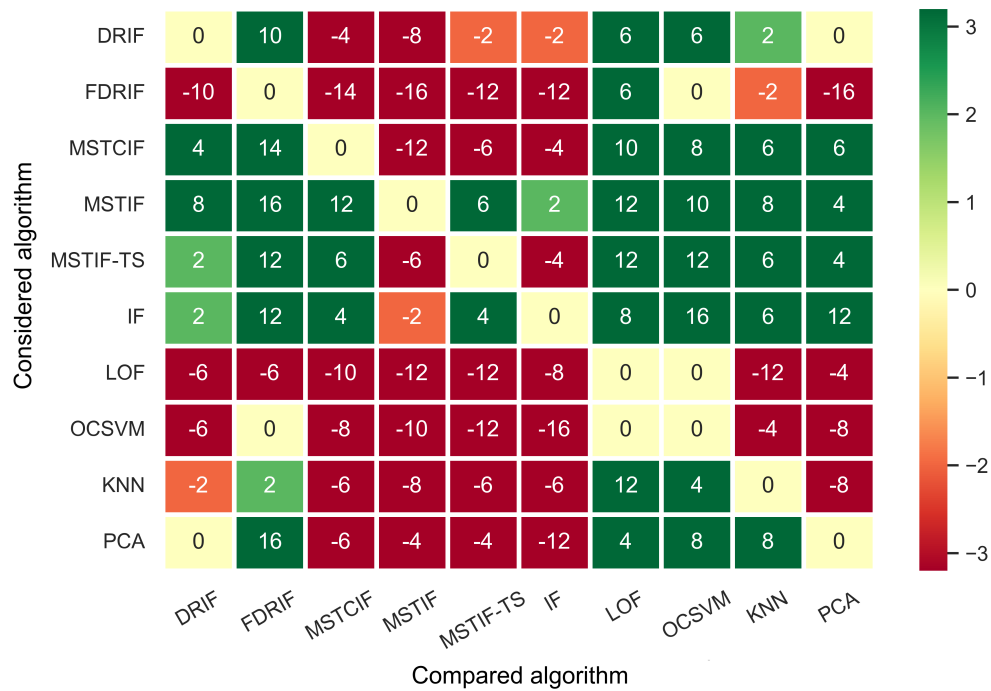


Figure 5.6: Heatmap presenting the differences in the number of better and worse PR AUC scores, determined between the compared algorithms.

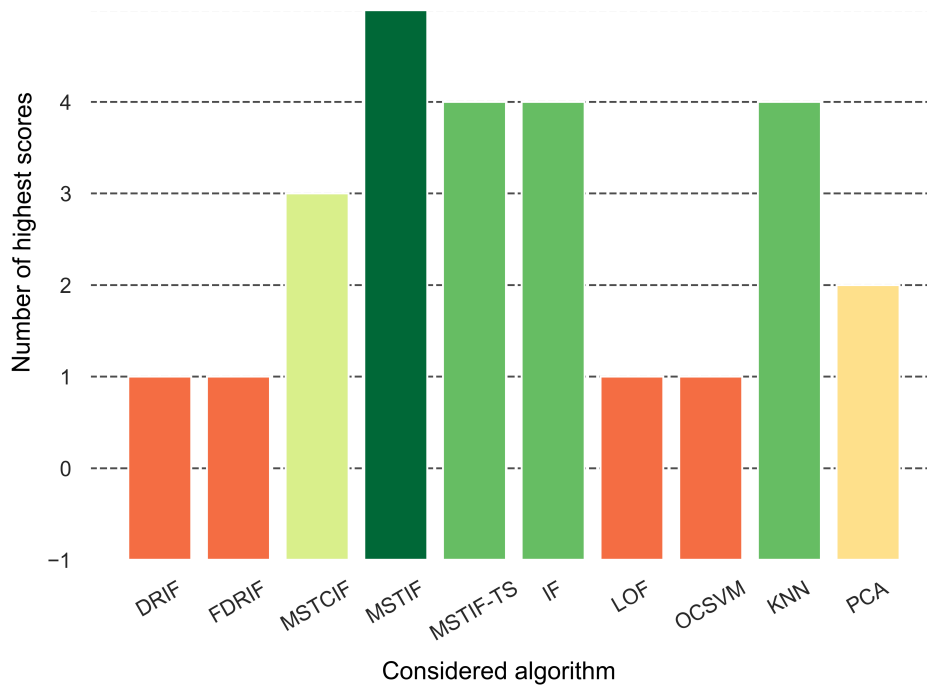


Figure 5.7: Chart presenting the number of cases in which individual algorithms achieved the highest ROC AUC scores for a given dataset among all analyzed methods.

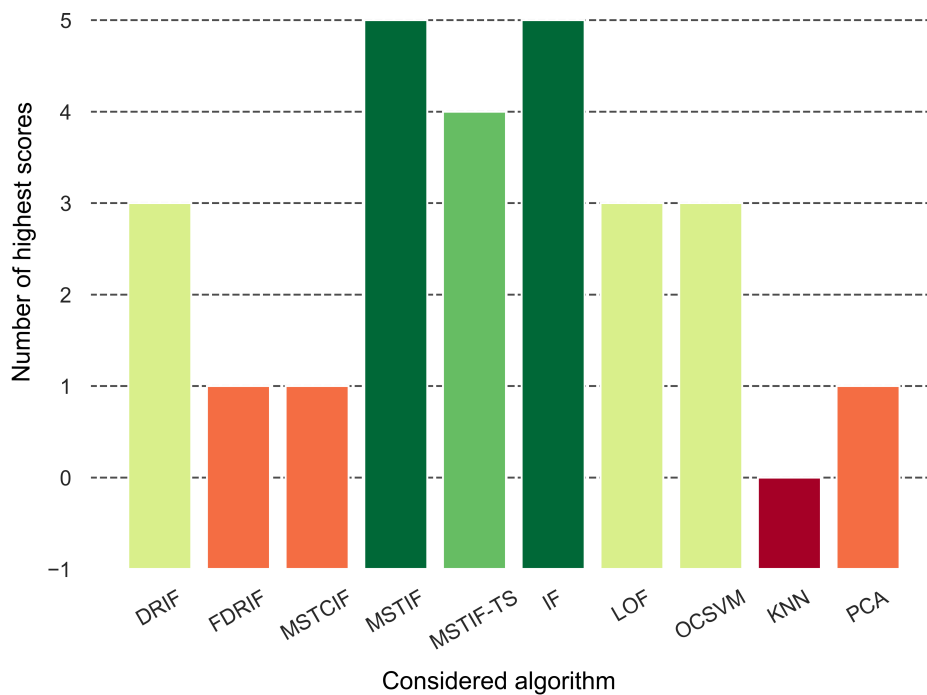


Figure 5.8: Chart presenting the number of cases in which the individual algorithms achieved the highest PR AUC scores for a given dataset, among all analyzed methods.

Table 5.6: Wilcoxon test p-values for the ROC AUC measure corresponding to the compared algorithm pairs. Part I. Significance level $\alpha = 0.05$. Statistically significant results are marked in bold.

Algorithm	DRIF	FDRIF	MSTCIF	MSTIF	MSTIF-TS
DRIF	-	0.2417	0.9960	0.9945	0.9972
FDRIF	0.7661	-	0.9997	0.9993	0.9997
MSTCIF	0.0043	0.0003	-	0.8865	0.9093
MSTIF	0.0060	0.0008	0.1185	-	0.6178
MSTIF-TS	0.0031	0.0003	0.0950	0.3918	-
IF	0.0167	0.0036	0.8362	0.8764	0.8815
LOF	0.9135	0.9050	0.9969	0.9994	0.9994
OCSVM	0.9093	0.9418	0.9994	0.9990	0.9995
KNN	0.3918	0.2577	0.9555	0.9753	0.9854
PCA	0.2741	0.0582	0.9960	0.9864	0.9912

Table 5.7: Wilcoxon test p-values for the ROC AUC measure corresponding to the compared algorithm pairs. Part II. Significance level $\alpha = 0.05$. Statistically significant results are marked in bold.

Algorithm	IF	LOF	OCSVM	KNN	PCA
DRIF	0.9844	0.0907	0.0950	0.6178	0.7342
FDRIF	0.9967	0.0994	0.0613	0.7504	0.9447
MSTCIF	0.1702	0.0033	0.0007	0.0470	0.0043
MSTIF	0.1289	0.0007	0.0011	0.0263	0.0146
MSTIF-TS	0.1236	0.0006	0.0006	0.0156	0.0095
IF	-	0.0102	0.0004	0.0376	0.0070
LOF	0.9905	-	0.3822	0.9898	0.9447
OCSVM	0.9996	0.6273	-	0.9175	0.9964
KNN	0.9645	0.0110	0.0865	-	0.6915
PCA	0.9935	0.0582	0.0040	0.3174	-

Table 5.8: Wilcoxon test p-values for the PR AUC measure corresponding to the compared algorithm pairs. Part I. Significance level $\alpha = 0.05$. Statistically significant results are marked in bold.

Algorithm	DRIF	FDRIF	MSTCIF	MSTIF	MSTIF-TS
DRIF	-	0.0376	0.8362	0.9898	0.9050
FDRIF	0.9645	-	0.9999	0.9999	0.9990
MSTCIF	0.1702	0.0001	-	0.9977	0.8424
MSTIF	0.0110	0.0001	0.0026	-	0.0355
MSTIF-TS	0.0994	0.0011	0.1638	0.9666	-
IF	0.1086	0.0009	0.3727	0.8167	0.4602
LOF	0.9555	0.9213	0.9949	0.9982	0.9977
OCSVM	0.9321	0.6178	0.9924	0.9957	0.9940
KNN	0.6553	0.3264	0.9579	0.9833	0.9624
PCA	0.4111	0.0009	0.8656	0.9418	0.8233

Table 5.9: Wilcoxon test p-values for the PR AUC measure corresponding to the compared algorithm pairs. Part II. Significance level $\alpha = 0.05$. Statistically significant results are marked in bold.

Algorithm	IF	LOF	OCSVM	KNN	PCA
DRIF	0.8961	0.0470	0.0714	0.3539	0.5986
FDRIF	0.9992	0.0825	0.3918	0.6826	0.9992
MSTCIF	0.6367	0.0055	0.0082	0.0445	0.1399
MSTIF	0.1901	0.0020	0.0047	0.0179	0.0613
MSTIF-TS	0.5497	0.0026	0.0065	0.0398	0.1833
IF	-	0.0065	0.0047	0.0865	0.0127
LOF	0.9940	-	0.6645	0.9982	0.9783
OCSVM	0.9957	0.3447	-	0.8601	0.9821
KNN	0.9175	0.0020	0.1457	-	0.9213
PCA	0.9882	0.0232	0.0191	0.0825	-

5.2.3 Discussion of the results

Let us begin by analyzing the measurement outcomes. For the ROC AUC metric (see Tables 5.2 and 5.3), the highest average scores are achieved by the MSTIF-TS method, while the highest median values are obtained using MSTIF. Similarly, MSTCIF also yielded high scores, only slightly lower than those achieved by MSTIF and MSTIF-TS. Among the Isolation Forest-based approaches, the methods based on dimensionality reduction performed the weakest. It is, however, worth noting that the datasets analyzed are designed for anomaly detection and thus had a reduced number of dimensions. Nonetheless, these methods outperformed LOF, OCSVM, and KNN. Notably, MSTCIF, MSTIF, and MSTIF-TS showed significantly lower standard deviations, indicating a higher stability of these approaches. This suggests that fewer isolation trees could be used while still achieving comparable results to the baseline IF. In the case of the PR AUC metric (see Tables 5.4 and 5.5), MSTIF demonstrated the best performance in terms of both average and median scores. All proposed methods had lower standard deviations compared to IF. Therefore, if for a fixed t the proposed methods already yield lower variability of results than IF, the same level of stability could, in principle, be achieved with a smaller number of trees. This observation suggests a potential for reducing the ensemble size without sacrificing robustness. As with ROC AUC, MSTCIF, MSTIF, and MSTIF-TS are the most effective. It is also worth noting that DRIF outperformed FDRIF in this measure, though both methods achieved lower scores than IF. LOF and OCSVM are the least effective based on this metric.

Analyzing raw values becomes increasingly difficult with a large number of datasets. Nevertheless, it enables assessment of the effectiveness of methods across individual collections. To provide a clearer overview, heatmaps are prepared to visualize algorithm performance characteristics. In this context, all possible combinations of techniques are compared. From the preliminary analysis of Figures 5.1 and 5.2, we observe that for both metrics, the newly proposed approaches outperform LOF, OCSVM, and KNN. Moreover, the baseline IF method is more effective than DRIF, FDRIF, LOF, OCSVM, KNN, and PCA. The results highlight MSTCIF, MSTIF, and MSTIF-TS as the top performers, with MSTIF-TS leading in all ROC AUC comparisons. However, MSTIF is only marginally behind. In terms of PR AUC, MSTIF clearly dominates, followed by MSTIF-TS. Next, we examine heatmaps of median differences (Figures 5.3 and 5.4). As with average values, LOF and OCSVM show the most red-marked fields, indicating inferior performance. The dimensionality reduction approaches outperform LOF and OCSVM and are on par with KNN in terms of ROC AUC. For PR AUC, DRIF slightly surpasses KNN, while FDRIF performs worse. Again, MSTCIF, MSTIF, and MSTIF-TS stand out, with MSTCIF performing slightly worse than the other two, which exhibit nearly identical scores. Importantly, their results are significantly better than those of the baseline IF.

The analysis of averages and medians should be complemented by evaluating how many datasets each algorithm outperforms others on. This provides insight into the number of datasets where an algorithm yields better results. Traditional measures might not reflect this nuance. In the case of attribute reduction approaches, they outperform LOF and OCSVM more frequently. Furthermore, KNN and PCA are clearly outperformed by IF, MSTCIF, MSTIF, and MSTIF-TS. For ROC AUC, MSTIF and MSTIF-TS display only green fields on the heatmap (see Figure 5.5), indicating better outcomes across all comparisons, and their effectiveness appears equivalent. MSTCIF ranks next, with red-marked fields appearing only in comparisons with the top methods. Regarding PR AUC, MSTIF clearly dominates, as shown by fully green fields in Figure 5.6. IF comes next with only one red field, followed by MSTIF-TS with two, and MSTCIF with three. These analyses covered as many as 26 real datasets, showing significant variability in method effectiveness and clearly identifying which algorithms performed best on more datasets. The final visualizations (Figures 5.7 and 5.8) show how many times each algorithm achieved the best result. Although each algorithm had some success, DRIF, FDRIF, LOF, and OCSVM obtained only one best result in ROC AUC. MSTCIF achieved three best results, while IF, MSTIF-TS, and KNN had four. MSTIF stood out as the most versatile solution. Regarding PR AUC, KNN had no best results, FDRIF, MSTCIF, and PCA had one, and DRIF, LOF, and OCSVM had three each. Ultimately, MSTIF, MSTIF-TS, and IF are the most effective.

The final aspect discussed is the formal statistical comparison. Tests are conducted to determine whether the differences in results are statistically significant. For ROC AUC, MSTCIF, MSTIF, MSTIF-TS, and IF demonstrated statistically superior performance compared to attribute reduction methods and other deterministic competitors like LOF, OCSVM, KNN, and PCA (Tables 5.6 and 5.7). Further p-value analysis reveals that MSTCIF, MSTIF, and MSTIF-TS offer marginal improvements over IF. For PR AUC, MSTIF again dominates (Tables 5.8 and 5.9). The difference is statistically significant compared to IF and borderline significant when compared to PCA. DRIF demonstrated statistically better performance than FDRIF and LOF. Although the p-value in comparison with OCSVM is low, it does not reach statistical significance. FDRIF, on the other hand, showed no statistically significant advantage over other methods. MSTCIF and MSTIF-TS yielded similar results, outperforming FDRIF, LOF, OCSVM, and KNN, while their comparisons with DRIF and PCA produced low but not always statistically significant p-values.

In conclusion, in-depth analysis using various statistical measures and perspectives reveals that each of the examined algorithms has potential in anomaly detection tasks. The most effective methods, i.e., MSTCIF, MSTIF, and MSTIF-TS, consistently achieved top performance. It is important to emphasize that algorithm performance varies significantly depending on the dataset characteristics, which should be considered when selecting the most suitable method. Both ROC AUC and PR AUC are useful metrics for evaluating classifier quality in scenarios without a clearly defined detection threshold, making them valuable for preliminary algorithm selection.

5.3 Threshold-dependent evaluation measures

5.3.1 Determination of the detection threshold value

The detection threshold value can be determined in many different ways, depending on the effect we want to achieve. Situations may arise where we particularly care about a specific behavior of the classifier, e.g., when false positive detection results have important consequences for us. Nevertheless, in our considerations we determine the detection threshold for each algorithm based on the ROC curve.

First, it is necessary to explain how the threshold is found, i.e., which point on the ROC curve indicates the optimal threshold value. It should be noted that the curve presents the relationship between the True Positive Rate (TPR) and the False Positive Rate (FPR) for various detection thresholds. From the classification quality perspective, an ideal point can be distinguished, i.e., $TPR = 1.0$ and $FPR = 0.0$. On the other hand, points lying on the line $y = x$ represent random detection. Therefore, two methods are used to determine the threshold. The first is to find the point closest to the ideal point. The second is to find the point furthest from the line $y = x$. In our study, we use the second method, meaning that the point on the ROC curve furthest from the line of random detections is determined. When such a point is found, it is necessary to check at which threshold value the corresponding TPR and FPR values are obtained. The found threshold is the optimal detection threshold for a given method. Examples of ROC curves for the Annthyroid dataset, with the point furthest from the $y = x$ line marked, are illustrated in Fig. 5.9. A red line indicates the segment connecting the selected point with the mentioned line. Additionally, the determined optimal threshold value is indicated.

Threshold value calculations are performed for each dataset and each method. For nondeterministic methods, the calculations are repeated 100 times, and the presented values are means and standard deviations. The threshold values are shown in Tables 5.10 and 5.11.

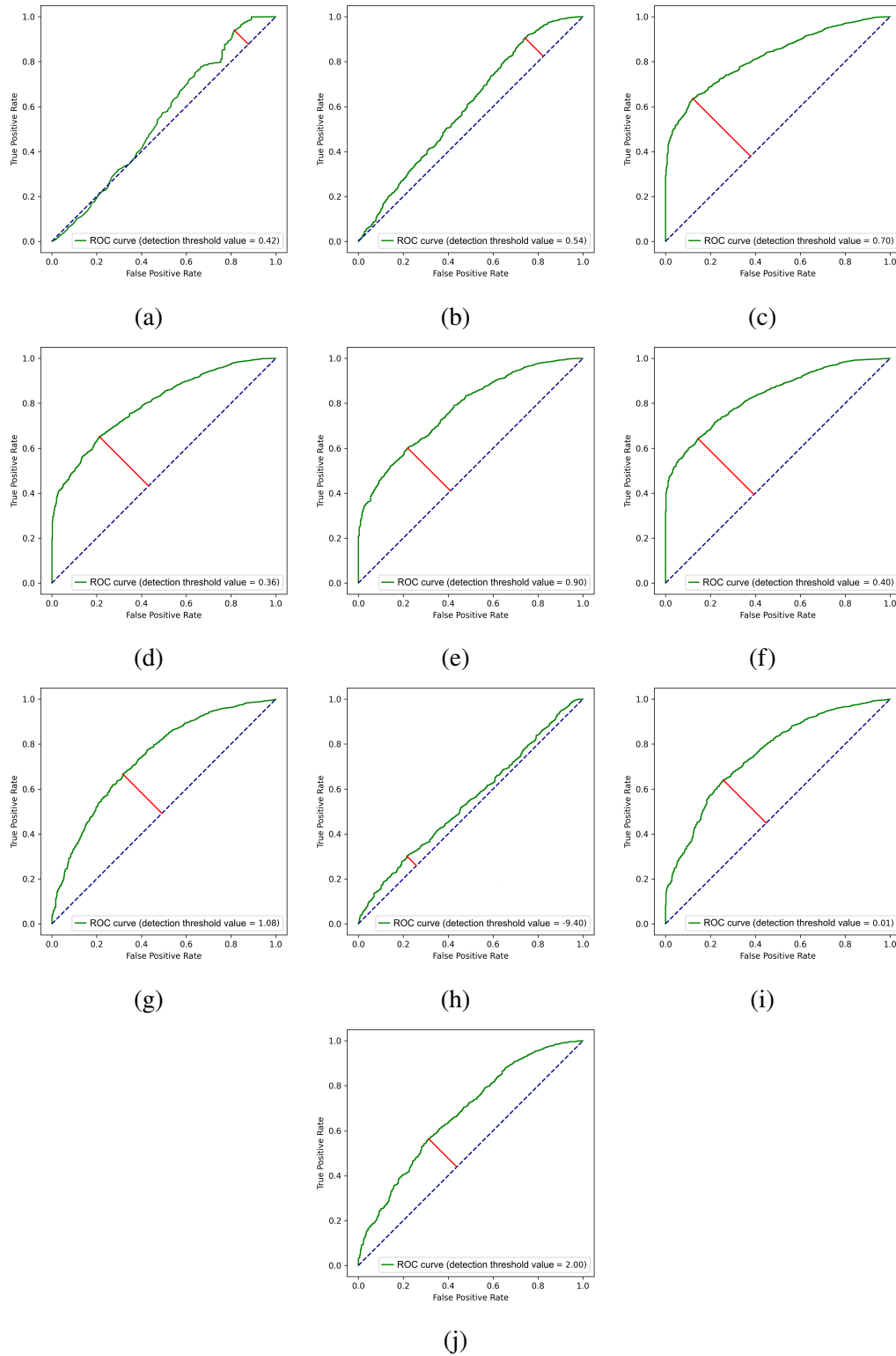


Figure 5.9: ROC curves for the Annthyroid dataset. Points most distant from the line $y = x$, representing the optimal detection thresholds for the algorithms, are indicated: (a) DRIF, (b) FDRIF, (c) MSTCIF, (d) MSTIF, (e) MSTIF-TS, (f) IF, (g) LOF, (h) OCSVM, (i) KNN, and (j) PCA.

Table 5.10: Detection threshold values for the new methods, along with standard deviations given in parentheses.

Dataset name	DRIF	FDRIF	MSTCIF	MSTIF	MSTIF-TS
Annthroid	0.42 (0.02)	0.54 (0.01)	0.72 (0.02)	0.36 (0.01)	0.89 (0.02)
Arrhythmia	0.38 (0.01)	0.50 (0.01)	0.92 (0.04)	1.00 (0.00)	2.16 (0.49)
BreastW	0.38 (0.00)	0.45 (0.01)	0.80 (0.02)	0.58 (0.02)	2.44 (0.17)
Cardio	0.39 (0.01)	0.46 (0.01)	0.89 (0.02)	1.00 (0.02)	3.11 (0.73)
ForestCover	0.39 (0.00)	0.48 (0.01)	0.95 (0.02)	0.69 (0.01)	1.67 (0.11)
Glass	0.34 (0.00)	0.53 (0.02)	0.62 (0.01)	0.47 (0.01)	1.48 (0.06)
Http	0.51 (0.02)	0.72 (0.01)	1.80 (0.02)	1.44 (0.04)	7.13 (0.15)
Ionosphere	0.37 (0.00)	0.50 (0.01)	0.83 (0.02)	0.88 (0.01)	1.93 (0.11)
Letter	0.39 (0.01)	0.48 (0.02)	0.86 (0.01)	0.97 (0.01)	1.21 (0.08)
Lympho	0.39 (0.01)	0.54 (0.02)	0.82 (0.03)	1.07 (0.04)	4.15 (0.18)
Mammography	0.37 (0.01)	0.48 (0.01)	0.66 (0.04)	0.41 (0.02)	2.80 (0.13)
Mnist	0.36 (0.01)	0.47 (0.02)	0.99 (0.01)	0.99 (0.00)	1.29 (0.11)
Musk	0.53 (0.01)	0.69 (0.01)	0.91 (0.02)	1.66 (0.02)	3.41 (0.04)
Optdigits	0.35 (0.00)	0.40 (0.01)	0.86 (0.02)	0.99 (0.01)	1.49 (0.06)
Pendigits	0.40 (0.00)	0.50 (0.01)	0.91 (0.02)	1.10 (0.02)	3.00 (0.10)
Pima	0.37 (0.00)	0.46 (0.02)	0.74 (0.02)	0.58 (0.01)	1.24 (0.17)
Satellite	0.39 (0.01)	0.43 (0.01)	0.77 (0.01)	0.80 (0.01)	1.43 (0.03)
Satimage-2	0.44 (0.01)	0.64 (0.01)	1.16 (0.03)	1.23 (0.09)	5.13 (0.78)
Shuttle	0.46 (0.01)	0.40 (0.02)	0.89 (0.01)	0.83 (0.02)	3.35 (0.04)
Smtpt	0.51 (0.08)	0.60 (0.01)	0.70 (0.10)	0.47 (0.06)	5.28 (0.96)
Speech	0.40 (0.01)	0.49 (0.02)	0.96 (0.12)	1.00 (0.00)	1.11 (0.06)
Thyroid	0.38 (0.01)	0.52 (0.01)	0.95 (0.04)	0.64 (0.03)	3.70 (0.24)
Vertebral	0.35 (0.00)	0.40 (0.01)	0.44 (0.01)	0.21 (0.00)	0.68 (0.01)
Vowels	0.39 (0.00)	0.48 (0.01)	0.83 (0.02)	0.81 (0.02)	2.57 (0.14)
WBC	0.39 (0.01)	0.52 (0.01)	0.86 (0.02)	0.94 (0.01)	2.92 (0.14)
Wine	0.42 (0.00)	0.48 (0.02)	0.59 (0.02)	0.71 (0.02)	2.67 (0.04)
Average:	0.40 (0.01)	0.51 (0.01)	0.86 (0.03)	0.84 (0.02)	2.62 (0.20)
Median:	0.39 (0.01)	0.48 (0.01)	0.86 (0.02)	0.85 (0.01)	2.50 (0.11)

Table 5.11: Detection threshold values for the reference methods, along with standard deviations given in parentheses.

Dataset name	IF	LOF	OCSVM	KNN	PCA
Annth thyroid	0.41 (0.01)	1.08	-9.39	0.01	2.00
Arrhythmia	0.36 (0.01)	1.17	0.00	151.10	14.28
BreastW	0.49 (0.01)	1.00	10.42	3.46	2.50
Cardio	0.44 (0.01)	1.16	37.40	2.37	4.79
ForestCover	0.48 (0.01)	1.04	0.00	55.18	3.91
Glass	0.40 (0.01)	1.14	14.97	0.75	1.75
Http	0.60 (0.02)	1.00	128784.00	0.05	8.68
Ionosphere	0.51 (0.02)	1.54	11.83	1.64	7.36
Letter	0.45 (0.01)	1.07	0.29	8.72	6.04
Lympho	0.52 (0.01)	1.10	4.77	3.00	5.61
Mammography	0.46 (0.01)	1.01	208.68	0.25	2.23
Mnist	0.39 (0.00)	1.06	0.00	458.24	8.36
Musk	0.53 (0.01)	1.03	0.00	616.62	18.38
Optdigits	0.44 (0.01)	1.08	-0.00	29.26	6.24
Pendigits	0.54 (0.01)	1.16	52.97	0.25	4.66
Pima	0.45 (0.01)	0.99	-0.00	20.74	2.78
Satellite	0.48 (0.01)	1.14	0.00	28.07	7.42
Satimage-2	0.54 (0.01)	1.36	-0.00	27.31	10.36
Shuttle	0.54 (0.01)	1.04	0.41	2.00	4.15
Smt p	0.54 (0.01)	1.05	16798.30	0.05	5.45
Speech	0.45 (0.02)	1.31	83.84	23.26	23.78
Thyroid	0.50 (0.01)	1.14	9.61	0.07	3.04
Vertebral	0.40 (0.01)	1.04	0.00	8.03	1.27
Vowels	0.47 (0.01)	1.14	3.24	1.90	4.38
WBC	0.45 (0.01)	1.19	1.61	0.42	5.56
Wine	0.48 (0.01)	1.67	0.00	70.07	4.55
Average:	0.47 (0.01)	1.14	5615.88	58.19	6.52
Median:	0.47 (0.01)	1.09	1.01	3.23	5.12

5.3.2 Results

For the detection thresholds determined in the previous subsection, statistics dependent on the threshold selection are measured. Specifically, these are: Accuracy, precision, recall, specificity, false alarm ratio, and F1 score. The measurements are carried out in the same manner as for the threshold-independent evaluation measures. The results are presented respectively for: Accuracy in Tables 5.12 and 5.13, precision in Tables 5.14 and 5.15, recall in Tables 5.16 and 5.17, specificity in Tables 5.18 and 5.19, false alarm ratio in Tables 5.20 and 5.21, and F1 score in Tables 5.22 and 5.23.

5.3.3 Analysis of the results

For a deeper analysis of the results, visualizations of the outcomes are presented using heatmaps and charts, namely for average value difference comparisons in Figures 5.10–5.15, for median differences in Figures 5.16–5.21, for differences in the number of better and worse results in Figures 5.22–5.27, and for the number of best results among all presented methods in Figures 5.28–5.33. Moreover, Wilcoxon statistical tests are conducted to compare results between all pairs of algorithms. The p-values are presented in Tables 5.24–5.35. The level of statistical significance is set to $\alpha = 0.05$. For the false alarm ratio metric, the alternative hypothesis assumed is that the distribution of differences between sample pairs is statistically smaller than a symmetric distribution around zero. For the remaining metrics, the alternative hypothesis assumed that the distribution of differences between sample pairs is statistically greater than a symmetric distribution around zero. The reason for adopting different hypotheses is that, for the false alarm ratio metric, lower values indicate better performance, i.e., fewer false alarms, while for the other metrics, higher values indicate better algorithm performance.

Table 5.12: Accuracy values for the new methods, along with standard deviations given in parentheses.

Dataset name	DRIF [%]	FDRIF [%]	MSTCIF [%]	MSTIF [%]	MSTIF-TS [%]
Annth thyroid	87.4 (1.3)	84.8 (0.9)	67.4 (0.6)	66.0 (0.4)	59.7 (0.3)
Arrhythmia	78.3 (1.0)	72.0 (1.6)	74.8 (0.9)	76.0 (0.6)	75.1 (0.5)
BreastW	94.9 (0.5)	91.3 (0.7)	94.0 (0.2)	95.0 (0.1)	95.2 (0.1)
Cardio	89.7 (1.0)	57.8 (3.1)	80.0 (0.7)	91.0 (0.2)	88.1 (0.2)
ForestCover	82.6 (1.9)	66.6 (1.8)	85.7 (0.5)	79.5 (0.3)	76.1 (0.3)
Glass	53.8 (2.5)	84.1 (0.9)	57.2 (1.0)	58.5 (0.4)	58.2 (0.5)
Http	99.4 (0.0)	99.3 (0.1)	99.9 (0.0)	99.4 (0.1)	99.4 (0.1)
Ionosphere	71.6 (1.7)	71.0 (1.4)	88.7 (0.7)	92.0 (0.4)	92.0 (0.2)
Letter	81.5 (1.7)	66.6 (1.6)	64.3 (1.1)	62.7 (0.9)	64.0 (0.4)
Lympho	85.6 (1.8)	89.3 (1.5)	87.1 (1.6)	76.8 (0.7)	77.5 (0.7)
Mammography	82.4 (1.3)	78.1 (1.1)	73.2 (0.4)	71.0 (0.3)	82.9 (0.2)
Mnist	60.4 (4.3)	69.3 (2.7)	71.5 (0.7)	75.6 (0.3)	74.8 (0.3)
Musk	99.6 (0.5)	99.8 (0.2)	98.1 (0.5)	99.9 (0.1)	99.3 (0.2)
Optdigits	10.5 (5.9)	5.5 (3.3)	36.5 (0.8)	66.0 (0.7)	80.3 (0.6)
Pendigits	79.2 (2.2)	71.3 (1.5)	84.6 (0.6)	90.5 (0.3)	89.9 (0.3)
Pima	60.0 (1.5)	52.5 (1.5)	64.0 (0.7)	65.9 (0.4)	67.6 (0.4)
Satellite	77.7 (0.5)	67.7 (1.3)	78.7 (0.6)	75.7 (0.3)	77.0 (0.6)
Satimage-2	98.8 (0.4)	98.8 (0.3)	99.5 (0.1)	99.2 (0.1)	99.0 (0.0)
Shuttle	98.8 (0.3)	69.5 (2.1)	94.2 (0.5)	97.4 (0.1)	94.7 (0.2)
Smt p	96.9 (0.2)	95.6 (0.3)	78.1 (0.5)	82.7 (0.4)	94.9 (0.4)
Speech	86.7 (1.6)	64.5 (2.5)	65.4 (0.6)	5.5 (0.3)	88.0 (0.4)
Thyroid	83.4 (0.7)	84.6 (1.2)	89.7 (0.3)	91.0 (0.1)	90.8 (0.1)
Vertebral	18.1 (4.8)	12.9 (0.8)	13.4 (0.6)	12.8 (0.2)	12.9 (0.2)
Vowels	76.9 (3.7)	64.1 (1.7)	72.7 (0.8)	91.4 (0.4)	92.2 (0.3)
WBC	82.9 (0.8)	84.4 (1.0)	81.1 (0.7)	82.6 (0.4)	81.4 (0.3)
Wine	90.6 (1.7)	60.5 (2.8)	78.7 (1.9)	90.6 (1.2)	88.6 (1.3)
Average:	78.0 (1.7)	71.6 (1.5)	76.1 (0.7)	76.7 (0.4)	80.8 (0.3)
Median:	82.8 (1.4)	71.1 (1.5)	78.7 (0.6)	81.1 (0.3)	85.4 (0.3)

Table 5.13: Accuracy values for the reference methods, along with standard deviations given in parentheses.

Dataset name	IF [%]	LOF [%]	OCSVM [%]	KNN [%]	PCA [%]
Annth thyroid	68.0 (2.1)	66.8	33.8	64.8	57.3
Arrhythmia	74.3 (2.9)	72.6	36.9	73.2	73.0
BreastW	96.1 (0.3)	52.0	95.2	95.0	94.9
Cardio	80.1 (1.9)	76.7	82.4	77.8	83.1
ForestCover	73.0 (2.3)	79.8	3.8	79.9	84.2
Glass	51.9 (1.5)	64.5	84.6	77.1	34.1
Http	99.6 (0.1)	27.7	99.4	99.1	99.5
Ionosphere	78.5 (1.1)	84.6	79.2	91.5	78.1
Letter	45.4 (3.9)	80.6	83.8	87.0	68.0
Lympho	96.7 (1.1)	92.6	94.6	90.5	92.6
Mammography	78.2 (1.7)	53.9	75.5	65.1	76.1
Mnist	68.7 (4.1)	61.5	9.2	76.5	74.0
Musk	99.6 (0.4)	71.0	75.5	95.3	100.0
Optdigits	52.9 (5.7)	76.5	40.9	96.8	32.3
Pendigits	85.8 (1.0)	84.8	82.7	52.3	83.2
Pima	64.4 (1.6)	46.5	56.0	63.8	65.4
Satellite	78.2 (1.0)	63.4	67.7	70.4	77.9
Satimage-2	97.2 (0.3)	94.5	3.6	75.0	98.6
Shuttle	99.2 (0.2)	61.7	66.4	57.4	99.1
Smt p	93.3 (0.4)	75.2	100.0	96.5	97.7
Speech	79.9 (4.7)	93.5	93.2	89.6	93.0
Thyroid	92.4 (0.6)	71.8	71.6	89.8	89.1
Vertebral	17.6 (2.7)	48.3	67.5	13.8	15.4
Vowels	63.3 (3.3)	84.3	58.8	94.8	87.3
WBC	80.5 (1.9)	81.0	89.7	83.6	83.6
Wine	77.7 (2.8)	98.4	76.0	98.4	86.8
Average:	76.6 (1.9)	71.7	66.5	79.0	77.9
Median:	78.4 (1.7)	73.9	75.5	81.7	83.4

Table 5.14: Precision values for the new methods, along with standard deviations given in parentheses.

Dataset name	DRIF [%]	FDRIF [%]	MSTCIF [%]	MSTIF [%]	MSTIF-TS [%]
Annthroid	19.4 (2.4)	16.9 (1.0)	16.5 (0.3)	15.4 (0.2)	13.3 (0.2)
Arrhythmia	37.1 (1.2)	26.6 (1.7)	32.8 (1.1)	33.3 (0.8)	32.4 (0.6)
BreastW	88.4 (0.9)	81.0 (1.1)	86.1 (0.4)	88.2 (0.1)	88.4 (0.2)
Cardio	48.3 (2.7)	16.7 (1.1)	31.7 (0.9)	53.0 (0.9)	41.8 (0.7)
ForestCover	5.0 (0.4)	2.4 (0.1)	5.9 (0.2)	4.0 (0.1)	3.3 (0.1)
Glass	8.0 (0.3)	17.8 (1.1)	8.2 (0.8)	8.7 (0.7)	8.4 (0.8)
Http	37.6 (0.9)	35.1 (8.1)	78.2 (5.7)	40.9 (4.7)	42.0 (10.2)
Ionosphere	56.7 (1.5)	58.9 (2.0)	84.1 (1.1)	91.5 (0.1)	91.8 (0.5)
Letter	12.6 (1.0)	9.9 (0.4)	12.6 (0.5)	13.4 (0.3)	14.0 (0.2)
Lympho	19.7 (2.5)	26.1 (2.9)	22.8 (3.2)	13.4 (0.9)	14.2 (1.1)
Mammography	7.8 (0.5)	7.9 (0.3)	6.3 (0.1)	6.1 (0.1)	8.8 (0.1)
Mnist	15.0 (1.3)	16.2 (2.2)	22.6 (0.6)	25.6 (0.3)	24.9 (0.2)
Musk	100.0 (0.0)	97.4 (1.8)	63.2 (6.2)	100.0 (0.0)	82.2 (2.4)
Optdigits	2.8 (0.3)	2.8 (0.1)	3.6 (0.2)	6.6 (0.5)	12.0 (1.3)
Pendigits	9.0 (0.8)	5.9 (0.3)	12.6 (0.5)	18.8 (0.6)	18.0 (0.6)
Pima	46.0 (1.0)	38.4 (1.2)	48.9 (0.7)	50.8 (0.4)	52.8 (0.4)
Satellite	80.3 (3.2)	49.4 (1.3)	69.3 (1.3)	61.6 (0.5)	62.5 (0.9)
Satimage-2	52.9 (7.9)	50.4 (5.7)	73.1 (2.9)	59.7 (1.5)	54.9 (0.9)
Shuttle	89.0 (2.4)	17.0 (0.8)	55.3 (2.0)	73.5 (1.0)	57.5 (0.9)
Smtpt	0.7 (0.0)	0.5 (0.0)	0.1 (0.0)	0.1 (0.0)	0.5 (0.0)
Speech	2.6 (0.4)	2.0 (0.2)	1.3 (0.1)	1.6 (0.0)	2.0 (0.2)
Thyroid	12.1 (0.4)	12.5 (0.8)	18.6 (0.5)	20.8 (0.3)	20.5 (0.2)
Vertebral	12.2 (0.6)	12.2 (0.4)	12.4 (0.3)	12.4 (0.2)	12.4 (0.2)
Vowels	8.8 (1.0)	8.3 (0.3)	9.7 (0.5)	27.8 (0.9)	29.6 (0.9)
WBC	23.0 (0.7)	21.6 (1.3)	22.2 (1.1)	23.2 (0.9)	22.2 (0.5)
Wine	43.3 (7.6)	14.6 (1.0)	25.5 (2.2)	44.9 (2.9)	39.9 (3.6)
Average:	32.3 (1.6)	24.9 (1.4)	31.7 (1.3)	34.4 (0.7)	32.7 (1.1)
Median:	19.6 (0.9)	16.8 (1.0)	22.4 (0.6)	24.4 (0.4)	23.6 (0.6)

Table 5.15: Precision values for the reference methods, along with standard deviations given in parentheses.

Dataset name	IF [%]	LOF [%]	OCSVM [%]	KNN [%]	PCA [%]
Annthroid	16.2 (0.9)	14.1	8.3	14.2	11.2
Arrhythmia	32.7 (2.5)	31.9	14.8	32.7	31.1
BreastW	91.3 (0.6)	40.1	89.9	88.7	91.5
Cardio	31.6 (2.2)	14.9	34.9	22.6	36.1
ForestCover	3.2 (0.3)	1.6	1.0	3.1	5.5
Glass	7.1 (1.1)	9.6	12.5	15.5	5.4
Http	45.9 (9.7)	0.5	40.7	2.5	43.5
Ionosphere	76.2 (3.0)	81.0	81.2	90.7	86.6
Letter	7.9 (0.4)	21.9	25.0	29.7	8.0
Lympho	57.8 (11.2)	35.3	41.7	30.0	33.3
Mammography	8.2 (0.5)	4.0	7.8	5.7	8.1
Mnist	20.1 (1.8)	13.4	9.2	23.2	23.7
Musk	91.7 (8.3)	3.5	6.8	5.6	99.0
Optdigits	4.7 (0.5)	5.2	3.1	9.1	3.5
Pendigits	13.3 (1.0)	3.6	11.1	3.8	11.5
Pima	49.3 (1.8)	38.1	40.3	48.3	50.3
Satellite	73.6 (3.1)	39.6	34.1	53.8	80.9
Satimage-2	30.1 (2.8)	6.0	1.2	4.6	47.1
Shuttle	91.8 (2.4)	9.1	13.7	11.6	92.0
Smtpt	0.4 (0.0)	0.1	74.1	0.6	0.9
Speech	1.6 (0.3)	3.6	3.0	2.1	2.4
Thyroid	23.8 (1.3)	6.5	6.9	18.6	17.6
Vertebral	12.3 (0.6)	14.4	20.0	12.7	12.6
Vowels	6.6 (0.6)	17.3	6.5	39.1	9.1
WBC	20.9 (1.7)	21.3	33.3	24.7	24.1
Wine	21.5 (3.0)	90.0	11.1	83.3	31.6
Average:	32.3 (2.4)	20.3	24.3	26.0	33.3
Median:	21.2 (1.5)	13.7	13.1	17.0	23.9

Table 5.16: Recall values for the new methods, along with standard deviations given in parentheses.

Dataset name	DRIF [%]	FDRIF [%]	MSTCIF [%]	MSTIF [%]	MSTIF-TS [%]
Annthroid	21.4 (1.6)	26.7 (1.3)	83.6 (2.0)	79.5 (1.0)	80.2 (1.1)
Arrhythmia	69.5 (2.4)	52.1 (4.0)	68.8 (2.4)	64.0 (1.5)	64.5 (0.9)
BreastW	98.5 (1.7)	98.2 (2.0)	98.8 (0.5)	99.2 (0.2)	99.4 (0.3)
Cardio	86.5 (2.6)	84.4 (5.6)	93.5 (2.4)	58.3 (0.7)	60.1 (0.4)
ForestCover	93.1 (3.9)	84.9 (3.8)	93.2 (2.1)	89.1 (1.0)	85.0 (1.4)
Glass	94.4 (6.4)	76.6 (3.5)	90.3 (9.3)	93.2 (7.8)	90.1 (9.1)
Http	99.9 (0.0)	94.3 (22.5)	99.9 (0.0)	99.9 (0.3)	99.7 (0.6)
Ionosphere	87.3 (4.7)	63.6 (3.3)	84.3 (1.3)	85.5 (1.0)	85.3 (0.5)
Letter	32.7 (2.6)	53.3 (3.1)	79.4 (3.1)	91.4 (1.3)	92.7 (1.2)
Lympho	81.7 (5.0)	87.2 (7.4)	90.5 (10.6)	85.8 (6.0)	90.3 (8.2)
Mammography	60.5 (2.7)	78.8 (2.3)	76.2 (1.6)	80.5 (1.0)	68.2 (0.5)
Mnist	70.4 (8.9)	55.7 (8.0)	86.3 (2.2)	87.0 (0.7)	86.4 (0.5)
Musk	86.9 (14.3)	96.3 (6.1)	96.9 (4.2)	98.4 (3.7)	98.2 (2.8)
Optdigits	88.3 (12.9)	94.8 (6.5)	82.8 (5.0)	82.3 (6.1)	92.8 (10.9)
Pendigits	89.4 (9.0)	77.5 (3.3)	96.9 (3.2)	95.5 (2.3)	96.5 (3.2)
Pima	85.3 (5.2)	59.9 (4.8)	73.0 (1.2)	72.8 (0.7)	68.1 (0.6)
Satellite	39.2 (1.0)	76.5 (3.9)	58.4 (1.3)	61.4 (0.7)	67.8 (1.4)
Satimage-2	95.4 (2.8)	96.5 (0.8)	96.5 (1.0)	97.2 (0.0)	97.2 (0.0)
Shuttle	95.8 (2.2)	83.7 (3.9)	99.1 (1.1)	99.6 (0.7)	99.8 (0.4)
Smtpt	70.0 (0.0)	74.6 (1.6)	77.3 (2.6)	82.0 (6.5)	76.5 (0.7)
Speech	19.5 (3.6)	42.7 (4.7)	26.8 (2.7)	91.4 (1.6)	12.9 (1.1)
Thyroid	90.9 (1.8)	87.2 (2.6)	93.8 (1.3)	94.3 (0.7)	94.8 (0.5)
Vertebral	90.1 (9.8)	96.5 (4.5)	97.7 (3.0)	98.3 (1.7)	98.3 (1.8)
Vowels	60.0 (7.2)	94.2 (3.5)	84.0 (3.8)	94.1 (0.7)	91.6 (1.6)
WBC	88.4 (3.8)	68.6 (3.1)	95.8 (5.2)	92.2 (4.2)	93.7 (2.6)
Wine	80.8 (21.7)	84.3 (6.4)	91.3 (7.2)	91.3 (9.6)	93.1 (8.3)
Average:	76.0 (5.3)	76.5 (4.7)	85.2 (3.1)	87.1 (2.4)	84.0 (2.3)
Median:	86.7 (3.7)	81.2 (3.8)	90.4 (2.4)	91.3 (1.0)	91.0 (1.1)

Table 5.17: Recall values for the reference methods, along with standard deviations given in parentheses.

Dataset name	IF [%]	LOF [%]	OCSVM [%]	KNN [%]	PCA [%]
Annthroid	79.3 (3.5)	68.4	78.5	74.3	68.7
Arrhythmia	70.3 (4.6)	77.3	69.7	78.8	69.7
BreastW	98.1 (0.4)	75.3	97.1	98.3	94.1
Cardio	90.3 (4.3)	30.1	96.0	54.0	98.3
ForestCover	91.0 (5.9)	33.0	99.6	66.5	95.4
Glass	87.2 (15.0)	88.9	44.4	100.0	88.9
Http	95.9 (18.8)	98.1	99.9	3.3	99.9
Ionosphere	58.7 (1.4)	74.6	54.8	84.9	46.0
Letter	72.8 (6.0)	82.0	80.0	79.0	39.0
Lympho	88.0 (8.5)	100.0	83.3	100.0	83.3
Mammography	81.2 (3.0)	82.3	88.1	90.0	89.2
Mnist	79.6 (6.3)	58.0	100.0	67.3	82.0
Musk	95.9 (6.3)	30.9	52.6	3.1	100.0
Optdigits	79.3 (11.0)	42.0	64.0	1.3	86.0
Pendigits	95.4 (5.8)	21.8	94.2	82.7	96.2
Pima	60.8 (4.0)	85.1	54.5	51.5	58.6
Satellite	48.7 (1.9)	29.8	2.2	45.4	39.4
Satimage-2	94.7 (1.3)	23.9	98.6	98.6	90.1
Shuttle	97.5 (1.0)	48.4	69.7	75.2	95.5
Smtpt	75.0 (1.8)	96.7	66.7	66.7	70.0
Speech	18.0 (5.4)	11.5	9.8	11.5	8.2
Thyroid	94.4 (1.6)	78.5	83.9	92.5	93.5
Vertebral	91.2 (7.8)	63.3	53.3	100.0	96.7
Vowels	73.9 (6.4)	94.0	82.0	90.0	30.0
WBC	89.2 (5.6)	90.5	85.7	95.2	90.5
Wine	69.7 (9.4)	90.0	30.0	100.0	60.0
Average:	79.9 (5.7)	64.4	70.7	69.6	75.7
Median:	84.2 (5.5)	75.0	79.2	78.9	87.4

Table 5.18: Specificity values for the new methods, along with standard deviations given in parentheses.

Dataset name	DRIF [%]	FDRIF [%]	MSTCIF [%]	MSTIF [%]	MSTIF-TS [%]
Annthroid	92.7 (1.4)	89.5 (1.0)	66.1 (0.7)	64.9 (0.4)	58.0 (0.3)
Arrhythmia	79.8 (1.5)	75.4 (2.1)	75.8 (1.1)	78.1 (0.7)	77.0 (0.5)
BreastW	93.0 (0.7)	87.6 (1.0)	91.4 (0.3)	92.8 (0.1)	93.0 (0.2)
Cardio	90.1 (1.2)	55.0 (3.6)	78.6 (0.8)	94.5 (0.2)	91.1 (0.3)
ForestCover	82.5 (2.0)	66.4 (1.9)	85.6 (0.5)	79.4 (0.3)	76.0 (0.3)
Glass	52.0 (2.8)	84.4 (1.0)	55.7 (1.0)	57.0 (0.3)	56.8 (0.3)
Http	99.4 (0.0)	99.3 (0.0)	99.9 (0.0)	99.4 (0.1)	99.4 (0.1)
Ionosphere	62.8 (1.1)	75.1 (2.3)	91.1 (0.7)	95.6 (0.0)	95.7 (0.3)
Letter	84.8 (2.0)	67.5 (1.9)	63.3 (1.2)	60.8 (1.0)	62.1 (0.4)
Lympho	85.7 (1.9)	89.4 (1.7)	86.9 (1.6)	76.5 (0.7)	76.9 (0.6)
Mammography	82.9 (1.4)	78.0 (1.2)	73.1 (0.4)	70.7 (0.4)	83.2 (0.3)
Mnist	59.4 (5.3)	70.7 (2.9)	70.0 (0.8)	74.4 (0.3)	73.6 (0.3)
Musk	100.0 (0.0)	99.9 (0.1)	98.1 (0.5)	100.0 (0.0)	99.3 (0.1)
Optdigits	8.2 (6.4)	2.9 (3.6)	35.1 (0.8)	65.6 (0.8)	80.0 (0.5)
Pendigits	79.0 (2.4)	71.2 (1.6)	84.4 (0.6)	90.4 (0.3)	89.8 (0.3)
Pima	46.4 (4.1)	48.4 (3.8)	59.1 (1.0)	62.2 (0.5)	67.3 (0.5)
Satellite	95.5 (1.0)	63.6 (3.5)	88.0 (0.8)	82.2 (0.4)	81.2 (0.7)
Satimage-2	98.9 (0.4)	98.8 (0.3)	99.6 (0.1)	99.2 (0.1)	99.0 (0.0)
Shuttle	99.1 (0.2)	68.4 (2.5)	93.8 (0.5)	97.2 (0.1)	94.3 (0.2)
Smtpt	97.0 (0.2)	95.6 (0.3)	78.1 (0.5)	82.7 (0.4)	94.9 (0.4)
Speech	87.8 (1.6)	64.8 (2.6)	66.1 (0.7)	4.1 (0.3)	89.3 (0.4)
Thyroid	83.3 (0.8)	84.5 (1.3)	89.6 (0.4)	90.9 (0.1)	90.7 (0.1)
Vertebral	7.9 (6.7)	1.0 (1.3)	1.4 (0.7)	0.6 (0.2)	0.7 (0.2)
Vowels	77.5 (4.0)	63.0 (1.9)	72.3 (0.8)	91.3 (0.4)	92.2 (0.3)
WBC	82.6 (1.0)	85.3 (1.1)	80.2 (0.7)	82.0 (0.3)	80.7 (0.3)
Wine	91.4 (0.7)	58.5 (3.3)	77.6 (2.1)	90.6 (1.8)	88.2 (1.2)
Average:	77.7 (2.0)	70.9 (1.8)	75.4 (0.7)	76.3 (0.4)	80.4 (0.4)
Median:	84.0 (1.4)	73.1 (1.8)	78.3 (0.7)	82.1 (0.3)	85.7 (0.3)

Table 5.19: Specificity values for the reference methods, along with standard deviations given in parentheses.

Dataset name	IF [%]	LOF [%]	OCSVM [%]	KNN [%]	PCA [%]
Annthroid	67.1 (2.4)	66.6	30.2	64.1	56.4
Arrhythmia	75.0 (3.9)	71.8	31.3	72.3	73.6
BreastW	94.9 (0.4)	39.4	94.1	93.2	95.3
Cardio	79.1 (2.2)	81.7	81.0	80.4	81.5
ForestCover	72.8 (2.4)	80.3	2.8	80.0	84.0
Glass	50.4 (1.6)	63.4	86.3	76.1	31.7
Http	99.6 (0.1)	27.4	99.4	99.5	99.5
Ionosphere	89.6 (1.7)	90.2	92.9	95.1	96.0
Letter	43.6 (4.4)	80.5	84.0	87.5	69.9
Lympho	97.1 (1.1)	92.3	95.1	90.1	93.0
Mammography	78.1 (1.8)	53.2	75.2	64.5	75.8
Mnist	67.6 (4.9)	61.9	0.0	77.5	73.2
Musk	99.7 (0.3)	72.3	76.3	98.3	100.0
Optdigits	52.1 (6.1)	77.5	40.2	99.6	30.7
Pendigits	85.5 (1.1)	86.3	82.5	51.6	82.9
Pima	66.4 (3.9)	25.8	56.8	70.4	69.0
Satellite	91.9 (1.2)	79.0	98.1	81.9	95.7
Satimage-2	97.2 (0.4)	95.3	2.4	74.7	98.7
Shuttle	99.3 (0.2)	62.8	66.2	56.0	99.4
Smtpt	93.3 (0.4)	75.2	100.0	96.5	97.7
Speech	81.0 (4.8)	94.9	94.6	90.9	94.5
Thyroid	92.3 (0.6)	71.7	71.3	89.8	88.9
Vertebral	7.1 (3.9)	46.2	69.5	1.4	3.8
Vowels	62.9 (3.5)	84.0	58.0	95.0	89.3
WBC	80.0 (2.1)	80.4	89.9	82.9	83.2
Wine	78.4 (3.2)	99.2	79.8	98.3	89.1
Average:	77.0 (2.3)	71.5	67.6	79.5	79.0
Median:	79.5 (2.0)	76.3	78.1	82.4	86.5

Table 5.20: False alarm ratio values for the new methods, along with standard deviations given in parentheses.

Dataset name	DRIF [%]	FDRIF [%]	MSTCIF [%]	MSTIF [%]	MSTIF-TS [%]
Annthroid	7.3 (1.4)	10.5 (1.0)	33.9 (0.7)	35.1 (0.4)	42.0 (0.3)
Arrhythmia	20.2 (1.5)	24.6 (2.1)	24.2 (1.1)	21.9 (0.7)	23.0 (0.5)
BreastW	7.0 (0.7)	12.4 (1.0)	8.6 (0.3)	7.2 (0.1)	7.0 (0.2)
Cardio	9.9 (1.2)	45.0 (3.6)	21.4 (0.8)	5.5 (0.2)	8.9 (0.3)
ForestCover	17.5 (2.0)	33.6 (1.9)	14.4 (0.5)	20.6 (0.3)	24.0 (0.3)
Glass	48.0 (2.8)	15.6 (1.0)	44.3 (1.0)	43.0 (0.3)	43.2 (0.3)
Http	0.6 (0.0)	0.7 (0.0)	0.1 (0.0)	0.6 (0.1)	0.6 (0.1)
Ionosphere	37.2 (1.1)	24.9 (2.3)	8.9 (0.7)	4.4 (0.0)	4.3 (0.3)
Letter	15.2 (2.0)	32.5 (1.9)	36.7 (1.2)	39.2 (1.0)	37.9 (0.4)
Lympho	14.3 (1.9)	10.6 (1.7)	13.1 (1.6)	23.5 (0.7)	23.1 (0.6)
Mammography	17.1 (1.4)	22.0 (1.2)	26.9 (0.4)	29.3 (0.4)	16.8 (0.3)
Mnist	40.6 (5.3)	29.3 (2.9)	30.0 (0.8)	25.6 (0.3)	26.4 (0.3)
Musk	0.0 (0.0)	0.1 (0.1)	1.9 (0.5)	0.0 (0.0)	0.7 (0.1)
Optdigits	91.8 (6.4)	97.1 (3.6)	64.9 (0.8)	34.4 (0.8)	20.0 (0.5)
Pendigits	21.0 (2.4)	28.8 (1.6)	15.6 (0.6)	9.6 (0.3)	10.2 (0.3)
Pima	53.6 (4.1)	51.6 (3.8)	40.9 (1.0)	37.8 (0.5)	32.7 (0.5)
Satellite	4.5 (1.0)	36.4 (3.5)	12.0 (0.8)	17.8 (0.4)	18.8 (0.7)
Satimage-2	1.1 (0.4)	1.2 (0.3)	0.4 (0.1)	0.8 (0.1)	1.0 (0.0)
Shuttle	0.9 (0.2)	31.6 (2.5)	6.2 (0.5)	2.8 (0.1)	5.7 (0.2)
Smtpt	3.0 (0.2)	4.4 (0.3)	21.9 (0.5)	17.3 (0.4)	5.1 (0.4)
Speech	12.2 (1.6)	35.2 (2.6)	33.9 (0.7)	95.9 (0.3)	10.7 (0.4)
Thyroid	16.7 (0.8)	15.5 (1.3)	10.4 (0.4)	9.1 (0.1)	9.3 (0.1)
Vertebral	92.1 (6.7)	99.0 (1.3)	98.6 (0.7)	99.4 (0.2)	99.3 (0.2)
Vowels	22.5 (4.0)	37.0 (1.9)	27.7 (0.8)	8.7 (0.4)	7.8 (0.3)
WBC	17.4 (1.0)	14.7 (1.1)	19.8 (0.7)	18.0 (0.3)	19.3 (0.3)
Wine	8.6 (0.7)	41.5 (3.3)	22.4 (2.1)	9.4 (1.8)	11.8 (1.2)
Average:	22.3 (2.0)	29.1 (1.8)	24.6 (0.7)	23.7 (0.4)	19.6 (0.4)
Median:	16.0 (1.4)	26.9 (1.8)	21.7 (0.7)	17.9 (0.3)	14.3 (0.3)

Table 5.21: False alarm ratio values for the reference methods, along with standard deviations given in parentheses.

Dataset name	IF [%]	LOF [%]	OCSVM [%]	KNN [%]	PCA [%]
Annthroid	32.9 (2.4)	33.4	69.8	35.9	43.6
Arrhythmia	25.0 (3.9)	28.2	68.7	27.7	26.4
BreastW	5.1 (0.4)	60.6	5.9	6.8	4.7
Cardio	20.9 (2.2)	18.3	19.0	19.6	18.5
ForestCover	27.2 (2.4)	19.7	97.2	20.0	16.0
Glass	49.6 (1.6)	36.6	13.7	23.9	68.3
Http	0.4 (0.1)	72.6	0.6	0.5	0.5
Ionosphere	10.4 (1.7)	9.8	7.1	4.9	4.0
Letter	56.4 (4.4)	19.5	16.0	12.5	30.1
Lympho	2.9 (1.1)	7.7	4.9	9.9	7.0
Mammography	21.9 (1.8)	46.8	24.8	35.5	24.2
Mnist	32.4 (4.9)	38.1	100.0	22.5	26.8
Musk	0.3 (0.3)	27.7	23.7	1.7	0.0
Optdigits	47.9 (6.1)	22.5	59.8	0.4	69.3
Pendigits	14.5 (1.1)	13.7	17.5	48.4	17.1
Pima	33.6 (3.9)	74.2	43.2	29.6	31.0
Satellite	8.1 (1.2)	21.0	1.9	18.1	4.3
Satimage-2	2.8 (0.4)	4.7	97.6	25.3	1.3
Shuttle	0.7 (0.2)	37.2	33.8	44.0	0.6
Smtpt	6.7 (0.4)	24.8	0.0	3.5	2.3
Speech	19.0 (4.8)	5.1	5.4	9.1	5.5
Thyroid	7.7 (0.6)	28.3	28.7	10.2	11.1
Vertebral	92.9 (3.9)	53.8	30.5	98.6	96.2
Vowels	37.1 (3.5)	16.0	42.0	5.0	10.7
WBC	20.0 (2.1)	19.6	10.1	17.1	16.8
Wine	21.6 (3.2)	0.8	20.2	1.7	10.9
Average:	23.0 (2.3)	28.5	32.4	20.5	21.0
Median:	20.5 (2.0)	23.7	21.9	17.6	13.5

Table 5.22: F1 score values for the new methods, along with standard deviations given in parentheses.

Dataset name	DRIF [%]	FDRIF [%]	MSTCIF [%]	MSTIF [%]	MSTIF-TS [%]
Annth thyroid	20.2 (1.4)	20.7 (0.8)	27.5 (0.6)	25.8 (0.3)	22.8 (0.3)
Arrhythmia	48.3 (0.6)	35.2 (2.1)	44.4 (1.4)	43.8 (1.0)	43.1 (0.7)
BreastW	93.2 (0.8)	88.8 (1.0)	92.0 (0.3)	93.3 (0.1)	93.6 (0.2)
Cardio	61.9 (2.2)	27.8 (1.7)	47.4 (1.2)	55.5 (0.6)	49.3 (0.5)
ForestCover	9.4 (0.8)	4.7 (0.2)	11.1 (0.4)	7.7 (0.1)	6.4 (0.1)
Glass	14.7 (0.6)	28.8 (1.6)	15.1 (1.5)	15.9 (1.2)	15.3 (1.4)
Http	54.7 (0.9)	51.2 (11.9)	87.6 (3.7)	57.9 (3.6)	58.4 (8.4)
Ionosphere	68.7 (2.6)	61.1 (2.0)	84.2 (0.9)	88.4 (0.6)	88.4 (0.3)
Letter	18.2 (1.1)	16.6 (0.7)	21.8 (0.8)	23.4 (0.4)	24.4 (0.3)
Lympho	31.7 (3.4)	40.0 (3.6)	36.4 (4.7)	23.1 (1.5)	24.5 (2.0)
Mammography	13.8 (0.7)	14.3 (0.5)	11.7 (0.3)	11.4 (0.1)	15.6 (0.2)
Mnist	24.7 (2.1)	25.0 (3.4)	35.8 (0.8)	39.6 (0.3)	38.7 (0.2)
Musk	92.3 (8.7)	96.8 (3.6)	76.4 (5.1)	99.1 (2.0)	89.5 (2.2)
Optdigits	5.3 (0.5)	5.5 (0.2)	7.0 (0.4)	12.2 (0.9)	21.3 (2.4)
Pendigits	16.4 (1.3)	10.9 (0.6)	22.3 (0.8)	31.4 (1.0)	30.4 (1.0)
Pima	59.7 (1.5)	46.7 (2.0)	58.6 (0.7)	59.8 (0.4)	59.4 (0.4)
Satellite	52.6 (0.7)	60.0 (0.9)	63.4 (1.0)	61.5 (0.4)	65.1 (0.9)
Satimage-2	67.7 (6.3)	66.0 (4.9)	83.2 (1.9)	74.0 (1.2)	70.1 (0.7)
Shuttle	92.2 (1.7)	28.2 (1.2)	71.0 (1.7)	84.6 (0.7)	72.9 (0.7)
Smt p	1.4 (0.1)	1.1 (0.1)	0.2 (0.0)	0.3 (0.0)	0.9 (0.1)
Speech	4.6 (0.6)	3.8 (0.3)	2.5 (0.2)	3.1 (0.1)	3.4 (0.3)
Thyroid	21.3 (0.6)	21.9 (1.1)	31.1 (0.7)	34.0 (0.3)	33.7 (0.3)
Vertebral	21.5 (1.2)	21.7 (0.7)	22.0 (0.6)	22.0 (0.3)	22.0 (0.3)
Vowels	15.3 (1.5)	15.3 (0.5)	17.5 (0.9)	42.9 (1.1)	44.7 (1.2)
WBC	36.5 (1.0)	32.9 (1.6)	36.0 (1.7)	37.1 (1.4)	35.9 (0.8)
Wine	56.2 (11.8)	24.9 (1.6)	39.8 (3.1)	60.0 (2.9)	55.8 (4.6)
Average:	38.6 (2.1)	32.7 (1.9)	40.2 (1.4)	42.6 (0.9)	41.8 (1.2)
Median:	28.2 (1.1)	26.4 (1.2)	35.9 (0.9)	38.3 (0.6)	37.3 (0.6)

Table 5.23: F1 score values for the reference methods, along with standard deviations given in parentheses.

Dataset name	IF [%]	LOF [%]	OCSVM [%]	KNN [%]	PCA [%]
Annthroid	26.9 (1.3)	23.4	14.9	23.9	19.3
Arrhythmia	44.5 (2.1)	45.1	24.4	46.2	43.0
BreastW	94.6 (0.4)	52.3	93.4	93.3	92.8
Cardio	46.7 (2.6)	19.9	51.2	31.9	52.8
ForestCover	6.1 (0.5)	3.0	1.9	6.0	10.4
Glass	13.2 (2.1)	17.4	19.5	26.9	10.2
Http	62.0 (12.5)	1.0	57.8	2.9	60.6
Ionosphere	66.3 (1.4)	77.7	65.4	87.7	60.1
Letter	14.3 (0.8)	34.5	38.1	43.2	13.2
Lympho	69.1 (8.7)	52.2	55.6	46.2	47.6
Mammography	14.8 (0.8)	7.7	14.3	10.7	14.8
Mnist	32.0 (2.3)	21.7	16.9	34.6	36.8
Musk	93.5 (6.2)	6.3	12.0	4.0	99.5
Optdigits	8.8 (0.9)	9.3	5.9	2.3	6.8
Pendigits	23.4 (1.6)	6.1	19.9	7.3	20.6
Pima	54.4 (1.5)	52.6	46.3	49.8	54.1
Satellite	58.6 (1.9)	34.0	4.1	49.2	53.0
Satimage-2	45.6 (3.2)	9.6	2.4	8.8	61.8
Shuttle	94.5 (1.3)	15.3	22.9	20.1	93.8
Smtpt	0.7 (0.0)	0.2	70.2	1.2	1.9
Speech	2.9 (0.6)	5.5	4.6	3.5	3.7
Thyroid	38.0 (1.7)	12.1	12.7	30.9	29.6
Vertebral	21.6 (1.2)	23.5	29.1	22.5	22.2
Vowels	12.2 (1.0)	29.2	12.0	54.5	14.0
WBC	33.8 (2.3)	34.5	48.0	39.2	38.0
Wine	32.8 (4.2)	90.0	16.2	90.9	41.4
Average:	38.9 (2.4)	26.3	29.2	32.2	38.5
Median:	33.3 (1.6)	20.8	19.7	28.9	37.4

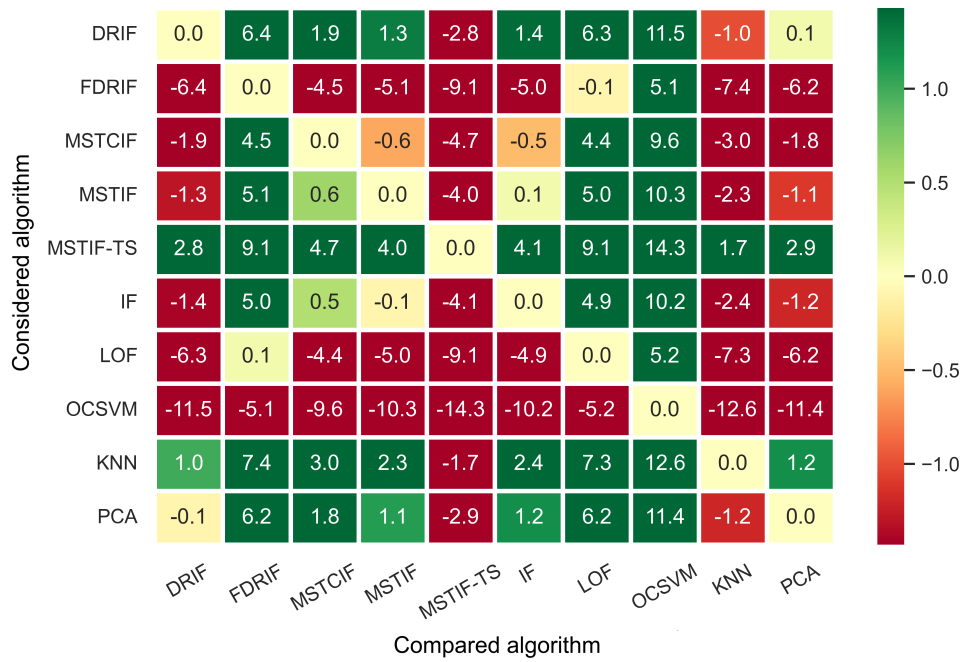


Figure 5.10: Heatmap presenting the difference in the average accuracy scores (in percentage) for all datasets, determined between the compared algorithms.

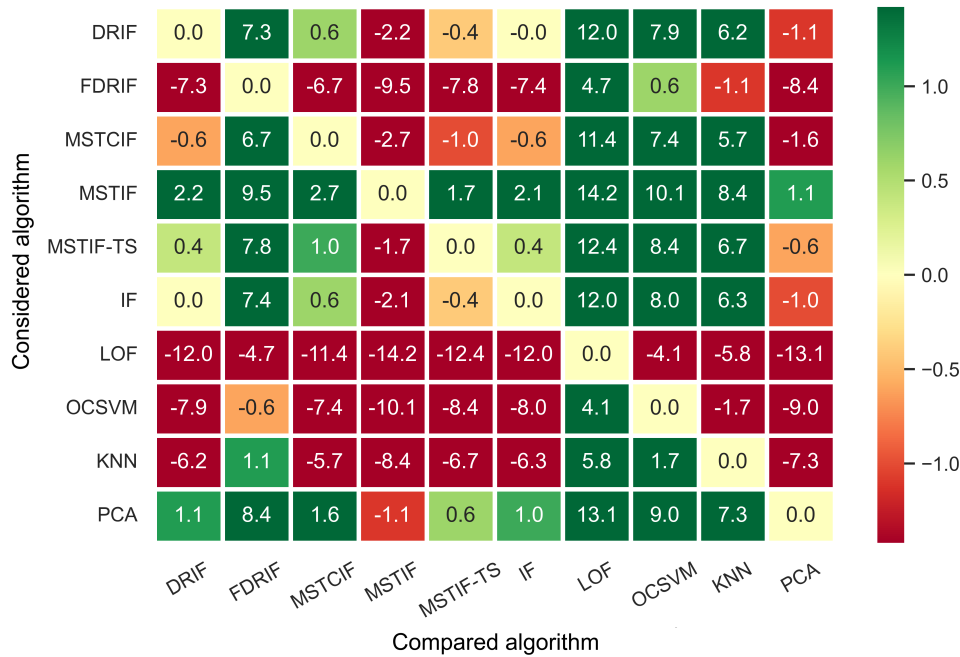


Figure 5.11: Heatmap presenting the difference in average precision scores (in percentage) for all datasets, determined between the compared algorithms.

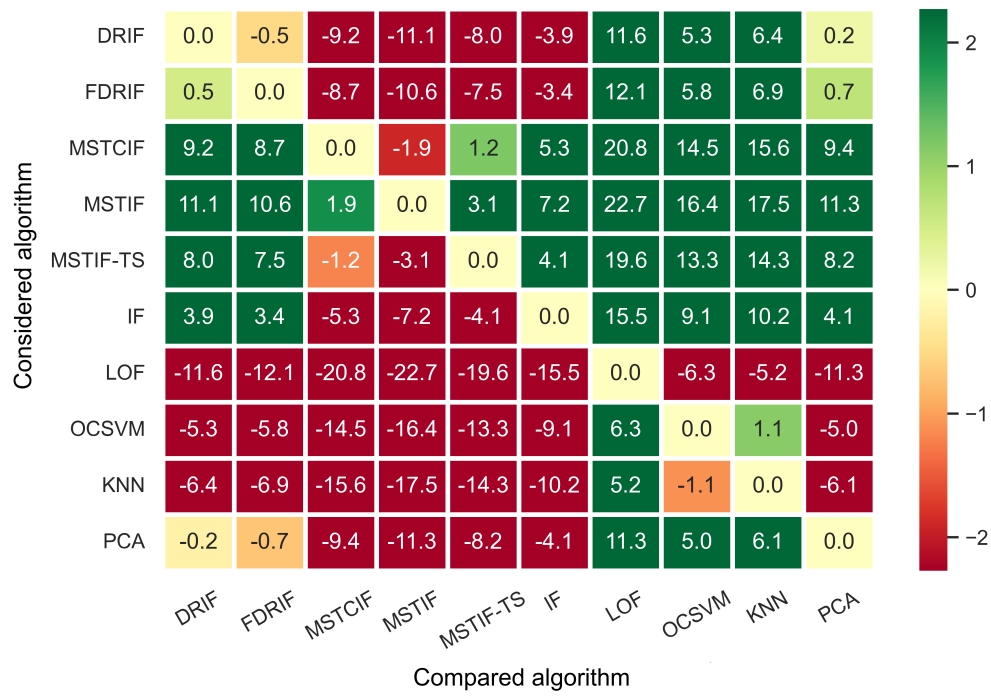


Figure 5.12: Heatmap presenting the difference in the average recall scores (in percentage) for all datasets, determined between the compared algorithms.

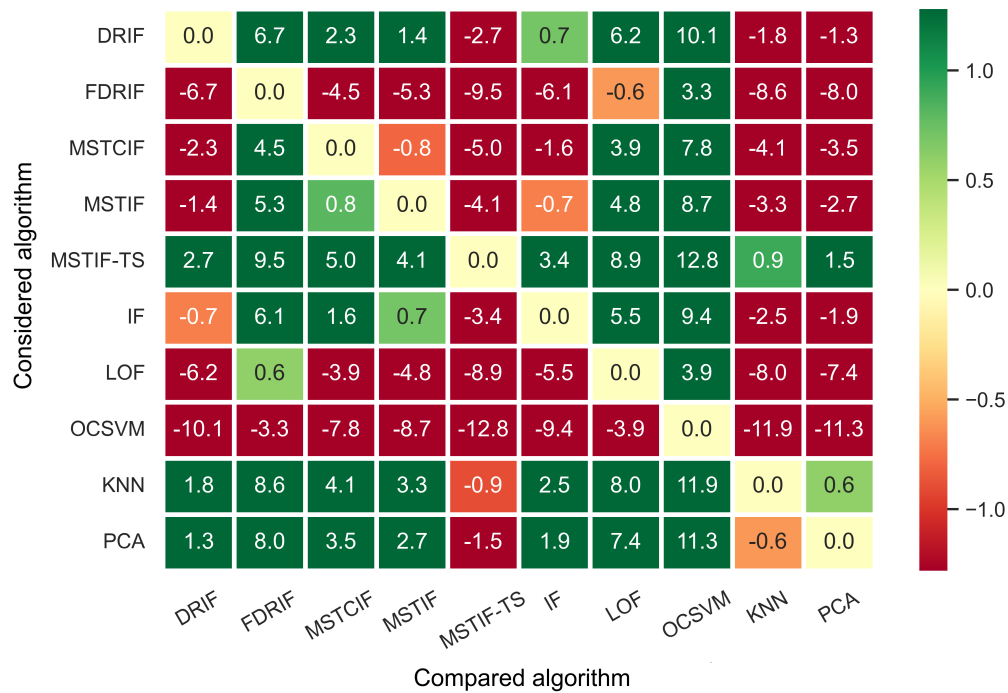


Figure 5.13: Heatmap presenting the difference in the mean specificity scores (in percentage) across all datasets, determined between the compared algorithms.

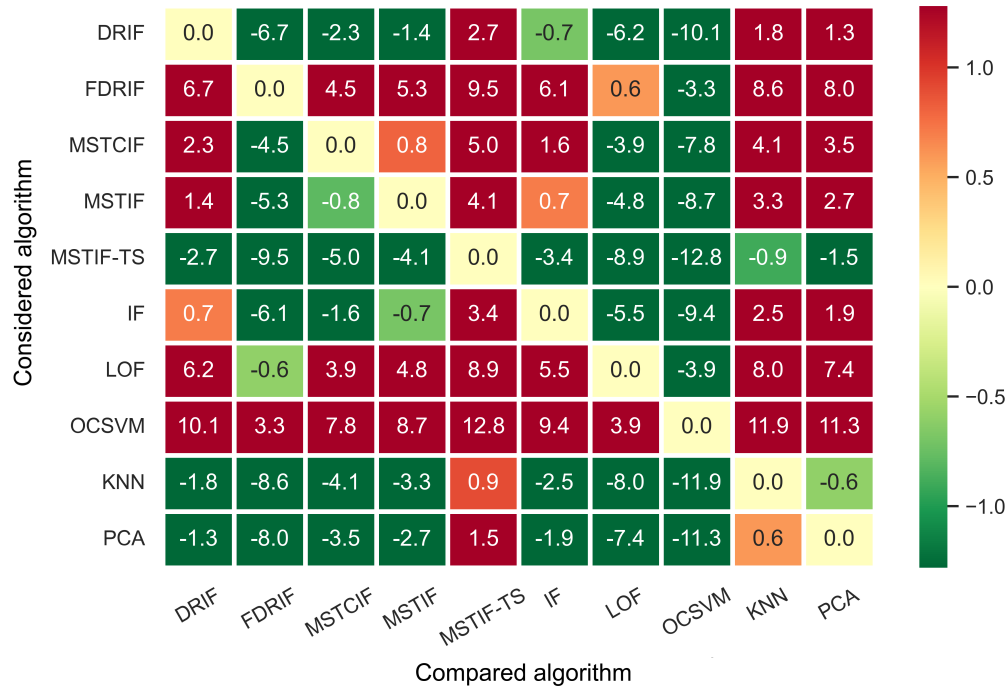


Figure 5.14: Heatmap presenting the difference in the average false alarm rate scores (in percentage) for all datasets, determined between the compared algorithms.

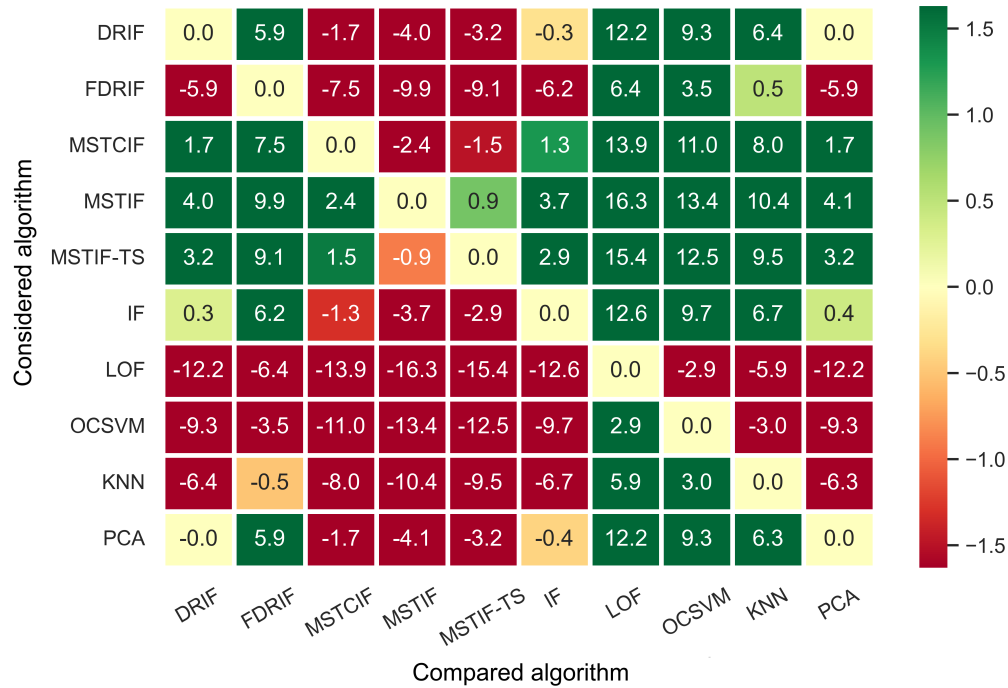


Figure 5.15: Heatmap presenting the difference in average F1 scores (in percentage) across all datasets, determined between the compared algorithms.

5.3 Threshold-dependent evaluation measures

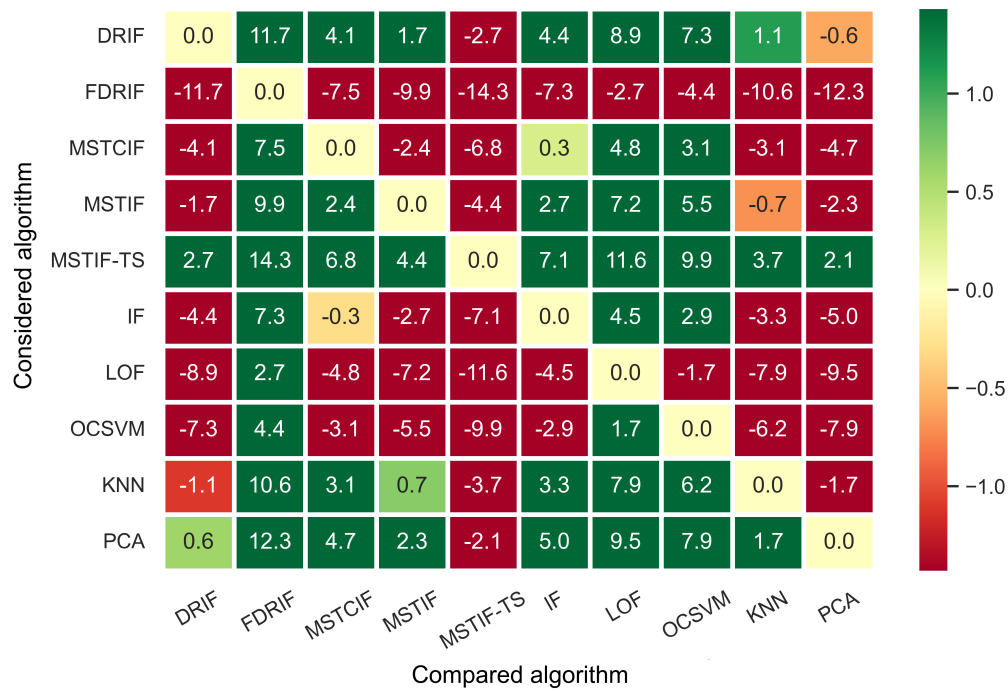


Figure 5.16: Heatmap presenting the difference in median accuracy scores (in percentage) across all datasets, determined between the compared algorithms.

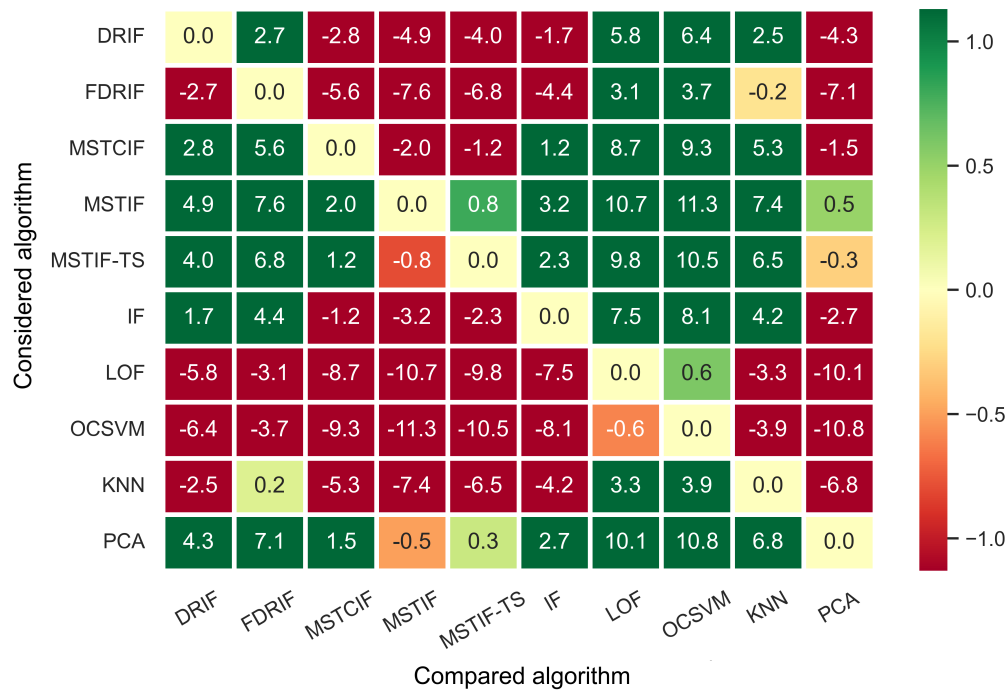


Figure 5.17: Heatmap presenting the difference in the median precision scores (in percentage) for all datasets, determined between the compared algorithms.

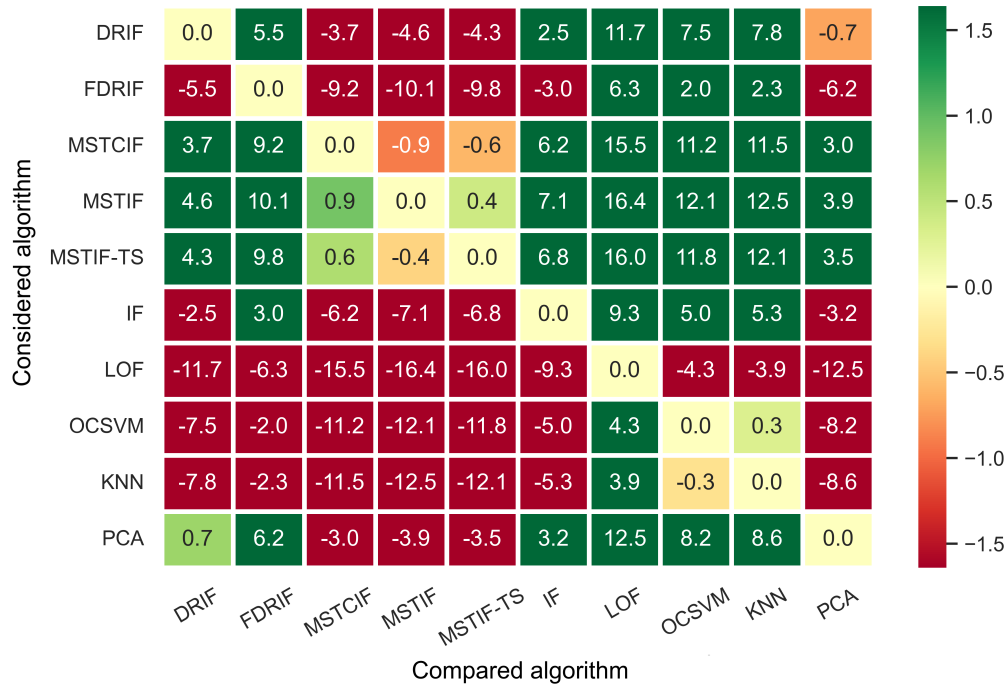


Figure 5.18: Heatmap presenting the difference in the median recall scores (in percentage) for all datasets, determined between the compared algorithms.

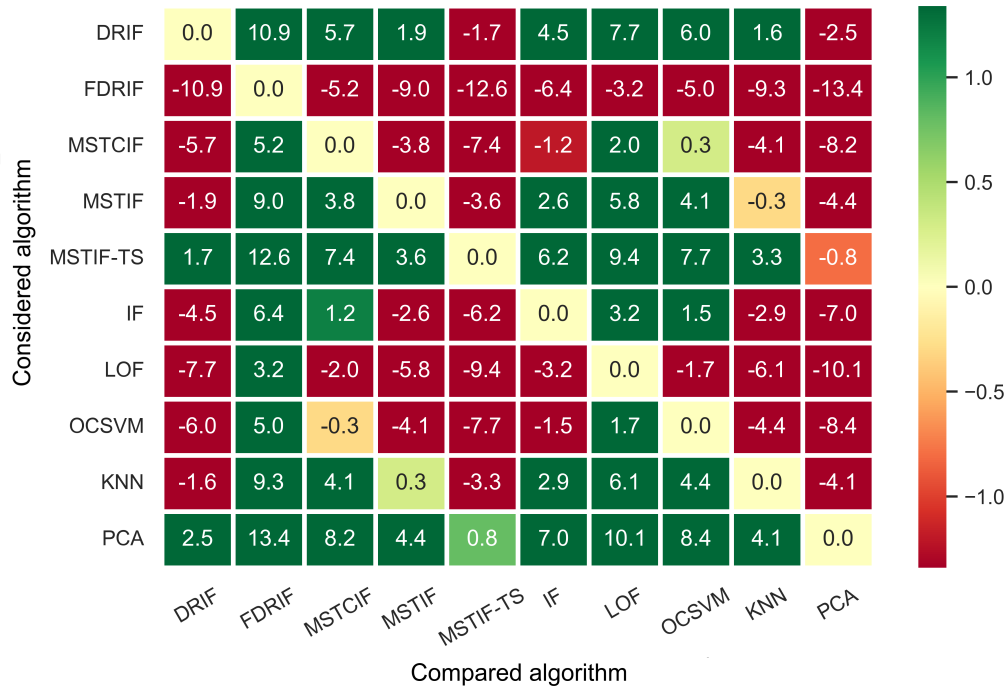


Figure 5.19: Heatmap presenting the difference in the median specificity scores (in percentage) across all datasets, determined between the compared algorithms.

5.3 Threshold-dependent evaluation measures

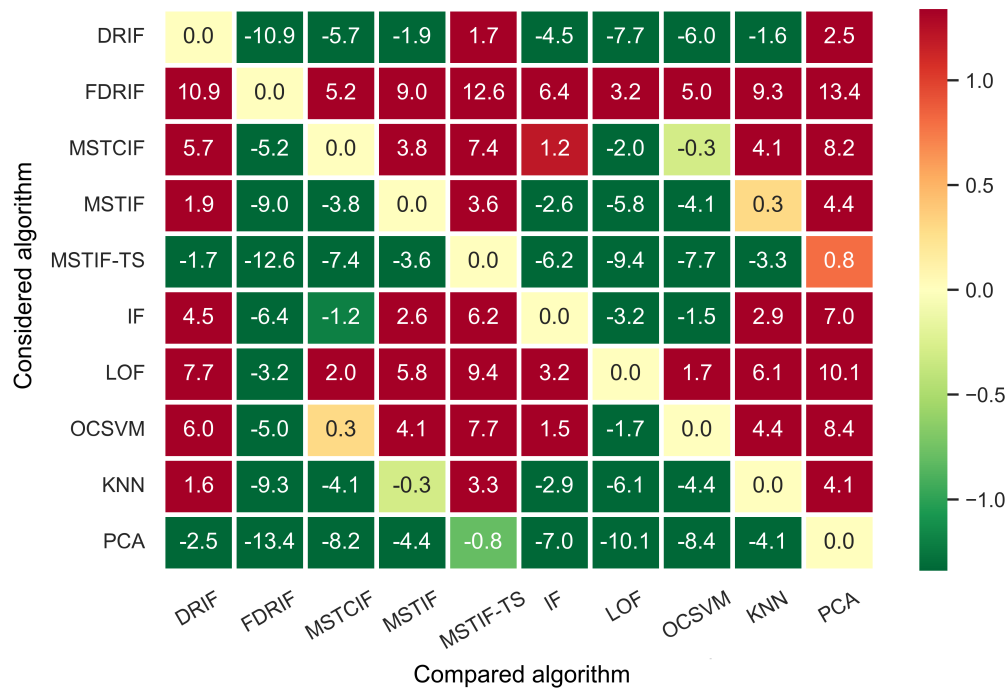


Figure 5.20: Heatmap presenting the difference in median false alarm ratio scores (in percentage) for all datasets, determined between the compared algorithms.

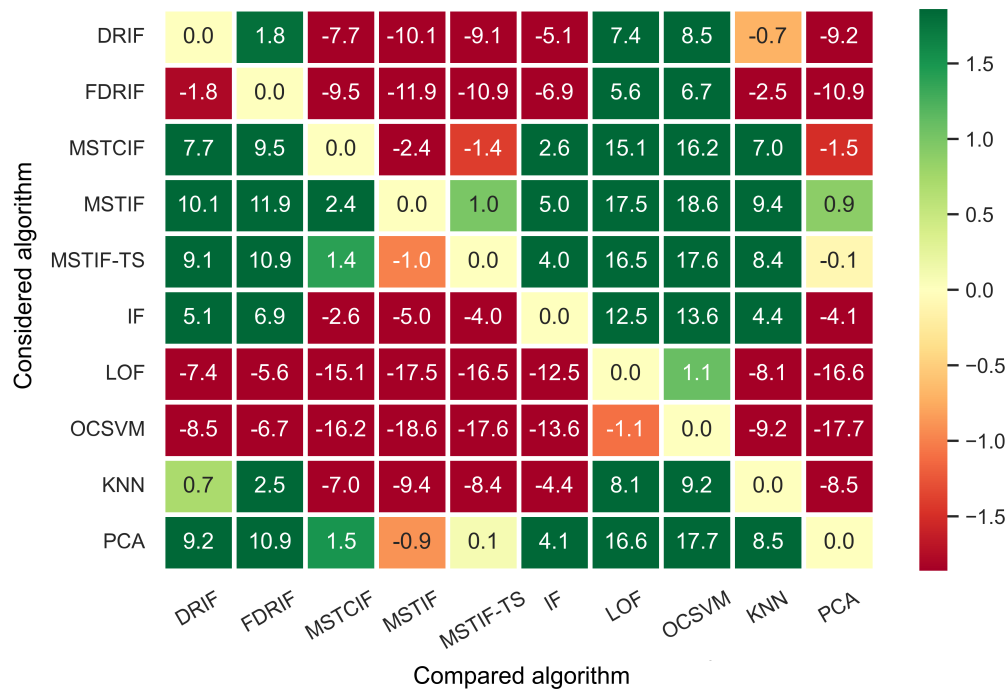


Figure 5.21: Heatmap presenting the difference in the median F1 scores (in percentage) for all datasets, determined between the compared algorithms.

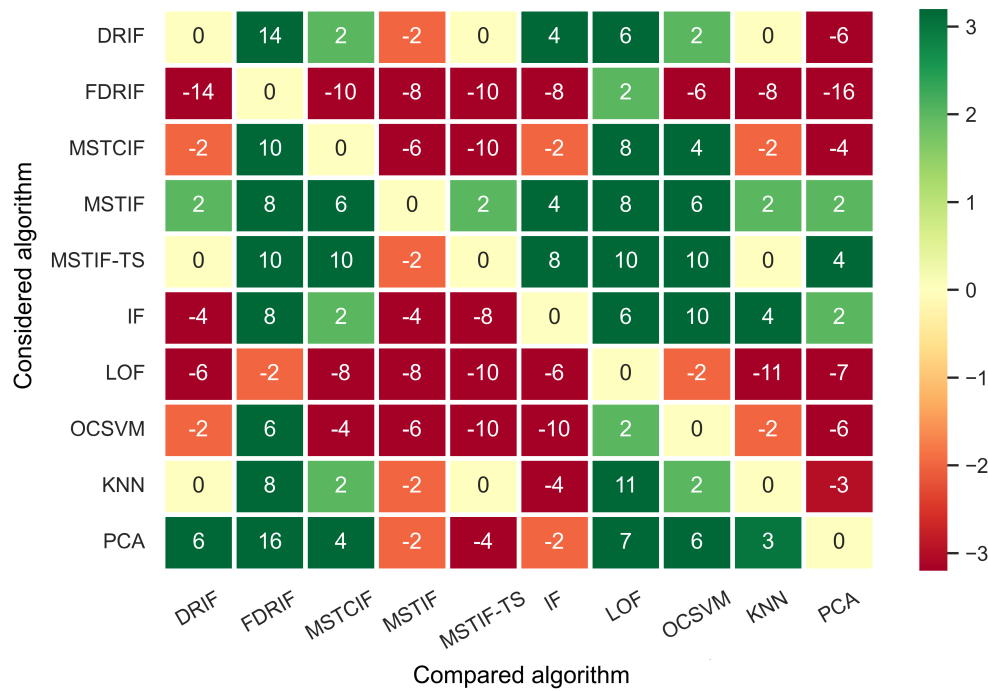


Figure 5.22: Heatmap presenting the difference in the number of better and worse accuracy scores, determined between the compared algorithms.

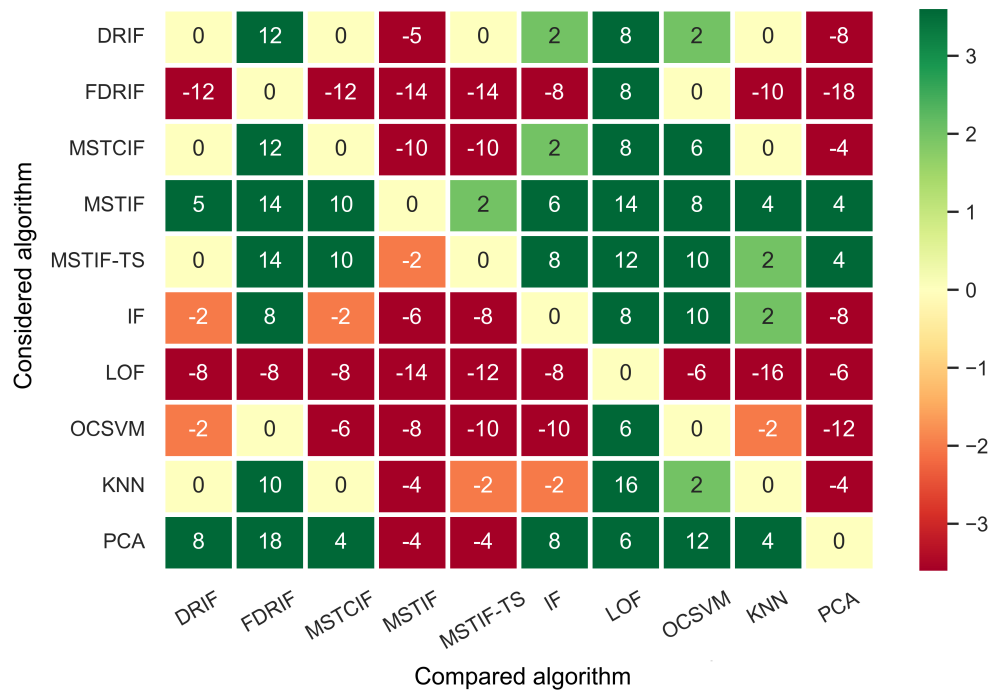


Figure 5.23: Heatmap presenting the differences in the number of better and worse precision scores determined between the compared algorithms.

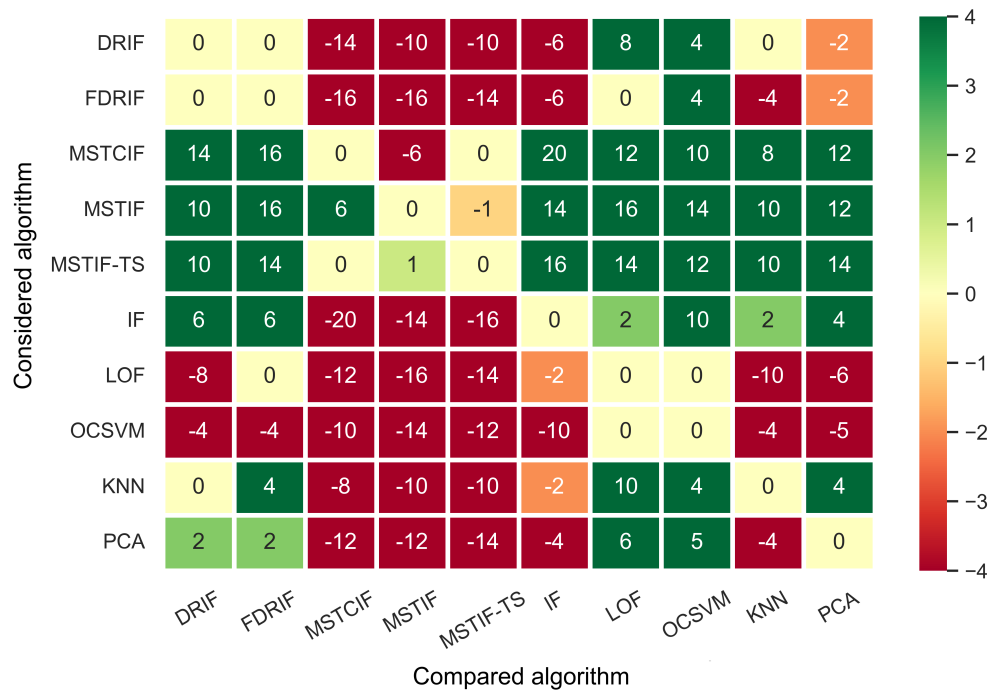


Figure 5.24: Heatmap presenting differences in the number of better and worse recall scores, determined between the compared algorithms.

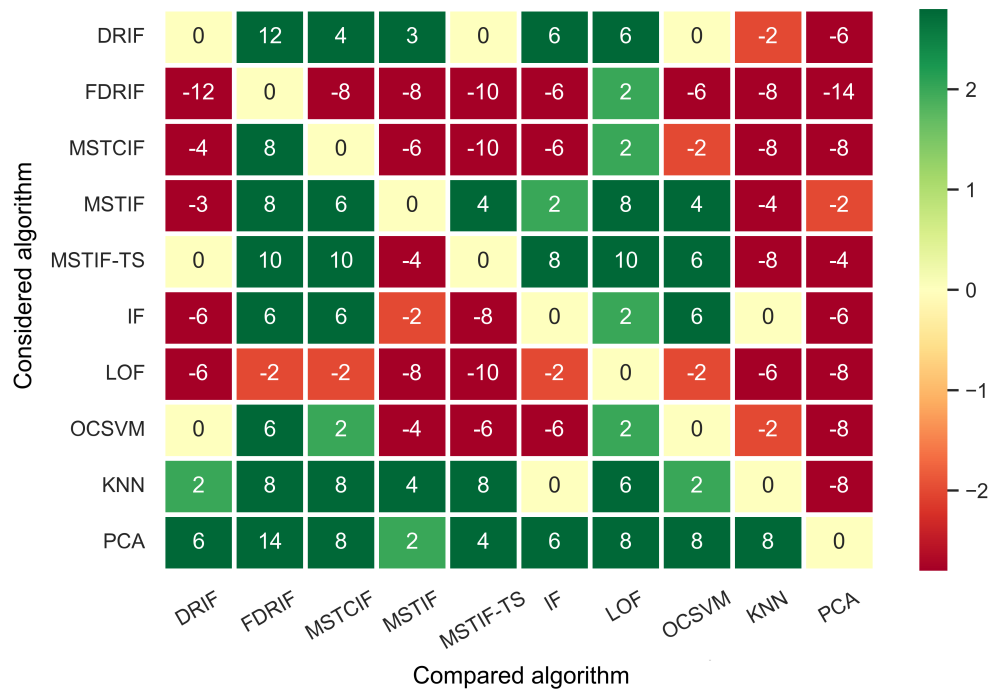


Figure 5.25: Heatmap presenting the difference in the number of better and worse specificity scores determined between the compared algorithms.

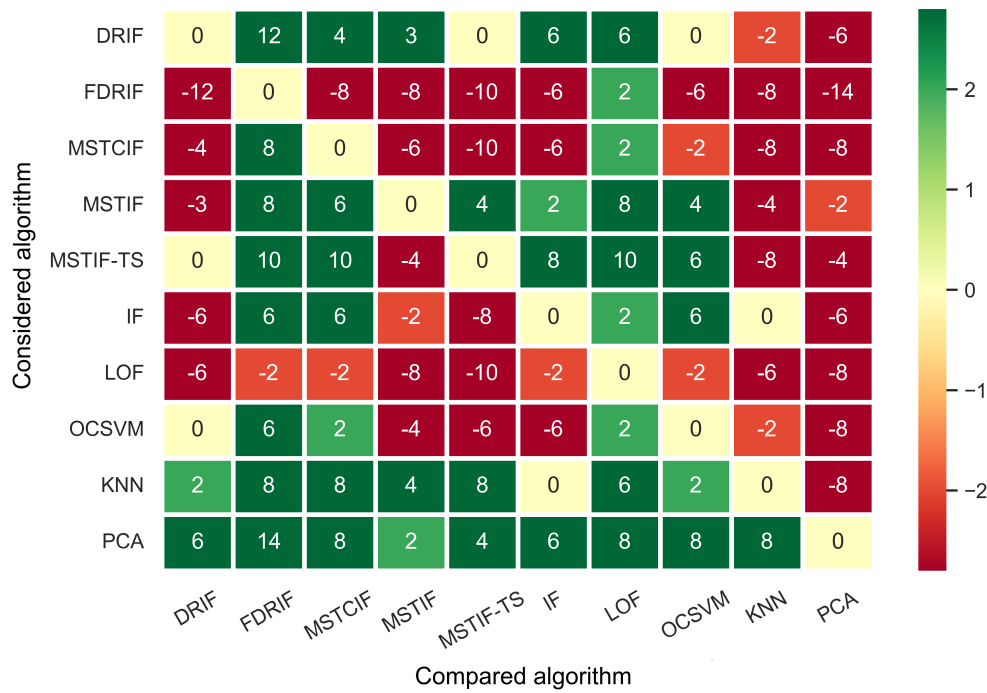


Figure 5.26: Heatmap presenting the difference in the number of better and worse false alarm ratio scores, determined between the compared algorithms.

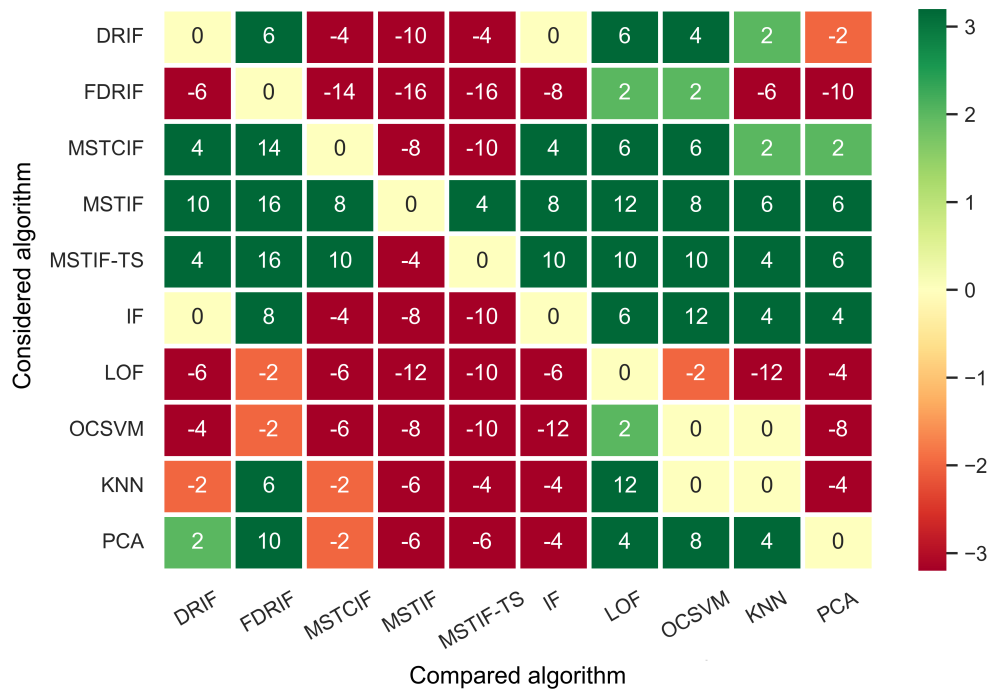


Figure 5.27: Heatmap presenting the differences in the number of better and worse F1 scores, determined between the compared algorithms.

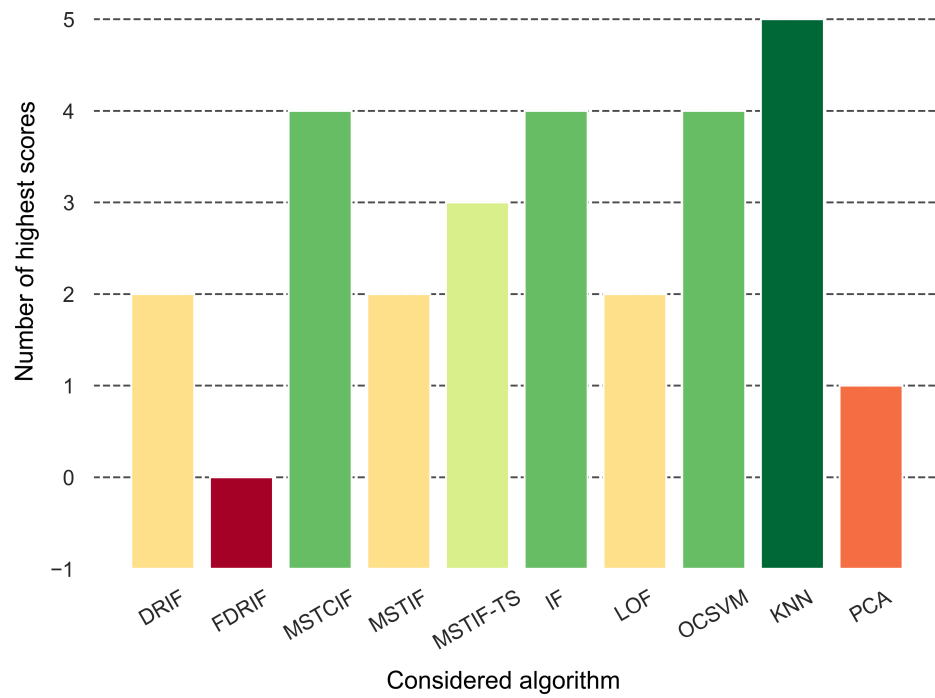


Figure 5.28: Chart presenting the number of cases in which particular algorithms achieved the highest accuracy scores for a given dataset, among all analyzed methods.

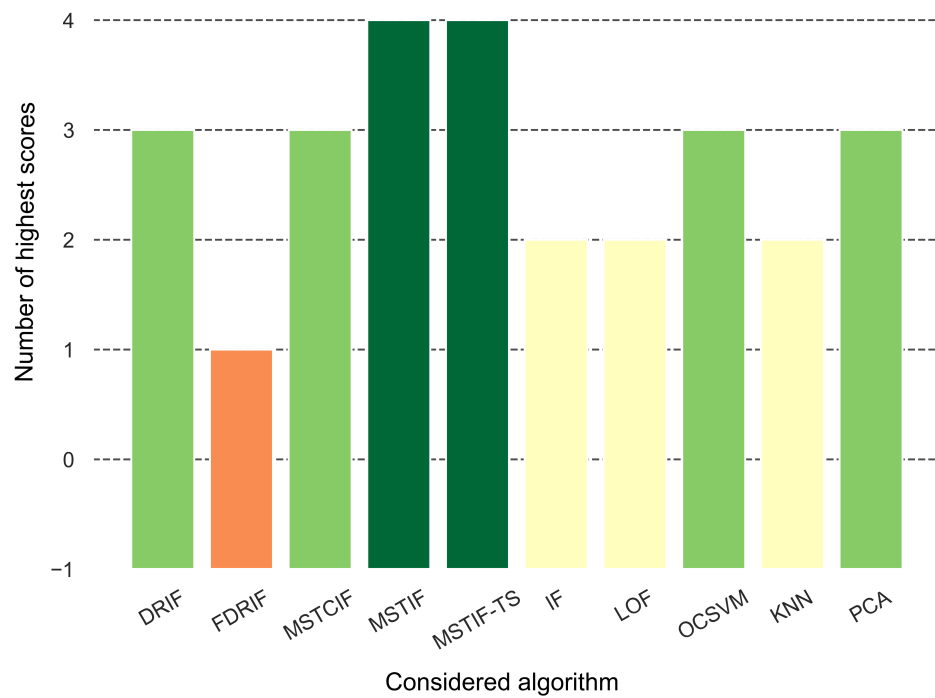


Figure 5.29: Chart presenting the number of cases in which individual algorithms achieved the highest precision scores for a given dataset among all analyzed methods.

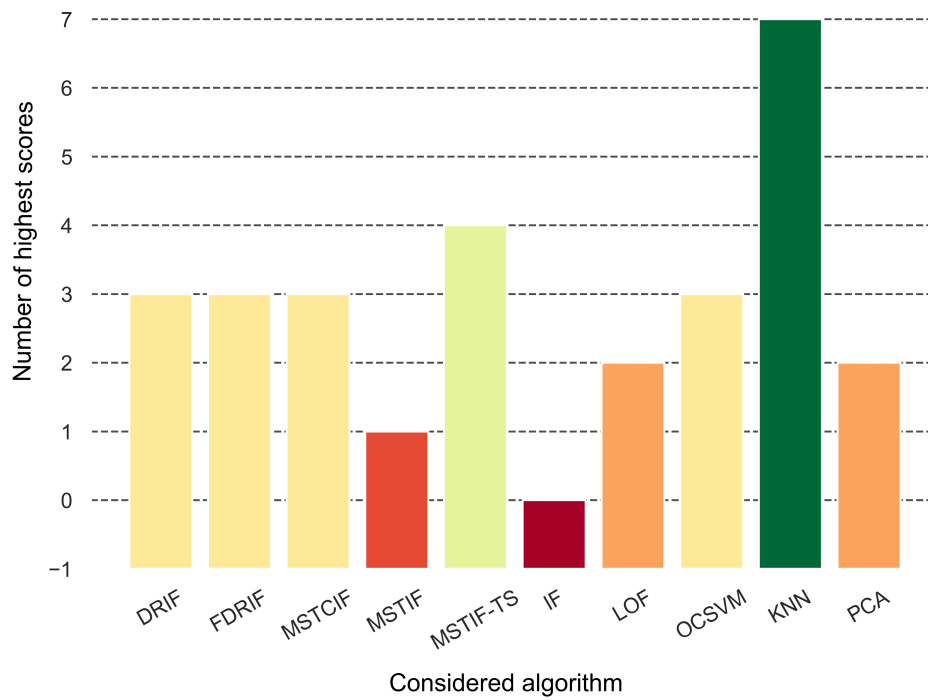


Figure 5.30: Chart presenting the number of cases in which particular algorithms achieved the highest recall scores for a given dataset, among all analyzed methods.

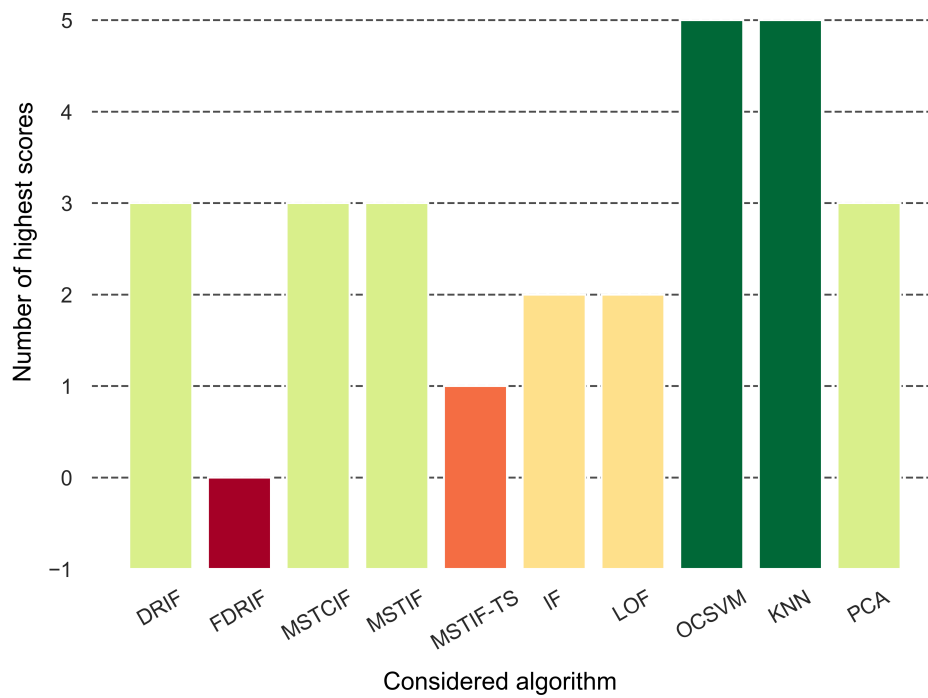


Figure 5.31: Chart presenting the number of cases in which the individual algorithms achieved the highest specificity scores for a given dataset among all analyzed methods.

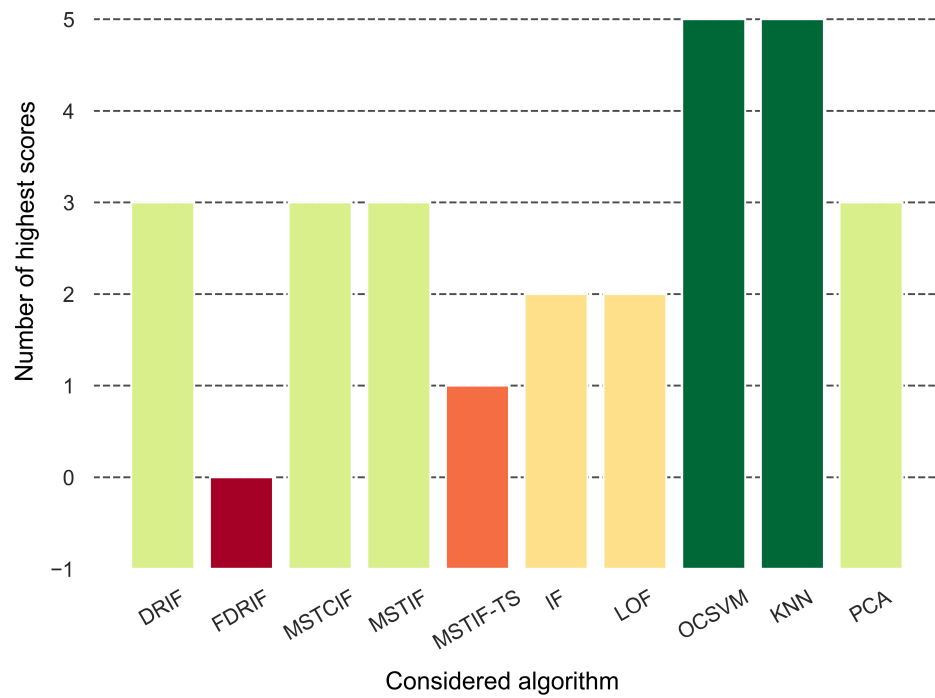


Figure 5.32: Chart presenting the number of cases in which particular algorithms achieved the lowest false alarm ratio scores for a given dataset among all analyzed methods.

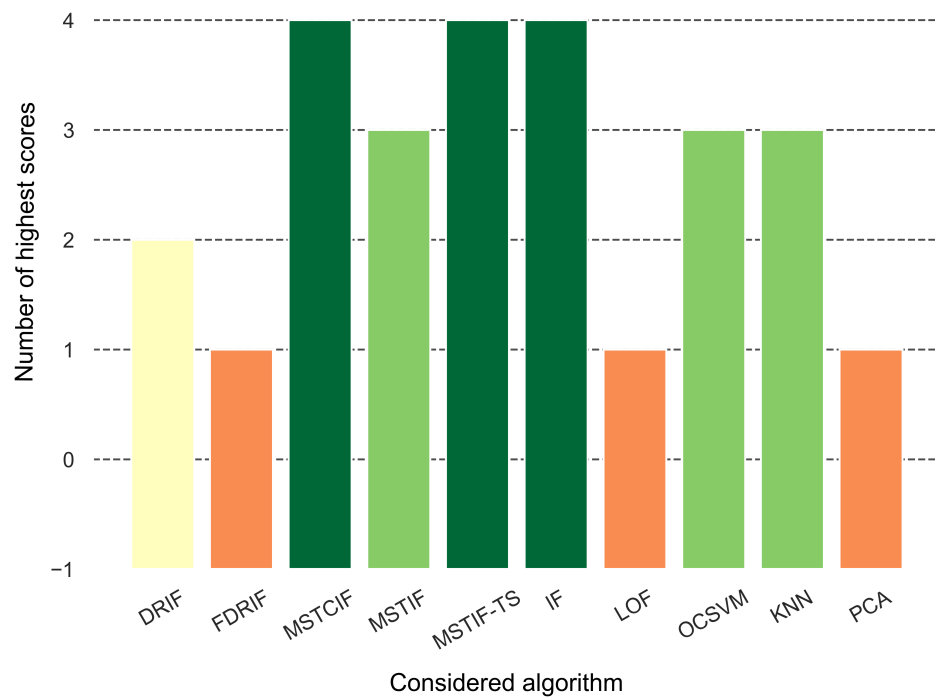


Figure 5.33: Chart presenting the number of cases in which particular algorithms achieved the highest F1 scores for a given dataset among all analyzed methods.

Table 5.24: Wilcoxon test p-values for the accuracy measure corresponding to the compared algorithm pairs. Part I. Significance level $\alpha = 0.05$. Statistically significant results are marked in bold.

Algorithm	DRIF	FDRIF	MSTCIF	MSTIF	MSTIF-TS
DRIF	-	0.0023	0.1702	0.4503	0.5497
FDRIF	0.9979	-	0.9645	0.9447	0.9935
MSTCIF	0.8362	0.0376	-	0.9286	0.9898
MSTIF	0.5595	0.0582	0.0750	-	0.5100
MSTIF-TS	0.4602	0.0070	0.0110	0.5000	-
IF	0.8099	0.0315	0.3174	0.8601	0.9844
LOF	0.9213	0.5398	0.8362	0.9387	0.9796
OCSVM	0.9476	0.4111	0.8865	0.9503	0.9905
KNN	0.5398	0.0497	0.3174	0.4900	0.5199
PCA	0.2262	0.0013	0.0825	0.6826	0.7886

Table 5.25: Wilcoxon test p-values for the accuracy measure corresponding to the compared algorithm pairs. Part II. Significance level $\alpha = 0.05$. Statistically significant results are marked in bold.

Algorithm	IF	LOF	OCSVM	KNN	PCA
DRIF	0.1970	0.0825	0.0553	0.4701	0.7813
FDRIF	0.9703	0.4701	0.5986	0.9530	0.9988
MSTCIF	0.6915	0.1702	0.1185	0.6915	0.9213
MSTIF	0.1457	0.0646	0.0524	0.5199	0.3264
MSTIF-TS	0.0167	0.0217	0.0102	0.4900	0.2187
IF	-	0.1901	0.0445	0.6178	0.6826
LOF	0.8167	-	0.4208	0.9789	0.9086
OCSVM	0.9579	0.5889	-	0.9447	0.9530
KNN	0.3918	0.0211	0.0582	-	0.6112
PCA	0.3264	0.0914	0.0497	0.3888	-

Table 5.26: Wilcoxon test p-values for the precision measure corresponding to the compared algorithm pairs. Part I. Significance level $\alpha = 0.05$. Statistically significant results are marked in bold.

Algorithm	DRIF	FDRIF	MSTCIF	MSTIF	MSTIF-TS
DRIF	-	0.0023	0.4801	0.8736	0.6273
FDRIF	0.9979	-	0.9974	0.9995	0.9984
MSTCIF	0.5299	0.0028	-	0.9624	0.9250
MSTIF	0.1264	0.0005	0.0398	-	0.2417
MSTIF-TS	0.3822	0.0018	0.0787	0.7661	-
IF	0.6178	0.0118	0.5299	0.9006	0.8424
LOF	0.9602	0.8484	0.9833	0.9974	0.9953
OCSVM	0.8484	0.3822	0.9354	0.9796	0.9833
KNN	0.5694	0.0524	0.5986	0.8865	0.8764
PCA	0.2339	0.0007	0.1236	0.8865	0.8167

Table 5.27: Wilcoxon test p-values for the precision measure corresponding to the compared algorithm pairs. Part II. Significance level $\alpha = 0.05$. Statistically significant results are marked in bold.

Algorithm	IF	LOF	OCSVM	KNN	PCA
DRIF	0.3918	0.0421	0.1576	0.4405	0.7738
FDRIF	0.9890	0.1576	0.6273	0.9503	0.9994
MSTCIF	0.4801	0.0179	0.0679	0.4111	0.8815
MSTIF	0.1039	0.0028	0.0217	0.1185	0.1185
MSTIF-TS	0.1638	0.0051	0.0179	0.1289	0.1901
IF	-	0.0167	0.0376	0.2041	0.9093
LOF	0.9844	-	0.8656	0.9995	0.9821
OCSVM	0.9645	0.1399	-	0.8099	0.9703
KNN	0.8030	0.0006	0.1970	-	0.8099
PCA	0.0950	0.0191	0.0315	0.1970	-

Table 5.28: Wilcoxon test p-values for the recall measure corresponding to the compared algorithm pairs. Part I. Significance level $\alpha = 0.05$. Statistically significant results are marked in bold.

Algorithm	DRIF	FDRIF	MSTCIF	MSTIF	MSTIF-TS
DRIF	-	0.4900	0.9996	0.9935	0.9854
FDRIF	0.5199	-	0.9972	0.9992	0.9890
MSTCIF	0.0005	0.0031	-	0.8099	0.4405
MSTIF	0.0070	0.0009	0.1970	-	0.4946
MSTIF-TS	0.0156	0.0118	0.5694	0.5054	-
IF	0.1086	0.1457	1.0000	0.9990	0.9890
LOF	0.9476	0.9213	0.9988	0.9996	0.9985
OCSVM	0.8484	0.7504	0.9989	0.9982	0.9940
KNN	0.5792	0.5497	0.9898	0.9949	0.9854
PCA	0.3539	0.3822	0.9989	0.9935	0.9940

Table 5.29: Wilcoxon test p-values for the recall measure corresponding to the compared algorithm pairs. Part II. Significance level $\alpha = 0.05$. Statistically significant results are marked in bold.

Algorithm	IF	LOF	OCSVM	KNN	PCA
DRIF	0.8961	0.0553	0.1576	0.4306	0.6553
FDRIF	0.8601	0.0825	0.2577	0.4602	0.6273
MSTCIF	0.0000	0.0013	0.0012	0.0110	0.0012
MSTIF	0.0011	0.0005	0.0020	0.0055	0.0070
MSTIF-TS	0.0118	0.0016	0.0065	0.0156	0.0065
IF	-	0.0355	0.0279	0.2114	0.0865
LOF	0.9666	-	0.7174	0.9149	0.9149
OCSVM	0.9737	0.2911	-	0.6341	0.8422
KNN	0.7959	0.0851	0.3659	-	0.5299
PCA	0.9175	0.0851	0.1578	0.4900	-

Table 5.30: Wilcoxon test p-values for the specificity measure corresponding to the compared algorithm pairs. Part I. Significance level $\alpha = 0.05$. Statistically significant results are marked in bold.

Algorithm	DRIF	FDRIF	MSTCIF	MSTIF	MSTIF-TS
DRIF	-	0.0088	0.1457	0.3888	0.5497
FDRIF	0.9918	-	0.9354	0.9250	0.9864
MSTCIF	0.8601	0.0679	-	0.9286	0.9873
MSTIF	0.6112	0.0787	0.0750	-	0.4111
MSTIF-TS	0.4602	0.0146	0.0136	0.5986	-
IF	0.7813	0.0398	0.0865	0.7259	0.9530
LOF	0.9093	0.4900	0.7661	0.9135	0.9579
OCSVM	0.8484	0.3447	0.6367	0.9050	0.9685
KNN	0.5100	0.0553	0.1399	0.2496	0.2114
PCA	0.2114	0.0031	0.0076	0.2741	0.3539

Table 5.31: Wilcoxon test p-values for the specificity measure corresponding to the compared algorithm pairs. Part II. Significance level $\alpha = 0.05$. Statistically significant results are marked in bold.

Algorithm	IF	LOF	OCSVM	KNN	PCA
DRIF	0.2262	0.0950	0.1576	0.5000	0.7959
FDRIF	0.9624	0.5199	0.6645	0.9476	0.9972
MSTCIF	0.9175	0.2417	0.3727	0.8656	0.9930
MSTIF	0.2826	0.0907	0.0994	0.7583	0.7342
MSTIF-TS	0.0497	0.0445	0.0334	0.7959	0.6553
IF	-	0.2339	0.1135	0.7089	0.8711
LOF	0.7738	-	0.4900	0.9666	0.9476
OCSVM	0.8914	0.5199	-	0.9135	0.9530
KNN	0.2998	0.0355	0.0907	-	0.7504
PCA	0.1344	0.0553	0.0497	0.2577	-

Table 5.32: Wilcoxon test p-values for the false alarm ratio corresponding to the compared algorithm pairs. Part I. Significance level $\alpha = 0.05$. Statistically significant results are marked in bold.

Algorithm	DRIF	FDRIF	MSTCIF	MSTIF	MSTIF-TS
DRIF	-	0.0088	0.1457	0.3888	0.5497
FDRIF	0.9918	-	0.9354	0.9250	0.9864
MSTCIF	0.8601	0.0679	-	0.9286	0.9873
MSTIF	0.6112	0.0787	0.0750	-	0.4111
MSTIF-TS	0.4602	0.0146	0.0136	0.5986	-
IF	0.7813	0.0398	0.0865	0.7259	0.9530
LOF	0.9093	0.4900	0.7661	0.9135	0.9579
OCSVM	0.8484	0.3447	0.6367	0.9050	0.9685
KNN	0.5100	0.0553	0.1399	0.2496	0.2114
PCA	0.2114	0.0031	0.0076	0.2741	0.3539

Table 5.33: Wilcoxon test p-values for the false alarm ratio corresponding to the compared algorithm pairs. Part II. Significance level $\alpha = 0.05$. Statistically significant results are marked in bold.

Algorithm	IF	LOF	OCSVM	KNN	PCA
DRIF	0.2262	0.0950	0.1576	0.5000	0.7959
FDRIF	0.9624	0.5199	0.6645	0.9476	0.9972
MSTCIF	0.9175	0.2417	0.3727	0.8656	0.9930
MSTIF	0.2826	0.0907	0.0994	0.7583	0.7342
MSTIF-TS	0.0497	0.0445	0.0334	0.7959	0.6553
IF	-	0.2339	0.1135	0.7089	0.8711
LOF	0.7738	-	0.4900	0.9666	0.9476
OCSVM	0.8914	0.5199	-	0.9135	0.9530
KNN	0.2998	0.0355	0.0907	-	0.7504
PCA	0.1344	0.0553	0.0497	0.2577	-

Table 5.34: Wilcoxon test p-values for the F1 score corresponding to the compared algorithm pairs. Part I. Significance level $\alpha = 0.05$. Statistically significant results are marked in bold.

Algorithm	DRIF	FDRIF	MSTCIF	MSTIF	MSTIF-TS
DRIF	-	0.0421	0.8030	0.9796	0.9050
FDRIF	0.9602	-	0.9985	0.9997	0.9995
MSTCIF	0.2041	0.0016	-	0.9476	0.9476
MSTIF	0.0217	0.0003	0.0553	-	0.1516
MSTIF-TS	0.0994	0.0005	0.0553	0.8543	-
IF	0.4801	0.0102	0.8543	0.9666	0.9530
LOF	0.9530	0.7583	0.9873	0.9979	0.9964
OCSVM	0.9213	0.5889	0.9737	0.9854	0.9864
KNN	0.6461	0.1833	0.8233	0.9213	0.9093
PCA	0.4602	0.0076	0.7738	0.9844	0.9354

Table 5.35: Wilcoxon test p-values for the F1 score corresponding to the compared algorithm pairs. Part II. Significance level $\alpha = 0.05$. Statistically significant results are marked in bold.

Algorithm	IF	LOF	OCSVM	KNN	PCA
DRIF	0.5299	0.0497	0.0825	0.3633	0.5497
FDRIF	0.9905	0.2496	0.4208	0.8233	0.9930
MSTCIF	0.1516	0.0136	0.0279	0.1833	0.2339
MSTIF	0.0355	0.0023	0.0156	0.0825	0.0167
MSTIF-TS	0.0497	0.0040	0.0146	0.0950	0.0679
IF	-	0.0279	0.0247	0.1344	0.3633
LOF	0.9737	-	0.6553	0.9972	0.9321
OCSVM	0.9768	0.3539	-	0.7342	0.9387
KNN	0.8711	0.0031	0.2741	-	0.8233
PCA	0.6461	0.0714	0.0646	0.1833	-

5.3.4 Discussion of the results

The discussion of the experimental results should begin with the observation that for the MSTCIF, MSTIF, and MSTIF-TS techniques, the obtained standard deviation values are the lowest. This indicates that the methods maintain stability, allowing for a reduction in the number of trees in the forest, which results in increased execution speed. The discussion has been structured in a way to analyze each metric separately.

Let us begin the discussion with an analysis of the accuracy metric. In the case of differences in average values, the MSTIF-TS method is unrivaled (Figure 5.10). In this category, KNN is the second-best performing solution. DRIF and PCA show similar values, placing them in third position. Looking from a different perspective, that is, median differences, the situation is similar. MSTIF-TS ranks highest in this comparison, while PCA and DRIF also achieve noteworthy results (Figure 5.16). On the other hand, when comparing the difference in the number of better and worse results for individual datasets, the situation changes slightly. Here, MSTIF is the most effective, followed by MSTIF-TS (Figure 5.22). DRIF and PCA also achieve good results. However, it can be noted that no algorithm definitively achieves the best results across all datasets. This indicates that, for optimal algorithm selection, each method should be considered. On the dimension of globally most effective methods, MSTCIF gains an advantage with four counts, as do IF and OCSVM, while KNN achieves the best result with five counts (Figure 5.28). Comparable values with three counts are achieved by MSTIF-TS. DRIF, MSTIF, and LOF also show strong performance, each with two best results.

The second metric under consideration is precision. For this measure, in average value comparisons, the leading algorithm is MSTIF (Figure 5.11). PCA ranks second, with one red field on the heatmap indicating lower values. DRIF, MSTIF-TS, and IF show similar results. From the perspective of median differences, MSTIF stands out with the best precision (Figure 5.17). MSTIF-TS and PCA also show good results. It is worth noting MSTCIF, which demonstrates good detection quality in this comparison that is less evident in the average analysis. Further conclusions regarding the number of higher results across datasets reveal interesting patterns. MSTIF achieves the highest effectiveness, with MSTIF-TS slightly behind (Figure 5.23). DRIF and PCA also belong among the top solutions. In this ranking, MSTCIF is just behind the previously mentioned solutions. The final bar chart in Figure 5.29, illustrating the number of globally best results, shows that every method deserves attention. Specifically, all algorithms recorded at least one count. In this category, MSTIF and MSTIF-TS achieved four top results each. DRIF, MSTCIF, OCSVM, and PCA also scored highly, holding significant positions in the analysis. Notably, although FDRIF had not shown strong results earlier, it achieved one count among the considered datasets.

When analyzing the recall metric, we observe that MSTIF has the highest average value (Figure 5.12). Close to this result are MSTCIF and MSTIF-TS. In this comparison, LOF records the lowest values. The interpretation of median differences confirms MSTIF's effectiveness (Figure 5.18). Interestingly, the ranking of the top algorithms changes: MSTIF-TS now shows better results than MSTCIF. Additionally, DRIF presents good results. Comparing the difference in the number of better and worse results for pairs shows the best performance when using MSTIF-TS (Figure 5.24). Comparable effectiveness is visible for MSTIF. MSTCIF also achieves significant results, placing it among the leading algorithms. Importantly, there is no clear distinction between FDRIF and DRIF. In the final context of the analyses, namely, the top results, when considering all methods, we observe interesting behavior. The baseline IF achieves no top result (Figure 5.30). Meanwhile, MSTIF is marked by only one count, and FDRIF stands out with three counts. DRIF and MSTCIF also obtained three best results each. LOF and PCA achieved two top results. The leader in this category among isolation tree-based algorithms is MSTIF-TS with four counts. It is also worth noting that the highest scores in this case are obtained by the KNN algorithm.

The next analyzed metric is specificity. For the series of average differences, as in previous statistics, MSTIF-TS shows high performance (Figure 5.13). Slightly lower values are observed for KNN and PCA. DRIF also demonstrates good results. However, the analysis of median differences indicates that PCA achieved the most effective detection (Figure 5.19). Slightly lower results are noted for MSTIF-TS. For both average and median value difference studies, DRIF scores highly. In the case of KNN and MSTIF, the median difference analysis revealed their favorable characteristics. Now focusing on the relationship of better and worse result counts for algorithm pairs, we observe the highest results for PCA, closely followed by KNN (Figure 5.25). Among the newly proposed techniques, DRIF achieves the best result. On the other hand, MSTIF and MSTIF-TS also show strong performance, placing them among the leading methods. In terms of specificity, the globally best results are presented by OCSVM and KNN. DRIF, MSTCIF, MSTIF, and PCA show slightly lower effectiveness with three counts each (Figure 5.31).

Let us now discuss the next metric, false alarm ratio. For average difference analysis, MSTIF-TS is the most effective method (Figure 5.14). Slightly lower results are achieved by the deterministic algorithms KNN and PCA. Among the newly introduced solutions, the results obtained with DRIF are also noteworthy. The analysis of median differences yields similar conclusions. Specifically, PCA demonstrates the best results, and MSTIF-TS ranks second among the most effective algorithms (Figure 5.20). As in the average analysis, DRIF performs well, while MSTIF also presents strong results. In the analysis of differences in better and worse result counts, PCA and KNN again demonstrate the best properties (Figure 5.26). Importantly, new proposals such as DRIF, MSTIF, and MSTIF-TS achieve only slightly lower

values. From the perspective of globally lowest results among all algorithms, OCSVM and KNN perform well (Figure 5.32). Interestingly, among the newly introduced methods, DRIF, MSTCIF, and MSTIF achieved the best quality in terms of false alarm ratio.

The final classification quality measure considered is the F1 score. In the study of average differences, the most effective algorithm is MSTIF (Figure 5.15). MSTIF-TS also achieves similarly high results, while MSTCIF ranks third. When focusing on the analysis of median differences, MSTIF confirms its top detection capabilities (Figure 5.21). PCA and MSTIF-TS, with similar effectiveness, share second place in this analysis. MSTCIF also deserves mention, with its result only slightly lower than those of the leading solutions. Interpreting the results based on differences in the number of best and worst values for the F1 score, MSTIF is the undisputed leader, with MSTIF-TS achieving very similar but slightly lower results (Figure 5.27). MSTCIF, in this case, gains prominence. When considering the globally best results, MSTCIF, MSTIF-TS, and IF have the highest number of counts (Figure 5.33). Furthermore, MSTIF, OCSVM, and PCA all show a high total number of top results. DRIF records two counts. At this point, it is worth emphasizing that all remaining methods achieved at least one best result, indicating the potential usefulness of all considered solutions.

Let us move on to a more formal aspect of comparing the discussed algorithms. We focus on the obtained p-values from the conducted statistical tests for algorithm pairs. In the case of the accuracy metric, MSTIF-TS has the largest number of statistically significantly better results (Tables 5.24 and 5.25). Next, MSTIF and DRIF show p-values close to the threshold of statistical significance in comparisons with competing algorithms. In the context of the precision metric, MSTIF stands out as the method with the largest number of statistically significant results (Tables 5.26 and 5.27). For the remaining new proposals, the p-values do not show statistical significance, although the values are low. Regarding the recall metric, the solutions with the best detection perspective are MSTCIF, MSTIF, and MSTIF-TS (Tables 5.28 and 5.29). Interestingly, for these methods, the tests demonstrated statistically significant differences compared to other methods. Further comparisons for the specificity metric highlight the advantage of the MSTIF-TS method (Tables 5.30 and 5.31). Meanwhile, for MSTIF, KNN, and PCA, low p-values also emphasize the effectiveness of these methods. Moving on to the analysis of the false alarm ratio, it is important to note that the alternative hypothesis tested assumed that the distribution of the differences between sample pairs is statistically smaller than a symmetric distribution around zero. In this situation, statistically significant results indicate better algorithm performance. In this context, MSTIF-TS again stands out as the leader, achieving five statistically significantly better results (Tables 5.32 and 5.33). Similar results, as observed for specificity, showing low p-values, are noted for MSTIF, KNN, and PCA. Finally, the statistical test results for the F1 score are analyzed. MSTIF achieved the highest number of statistically significantly better results, with MSTIF-TS presenting results

very close to this algorithm (Tables 5.34 and 5.35). When testing the hypothesis for the MSTIF and MSTIF-TS pair, the p-value suggests higher effectiveness of MSTIF. However, this conclusion is not supported by achieving statistical significance. Additionally, MSTCIF also obtains low p-values, but is clearly less effective compared to the top solutions mentioned.

In the final part of our discussion, we summarize all obtained results and analyses. Reviewing various aspects of algorithm performance, it becomes evident that each of the considered solutions has potential applications. It is also important that the anomaly detection performance analyses include a wide range of different classification quality metrics. This is especially important when optimizing for a specific behavior of the method. It is also crucial to balance the benefits and losses resulting from the use of a given solution. Therefore, it is important to consider selected evaluation metrics for the algorithm. In the case of good results in metrics independent of the detection threshold, such as ROC AUC and PR AUC, the metrics dependent on the detection threshold usually also demonstrated good properties. The most frequently appearing among the top algorithms is MSTIF-TS. However, considering the results of DRIF, high accuracy is noticeable, which is not so evident for threshold-independent metrics. It is worth emphasizing that the Isolation Forest-based methods competed with each other, which may have lowered their scores in the case of the number of globally best results. Another conclusion of our analysis is that the developed methods effectively handle anomaly detection in real-world datasets. However, it is worth considering the use of different techniques to tailor the approach to a specific dataset. The search for an optimal method can begin with those that showed the best results in our studies.

5.4 Response time

One of the important aspects in choosing an anomaly detection algorithm is its response time. A distinction can be made between the training time of the solution and the time required to evaluate the elements. It is essential to choose an algorithm in such a way that it meets the expectations regarding the execution time of its individual phases. This is particularly important when the decision about an element's anomaly must be made in real time, especially without significant delay. The Isolation Forest is listed in the literature as one of the fastest techniques. Therefore, improving it while maintaining a short response time is a particularly important aspect of our research. Execution times for each phase of the methods are measured. Time registration is carried out using a single thread, i.e., each time measurement is performed sequentially. To achieve precise results, the experiment is repeated one hundred times for the methods DRIF, FDRIF, MSTCIF, MSTIF, MSTIF-TS, IF, and PCA. For the remaining considered solutions, five repetitions are performed due to longer execution times. The measurement procedure is carried out on a laptop with the following specifications: Intel

Core i7-10750H processor, 32GB RAM, GTX 1660 Ti 6GB graphics card, running the Ubuntu 20.04.3 LTS 64-bit operating system. All anomaly detection algorithms, as well as the application responsible for registering the results, are written in C++.

5.4.1 Response time for the training phase

5.4.1.1 Results

First, the results recorded for the execution time of the training phase for the newly introduced solutions and competing methods are presented. Let us emphasize that this phase is invoked only once for the considered algorithms. Therefore, its impact may not be as significant, since training can be performed over a longer period. The most important time is the time to evaluate individual elements. The results of the training phase time measurements are presented in Tables 5.36 and 5.37.

5.4.1.2 Discussion of the results

In the comparison of training phase execution times, the fastest solutions are KNN and PCA. The baseline IF also features a short training time, although it is slightly slower. This results from the need to prepare a large number of trees to ensure detection stability. The proposals introduced in this dissertation are characterized by longer algorithm preparation times. Among these solutions, MSTCIF is the fastest. Attribute reduction methods have the longest execution times. DRIF records particularly long times for datasets with a large number of elements, which results from the high computational complexity of the clustering analysis. The algorithm using the FCM technique for data dimensionality reduction turns out to be the most time-consuming. In this comparison, MSTIF and MSTIF-TS present similar execution times. It should be noted that despite the extended training phase times, the obtained results are competitive in comparison with other presented techniques. Methods such as LOF and OCSVM show significantly prolonged training phases. This is especially visible when considering datasets with a very large number of elements. In such cases, the processing time can reach even several hours.

Table 5.36: Execution times of the training phase for the new methods, with standard deviations given in parentheses.

Dataset name	DRIF [ms]	FDRIF [ms]	MSTCIF [ms]	MSTIF [ms]	MSTIF-TS [ms]
Annthroid	255 (2)	1.3e4 (227)	59 (1)	649 (134)	589 (4)
Arrhythmia	146 (1)	3068 (223)	250 (10)	1293 (2)	1328 (35)
BreastW	20 (0)	1119 (100)	31 (0)	279 (2)	287 (1)
Cardio	63 (0)	7184 (168)	53 (1)	581 (3)	585 (2)
ForestCover	1.6e4 (194)	7.9e5 (2.3e4)	47 (1)	408 (10)	424 (6)
Glass	13 (0)	189 (10)	47 (0)	367 (1)	380 (1)
Http	1.5e4 (159)	7.3e5 (2.0e4)	41 (1)	386 (5)	413 (2)
Ionosphere	21 (0)	1227 (75)	56 (1)	485 (2)	499 (1)
Letter	71 (0)	8355 (703)	34 (0)	310 (4)	314 (5)
Lympho	15 (0)	379 (12)	27 (0)	390 (2)	401 (1)
Mammography	216 (1)	1.5e4 (891)	38 (0)	332 (4)	383 (6)
Mnist	1680 (2)	2.6e4 (369)	44 (0)	890 (6)	1029 (124)
Musk	437 (1)	9.1e4 (3881)	58 (1)	809 (159)	786 (3)
Optdigits	543 (1)	1.1e4 (201)	40 (0)	740 (134)	687 (1)
Pendigits	175 (1)	1.3e4 (1420)	33 (0)	249 (3)	270 (6)
Pima	25 (0)	1561 (156)	42 (1)	474 (4)	555 (106)
Satellite	591 (1)	2.9e4 (2435)	55 (1)	393 (94)	411 (77)
Satimage-2	450 (1)	2.7e4 (2299)	58 (1)	487 (88)	463 (6)
Shuttle	825 (34)	9.0e4 (6505)	40 (2)	495 (4)	529 (4)
Smtpt	2264 (5)	7.7e4 (8932)	32 (0)	343 (2)	404 (84)
Speech	2696 (84)	1.4e4 (185)	704 (69)	1805 (80)	1881 (9)
Thyroid	120 (1)	6259 (349)	58 (0)	609 (41)	629 (1)
Vertebral	15 (0)	339 (105)	49 (1)	413 (8)	437 (8)
Vowels	36 (0)	2052 (194)	64 (1)	239 (2)	260 (2)
WBC	28 (0)	2482 (180)	87 (1)	585 (3)	631 (3)
Wine	13 (0)	300 (38)	42 (0)	289 (2)	314 (2)
Average:	1578 (19)	7.5e4 (2798)	80 (4)	550 (31)	573 (19)
Median:	161 (1)	9439 (225)	47 (1)	444 (4)	450 (4)

Table 5.37: Execution times of the training phase for the reference methods, with standard deviations given in parentheses.

Dataset name	IF [ms]	LOF [ms]	OCSVM [ms]	KNN [ms]	PCA [ms]
Annth thyroid	13 (0)	2378 (54)	1223 (15)	0 (0)	0 (0)
Arrhythmia	33 (0)	54 (1)	1.9e4 (82)	0 (0)	51 (3)
BreastW	14 (0)	17 (0)	1009 (33)	0 (0)	0 (0)
Cardio	14 (0)	156 (2)	889 (9)	0 (0)	0 (0)
ForestCover	16 (3)	5.0e6 (8.1e4)	6.4e7 (8.5e5)	6 (4)	38 (5)
Glass	14 (0)	2 (0)	1 (0)	0 (0)	0 (0)
Http	13 (0)	2.2e7 (4.1e5)	1.2e7 (6.7e5)	9 (6)	21 (1)
Ionosphere	18 (1)	7 (0)	647 (7)	0 (0)	0 (0)
Letter	15 (0)	141 (2)	2.9e4 (252)	0 (0)	1 (0)
Lympho	16 (2)	1 (0)	939 (16)	0 (0)	0 (0)
Mammography	12 (0)	5291 (232)	2151 (100)	0 (0)	1 (0)
Mnist	21 (0)	6550 (227)	1.1e4 (153)	1 (1)	17 (1)
Musk	27 (0)	1678 (85)	1.6e5 (850)	0 (0)	27 (2)
Optdigits	18 (0)	2264 (90)	4.6e5 (6603)	0 (0)	6 (0)
Pendigits	16 (2)	2615 (169)	1289 (9)	0 (0)	1 (0)
Pima	13 (0)	24 (1)	3.6e4 (2431)	0 (0)	0 (0)
Satellite	17 (0)	2632 (138)	4.6e5 (2.2e4)	0 (0)	3 (0)
Satimage-2	17 (1)	2090 (18)	3.0e5 (307)	0 (0)	3 (0)
Shuttle	14 (0)	1.3e5 (1448)	9.7e5 (2.2e4)	1 (1)	4 (0)
Smt p	13 (1)	5.3e5 (2091)	2.2e5 (2.0e4)	1 (1)	3 (0)
Speech	43 (0)	5673 (191)	8996 (96)	2 (1)	304 (15)
Thyroid	13 (0)	704 (56)	304 (13)	0 (0)	0 (0)
Vertebral	13 (0)	2 (0)	1.0e4 (165)	0 (0)	0 (0)
Vowels	14 (0)	96 (4)	998 (7)	0 (0)	0 (0)
WBC	15 (0)	7 (0)	696 (35)	0 (0)	0 (0)
Wine	15 (0)	1 (0)	5466 (195)	0 (0)	0 (0)
Average:	17 (1)	1.1e6 (1.9e4)	3.1e6 (6.1e4)	1 (0)	18 (1)
Median:	15 (0)	1191 (36)	9592 (127)	0 (0)	1 (0)

5.4.2 Response time for the evaluation phase

5.4.2.1 Results

The second parameter measured for all solutions is the time of the evaluation phase. This parameter is crucial, as it directly affects the speed of determining whether the considered element is an anomaly or a normal element. For the evaluation phase execution time, the time required to evaluate all elements of the individual detection datasets is measured. The results of the measurements are presented in Tables 5.38 and 5.39.

5.4.2.2 Discussion of the results

Analyzing the execution times of the evaluation phase, it can be stated that the PCA algorithm is the fastest; similar behavior is observed in the case of the first phase. The baseline IF and DRIF methods achieve similar time values. It is worth noting the FDRIF technique, based on fuzzy clustering analysis, which also features short execution times. In contrast, the MSTCIF algorithm is slower, which results from the need to traverse complete isolation trees. Regarding methods based on the new construction of isolation trees, i.e., MSTIF and MSTIF-TS, their times are similar, with MSTIF-TS being slightly slower. This is caused by additional fuzzy inference model computations. For competing techniques such as LOF, OCSVM, and KNN, the execution speed significantly decreases when the training dataset contains a large number of samples. Let us emphasize that the execution times of the evaluation phase are considered for the entire analyzed dataset. Consequently, the execution times for larger collections are longer compared to those for datasets with fewer elements.

Table 5.38: Execution times of the evaluation phase for the new methods, with standard deviations given in parentheses.

Dataset name	DRIF [ms]	FDRIF [ms]	MSTCIF [ms]	MSTIF [ms]	MSTIF-TS [ms]
Annth thyroid	52 (3)	87 (22)	253 (7)	952 (5)	987 (2)
Arrhythmia	6 (0)	6 (0)	10 (0)	2961 (1)	3057 (120)
BreastW	5 (0)	8 (2)	9 (0)	119 (1)	124 (1)
Cardio	13 (0)	18 (0)	45 (1)	623 (2)	638 (1)
ForestCover	1991 (104)	2898 (260)	6293 (138)	5.5e4 (228)	5.9e4 (192)
Glass	2 (0)	2 (0)	5 (0)	37 (0)	38 (0)
Http	3673 (305)	5452 (244)	1.2e4 (200)	5.4e4 (223)	5.8e4 (248)
Ionosphere	3 (0)	4 (0)	9 (0)	196 (2)	199 (1)
Letter	11 (0)	20 (5)	23 (0)	857 (3)	874 (2)
Lympho	1 (0)	2 (0)	2 (0)	45 (0)	47 (1)
Mammography	77 (3)	107 (3)	186 (3)	1905 (8)	2252 (6)
Mnist	67 (3)	93 (1)	86 (1)	1.6e4 (129)	1.8e4 (129)
Musk	28 (1)	37 (2)	37 (0)	1.2e4 (166)	1.3e4 (156)
Optdigits	46 (2)	57 (1)	65 (0)	6802 (90)	7404 (159)
Pendigits	57 (2)	76 (2)	111 (1)	1989 (147)	2101 (5)
Pima	7 (0)	10 (3)	17 (0)	126 (0)	152 (23)
Satellite	48 (1)	61 (2)	147 (3)	4199 (36)	4593 (132)
Satimage-2	42 (1)	59 (0)	141 (3)	3777 (60)	4133 (129)
Shuttle	405 (70)	481 (13)	1017 (8)	8674 (180)	9493 (154)
Smt p	632 (26)	865 (25)	1585 (16)	9172 (184)	9826 (127)
Speech	52 (12)	58 (1)	184 (2)	3.8e4 (1421)	4.0e4 (27)
Thyroid	28 (1)	40 (1)	128 (1)	571 (105)	576 (2)
Vertebral	2 (0)	4 (1)	7 (0)	33 (0)	36 (0)
Vowels	11 (0)	16 (0)	51 (2)	316 (1)	348 (3)
WBC	3 (0)	5 (0)	16 (0)	190 (2)	210 (3)
Wine	1 (0)	2 (0)	3 (0)	30 (0)	33 (0)
Average:	279 (21)	403 (23)	870 (15)	8416 (115)	9044 (62)
Median:	28 (1)	38 (1)	58 (1)	1428 (7)	1544 (6)

Table 5.39: Execution times of the evaluation phase for the reference methods, with standard deviations given in parentheses.

Dataset name	IF [ms]	LOF [ms]	OCSVM [ms]	KNN [ms]	PCA [ms]
Annthroid	64 (2)	2414 (57)	363 (1)	2558 (39)	1 (0)
Arrhythmia	3 (0)	54 (1)	58 (0)	63 (5)	26 (2)
BreastW	6 (0)	17 (0)	4 (0)	22 (2)	0 (0)
Cardio	15 (0)	153 (3)	49 (0)	180 (9)	1 (0)
ForestCover	2174 (82)	5.1e6 (9.8e4)	1.7e6 (1.8e5)	4.9e6 (9.7e4)	59 (7)
Glass	2 (0)	2 (0)	0 (0)	2 (0)	0 (0)
Http	3825 (214)	2.2e7 (2.8e5)	2.6e6 (1.2e5)	2.1e7 (7.3e5)	44 (3)
Ionosphere	4 (0)	7 (0)	3 (0)	7 (0)	0 (0)
Letter	15 (0)	138 (2)	75 (0)	177 (14)	1 (0)
Lympho	2 (0)	1 (0)	0 (0)	1 (0)	0 (0)
Mammography	81 (1)	5325 (296)	925 (46)	5641 (148)	1 (0)
Mnist	69 (2)	6513 (262)	7505 (217)	7077 (132)	50 (4)
Musk	27 (0)	1624 (66)	1664 (27)	1770 (145)	71 (5)
Optdigits	51 (0)	2255 (109)	1959 (31)	2583 (151)	16 (1)
Pendigits	72 (0)	2706 (158)	635 (139)	3121 (139)	2 (0)
Pima	8 (0)	24 (1)	14 (1)	28 (0)	0 (0)
Satellite	55 (0)	2585 (31)	2020 (193)	3190 (119)	7 (1)
Satimage-2	50 (1)	2181 (186)	1604 (95)	2519 (7)	6 (0)
Shuttle	427 (5)	1.3e5 (1038)	3.0e4 (930)	1.4e5 (3159)	9 (1)
Smtpt	762 (61)	5.4e5 (1.4e4)	5.7e4 (6046)	5.5e5 (2.1e4)	7 (1)
Speech	33 (2)	5513 (23)	3391 (133)	5711 (157)	514 (37)
Thyroid	33 (1)	710 (94)	99 (2)	661 (9)	0 (0)
Vertebral	3 (0)	2 (0)	1 (0)	2 (0)	0 (0)
Vowels	13 (0)	96 (8)	21 (0)	98 (6)	0 (0)
WBC	4 (0)	8 (0)	3 (0)	9 (1)	0 (0)
Wine	1 (0)	1 (0)	0 (0)	1 (0)	0 (0)
Average:	300 (14)	1.1e6 (1.5e4)	1.7e5 (1.2e4)	1.0e6 (3.3e4)	31 (2)
Median:	30 (0)	1167 (27)	231 (1)	1216 (12)	1 (0)

5.5 Hyperparameter Analysis

Another issue subjected to detailed analysis in this dissertation is the study of hyperparameters of the newly developed methods. These hyperparameters correspond to those applied in the baseline IF method. Specifically, they include the number of samples and the number of trees forming the isolation forest. Moreover, the algorithms in their design allow for the modification of the aforementioned hyperparameters in order to meet specific requirements.

5.5.1 Results

The presentation includes a detailed overview of the results obtained for three different numbers of samples forming a single isolation tree, i.e., 64, 128, and 256 samples. Additionally, bar charts illustrate the average training phase times relative to the average execution time of the baseline IF method. An analogous analysis is also carried out for the evaluation phase. The final examined characteristic is the comparison of the average values of the ROC AUC measure. On the charts, the level achieved by the baseline IF method with its default hyperparameters is clearly marked. The analysis took into account different numbers of trees forming the isolation forest, covering a range from a single tree up to one hundred isolation trees, specifically: 1, 2, 3, 4, 5, 10, 15, 20, 50, and 100 trees. The results of individual comparisons are presented in Figures 5.34, 5.35, and 5.36.

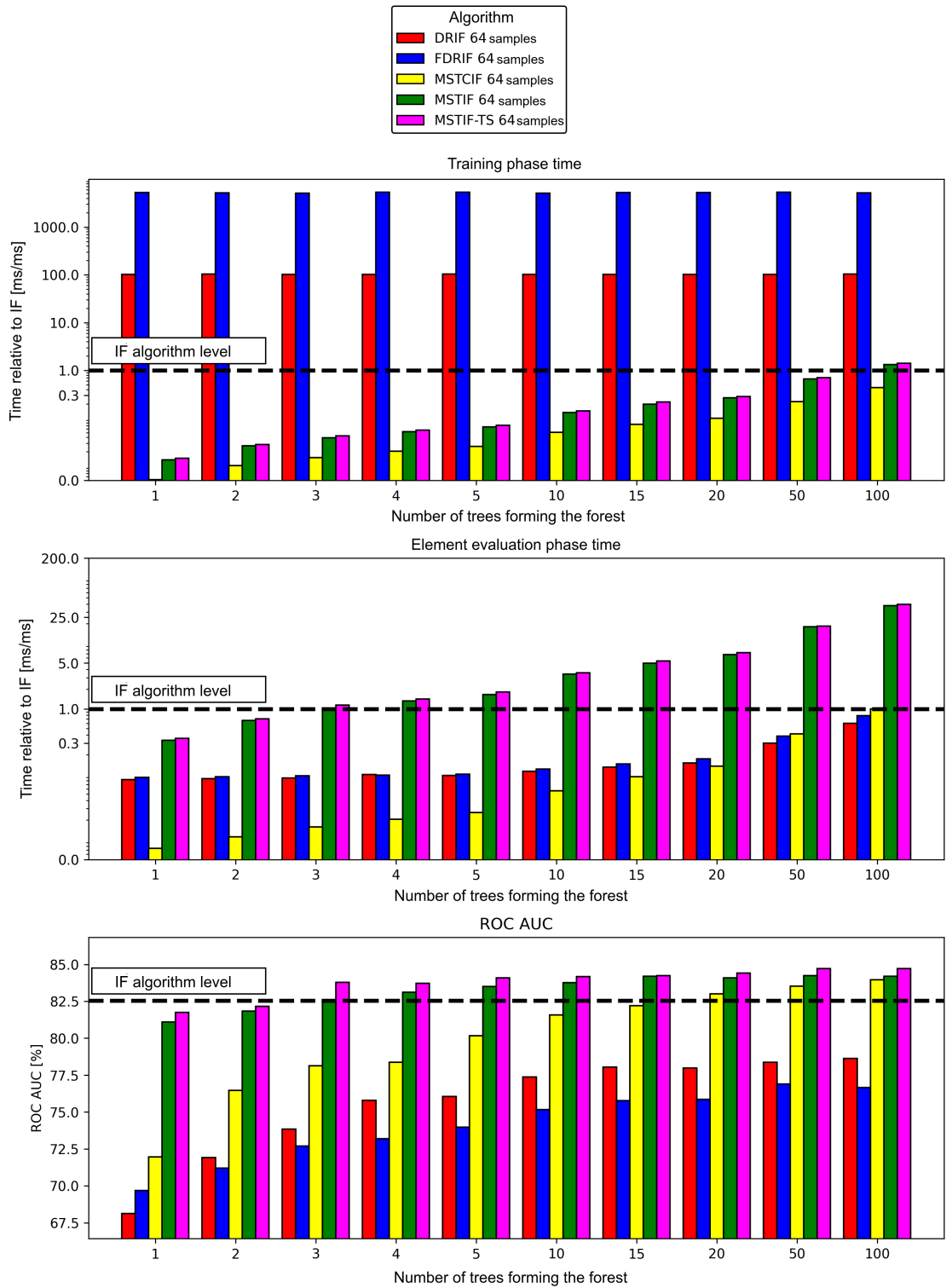


Figure 5.34: Plots of the properties of the proposed methods for different hyperparameter settings. Part I. The plots present the average values for the considered datasets. The time measurement results are presented on a logarithmic scale relative to the times of the baseline IF method with default hyperparameters, while the ROC AUC measure results are presented on a linear scale. The dashed line indicates the level of the baseline IF method.

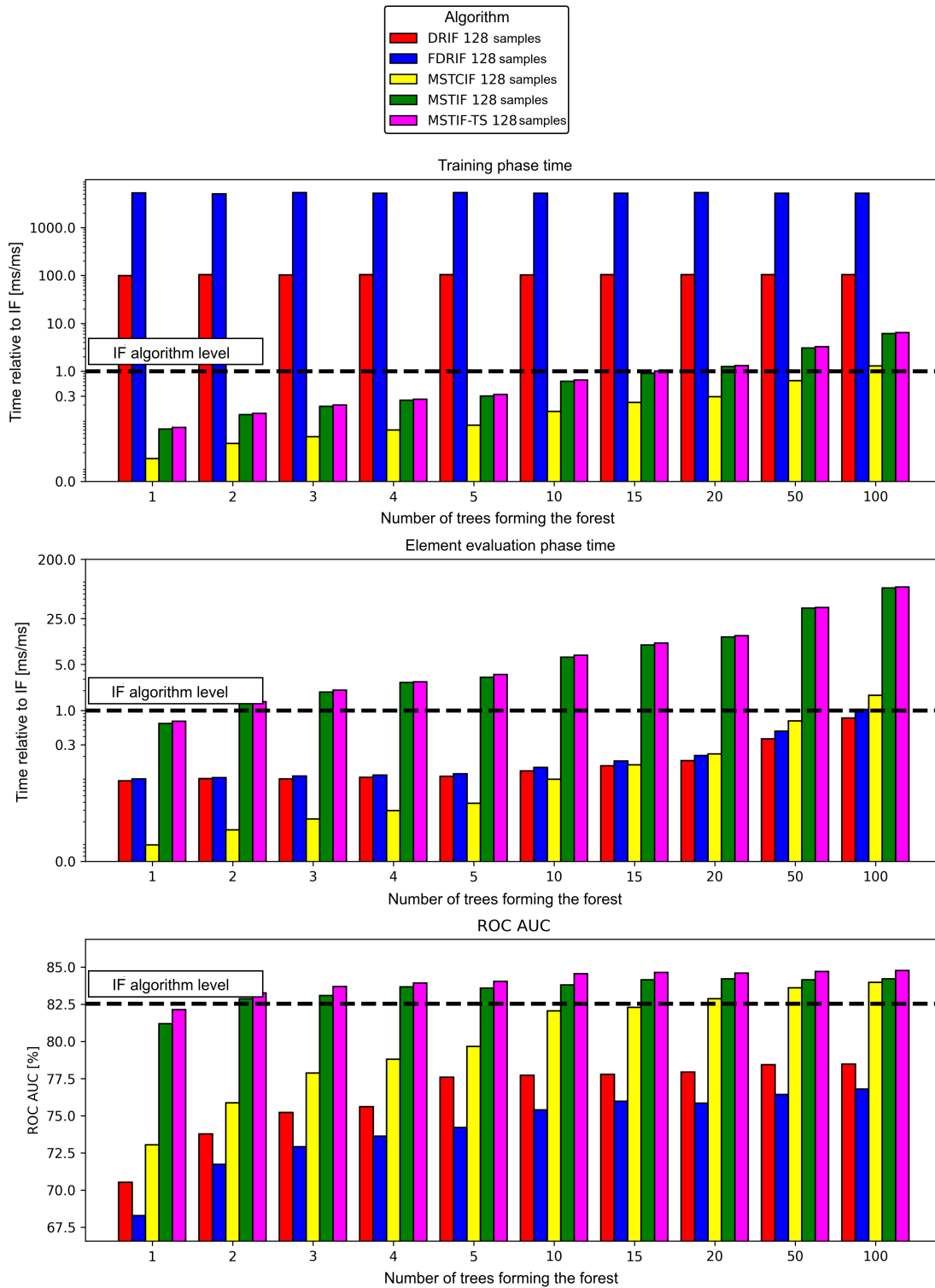


Figure 5.35: Plots of the properties of the proposed methods for different hyperparameter settings. Part II. The plots present the average values for the considered datasets. The time measurement results are presented on a logarithmic scale relative to the times of the baseline IF method with default hyperparameters, while the ROC AUC measure results are presented on a linear scale. The dashed line indicates the level of the baseline IF method.

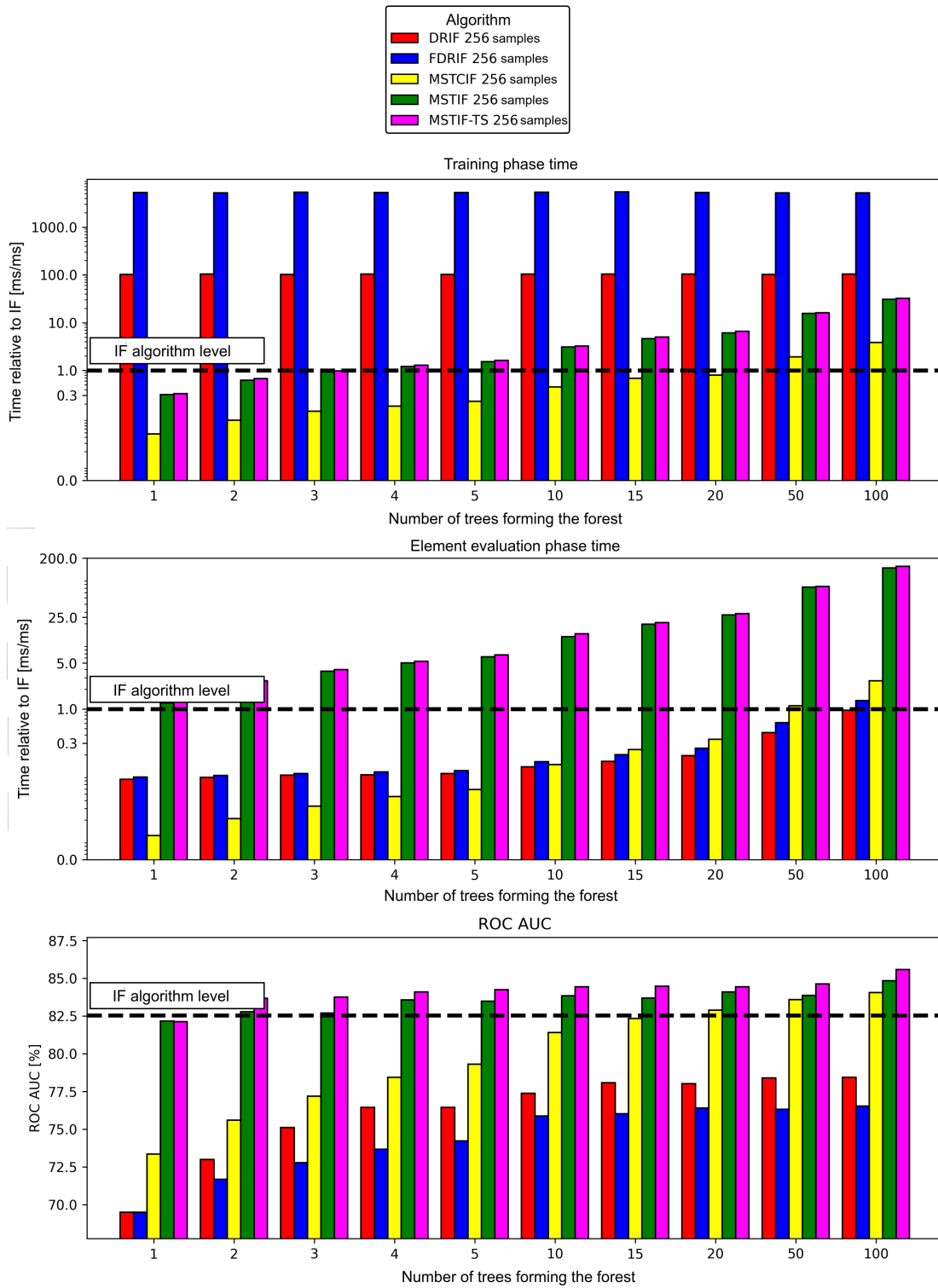


Figure 5.36: Plots of the properties of the proposed methods for different hyperparameter settings. Part III. The plots present the average values for the considered datasets. The time measurement results are presented on a logarithmic scale relative to the times of the baseline IF method with default hyperparameters, while the ROC AUC measure results are presented on a linear scale. The dashed line indicates the level of the baseline IF method.

5.5.2 Discussion of the results

The discussion should begin by emphasizing that the presented results are average values for the considered real-world datasets. Certain characteristic features of individual solutions can be observed. For attribute reduction algorithms, the most significant is the training phase time, i.e., cluster preparation. In this case, the execution time of the baseline IF method after element transformation is negligible. Therefore, on all charts, the training phase duration for DRIF and FDRIF remains unchanged. On the other hand, for the remaining proposals, it can be seen that the training time increases with an increasing number of samples or with an increasing number of trees forming the forest. The fastest method in this comparison turns out to be MSTCIF, whereas MSTIF and MSTIF-TS exhibit similar training times. When considering the evaluation phase, the longest execution times are recorded for MSTIF and MSTIF-TS, although it should be noted that their values are very close to each other, similar to the training phase analysis. For attribute reduction methods, the charts show that up to a certain number of trees — namely up to 10 trees — the times remain at the same level. This results from a constant time for sample processing to reduce its dimensionality, while the time associated with traversing isolation trees becomes significant only when the number of trees exceeds 10. The performance analysis using the ROC AUC metric leads to the conclusion that as the number of trees and the number of samples increase, the effectiveness of the methods improves. However, above a certain level of hyperparameters, the results stabilize, and no further significant improvement in performance is observed. Attribute reduction methods in this comparison achieve lower values than the baseline IF method. In contrast, the remaining proposals demonstrate better effectiveness. It should be emphasized that MSTIF and MSTIF-TS achieve results comparable to IF already with a few isolation trees, especially when considering 256 samples, where this is noticeable even for a single tree. For two trees, the results already exceed the performance of the reference IF algorithm. Similarly, for MSTCIF, it can be observed that it achieves high effectiveness with lower hyperparameter values than the default ones. The behavior of MSTCIF is particularly interesting when looking at the results overall. Namely, while maintaining a level of effectiveness close to IF, this method is able to perform training and evaluation in a shorter time. Conversely, MSTIF and MSTIF-TS, although characterized by higher effectiveness, do not achieve such short execution times. In summary, in the context of hyperparameter analysis of the proposed methods, attention should be paid to the possibility of their modification, which leads to a change in the characteristics of the solutions. This opens the perspective of optimizing the methods to adapt either to desired execution times or to desired effectiveness.

5.6 Separability of the score function results

The final aspect analyzed for detection results on real-world datasets is the analysis of the distributions of normalized score function values.

5.6.1 Results

Plots are prepared to visualize the distributions of normalized score function values. Additionally, the plots are constructed in such a way that the area under the distribution of normal samples, marked in blue, is equal to 1. Analogously, the area under the distribution of anomalous samples, marked in red, is also equal to 1. Furthermore, overlapping areas between these distributions are indicated with a crossed region. The plots for all methods and datasets are presented in Figures 5.37–5.43. The area of the intersection region is calculated, and the obtained values are shown in Tables 5.40 and 5.41.

5.6.2 Discussion of the results

The visualization of density distributions for both normal and anomalous points allowed for a better understanding of how the methods operate. Thus, we can assess how well the compared classifiers label the data. In the ideal case, the density distributions for normal and anomalous points should not exhibit any overlapping areas. This is because, under such conditions, it is possible to precisely distinguish between anomalous and normal points. However, in situations where there are areas where the score function values are identical for both classes of samples, it becomes impossible to unambiguously determine class membership based on the score value. This leads to classification errors. Therefore, it should be clearly emphasized that overlapping areas between these two distribution plots should be minimized, as the presence of such overlap is undesirable.

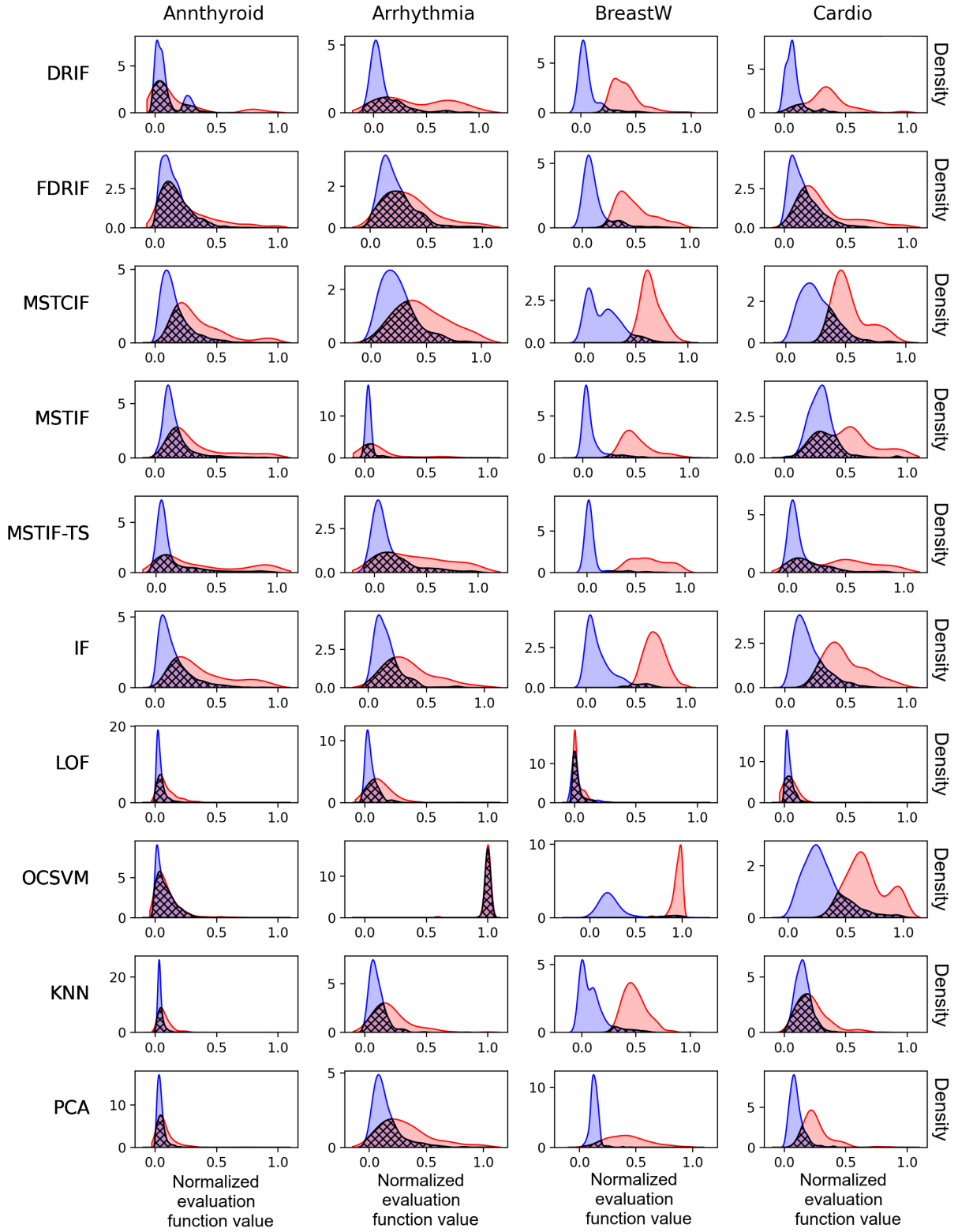


Figure 5.37: Normalized density plots of the score function values for the considered methods. Part I. The intersection of the density plots is marked with a crossed area. Blue and red areas indicate the densities of normal points and anomalies, respectively.

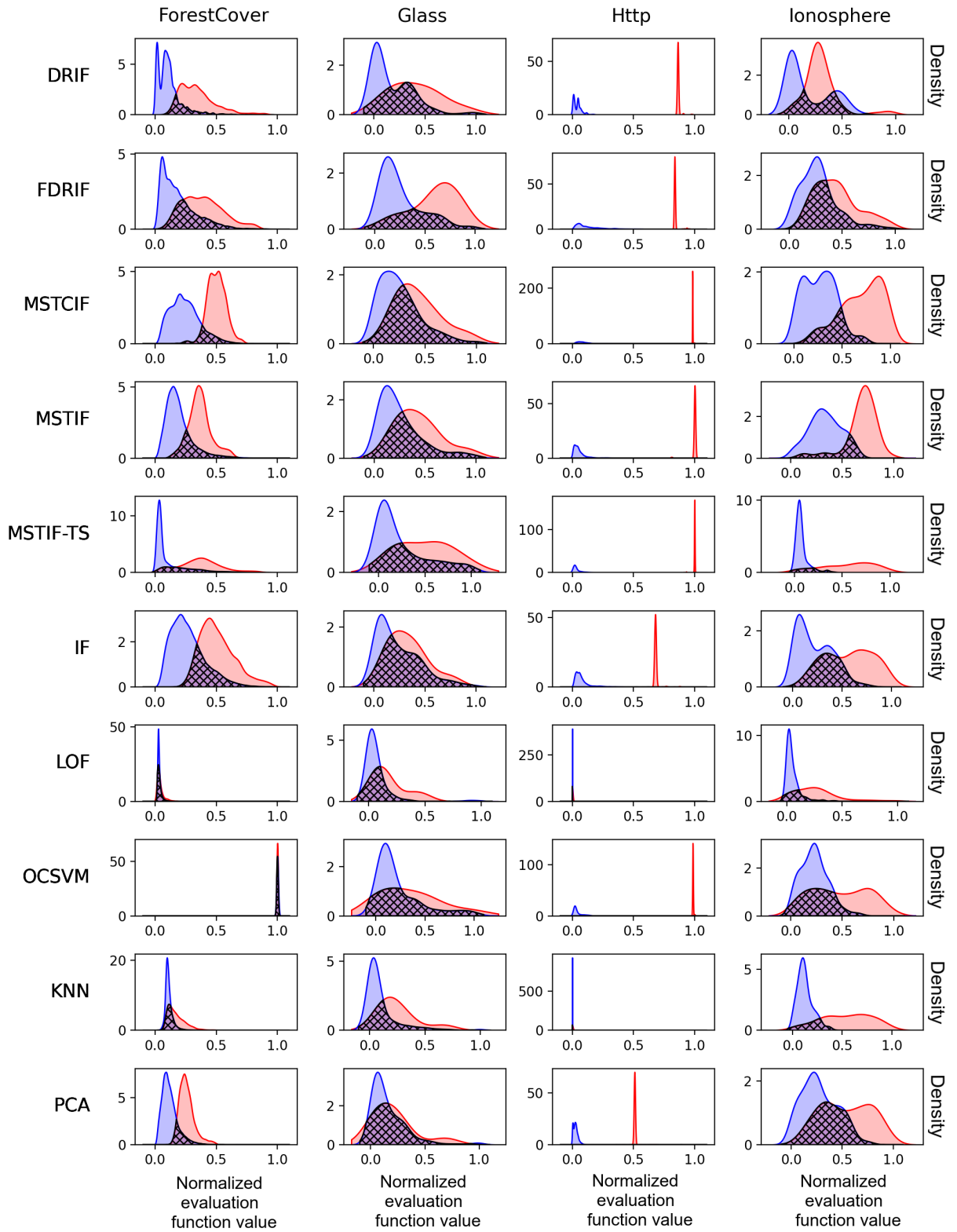


Figure 5.38: Normalized density plots of the score function values for the considered methods. Part II. The intersection of the density plots is marked with a crossed area. Blue and red areas indicate the densities of normal points and anomalies, respectively.

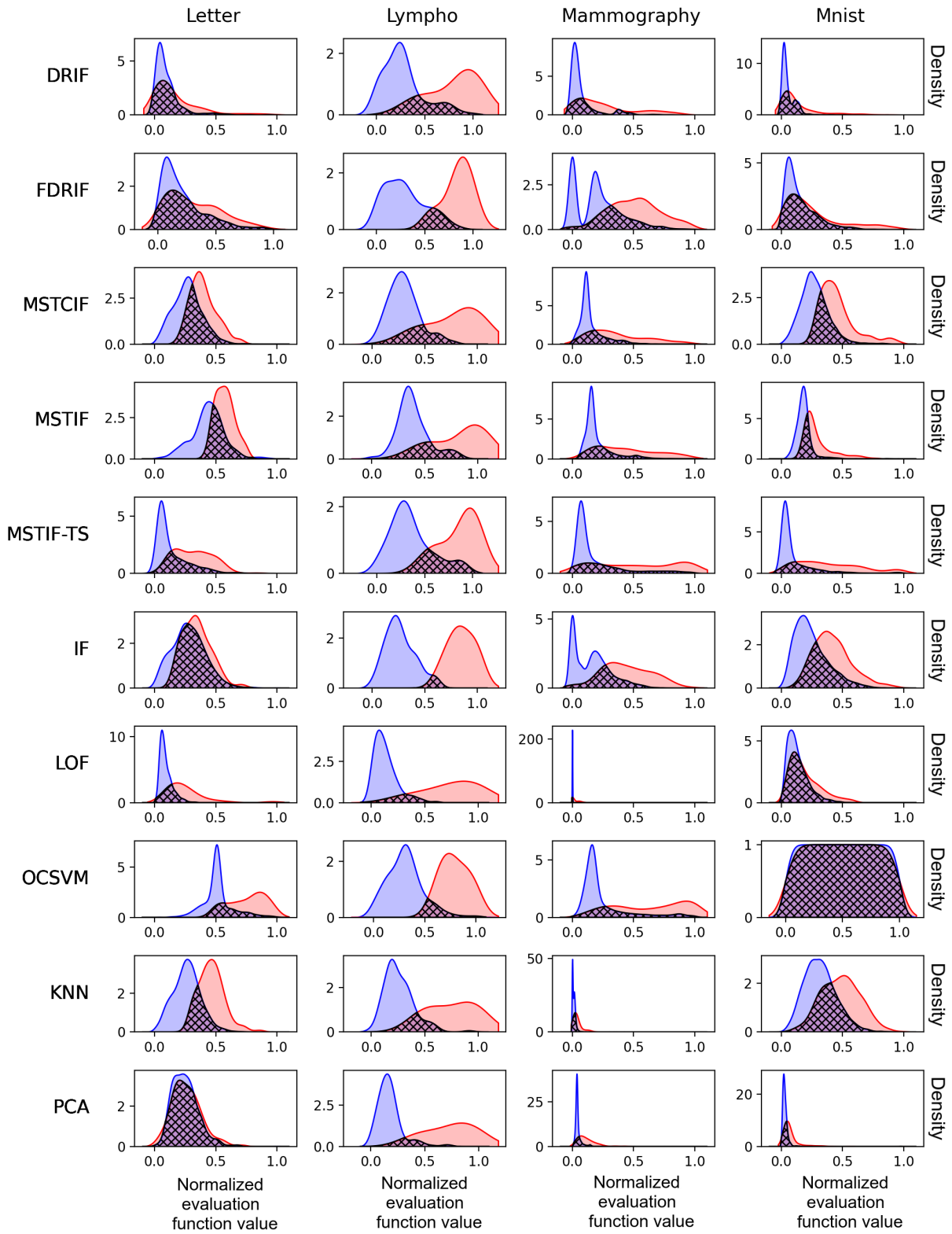


Figure 5.39: Normalized density plots of the score function values for the considered methods. Part III. The intersection of the density plots is marked with a crossed area. Blue and red areas indicate the densities of normal points and anomalies, respectively.

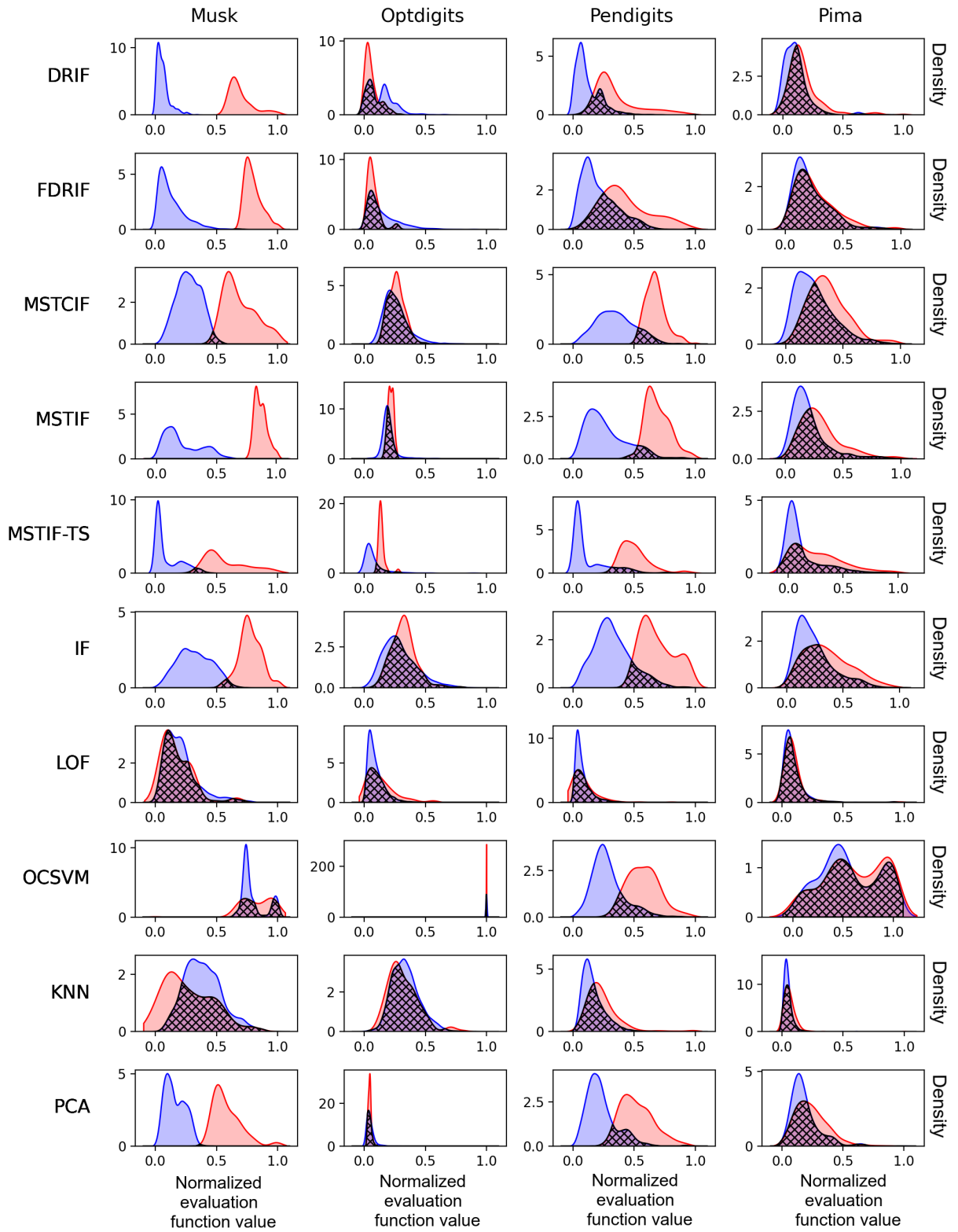


Figure 5.40: Normalized density plots of the score function values for the considered methods. Part IV. The intersection of the density plots is marked with a crossed area. Blue and red areas indicate the densities of normal points and anomalies, respectively.

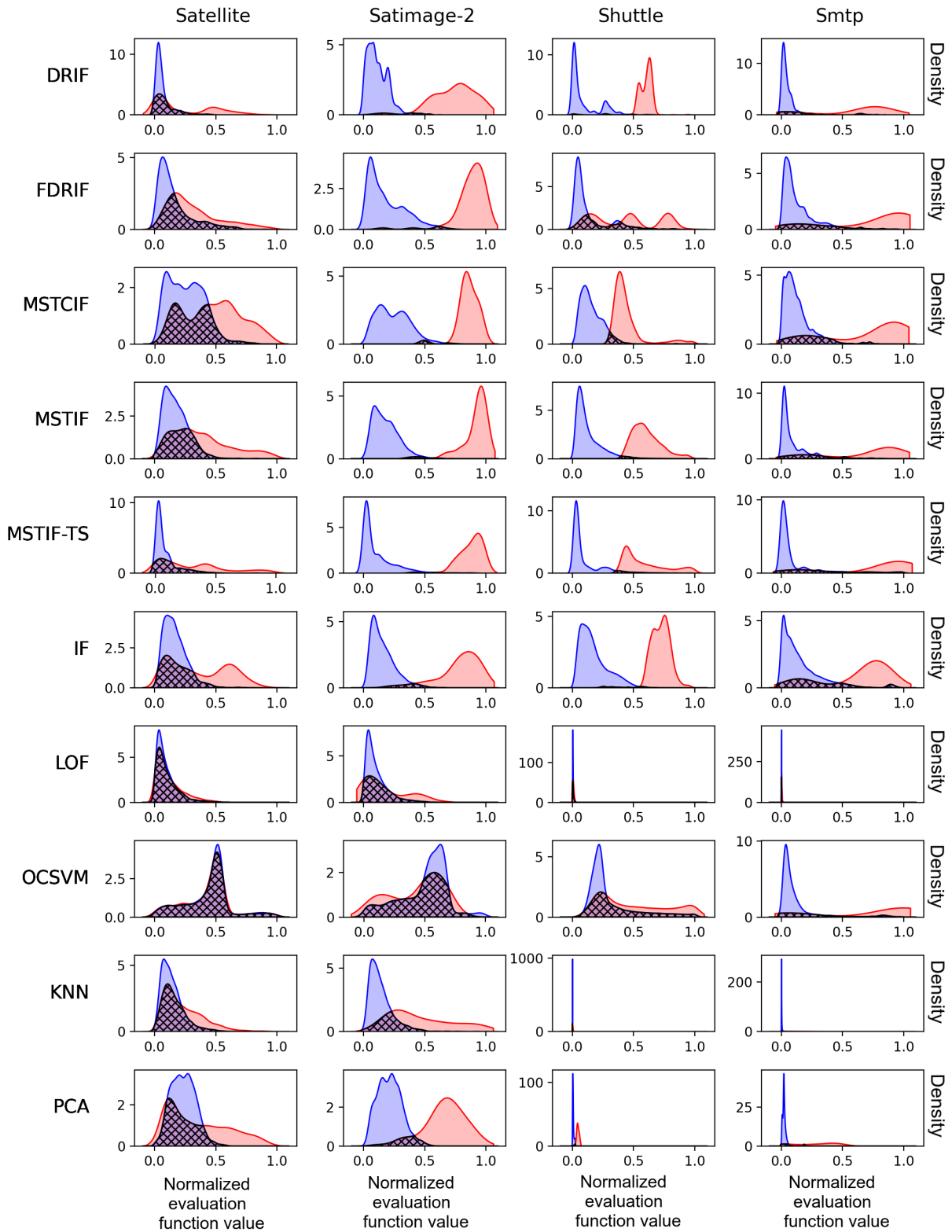


Figure 5.41: Normalized density plots of the score function values for the considered methods. Part V. The intersection of the density plots is marked with a crossed area. Blue and red areas indicate the densities of normal points and anomalies, respectively.

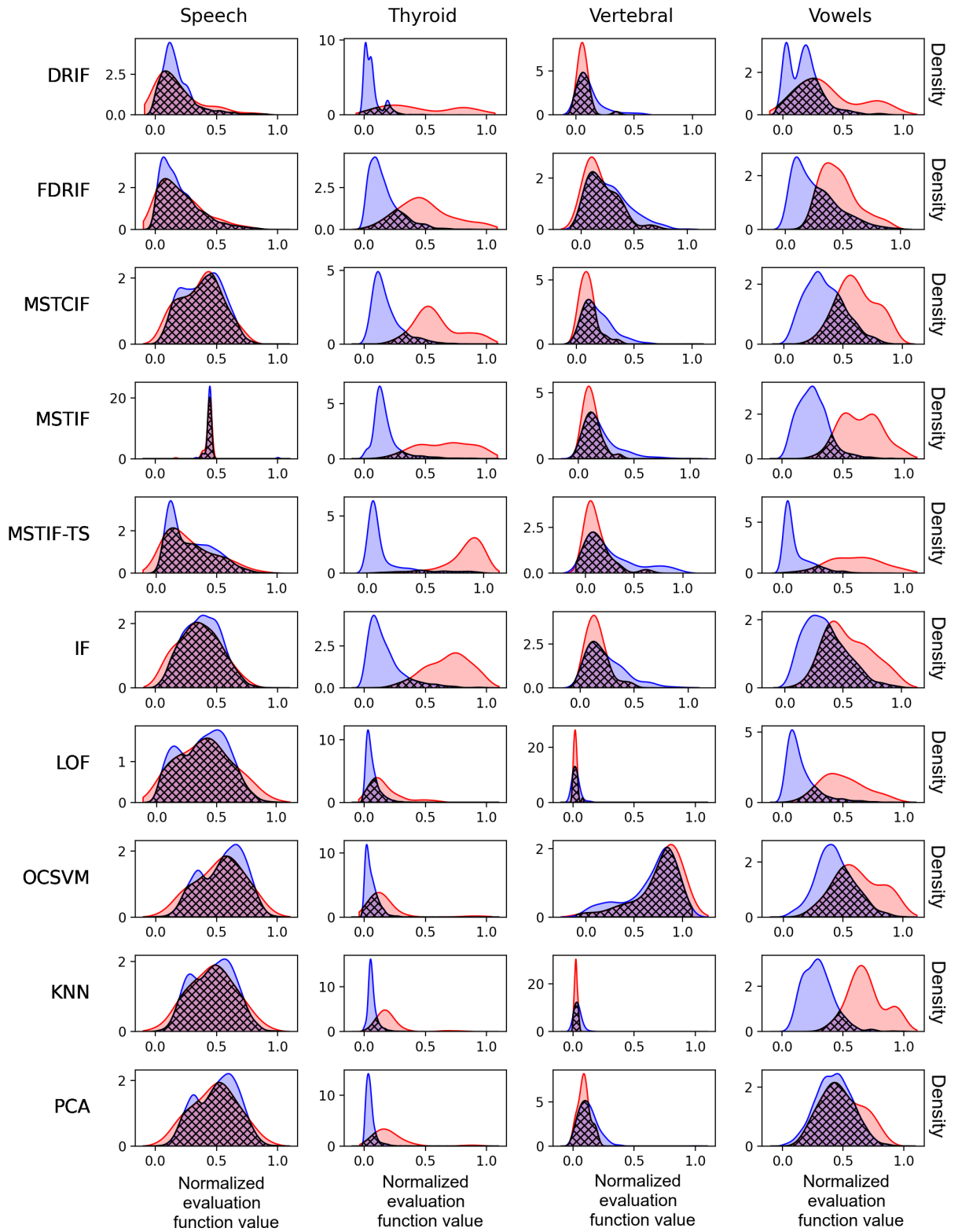


Figure 5.42: Normalized density plots of the score function values for the considered methods. Part VI. The intersection of the density plots is marked with a crossed area. Blue and red areas indicate the densities of normal points and anomalies, respectively.

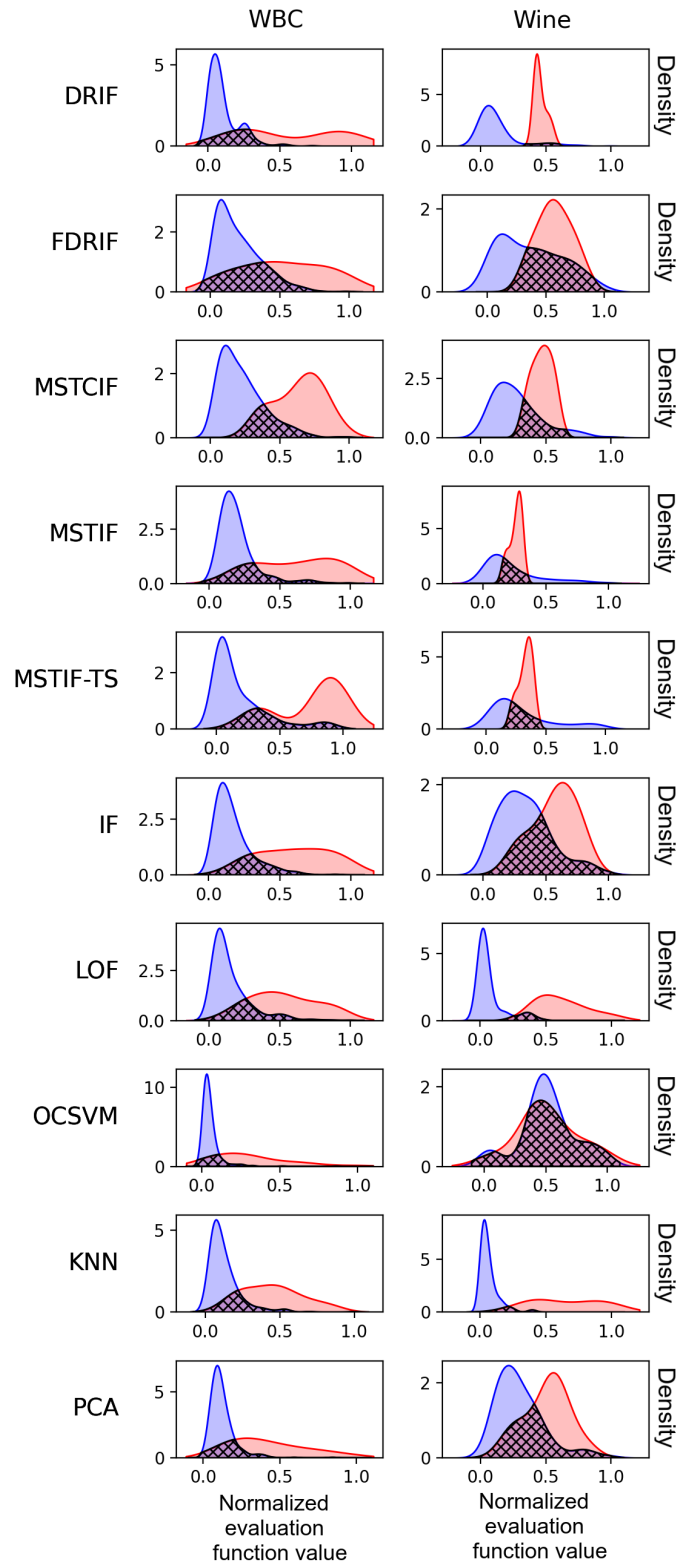


Figure 5.43: Normalized density plots of the score function values for the considered methods. Part VII. The intersection of the density plots is marked with a crossed area. Blue and red areas indicate the densities of normal points and anomalies, respectively.

Table 5.40: Intersection area values of the density distributions of normal and anomalous elements for the new methods.

Dataset name	DRIF [%]	FDRIF [%]	MSTCIF [%]	MSTIF [%]	MSTIF-TS [%]
Annth thyroid	62.1	78.1	53.2	57.8	53.9
Arrhythmia	46.9	71.5	57.4	38.3	53.7
BreastW	12.3	16.7	10.5	9.4	8.5
Cardio	23.9	64.0	32.2	49.8	47.7
ForestCover	25.7	50.5	18.1	31.4	36.7
Glass	61.0	44.4	66.7	63.2	57.7
Http	0.6	0.6	0.1	0.5	0.5
Ionosphere	44.3	64.3	29.4	24.4	18.6
Letter	70.3	76.4	57.3	49.2	49.0
Lympho	32.3	22.9	30.4	33.0	31.4
Mammography	48.6	45.4	48.7	46.8	46.5
Mnist	61.3	75.6	45.9	46.0	40.9
Musk	0.0	0.5	4.3	0.0	7.9
Optdigits	64.8	67.6	81.1	58.9	19.0
Pendigits	36.1	54.8	17.5	14.0	16.8
Pima	76.0	91.5	70.0	68.8	64.1
Satellite	54.1	60.6	55.7	55.9	48.2
Satimage-2	5.2	5.5	3.8	3.5	3.5
Shuttle	5.0	42.1	9.9	4.1	8.2
Smtpt	19.8	25.7	26.4	26.0	28.3
Speech	78.5	87.2	92.3	88.7	84.7
Thyroid	27.1	34.0	18.6	17.0	14.6
Vertebral	72.8	84.4	67.3	74.8	71.3
Vowels	60.1	51.0	44.4	18.3	19.0
WBC	34.6	48.5	29.8	33.0	33.1
Wine	7.3	57.2	33.1	35.8	36.3
Average:	39.6	50.8	38.6	36.5	34.6
Median:	40.2	52.9	32.6	34.4	34.7

Table 5.41: Intersection area values of the density distributions of normal and anomalous elements for the reference methods.

Dataset name	IF [%]	LOF [%]	OCSVM [%]	KNN [%]	PCA [%]
Annth thyroid	51.6	60.1	86.9	56.4	63.0
Arrhythmia	51.1	53.0	96.5	52.5	56.4
BreastW	6.9	80.1	7.8	9.6	14.0
Cardio	31.8	62.0	26.0	67.3	28.1
ForestCover	36.3	75.4	90.0	53.1	21.8
Glass	76.0	56.9	65.0	50.1	77.4
Http	0.6	14.2	0.6	19.2	0.4
Ionosphere	49.4	33.4	53.2	23.1	60.5
Letter	80.7	40.1	36.1	36.8	90.9
Lympho	7.0	17.6	14.9	24.0	16.6
Mammography	42.7	15.4	38.7	46.6	36.6
Mnist	53.2	78.9	97.6	56.5	48.4
Musk	5.9	84.6	59.3	65.5	0.8
Optdigits	76.9	72.4	53.3	87.2	67.5
Pendigits	20.8	72.2	26.0	70.0	24.6
Pima	73.2	92.0	88.1	78.4	75.3
Satellite	53.9	88.3	95.9	73.0	58.4
Satimage-2	8.4	63.5	80.3	35.4	13.3
Shuttle	2.7	58.6	62.5	25.8	5.7
Smt p	30.9	39.7	24.4	6.0	13.6
Speech	90.8	88.5	87.8	90.5	88.8
Thyroid	16.0	52.0	50.4	30.0	28.8
Vertebral	75.0	65.9	84.6	58.7	78.9
Vowels	62.3	25.9	61.0	16.3	84.5
WBC	30.7	32.7	26.9	29.9	32.0
Wine	53.6	12.1	85.8	10.2	52.4
Average:	41.9	55.2	57.7	45.1	43.8
Median:	46.0	59.3	60.2	48.3	42.5

Starting the discussion of results, let us first focus on analyzing the shapes of the density distributions. It can be observed that the density distributions of scores for normal samples are characterized by concentration, which is manifested in narrow plots with a clearly marked, high peak. On the other hand, the distributions for anomalous samples are typically flatter and spread over a wider range. Differences in the shapes of the plots for specific algorithms and datasets reflect diverse characteristics and classification properties. Let us analyze the plots for individual datasets. The Anthyroid dataset is an example of a difficult case in the context of anomaly detection, where no situation is observed in which the density plots are clearly separated. In this specific case, the best results are achieved by the IF and MSTCIF methods. Arrhythmia is characterized by the best performance of MSTCIF. BreastW is a dataset for which most solutions produce good results, with the exception of the LOF algorithm, whose score values do not allow for a clear distinction between anomalies and normal elements. In the case of the Cardio dataset, the best results are achieved using the DRIF algorithm. ForestCover shows effective separation of density plots when using MSTCIF, MSTIF, and MSTIF-TS. OCSVM does not prove effective in this case. This is illustrated by the plot, which is dominated by a large number of identical score values, with other values being barely noticeable. Such a characteristic hinders the differentiation of samples and the assessment of their anomaly level. With regard to the Glass dataset, the FDRIF method demonstrates effective separation of density plots. For the Http dataset, the visualizations suggest very good separation of samples based on score values, with the exception of LOF and KNN, which do not perform well on this type of data. The Ionosphere dataset presents satisfactory separation when using the MSTCIF, MSTIF, MSTIF-TS, and KNN algorithms. Most methods struggle to effectively separate samples in the Letter dataset. In this case, the OCSVM and KNN algorithms show acceptable separation. The results for Lympho indicate good performance across all considered methods. Significantly more challenging cases are seen in the Mammography and Mnist datasets, where none of the methods demonstrate clear separation of score values. The Musk dataset achieves good element separation using the DRIF, FDRIF, and MSTIF algorithms. MSTIF-TS exhibits the best classification properties for Optdigits. For Pendigits, effective separation of samples can be obtained using MSTCIF, MSTIF, or MSTIF-TS. The Pima dataset, however, presents a unique challenge. None of the methods provides satisfactory separation between anomalous and normal samples, and the intersection area of the density plots spans almost the entire domain. Slightly better results are observed for the Satellite dataset. Meanwhile, Satimage-2 allows for clear distinction using all Isolation Forest-based methods. In the case of Shuttle, the situation is similar to that of Satimage-2, although the FDRIF method is less effective. The Smt dataset is also well separated by methods utilizing isolation forests. All methods, however, encountered difficulties in separating samples for Speech, similarly to Vertebral. The best results for Thyroid and Vowels

are recorded by MSTIF-TS. Regarding the WBC dataset, all methods yield density plots for anomalous regions that are spread over a large range. This leads to mixing and makes separation difficult. The best performance on the WBC dataset is achieved by IF, MSTCIF, MSTIF, and MSTIF-TS. Finally, for the Wine dataset, the leading method turns out to be DRIF.

The final aspect analyzed is the area of the intersections of the density plots, presented in Tables 5.40 and 5.41. These results correlate with the analysis of the plots themselves, suggesting that algorithms that effectively separate samples are characterized by lower intersection area values. Analyzing the mean values across all considered datasets, the new proposals—except FDRIF—demonstrate a leading position. However, when considering the median of these values, the situation is different. MSTCIF achieves the highest effectiveness, slightly outperforming MSTIF and MSTIF-TS.

In conclusion, the analysis of the density plots for anomalous and normal elements highlights the significance of the newly developed methods. Not only are the intersection areas low, but the shapes of the plots also favor high-quality separation of score function values. This, in turn, translates into effective classification of the elements themselves.

Chapter 6

Anomaly detection on artificially generated datasets

This chapter presents an analysis of the performance of algorithms on artificially generated anomaly detection datasets. These datasets are constructed in a way that allows for their graphical visualization in a two-dimensional space. The analysis focuses on the baseline IF method and its enhancements proposed in this study.

6.1 Dataset characteristics

Four artificially generated anomaly detection datasets are prepared. Each of them is characterized by a specific distribution of elements—samples labeled as normal are arranged in shapes such as circles, a rectangle, rings, and a spiral. Anomalies are uniformly distributed such that they do not overlap with the region occupied by normal samples. A detailed description of the datasets is provided in Table 6.1 and visualized in Figure 6.1.

Table 6.1: Characteristics of the generated anomaly detection datasets.

Dataset name	Number of observations	Number of attributes	Number of anomalies
Circles	5100	2	100 (1.96%)
Rectangle	5100	2	100 (1.96%)
Target	5100	2	100 (1.96%)
Spiral	5100	2	100 (1.96%)

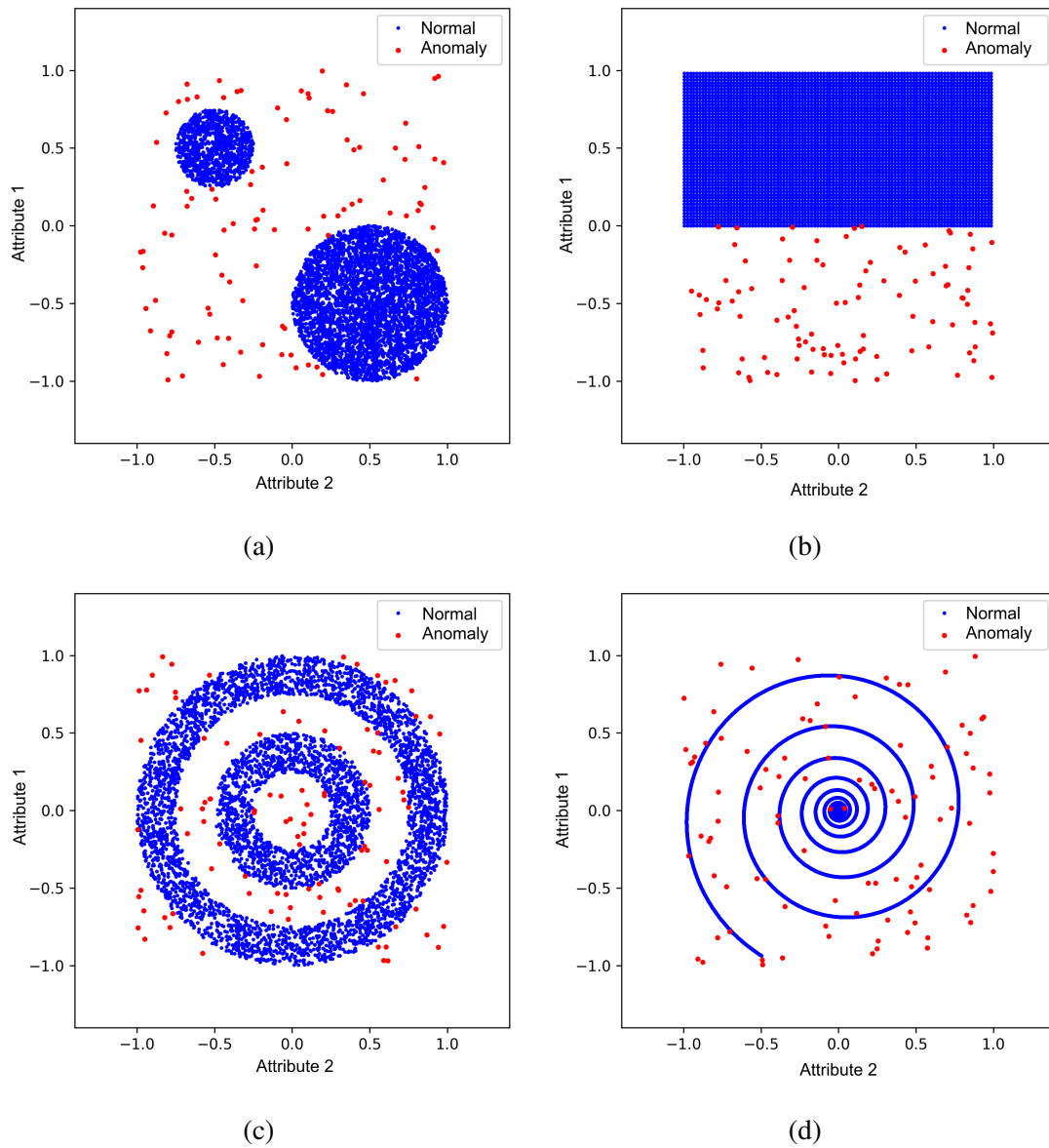


Figure 6.1: Graphical presentation of the generated anomaly detection datasets. (a) Circles, (b) Rectangle, (c) Target, (d) Spiral. Normal samples are marked in blue, while anomalies are marked in red.

6.2 Heatmaps of score function values

To provide intuition about the operation of the algorithms, heatmaps of the score function values are presented.

6.2.1 Results

Figures 6.2–6.5 illustrate the score function values for a single isolation tree, reflecting its detection capabilities when operating autonomously. Figures 6.6–6.9 show the same scenario, but using a forest of one hundred isolation trees.

6.2.2 Discussion of the results

We begin the analysis of the results using heatmaps created from the presented methods based on a single isolation tree. The results illustrate the quality of element assessment performed by such a tree. Thus, for trees with good detection properties, it can be assumed that it is possible to form a forest with fewer trees without losing detection accuracy. This translates to reduced training and evaluation time. Let us consider the individual algorithms and their heatmaps. The IF method clearly fails to outline the original normal region. Attribute reduction methods, namely DRIF and FDRIF, show better results, highlighting a trace of the region—particularly evident in the Circles dataset. Meanwhile, score values for the Rectangle dataset are less satisfactory, and the attribute reduction method using fuzzy FCM gained an advantage. For the Target and Spiral datasets, cluster contours can be observed; however, anomalies embedded within normal points are undetectable by these methods. On the other hand, anomalies located outside the clusters can be correctly classified. Now let us focus on the MSTCIF approach. Among the examined solutions, MSTCIF generates heatmaps most similar to those produced by IF. This results from its operational scheme being very close to that of the baseline IF. Nevertheless, MSTCIF results are more precise: it shows a delicate outline of the normal region and characteristic bands due to the technique of constructing isolation nodes through partitioning. The best results are achieved by approaches with a newly introduced tree construction strategy—specifically MSTIF and MSTIF-TS. The quality of training sample representation is already high with a single isolation tree. Moreover, the outline of the normal region is clearly visible. Anomalous elements sampled during training appear on the maps as small areas with elevated score values. With increasing distance from such samples, the anomaly level increases—an expected and desirable effect. The MSTIF-TS algorithm undoubtedly presents the best results. Even using only a single isolation tree, the normal region is well outlined.

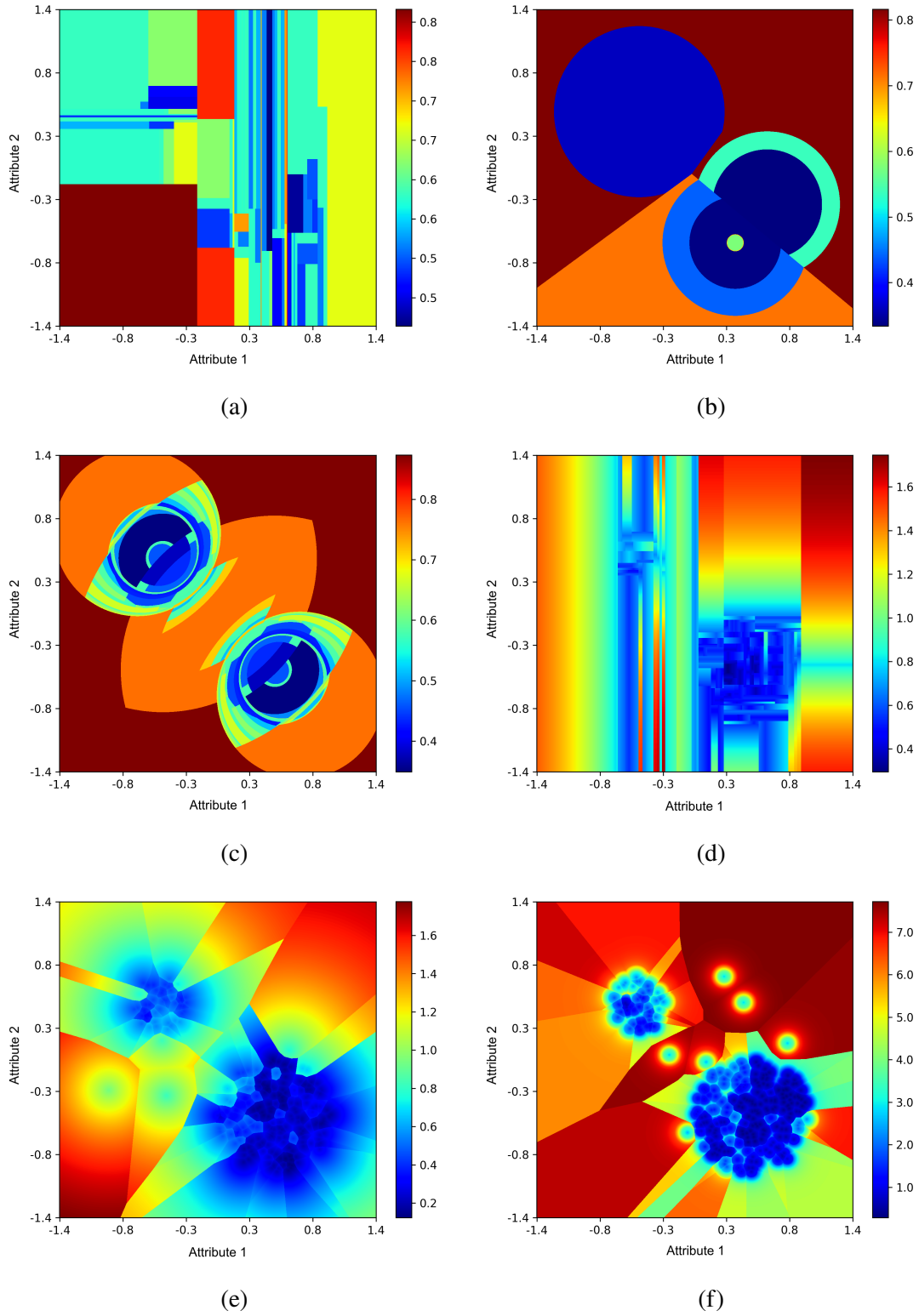


Figure 6.2: Heatmaps of the score function values for the Circles dataset using a forest composed of a single isolation tree. Methods: (a) IF, (b) DRIF, (c) FDRIF, (d) MSTCIF, (e) MSTIF, and (f) MSTIF-TS.

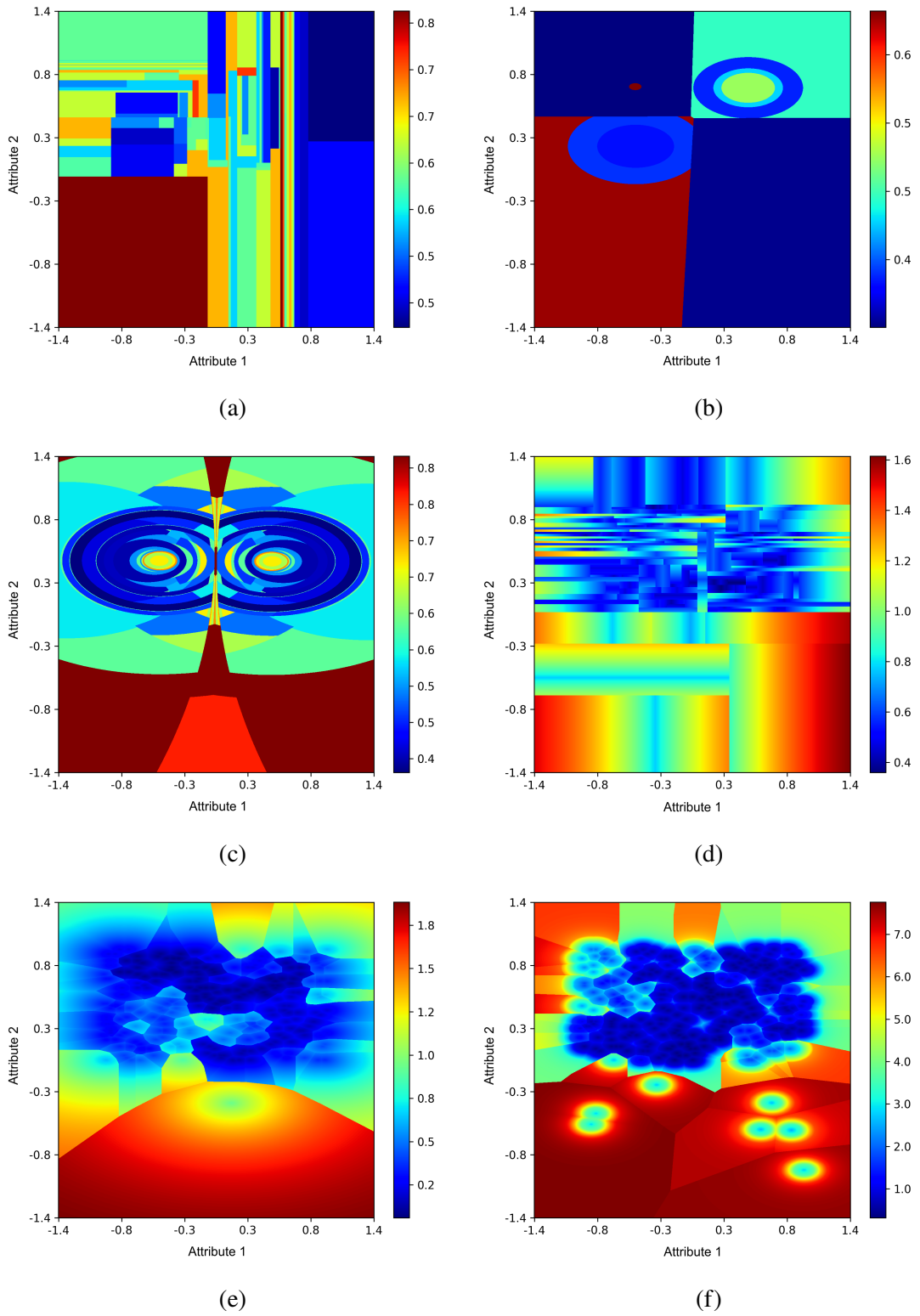


Figure 6.3: Heatmaps of the score function values for the Rectangle dataset using a forest composed of a single isolation tree. Methods: (a) IF, (b) DRIF, (c) FDRIF, (d) MSTCIF, (e) MSTIF, and (f) MSTIF-TS.

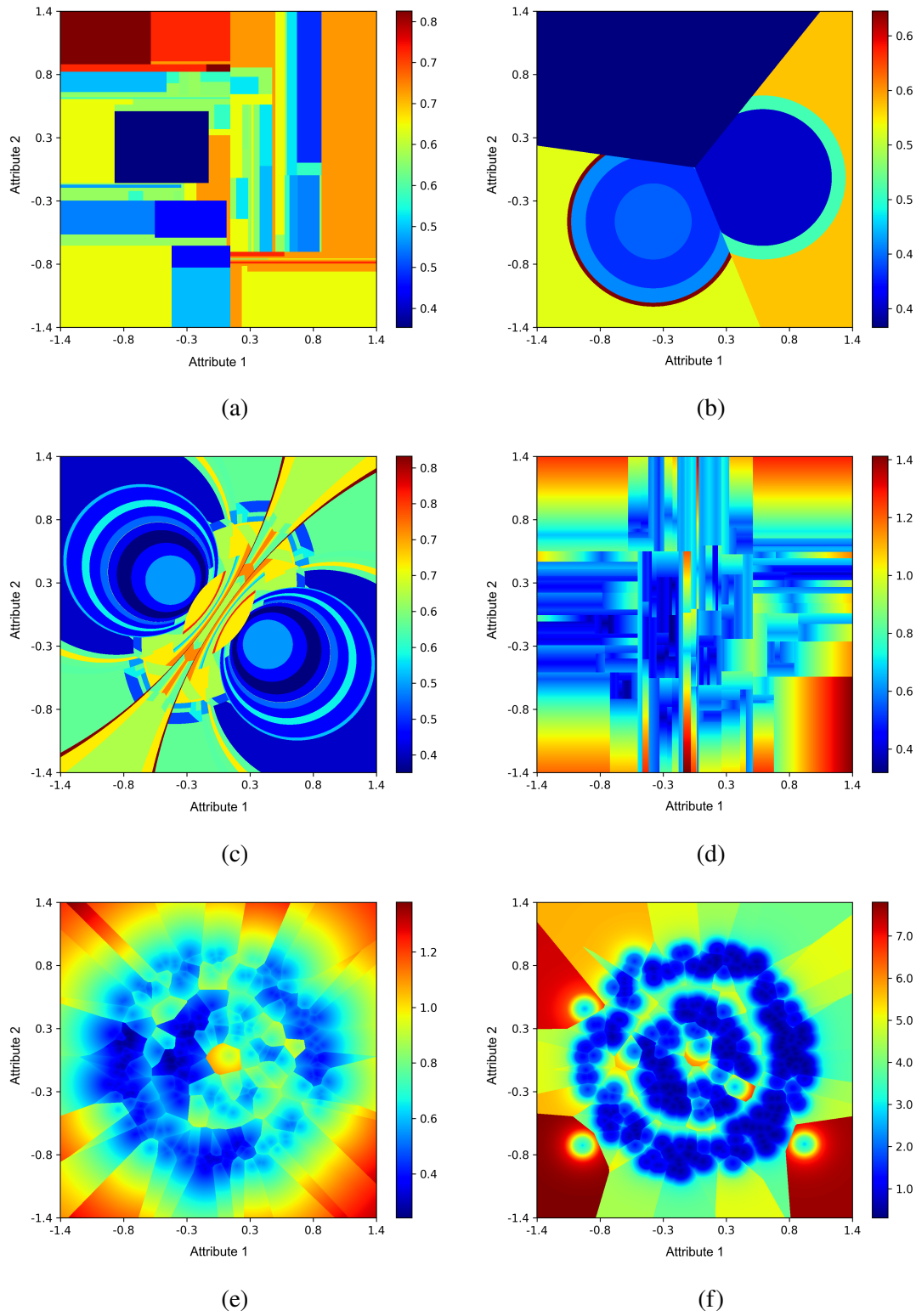


Figure 6.4: Heatmaps of the score function values for the Target dataset using a forest composed of a single isolation tree. Methods: (a) IF, (b) DRIF, (c) FDRIF, (d) MSTCIF, (e) MSTIF, and (f) MSTIF-TS.

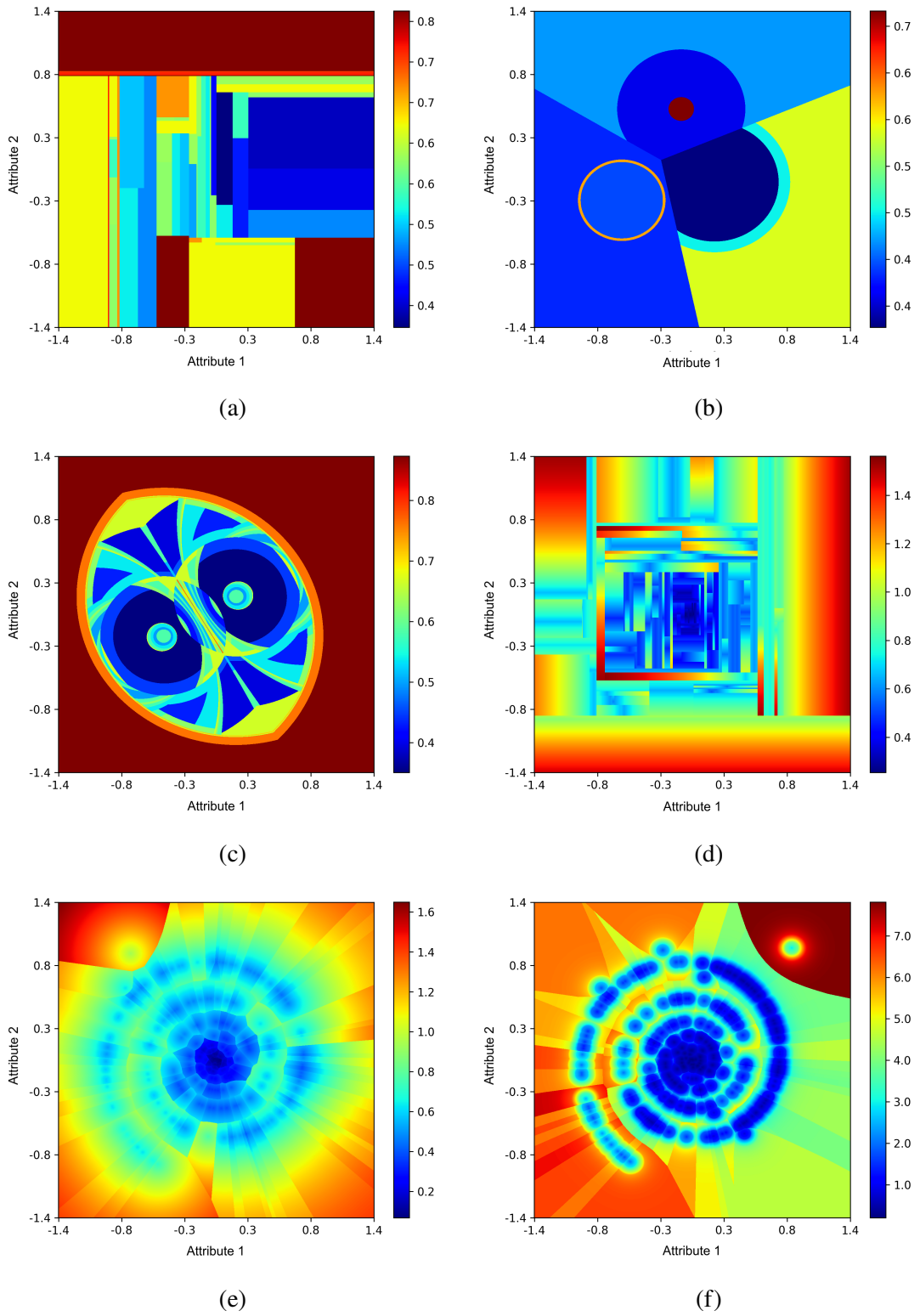


Figure 6.5: Heatmaps of the score function values for the Spiral dataset using a forest composed of a single isolation tree. Methods: (a) IF, (b) DRIF, (c) FDRIF, (d) MSTCIF, (e) MSTIF, and (f) MSTIF-TS.

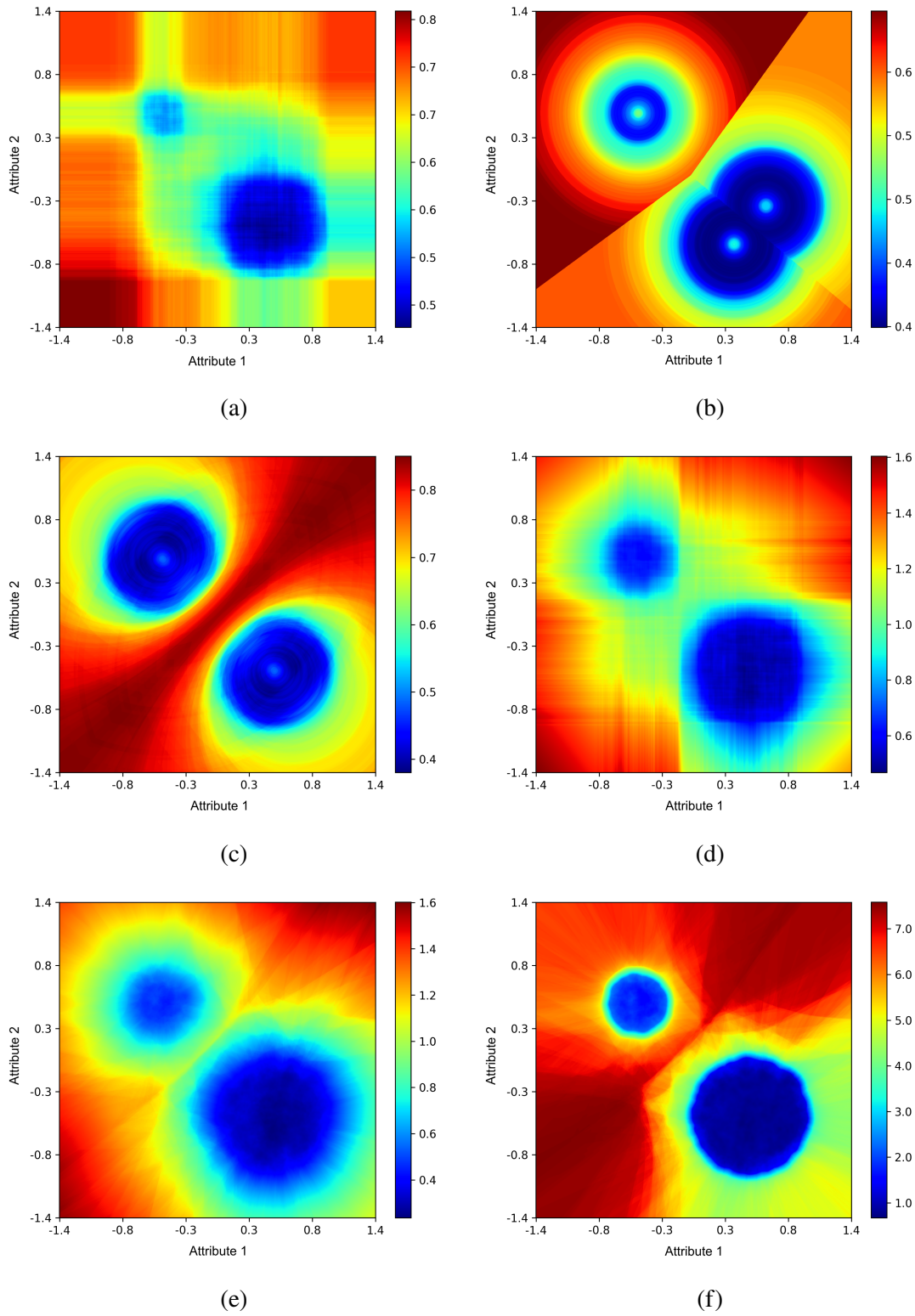


Figure 6.6: Heatmaps of the score function values for the Circles dataset using a forest of one hundred isolation trees. Methods: (a) IF, (b) DRIF, (c) FDRIF, (d) MSTCIF, (e) MSTIF, and (f) MSTIF-TS.

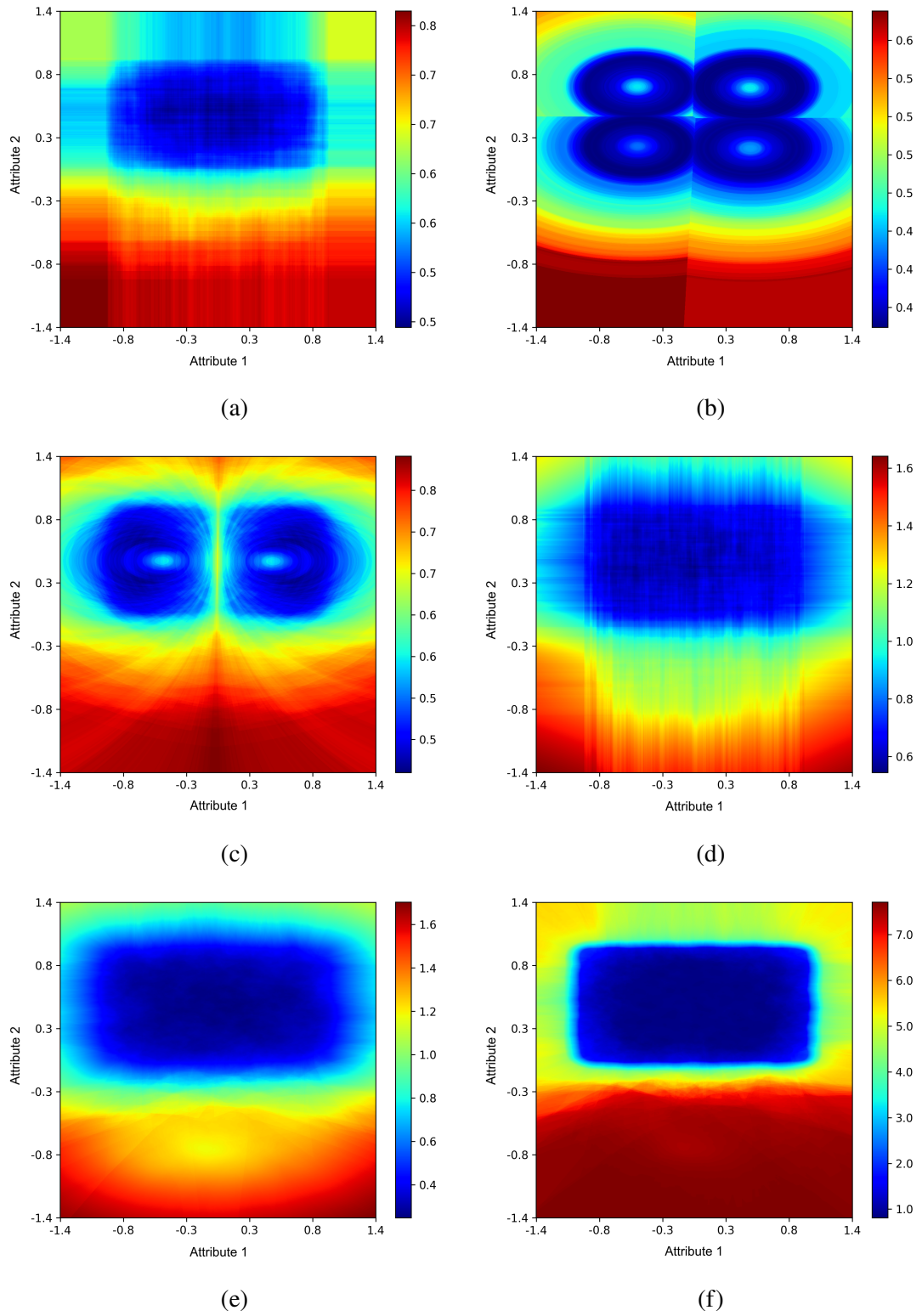


Figure 6.7: Heatmaps of the score function values for the Rectangle dataset using a forest of one hundred isolation trees. Methods: (a) IF, (b) DRIF, (c) FDRIF, (d) MSTCIF, (e) MSTIF, and (f) MSTIF-TS.

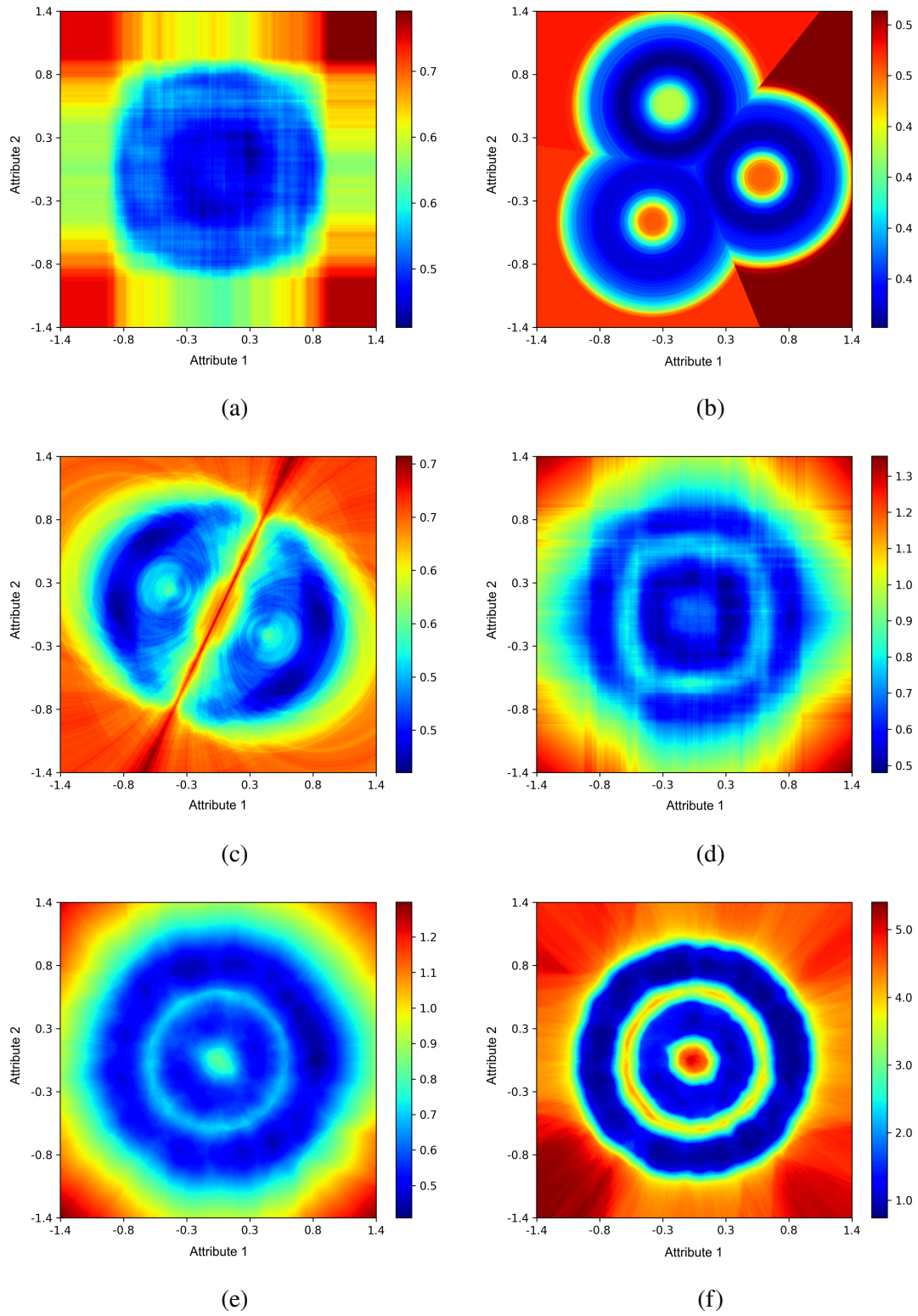


Figure 6.8: Heatmaps of the score function values for the Target dataset using a forest of one hundred isolation trees. Methods: (a) IF, (b) DRIF, (c) FDRIF, (d) MSTCIF, (e) MSTIF, and (f) MSTIF-TS.

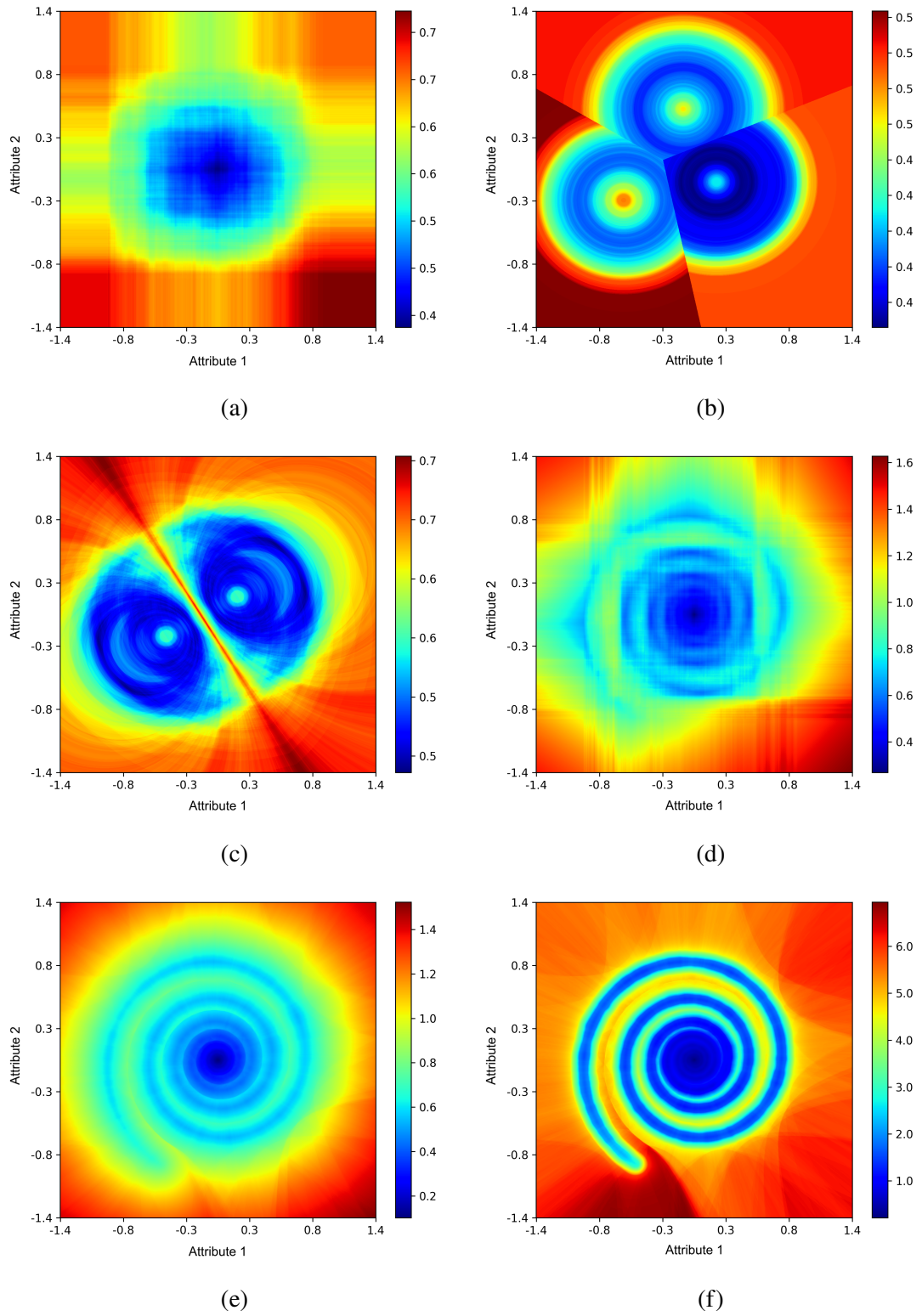


Figure 6.9: Heatmaps of the score function values for the Spiral dataset using a forest of one hundred isolation trees. Methods: (a) IF, (b) DRIF, (c) FDRIF, (d) MSTCIF, (e) MSTIF, and (f) MSTIF-TS.

Isolation Forest-based methods show their efficiency when the forest contains more trees. Therefore, results for forests of one hundred trees are presented. A clear improvement in detection quality is visible compared to single-tree forests. In each case, the normal region is properly delineated, although contours and score values differ between approaches. The IF algorithm is characterized by noticeable rectangular areas, whereas the actual set of normal samples resembles a circular shape. Moreover, IF heatmaps show specific low-score bands in locations where anomalies are present, potentially leading to misclassification. Notably, the difference in quality between single-tree and hundred-tree forests is greatest for IF. Now let us evaluate cluster-based techniques, i.e., DRIF and FDRIF. Distinct low-score clusters are observed due to the high sample concentration. For more challenging datasets such as Target and Spiral, the contour areas diverge from the true positions of the objects. This behavior—the inability to detect anomalies within normal areas—is also present in the single-tree forest analysis. Different behavior is seen in heatmaps from the remaining three methods: MSTCIF, MSTIF, and MSTIF-TS. Each sample set is effectively outlined. However, for MSTCIF, strip-like artifacts—similar to those in IF—appear. The Rectangle dataset illustrates this issue best. It is worth noting that MSTCIF, MSTIF, and MSTIF-TS effectively identify anomalies located between normal areas. Additionally, the methods based on the new MST-based tree construction perform best. It should be noted that the forest construction phase is the same for these approaches, with the difference being in the score function calculation. Analyzing MSTIF-TS with its TS-model-based score function, we observe that the shape of the normal region nearly fully overlaps with the training data distribution. MSTIF yields similar results, though the region borders are slightly more diffused. There is a risk that points near these borders may be misclassified. An interesting phenomenon for MSTIF and MSTIF-TS is the complete disappearance of ring-like artifacts when using a forest of many trees. This behavior is attributed to the averaging process used in the score function, which suppresses isolated artifacts.

By examining score function heatmaps, we assessed classification quality both for individual isolation trees and for forests of one hundred trees. The visual analysis of results shows higher detection quality for the newly proposed methods compared to the baseline IF. This leads to the conclusion that comparable detection performance can be achieved with fewer trees, resulting in faster response time in both training and evaluation phases. Cluster analysis-based methods perform effectively on datasets where clusters can be distinctly identified.

For more challenging collections—Target and Spiral—very good detection results are achieved by three algorithms: MSTCIF, MSTIF, and MSTIF-TS. Undoubtedly, the best solution in this type of heatmap comparison is the MSTIF-TS algorithm. In each case under analysis, both normal and anomalous areas are accurately and correctly identified. Moreover, it is crucial that the boundary zones between these areas are clearly marked, indicating the presence of a sharp transition in score function values. Such precision in boundary determination allows for more accurate classification of points located in those regions.

6.3 Graphical representation of the anomaly detection process

The second aspect considered in our analysis is the graphical representation of the anomaly detection process. In this subsection, we focus on visualizing the classification and identification of normal and anomalous elements. The detection threshold for each method is determined by analyzing the ROC curve; see Subsection 5.3.

6.3.1 Results

The classification results are illustrated in Figures 6.10–6.13. Additionally, elements that are correctly classified as anomalies or normal are marked. Particular attention is also given to elements that are misclassified. Detailed counts of these elements are included in the legends of the respective figures.

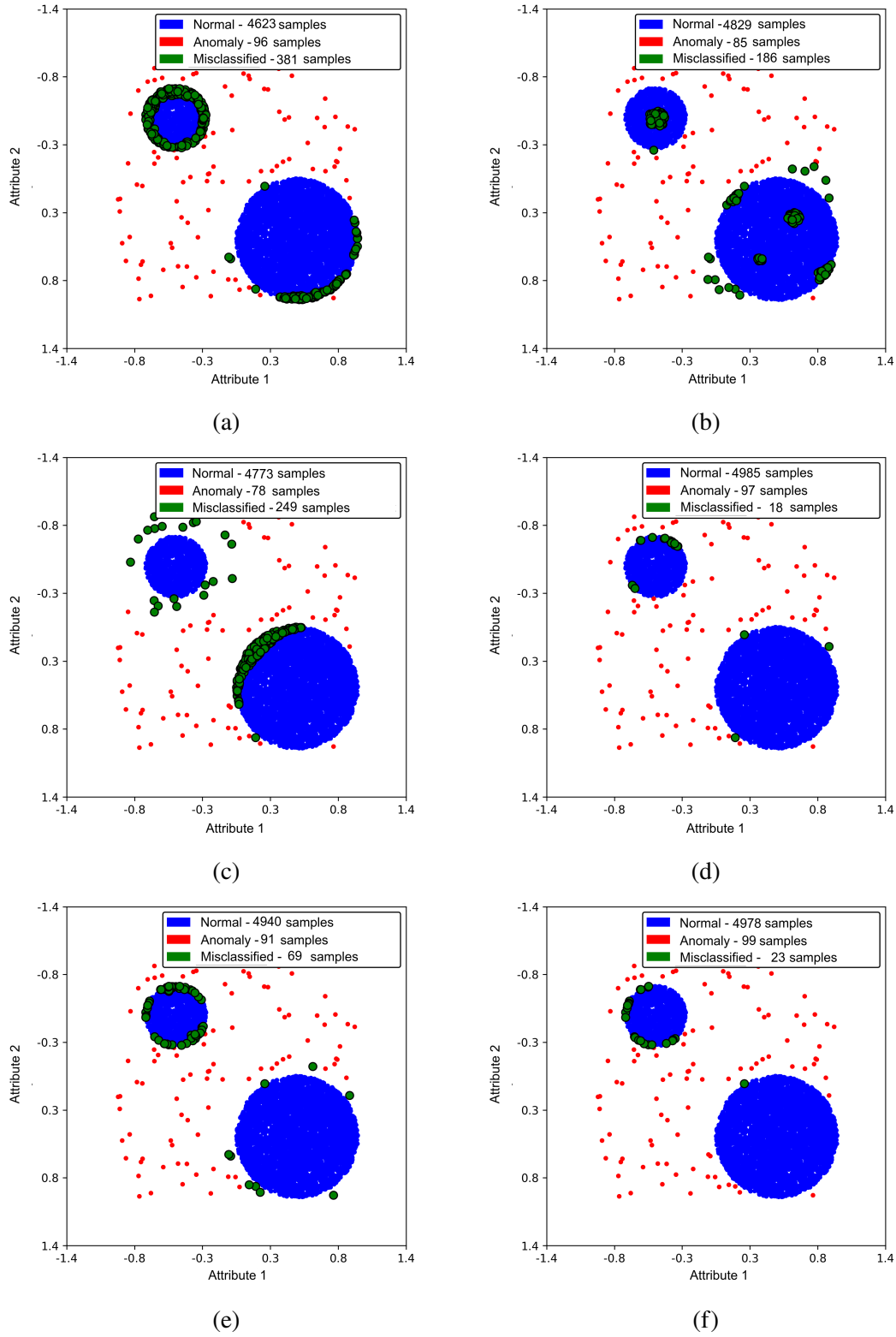


Figure 6.10: Graphical representation of the anomaly detection process for the Circles dataset. Methods: (a) IF, (b) DRIF, (c) FDRIF, (d) MSTCIF, (e) MSTIF, and (f) MSTIF-TS.

6.3 Graphical representation of the anomaly detection process

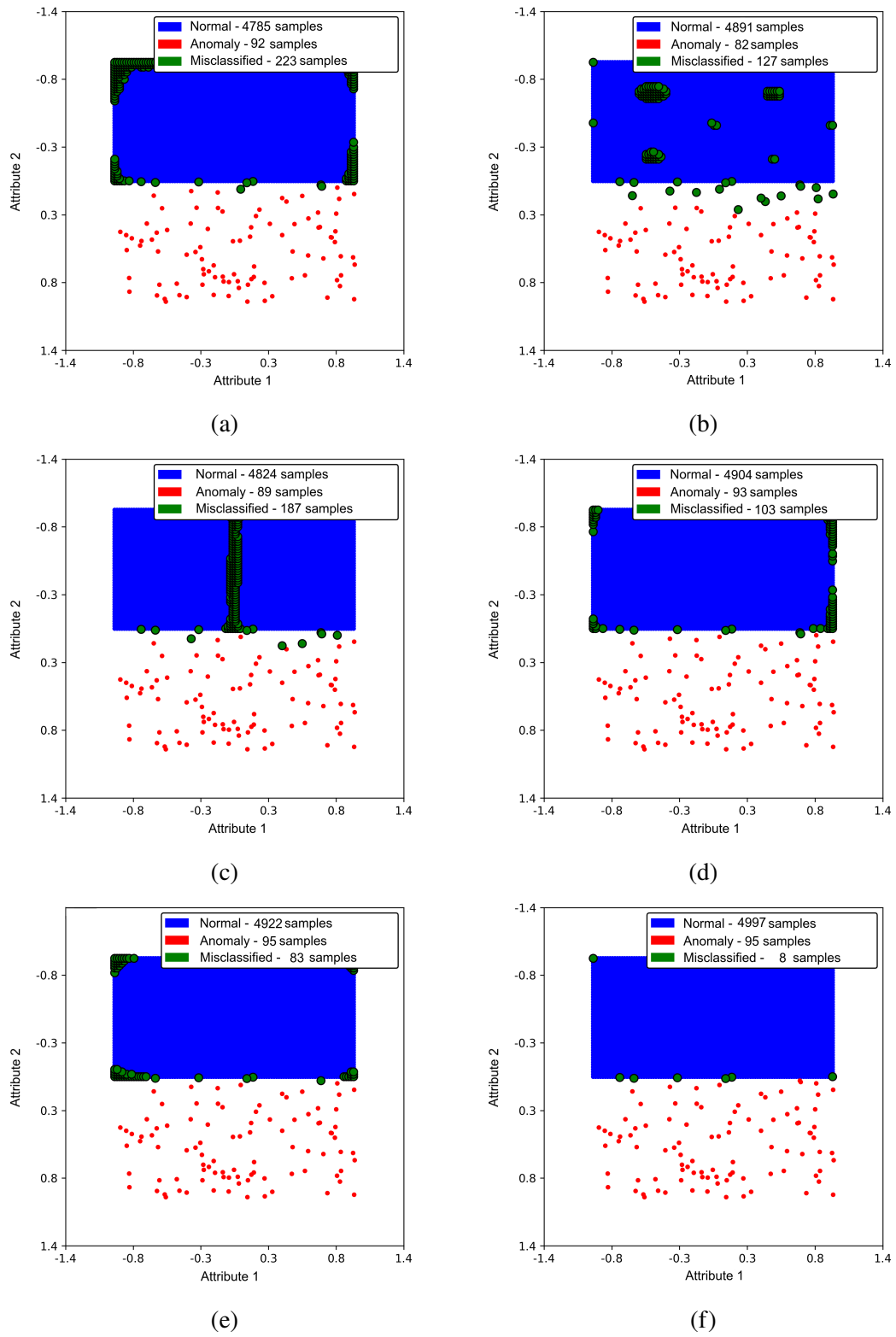


Figure 6.11: Graphical representation of the anomaly detection process for the Rectangle dataset. Methods: (a) IF, (b) DRIF, (c) FDRIF, (d) MSTCIF, (e) MSTIF, and (f) MSTIF-TS.

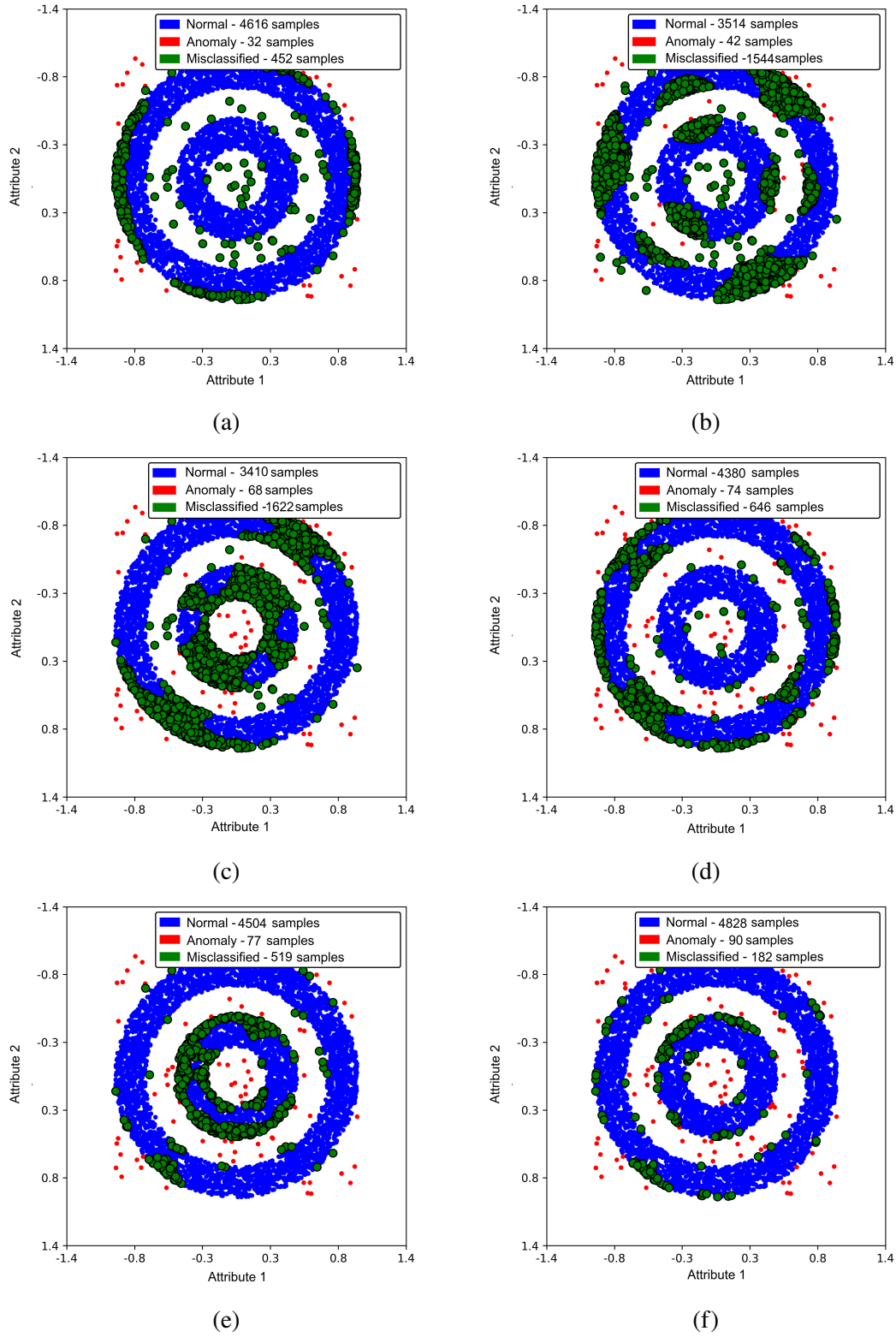


Figure 6.12: Graphical representation of the anomaly detection process for the Target dataset. Methods: (a) IF, (b) DRIF, (c) FDRIF, (d) MSTCIF, (e) MSTIF, and (f) MSTIF-TS.

6.3 Graphical representation of the anomaly detection process

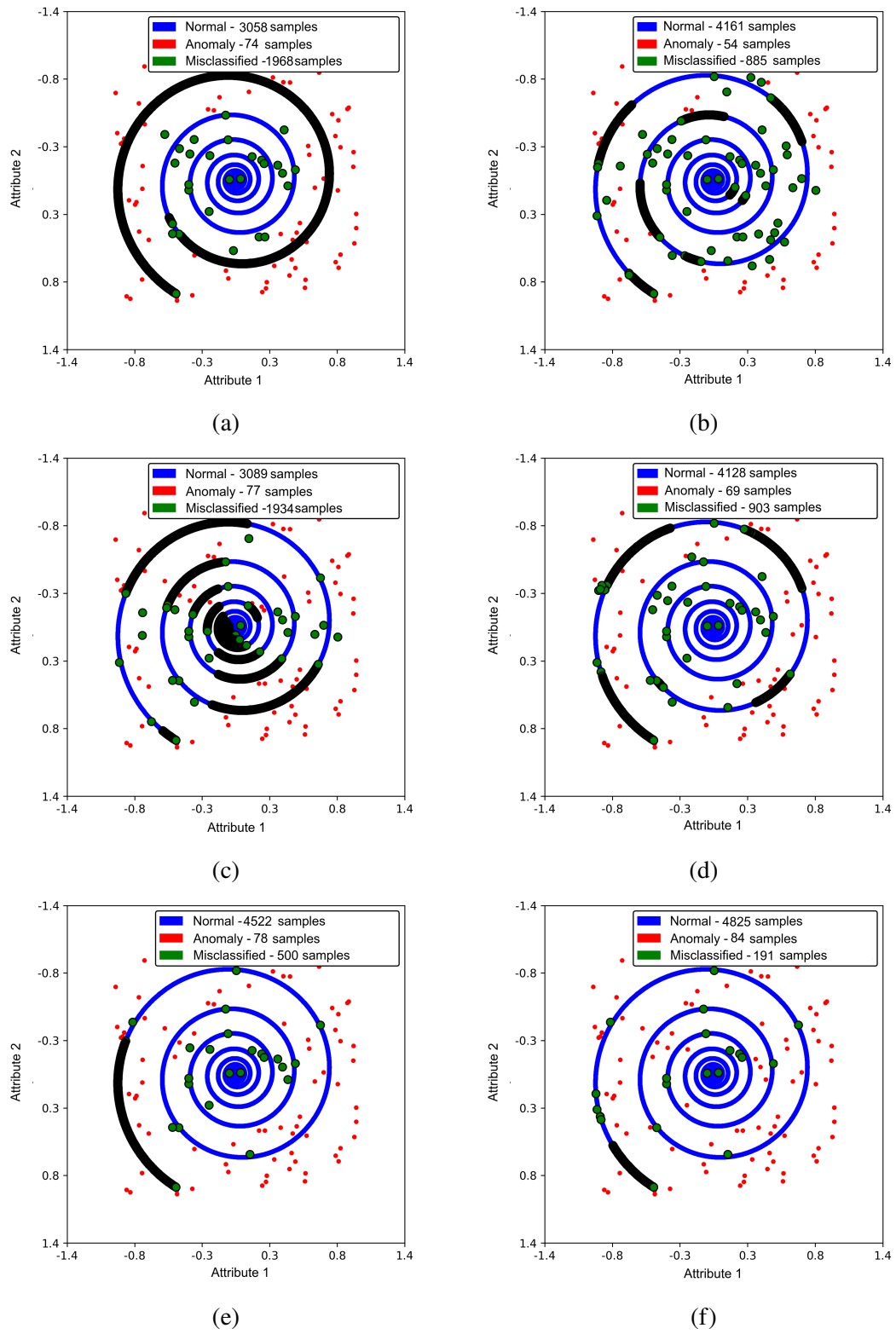


Figure 6.13: Graphical representation of the anomaly detection process for the Spiral dataset. Methods: (a) IF, (b) DRIF, (c) FDRIF, (d) MSTCIF, (e) MSTIF, and (f) MSTIF-TS.

6.3.2 Discussion of the results

We analyze the results of the anomaly detection process for the generated datasets. Let us first focus on the Circle dataset. Using the IF method, classification errors are primarily observed along the edges of the denser sample regions. Focusing on the smaller circle representing a normal region, the results show misclassifications around its entire perimeter. Regarding the larger circle, anomalies mainly occur near sparse outlier points adjacent to the normal region. The second method, based on k -Means, exhibits misclassified values near cluster centers. For the larger circle representing typical elements, two clusters are formed. Here, incorrect labels appear on the boundary between normal and anomalous regions, unlike in the smaller circle. In the case of the third approach, based on attribute reduction using FCM, inconsistencies are observed near the smaller circle and, for the larger one, along the edge facing the other normal region. The highest detection accuracy is achieved by the MSTCIF algorithm. Misclassifications appear on the border of the normal region, especially for the smaller circle; however, such cases are rare. Detection using the final two methods, namely MSTIF and MSTIF-TS, results in a greater number of misclassifications in the smaller circle area, although they are significantly fewer for MSTIF-TS. For the belt separating anomalies from normal points in the larger circle, the detectors produce interesting results. MSTIF shows errors mainly for points slightly away from the normal region, whereas MSTIF-TS achieves nearly perfect detection, with just a single misclassified sample. Now, let us consider the next dataset, Rectangle. The results obtained show many similarities to those from the Circle dataset. For the IF method, mislabeling primarily affects the edges of the normal region, which, in the case of the rectangular shape, manifests at the corners. The attribute-reduction-based algorithms differ in their behavior. DRIF produces misclassifications in the cluster center, while FDRIF makes mistakes at the cluster boundaries. Looking at the MSTCIF and MSTIF approaches, behavior similar to IF can be observed, with incorrect labels at the corners of the normal region. Finally, the MSTIF-TS algorithm yields the most favorable results. Errors appear along the shape's boundary but are minor—only 8 samples. The last two datasets analyzed—Target and Spiral—present more complex structures and thus more challenging data models. More specifically, the number of misclassifications is significantly higher than in the previously discussed datasets. Summarizing the performance of the baseline IF algorithm, it divides the space into two main zones: A central one where all samples are considered typical and an outer one where they are treated as outliers. This approach leads to many classification errors, particularly in the Spiral collection. In this case, a substantial portion of the outer spiral is mistakenly identified as anomalous. Clustering-based techniques show different results. Although their performance is similar to that observed for the first two collections, a key difference lies in the significantly higher number of classification errors. For DRIF, errors occur mainly in the central parts of clusters

and at their peripheries. FDRIF is characterized by misclassifications in the zone separating the clusters, consistent with observations from the first two collections. Next, the MSTCIF algorithm effectively identifies some anomalies located between normal zones. However, in the Spiral dataset, the central part of the outlier samples remains undetected. Additionally, bands of misclassified points within the normal region are visible. As observed for the Circles and Rectangle datasets, the MSTIF and MSTIF-TS strategies—featuring an innovative method of forming isolation trees—demonstrate the highest detection effectiveness. While both approaches effectively capture anomalous samples, MSTIF-TS proves to be more accurate. Some misclassifications of normal elements are observed, especially at the boundaries of the normal regions. In this specific case, MSTIF-TS once again outperforms the competing techniques, providing the most accurate sample labeling.

The graphical visualization of detection outcomes for the presented datasets reveals the varying responses of the algorithms to data. We obtained observations comparable to those in the heatmap study described in Subsection 6.2.2. Moreover, this analysis provided more specific data regarding detection effectiveness, allowing us to identify both the strengths and weaknesses of the discussed methods. It is worth emphasizing that the detection threshold for each algorithm is determined based on the ROC curve. Thus, there is reason to believe that a more precise threshold adjustment could potentially improve the obtained results. Clustering-based techniques perform better in situations where normal samples form clearly defined groups, which is a consequence of the nature of k -Means or FCM grouping. The most promising results are obtained for MSTCIF, MSTIF, and MSTIF-TS, with MSTIF-TS standing out for its highest accuracy.

Chapter 7

Summary

The final chapter of this work focuses on synthesizing the research carried out and drawing key conclusions. In addition, it outlines the prospects for future work and highlights potential directions for further studies.

7.1 Outcomes of the work

This study introduced five enhancements to the Isolation Forest algorithm:

1. Dimensionality reduction using the k -Means method, Dimensionality Reduction-Based Isolation Forest (DRIF),
2. Dimensionality reduction using the fuzzy FCM method, Fuzzy Dimensionality Reduction-Based Isolation Forest (FDRIF),
3. Selection of an optimal split point in isolation nodes using clustering with the minimal spanning tree algorithm, Minimal Spanning Tree Clustering for Isolation Forest (MSTCIF),
4. Construction of isolation trees using the minimal spanning tree algorithm, Minimal Spanning Tree-Based Isolation Forest (MSTIF),
5. Construction of isolation trees using the minimal spanning tree algorithm with improved anomaly scoring based on the Takagi-Sugeno fuzzy inference model, Minimal Spanning Tree-Based Isolation Forest with Evaluation Function Built on the Fuzzy Rules in Takagi-Sugeno Model (MSTIF-TS).

The work included a series of numerical experiments conducted on both real-world anomaly detection datasets and synthetically generated datasets. Comparisons with other competitive anomaly detection techniques are performed. A wide range of classification quality measures is analyzed, including those dependent on a detection threshold and those independent

of its selection. The runtime of both algorithm phases is recorded. Additionally, the possibilities of modifying the hyperparameters of the proposed solutions are examined. The obtained results are illustrated through graphical representations, including plots of score separability, heatmaps of the scoring functions, and visualizations of anomaly detection.

7.2 Conclusions

The analyses of the performance of the newly proposed methods indicate their significant effectiveness and potential applicability. Furthermore, a detailed examination of threshold-independent metrics, such as ROC AUC and PR AUC, allowed for the identification of methods with the greatest potential effectiveness. On the other hand, the evaluation of threshold-dependent measures revealed slightly different results, yet the general performance trends remained consistent. Particularly noteworthy is the MSTIF-TS method, which most frequently ranked among the top performers. It is important to highlight that better average results are achieved compared to the original IF method—for instance, an improvement of 3% in ROC AUC and 2% in PR AUC. Determining threshold values based on the ROC curve made it possible to identify optimal thresholds for different methods. Nevertheless, there is potential for further fine-tuning of these thresholds to improve performance. It is also worth emphasizing that the diversity of the evaluated metrics reflects a wide range of possible applications, depending on the specific goal to be achieved. Therefore, it is recommended to select the metric that best suits the characteristics of the problem at hand. Notable examples of significant improvements with MSTIF-TS include an increase of over 4% in average accuracy compared to IF, and nearly 3% in the F1 score.

This study demonstrated that applying dimensionality reduction and transforming the data using clustering techniques can significantly enhance the performance of the Isolation Forest method, particularly without the need to modify the core IF algorithm itself. Heatmap analyses showed that reduction techniques are especially effective when clusters are well separated. This aligns with the expected results of clustering methods such as k -Means and FCM. At the same time, the observations suggest that alternative clustering techniques—such as DBSCAN—could be used to identify more complex and unevenly distributed clusters. This approach may prove beneficial in specific contexts, especially in datasets where anomalous regions are interspersed with normal ones. Anomalies located between normal regions are often difficult to isolate. This is especially important when the anomalies take more complex forms, as in datasets such as Target or Spiral. Additionally, performing clustering using the MST algorithm may serve as a valuable tool in the dimensionality reduction process.

The analysis of the improvement concerning the choice of split value in isolation nodes during the tree-building phase showed that, although the randomness of the IF method is often

emphasized in the literature, a well-chosen split value may actually be preferable. Furthermore, incorporating additional factors into the scoring function led to improved algorithm performance. Therefore, it is justified to consider other characteristics of isolation trees that could be incorporated into the scoring function to potentially enhance method effectiveness.

The innovative approach of constructing isolation trees by integrating elements using the MST algorithm proved to be effective in forming isolation nodes. Importantly, this technique produces isolation trees that are very similar to the original IF tree; however, during their construction, a second tree is also generated—namely, the minimal spanning tree. The scoring function that considers both trees enables highly effective anomaly detection. It should be noted that this configuration yields two components in the scoring function: The first is related to the isolation tree (i.e., the node depth in the tree), and the second is associated with the minimal spanning tree (i.e., the distance to the nearest node—more precisely, the edge weight connecting the analyzed vertex to the MST). The use of the Takagi-Sugeno fuzzy inference model enables more precise label assignment than a simple aggregation of these components.

The runtime analysis revealed a variety of performance characteristics among the proposed solutions. The selection of a specific algorithm for a given task may depend on particular features, both those related to model training and those related to the speed of labeling new samples. Moreover, a detailed analysis of hyperparameters in the presented methods revealed opportunities for optimization, allowing for either faster execution or greater effectiveness.

Graphical visualizations of synthetically generated two-dimensional datasets enabled an in-depth analysis of the methods' behavior. Although the data are artificially generated, it is possible to observe the behavior of individual algorithms. This type of analysis can reveal specific elements and characteristics that require optimization. It is worth emphasizing that visual evaluation helps identify regions in which the methods exhibit lower classification quality. This, in turn, makes it possible to focus improvement efforts precisely on those regions. When considering a selected technique in the context of high-dimensional data, it may be beneficial to project it into two-dimensional space. This allows for a more comprehensible analysis of the method's behavior.

7.3 Directions for future research

This work introduced significant innovations in anomaly detection algorithms based on the Isolation Forest. However, there is still room for improving the results and developing new algorithms and techniques. Therefore, it is essential to identify potential directions for future research.

Several areas can be pointed out where further studies may introduce new ideas. On the one hand, there is an opportunity to reduce the execution time of the methods, for example, by sampling a subset of elements during training or during clustering in dimensionality

reduction techniques. On the other hand, the potential for optimizing clustering algorithms has not yet been fully exploited. For example, in the literature [89], reduced computational complexity and runtime are achieved for the FCM clustering algorithm. In the domain of data preprocessing, attention should also be paid to methods such as MSTCIF, which enhances the selection of attribute values in isolation tree nodes. It is also important to reconsider the selection of the attribute itself, whose value serves as the split point. Perhaps a deterministic choice of attribute could be correlated with the determination of its value, similar to MSTCIF. Furthermore, the innovation of modifying the tree-building technique by using a minimal spanning tree opens new possibilities. It is worth exploring whether other graph structures or approaches could support the construction of isolation trees. In addition, there is potential to optimize the MST-building algorithm itself [121], for instance, by parallelizing the construction process. Alternatively, instead of Kruskal's algorithm, other MST algorithms could be used, as suggested in [131]. Another promising direction is the exploration of alternative dissimilarity measures. Rather than relying on Euclidean distance, one might examine metrics such as Chebyshev, Mahalanobis, or Manhattan distance.

The methods developed in this study are fully unsupervised. However, the literature includes attempts to introduce weak supervision into the Isolation Forest. In the work [18], weak supervision is introduced to the Isolation Forest in order to improve classification accuracy and speed up the evaluation process. Therefore, it is worth considering the possibility of incorporating supervision elements into the presented methods or developing a solution based on weakly supervised training. This may potentially contribute to further improvements in detection quality and speed.

Although two independent clustering-based methods are developed as part of this work, an especially interesting idea would be to use two different clustering techniques as the foundation for anomaly detection. In particular, elements suspected of being anomalies would be those for which the clustering results disagree—i.e., they are assigned to different clusters by different clustering techniques. It is also worth noting that in the case of k -Means we are dealing with “hard” cluster membership, while in FCM the parameter m determines the fuzziness of the clusters. In this work, the value of parameter m is adopted based on the literature [128], [174], which indicated an optimal value. However, it should be noted that modifying this parameter may significantly influence the adaptation of the method to the data characteristics. This results from the fact that clusters can be either more compact or more dispersed.

Therefore, when choosing this technique, a thorough analysis is required—namely, answering the question of what value of parameter m should be adopted to ensure that the clusters are best suited to the structure of the data. It should be emphasized that the proposed method is developed as a fully unsupervised one, and thus no tuning of the parameter m is performed—an optimal value is adopted as commonly accepted in the literature. Future work should further explore this aspect to optimize anomaly detection performance.

A final, highly interesting direction for research is the optimization of the Isolation Forest in terms of selecting trees that provide the best results. The essence of the Isolation Forest and related methods is the use of a large ensemble of trees constructed from random samples. However, not all trees are equally effective, which justifies the need for a large number of them. Considering the use of a genetic algorithm that would eliminate trees with weaker detection properties—replacing them with new, randomly generated trees—appears to be a promising approach. The possibility of mutating existing trees in the forest should also be considered, for instance, by replacing part of the sample set with new data and reconstructing the isolation tree based on this modified set. Another interesting idea is to analyze the classification performance of trees based on their node balance. Less balanced trees may turn out to be more effective for detection than balanced ones. Moreover, reflecting on the distribution of samples across tree levels could prove beneficial. Optimal trees may be characterized by upper levels populated mostly by normal elements, while anomalies may be concentrated in a small number of lower levels, with a minimized number of samples on intermediate levels.

Bibliography

- [1] N. Abe, B. Zadrozny, and J. Langford, “Outlier detection by active learning,” in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2006, pp. 504–509.
- [2] O. Abualghanam, H. Alazzam, E. Alhenawi, M. Qatawneh, and O. Adwan, “Fusion-based anomaly detection system using modified isolation forest for internet of things,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 1, pp. 131–145, 2023.
- [3] C. C. Aggarwal, *Data mining*. Springer, 2015.
- [4] C. C. Aggarwal and S. Sathe, “Theoretical foundations and algorithms for outlier ensembles,” *ACM SIGKDD Explorations Newsletter*, vol. 17, no. 1, pp. 24–47, 2015.
- [5] W. S. Al Farizi, I. Hidayah, and M. N. Rizal, “Isolation forest based anomaly detection: A systematic literature review,” in *2021 8th International Conference on Information Technology, Computer and Electrical Engineering*, IEEE, 2021, pp. 118–122.
- [6] E. Anello, C. Masiero, F. Ferro, F. Ferrari, B. Mukaj, A. Beghi, and G. A. Susto, “Anomaly detection for the industrial internet of things: An unsupervised approach for fast root cause analysis,” in *2022 IEEE Conference on Control Technology and Applications*, IEEE, 2022, pp. 1366–1371.
- [7] F. Angiulli, S. Basta, and C. Pizzuti, “Distance-based detection and prediction of outliers,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 2, pp. 145–160, 2005.
- [8] F. Angiulli and C. Pizzuti, “Fast outlier detection in high dimensional spaces,” in *European Conference on Principles of Data Mining and Knowledge Discovery*, Springer, 2002, pp. 15–27.
- [9] L. A. S. Arias, C. W. Oosterlee, and P. Cirillo, “AIDA: Analytic isolation and distance-based anomaly detection algorithm,” *Pattern Recognition*, vol. 141, 2023, Art. no. 109607.
- [10] S. Aryal, K. M. Ting, J. R. Wells, and T. Washio, “Improving iforest with relative mass,” in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2014, pp. 510–521.

-
- [11] P. Augustyniak and G. Ślusarczyk, “Quantifying coincidence in non-uniform time series with mutual graph approximation: Speech and ecg examples,” *Electronics*, vol. 12, no. 20, 2023, Art. no. 4228.
 - [12] P. Ayegba, J. Ayoola, E. Asani, and A. Okeyinka, “A comparative study of minimal spanning tree algorithms,” in *2020 International Conference in Mathematics, Computer Engineering and Computer Science*, IEEE, 2020, pp. 1–4.
 - [13] M. Ayoub, J. Khalid, and P. Karczmarek, “Fuzzy c-means based extended isolation forest for anomaly detection,” in *International Conference on Advanced Intelligent Systems for Sustainable Development*, Springer, 2022, pp. 411–418.
 - [14] T. R. Bandaragoda, K. M. Ting, D. Albrecht, F. T. Liu, and J. R. Wells, “Efficient anomaly detection by isolation using nearest neighbour ensemble,” in *2014 IEEE International Conference on Data Mining Workshop*, IEEE, 2014, pp. 698–705.
 - [15] T. R. Bandaragoda, K. M. Ting, D. Albrecht, F. T. Liu, Y. Zhu, and J. R. Wells, “Isolation-based anomaly detection using nearest-neighbor ensembles,” *Computational Intelligence*, vol. 34, no. 4, pp. 968–998, 2018.
 - [16] S. Bandyopadhyay and S. Santra, “A genetic approach for efficient outlier detection in projected space,” *Pattern Recognition*, vol. 41, no. 4, pp. 1338–1349, 2008.
 - [17] T. Barbariol, F. D. Chiara, D. Marcato, and G. A. Susto, “A review of tree-based approaches for anomaly detection,” *Control Charts and Machine Learning for Anomaly Detection in Manufacturing*, pp. 149–185, 2022.
 - [18] T. Barbariol and G. A. Susto, “TiWS-iForest: Isolation forest in weakly supervised and tiny ML scenarios,” *Information Sciences*, vol. 610, pp. 126–143, 2022.
 - [19] P. Berezinski, B. Jasiul, and M. Szpyrka, “An entropy-based network anomaly detection method,” *Entropy*, vol. 17, no. 4, pp. 2367–2408, 2015.
 - [20] J. C. Bezdek, *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media, 2013.
 - [21] J. C. Bezdek, R. Ehrlich, and W. Full, “FCM: The fuzzy c-means clustering algorithm,” *Computers & Geosciences*, vol. 10, no. 2-3, pp. 191–203, 1984.
 - [22] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “LOF: Identifying density-based local outliers,” in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, ACM, 2000, pp. 93–104.
 - [23] S. Buschjäger, P.-J. Honysz, and K. Morik, “Randomized outlier detection with trees,” *International Journal of Data Science and Analytics*, vol. 13, no. 2, pp. 91–104, 2022.

- [24] R. N. Calheiros, K. Ramamohanarao, R. Buyya, C. Leckie, and S. Versteeg, "On the effectiveness of isolation-based anomaly detection in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 18, 2017, Art. no. e4169.
- [25] R. J. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical density estimates for data clustering, visualization, and outlier detection," *ACM Transactions on Knowledge Discovery from Data*, vol. 10, no. 1, pp. 1–51, 2015.
- [26] M. Carletti, M. Terzi, and G. A. Susto, "Interpretable anomaly detection with DIFFI: Depth-based feature importance of isolation forest," *Engineering Applications of Artificial Intelligence*, vol. 119, 2023, Art. no. 105730.
- [27] M. Çelik, F. Dadaşer-Çelik, and A. Ş. Dokuz, "Anomaly detection in temperature data using DBSCAN algorithm," in *2011 International Symposium on Innovations in Intelligent Systems and Applications*, IEEE, 2011, pp. 91–95.
- [28] Y. Chabchoub, M. U. Togbe, A. Boly, and R. Chiky, "An in-depth study and improvement of isolation forest," *IEEE Access*, vol. 10, pp. 10 219–10 237, 2022.
- [29] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.
- [30] M. Chater, A. Borgi, M. T. Slama, K. Sfar-Gandoura, and M. I. Landoulsi, "Fuzzy isolation forest for anomaly detection," *Procedia Computer Science*, vol. 207, pp. 916–925, 2022.
- [31] J. Chen, J. Zhang, R. Qian, J. Yuan, and Y. Ren, "An anomaly detection method for wireless sensor networks based on the improved isolation forest," *Applied Sciences*, vol. 13, no. 2, 2023, Art. no. 702.
- [32] X. Chen, W. Xu, S. Wang, Y. Li, and Z. Lin, "An anomaly detection scheme with k-means aided extended isolation forest in RSS-based wireless positioning system," in *2022 IEEE Wireless Communications and Networking Conference*, IEEE, 2022, pp. 1910–1915.
- [33] Y. Chen, J. Zhang, and C. K. Yeo, "Privacy-preserving knowledge transfer for intrusion detection with federated deep autoencoding gaussian mixture model," *Information Sciences*, vol. 609, pp. 1204–1220, 2022.
- [34] Y. Chen, S. Wang, Q. Zhao, and G. Sun, "Detection of multivariate geochemical anomalies using the bat-optimized isolation forest and bat-optimized elliptic envelope models," *Journal of Earth Science*, vol. 32, no. 2, pp. 415–426, 2021.
- [35] J. Choudhury, P. Ky, Y. Ren, and C. Shi, "Hypersphere for branching node for the family of isolation forest algorithms," in *2021 IEEE International Conference on Smart Computing*, IEEE, 2021, pp. 418–423.

-
- [36] J. Choudhury and C. Shi, “Enhanced performance of finite boundary isolation forest (FBIF) for datasets with standard distribution properties,” in *2022 International Conference on Electrical, Computer and Energy Technologies*, IEEE, 2022, pp. 1–5.
 - [37] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT Press, 2022.
 - [38] A. Degirmenci and O. Karal, “Efficient density and cluster based incremental outlier detection in data streams,” *Information Sciences*, vol. 607, pp. 901–920, 2022.
 - [39] Z. Ding, M. Fei, and D. Du, “An online anomaly detection method for stream data using isolation principle and statistic histogram,” *International Journal of Modeling, Simulation, and Scientific Computing*, vol. 6, no. 02, 2015, Art. no. 1550017.
 - [40] Z. Ding, Y. Mo, and Z. Pan, “A novel software defect prediction method based on isolation forest,” in *2019 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering*, IEEE, 2019, pp. 882–887.
 - [41] Z. Ding and L. Xing, “Improved software defect prediction using pruned histogram-based isolation forest,” *Reliability Engineering & System Safety*, vol. 204, 2020, Art. no. 107170.
 - [42] M. Dolecki, P. Karczmarek, Ł. Gałka, M. Plechawska-Wójcik, M. Kaczorowska, M. Tokovarov, and D. Czerwinski, “On the understanding of anomalies in the oculography data and their classification with an application of fuzzy aggregators,” in *2022 IEEE International Conference on Fuzzy Systems*, IEEE, 2022, pp. 1–6.
 - [43] M. Dolecki, P. Karczmarek, Ł. Gałka, M. Zawadka, J. Smolka, M. Skublewska-Paszkowska, E. Lukasik, P. Powroznik, P. Gawda, and D. Czerwinski, “On the detection of anomalies with the use of Choquet integral and their interpretability in motion capture data,” in *2022 IEEE International Conference on Fuzzy Systems*, IEEE, 2022, pp. 1–9.
 - [44] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui, “A comparative evaluation of outlier detection algorithms: Experiments and analyses,” *Pattern Recognition*, vol. 74, pp. 406–421, 2018.
 - [45] X. Duan, H. Yan, D. Tian, J. Zhou, J. Su, and W. Hao, “In-vehicle CAN bus tampering attacks detection for connected and autonomous vehicles using an improved isolation forest method,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 2, pp. 2122–2134, 2021.
 - [46] W. Dudzik, J. Nalepa, and M. Kawulok, “Evolving data-adaptive support vector machines for binary classification,” *Knowledge-Based Systems*, vol. 227, 2021, Art. no. 107221.

- [47] V. Dutta, M. Choraś, M. Pawlicki, and R. Kozik, “A deep learning ensemble for network anomaly and cyber-attack detection,” *Sensors*, vol. 20, no. 16, 2020, Art. no. 4583.
- [48] S. Dwivedi, M. Vardhan, S. Tripathi, and A. K. Shukla, “Implementation of adaptive scheme in evolutionary technique for anomaly-based intrusion detection,” *Evolutionary Intelligence*, vol. 13, no. 1, pp. 103–117, 2020.
- [49] M. Ebrahimi, C. Y. Suen, O. Ormandjieva, and A. Krzyzak, “Recognizing predatory chat documents using semi-supervised anomaly detection,” *Electronic Imaging*, vol. 28, pp. 1–9, 2016.
- [50] L. Eljabu, M. Etemad, and S. Matwin, “Anomaly detection in maritime domain based on spatio-temporal analysis of ais data using graph neural networks,” in *2021 5th International Conference on Vision, Image and Signal Processing*, IEEE, 2021, pp. 142–147.
- [51] T. Ergen and S. S. Kozat, “Unsupervised anomaly detection with LSTM neural networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 8, pp. 3127–3141, 2019.
- [52] Ł. Gałka and P. Karczmarek, “Minimal spanning tree-based isolation forest with anomaly score function built on the basis of fuzzy rules,” *Applied Soft Computing*, vol. 148, 2023, Art. no. 110935.
- [53] Ł. Gałka and P. Karczmarek, “Deterministic attribute selection for isolation forest,” *Pattern Recognition*, vol. 151, 2024, Art. no. 110395.
- [54] Ł. Gałka, P. Karczmarek, and M. Tokovarov, “Isolation forest based on minimal spanning tree,” *IEEE Access*, vol. 10, pp. 74 175–74 186, 2022.
- [55] Ł. Gałka, P. Karczmarek, and M. Tokovarov, “Effective enhancement of isolation forest method based on minimal spanning tree clustering,” *Information Sciences*, vol. 628, pp. 320–338, 2023.
- [56] S. Ghosh and S. K. Dubey, “Comparative analysis of k-means and fuzzy c-means algorithms,” *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 4, pp. 35–39, 2013.
- [57] R. L. Graham and P. Hell, “On the history of the minimum spanning tree problem,” English, *Annals of the History of Computing*, vol. 7, no. 1, pp. 43–57, 1985.
- [58] O. Grygorash, Y. Zhou, and Z. Jorgensen, “Minimum spanning tree based clustering algorithms,” in *2006 18th IEEE International Conference on Tools with Artificial Intelligence*, IEEE, 2006, pp. 73–81.

- [59] Y. Guo, X. Jiang, L. Tao, L. Meng, C. Dai, X. Long, F. Wan, Y. Zhang, J. Van Dijk, R. M. Aarts, et al., “Epileptic seizure detection by cascading isolation forest-based anomaly screening and EasyEnsemble,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 30, pp. 915–924, 2022.
- [60] S.-J. Han and S.-B. Cho, “Evolutionary neural networks for anomaly detection based on the behavior of a program,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 36, no. 3, pp. 559–570, 2006.
- [61] S. Hariri and M. C. Kind, “Batch and online anomaly detection for scientific applications in a Kubernetes environment,” in *Proceedings of the 9th Workshop on Scientific Cloud Computing*, ACM, 2018, pp. 1–7.
- [62] S. Hariri, M. C. Kind, and R. J. Brunner, “Extended isolation forest,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1479–1489, 2019.
- [63] J. A. Hartigan, *Clustering algorithms*. John Wiley & Sons, Inc., 1975.
- [64] A. J. Hashim, M. Balafar, J. Tanha, and A. Baradarani, “AEVAE: Adaptive evolutionary autoencoder for anomaly detection in time series,” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–12, 2023.
- [65] V. Hautamaki, I. Karkkainen, and P. Franti, “Outlier detection using k-nearest neighbour graph,” in *Proceedings of the 17th International Conference on Pattern Recognition*, vol. 3, IEEE, 2004, pp. 430–433.
- [66] Z. He, X. Xu, and S. Deng, “Discovering cluster-based local outliers,” *Pattern Recognition Letters*, vol. 24, no. 9-10, pp. 1641–1650, 2003.
- [67] S. Hido, Y. Tsuboi, H. Kashima, M. Sugiyama, and T. Kanamori, “Statistical outlier detection using direct density ratio estimation,” *Knowledge and Information Systems*, vol. 26, pp. 309–336, 2011.
- [68] H. Izakian and W. Pedrycz, “Anomaly detection and characterization in spatial time series data: A cluster-centric approach,” *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 6, pp. 1612–1624, 2014.
- [69] P. Jain, S. Jain, O. R. Zaïane, and A. Srivastava, “Anomaly detection in resource constrained environments with streaming data,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 3, pp. 649–659, 2021.
- [70] J.-S. Jang, “ANFIS: Adaptive-network-based fuzzy inference system,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.
- [71] Y.-S. Jeong, M. K. Jeong, and O. A. Omitaomu, “Weighted dynamic time warping for time series classification,” *Pattern Recognition*, vol. 44, no. 9, pp. 2231–2240, 2011.

- [72] P. Jiao, T. Li, Y. Xie, Y. Wang, W. Wang, D. He, and H. Wu, “Generative evolutionary anomaly detection in dynamic networks,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 12, pp. 12 234–12 248, 2023.
- [73] M. Jin, A. Lv, Y. Zhu, Z. Wen, Y. Zhong, Z. Zhao, J. Wu, H. Li, H. He, and F. Chen, “An anomaly detection algorithm for microservice architecture based on robust principal component analysis,” *IEEE Access*, vol. 8, pp. 226 397–226 408, 2020.
- [74] P. R. G. H. Junior, R. Scherer, L. B. Januario, D. Rodrigues, J. P. Papa, and K. A. Costa, “From network package flow to images: How to accurately detect anomalies in computer networks,” in *2022 29th International Conference on Systems, Signals and Image Processing*, IEEE, 2022, pp. 1–4.
- [75] P. Karczmarek, A. Kiersztyn, and W. Pedrycz, “N-ary isolation forest: An experimental comparative analysis,” in *International Conference on Artificial Intelligence and Soft Computing*, Springer, 2020, pp. 188–198.
- [76] P. Karczmarek, M. Dolecki, Ł. Gałka, W. Pedrycz, and D. Czerwinski, “Smooth quadrature-inspired generalized Choquet integral in an application to anomaly detection,” in *2023 IEEE International Conference on Fuzzy Systems*, IEEE, 2023, pp. 1–8.
- [77] P. Karczmarek, Ł. Gałka, M. Dolecki, W. Pedrycz, D. Czerwinski, A. Kiersztyn, and R. Stęgiński, “Enhanced tree-based anomaly detection,” in *2022 IEEE International Conference on Fuzzy Systems*, IEEE, 2022, pp. 1–7.
- [78] P. Karczmarek, Ł. Gałka, A. Kiersztyn, M. Dolecki, K. Kiersztyn, and W. Pedrycz, “Choquet integral-based aggregation for the analysis of anomalies occurrence in sustainable transportation systems,” *IEEE Transactions on Fuzzy Systems*, vol. 31, no. 2, pp. 536–546, 2022.
- [79] P. Karczmarek, A. Kiersztyn, and W. Pedrycz, “Fuzzy set-based isolation forest,” English, in *IEEE International Conference on Fuzzy Systems*, IEEE, 2020, pp. 1–6.
- [80] P. Karczmarek, A. Kiersztyn, W. Pedrycz, and E. Al, “K-means-based isolation forest,” English, *Knowledge-Based Systems*, vol. 195, 2020, Art. no. 105659.
- [81] P. Karczmarek, A. Kiersztyn, W. Pedrycz, M. Badurowicz, D. Czerwiński, and J. Montusiewicz, “K-medoids clustering and fuzzy sets for isolation forest,” in *2021 IEEE International Conference on Fuzzy Systems*, IEEE, 2021, pp. 1–8.
- [82] P. Karczmarek, A. Kiersztyn, W. Pedrycz, and D. Czerwiński, “Fuzzy c-means-based isolation forest,” English, *Applied Soft Computing*, vol. 106, 2021, Art. no. 107354.

-
- [83] F. Keller, E. Muller, and K. Bohm, “HiCS: High contrast subspaces for density-based outlier ranking,” in *2012 IEEE 28th International Conference on Data Engineering*, IEEE, 2012, pp. 1037–1048.
 - [84] D. J. Ketchen and C. L. Shook, “The application of cluster analysis in strategic management research: An analysis and critique,” *Strategic Management Journal*, vol. 17, no. 6, pp. 441–458, 1996.
 - [85] L. Khan, M. Awad, and B. Thuraisingham, “A new intrusion detection system using support vector machines and hierarchical clustering,” *The VLDB journal*, vol. 16, pp. 507–521, 2007.
 - [86] A. Kiersztyn, P. Karczmarek, K. Kiersztyn, and W. Pedrycz, “Detection and classification of anomalies in large datasets on the basis of information granules,” *IEEE Transactions on Fuzzy Systems*, vol. 30, no. 8, pp. 2850–2860, 2021.
 - [87] J. Kim and C. D. Scott, “Robust kernel density estimation,” *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 2529–2565, 2012.
 - [88] E. M. Knorr, R. T. Ng, and V. Tucakov, “Distance-based outliers: Algorithms and applications,” *The VLDB Journal*, vol. 8, no. 3, pp. 237–253, 2000.
 - [89] J. F. Kolen and T. Hutcheson, “Reducing the time complexity of the fuzzy c-means algorithm,” *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 263–267, 2002.
 - [90] M. Koziara and P. Karczmarek, “On a combination of clustering methods and isolation forest,” in *International Conference on Artificial Intelligence and Soft Computing*, Springer, 2022, pp. 114–126.
 - [91] G. Krishnamoorthy, P. Ashok, and D. Tesar, “Simultaneous sensor and process fault detection and isolation in multiple-input–multiple-output systems,” *IEEE Systems Journal*, vol. 9, no. 2, pp. 335–349, 2015.
 - [92] J. B. Kruskal, “On the shortest spanning subtree of a graph and the traveling salesman problem,” *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48–50, 1956.
 - [93] A. Lazarevic and V. Kumar, “Feature bagging for outlier detection,” in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, ACM, 2005, pp. 157–166.
 - [94] K. Lee, D.-W. Kim, D. Lee, and K. H. Lee, “Improving support vector data description using local density degree,” *Pattern Recognition*, vol. 38, no. 10, pp. 1768–1771, 2005.
 - [95] K. Lee, D.-W. Kim, K. H. Lee, and D. Lee, “Density-induced support vector data description,” *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 284–289, 2007.

- [96] J. Lesouple, C. Baudoin, M. Spigai, and J.-Y. Tournieret, “Generalized isolation forest for anomaly detection,” *Pattern Recognition Letters*, vol. 149, pp. 109–119, 2021.
- [97] F. Li, D. Zheng, T. Zhao, and W. Pedrycz, “A novel approach for anomaly detection in data streams: Fuzzy-statistical detection mode,” *Journal of Intelligent & Fuzzy Systems*, vol. 30, no. 5, pp. 2611–2622, 2016.
- [98] J. Li, H. Izakian, W. Pedrycz, and I. Jamal, “Clustering-based anomaly detection in multivariate time series data,” *Applied Soft Computing*, vol. 100, 2021, Art. no. 106919.
- [99] J. Li, W. Pedrycz, and I. Jamal, “Multivariate time series anomaly detection: A framework of hidden Markov models,” *Applied Soft Computing*, vol. 60, pp. 229–240, 2017.
- [100] N. Li, F. Chang, and C. Liu, “Spatial-temporal cascade autoencoder for video anomaly detection in crowded scenes,” *IEEE Transactions on Multimedia*, vol. 23, pp. 203–215, 2020.
- [101] Y. Li, X. Peng, Z. Wu, F. Yang, X. He, and Z. Li, “M3GAN: A masking strategy with a mutable filter for multidimensional anomaly detection,” *Knowledge-Based Systems*, vol. 271, 2023, Art. no. 110585.
- [102] L. Liao and B. Luo, “Entropy isolation forest based on dimension entropy for anomaly detection,” in *International Symposium on Intelligence Computation and Applications*, Springer, 2018, pp. 365–376.
- [103] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, “CANN: An intrusion detection system based on combining cluster centers and nearest neighbors,” *Knowledge-based systems*, vol. 78, pp. 13–21, 2015.
- [104] B. Lindemann, B. Maschler, N. Sahlab, and M. Weyrich, “A survey on anomaly detection for technical systems using LSTM networks,” *Computers in Industry*, vol. 131, 2021, Art. no. 103498.
- [105] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “On detecting clustered anomalies using sciforest,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2010, pp. 274–290.
- [106] F. Liu, K. M. Ting, and Z. Zhou, “Isolation forest,” in *Proceedings - IEEE International Conference on Data Mining*, IEEE, 2008, pp. 413–422.
- [107] F. Liu, K. M. Ting, and Z. Zhou, “Isolation-based anomaly detection,” *ACM Transactions on Knowledge Discovery from Data*, vol. 6, no. 1, pp. 1–39, 2012.
- [108] L. Livi, A. Sadeghian, and W. Pedrycz, “Entropic one-class classifiers,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 12, pp. 3187–3200, 2015.

-
- [109] P. R. Lorenzo, L. Tulczyjew, M. Marcinkiewicz, and J. Nalepa, “Hyperspectral band selection using attention-based convolutional neural networks,” *IEEE Access*, vol. 8, pp. 42 384–42 403, 2020.
 - [110] S. Luo, L. Luan, Y. Cui, X. Chai, Z. Wang, and Y. Kong, “An attribute associated isolation forest algorithm for detecting anomalous electro-data,” in *2019 Chinese Control Conference*, IEEE, 2019, pp. 3788–3792.
 - [111] Z. Lyu and Z. Pan, “HAD-IDC: A hybrid framework for data anomaly detection based on isolation, density, and clustering,” in *2022 2nd International Conference on Intelligent Technologies*, IEEE, 2022, pp. 1–6.
 - [112] H. Ma, B. Ghogh, M. N. Samad, D. Zheng, and M. Crowley, “Isolation Mondrian forest for batch and online anomaly detection,” in *2020 IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, 2020, pp. 3051–3058.
 - [113] X. Ma, J. Wu, J. Yang, and Q. Z. Sheng, “Towards graph-level anomaly detection via deep evolutionary mapping,” in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ACM, 2023, pp. 1631–1642.
 - [114] P. S. Maciąg, M. Kryszkiewicz, R. Bembenik, J. L. Lobo, and J. Del Ser, “Unsupervised anomaly detection in stream data with online evolving spiking neural networks,” *Neural Networks*, vol. 139, pp. 118–139, 2021.
 - [115] G. Madhukar Rao and D. Ramesh, “A hybrid and improved isolation forest algorithm for anomaly detection,” in *Proceedings of International Conference on Recent Trends in Machine Learning, IoT, Smart Cities and Applications*, Springer, 2021, pp. 589–598.
 - [116] E. H. Mamdani and S. Assilian, “An experiment in linguistic synthesis with a fuzzy logic controller,” *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, 1975.
 - [117] E. Marcelli, T. Barbariol, V. Savarino, A. Beghi, and G. A. Susto, “A revised isolation forest procedure for anomaly detection with high number of data points,” in *2022 IEEE 23rd Latin American Test Symposium*, IEEE, 2022, pp. 1–5.
 - [118] C. Melquiades and F. B. de Lima Neto, “Isolation forest-based semi-supervised anomaly detection of multiple classes,” in *2022 17th Iberian Conference on Information Systems and Technologies*, IEEE, 2022, pp. 1–6.
 - [119] A. Mensi and M. Bicego, “A novel anomaly score for isolation forests,” in *International Conference on Image Analysis and Processing*, Springer, 2019, pp. 152–163.
 - [120] B. Micenková, B. McWilliams, and I. Assent, “Learning outlier ensembles: The best of both worlds—supervised and unsupervised,” in *Proceedings of the ACM SIGKDD 2014 Workshop on Outlier Detection and Description under Data Diversity*, ACM, 2014, pp. 51–54.

- [121] C. Mohapatra and B. B. Ray, "A survey on large datasets minimum spanning trees," in *Artificial Intelligence: First International Symposium*, Springer, 2023, pp. 26–35.
- [122] N. Moustafa, K.-K. R. Choo, I. Radwan, and S. Camtepe, "Outlier dirichlet mixture mechanism: Adversarial statistical learning for anomaly detection in the fog," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 8, pp. 1975–1987, 2019.
- [123] Z. Mu, M. Wang, Y. Wang, R. Song, and X. Wang, "SI2FM: SID isolation double forest model for hyperspectral anomaly detection," *Remote Sensing*, vol. 15, no. 3, 2023, Art. no. 612.
- [124] T. H. A. Musa and A. Bouras, "Anomaly detection: A survey," in *Proceedings of Sixth International Congress on Information and Communication Technology*, Springer, 2022, pp. 391–401.
- [125] K. Negi, G. P. Kumar, G. Raj, S. Sahana, and V. Jain, "Degree of accuracy in credit card fraud detection using local outlier factor and isolation forest algorithm," in *2022 12th International Conference on Cloud Computing, Data Science & Engineering*, IEEE, 2022, pp. 240–245.
- [126] J. Nowak, M. Korytkowski, and R. Scherer, "Convolutional recurrent neural networks for computer network analysis," in *Artificial Neural Networks and Machine Learning–ICANN 2019: Text and Time Series: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings, Part IV* 28, Springer, 2019, pp. 747–757.
- [127] R. L. Page, "Algorithm 479: A minimal spanning tree clustering method," *Communications of the ACM*, vol. 17, no. 6, pp. 321–323, 1974.
- [128] N. R. Pal and J. C. Bezdek, "On cluster validity for the fuzzy c-means model," *IEEE Transactions on Fuzzy Systems*, vol. 3, no. 3, pp. 370–379, 1995.
- [129] W. Pedrycz, "An identification algorithm in fuzzy relational systems," *Fuzzy Sets and Systems*, vol. 13, no. 2, pp. 153–167, 1984.
- [130] W. Pedrycz, "Fuzzy multimodels," *IEEE Transactions on Fuzzy Systems*, vol. 4, no. 2, pp. 139–148, 1996.
- [131] S. Pettie and V. Ramachandran, "An optimal minimum spanning tree algorithm," *Journal of the ACM*, vol. 49, no. 1, pp. 16–34, 2002.
- [132] B. R. Preiss, *Data structures and algorithms with object-oriented design patterns in Java*. Wiley, 1999.
- [133] R. C. Prim, "Shortest connection networks and some generalizations," *The Bell System Technical Journal*, vol. 36, no. 6, pp. 1389–1401, 1957.

-
- [134] C. Y. Priyanto and H. D. Purnomo, "Combination of isolation forest and LSTM autoencoder for anomaly detection," in *2021 2nd International Conference on Innovative and Creative Information Technology*, IEEE, 2021, pp. 35–38.
- [135] L. Qi, Y. Yang, X. Zhou, W. Rafique, and J. Ma, "Fast anomaly identification based on multiaspect data streams for intelligent intrusion detection toward secure industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 9, pp. 6503–6511, 2022.
- [136] M. Radovanović, A. Nanopoulos, and M. Ivanović, "Reverse nearest neighbors in unsupervised distance-based outlier detection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1369–1382, 2014.
- [137] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, ACM, 2000, pp. 427–438.
- [138] A. R. Ramos, O. Llanes-Santiago, J. B. de Lázaro, C. C. Corona, A. S. Neto, and J. V. Galdeano, "A novel fault diagnosis scheme applying fuzzy clustering algorithms," *Applied Soft Computing*, vol. 58, pp. 605–619, 2017.
- [139] S. Rayana, *ODDS library*, Stony Brook University, Department of Computer Sciences, 2016. Accessed: Jun. 28, 2023. [Online]. Available: <http://odds.cs.stonybrook.edu>
- [140] S. Rayana and L. Akoglu, "Less is more: Building selective anomaly ensembles," *ACM Transactions on Knowledge Discovery from Data*, vol. 10, no. 4, pp. 1–33, 2016.
- [141] H. Ren, M. Liu, Z. Li, and W. Pedrycz, "A piecewise aggregate pattern representation approach for anomaly detection in time series," *Knowledge-Based Systems*, vol. 135, pp. 29–39, 2017.
- [142] H. Ren, Z. Ye, and Z. Li, "Anomaly detection based on a dynamic Markov model," *Information Sciences*, vol. 411, pp. 52–65, 2017.
- [143] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.
- [144] D. M. Roy and Y. W. Teh, "The Mondrian process," in *Advances in Neural Information Processing Systems 21 – Proceedings of the 2008 Conference*, NIPS, 2009, pp. 1377–1384.
- [145] D. Samariya and A. Thakkar, "A comprehensive survey of anomaly detection algorithms," *Annals of Data Science*, vol. 10, no. 3, pp. 829–850, 2023.
- [146] S. Sathe and C. Aggarwal, "LODES: Local density meets spectral outlier detection," in *Proceedings of the 2016 SIAM International Conference on Data Mining*, SIAM, 2016, pp. 171–179.

- [147] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, “Estimating the support of a high-dimensional distribution,” *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.
- [148] K. R. Shahapure and C. Nicholas, “Cluster quality analysis using silhouette score,” in *2020 IEEE 7th International Conference on Data Science and Advanced Analytics*, IEEE, 2020, pp. 747–748.
- [149] C. Shao, X. Du, J. Yu, and J. Chen, “Cluster-based improved isolation forest,” *Entropy*, vol. 24, no. 5, 2022, Art. no. 611.
- [150] Y. Shen, H. Liu, Y. Wang, Z. Chen, and G. Sun, “A novel isolation-based outlier detection method,” in *PRICAI 2016: Trends in Artificial Intelligence*, Springer, 2016, pp. 446–456.
- [151] L. Shouyu, K. Zhang, W. Fang, Z. Zhou, R. Hu, W. Zhu, Y. Li, Y. Wang, and J. Hou, “Anomaly detection of power grid dispatching platform based on isolation forest and k-means fusion algorithm,” in *Journal of Physics: Conference Series*, vol. 1601, IOP Publishing, 2020, 022010.
- [152] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, “A novel anomaly detection scheme based on principal component classifier,” in *Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop*, IEEE Press, 2003, pp. 172–179.
- [153] P. Song, C. Zhao, B. Huang, and J. Ding, “Explicit representation and customized fault isolation framework for learning temporal and spatial dependencies in industrial processes,” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2023.
- [154] W. Song, W. Dong, and L. Kang, “Group anomaly detection based on Bayesian framework with genetic algorithm,” *Information Sciences*, vol. 533, pp. 138–149, 2020.
- [155] X. Song, S. Aryal, K. M. Ting, Z. Liu, and B. He, “Spectral–spatial anomaly detection of hyperspectral data based on improved isolation forest,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–16, 2021.
- [156] E. Stripling, B. Baesens, B. Chizi, and S. vanden Broucke, “Isolation-based conditional anomaly detection on mixed-attribute data to uncover workers’ compensation fraud,” *Decision Support Systems*, vol. 111, pp. 13–26, 2018.
- [157] M. Sugeno and G. Kang, “Structure identification of fuzzy model,” *Fuzzy Sets and Systems*, vol. 28, no. 1, pp. 15–33, 1988.
- [158] T. Takagi and M. Sugeno, “Fuzzy identification of systems and its applications to modeling and control,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 1, pp. 116–132, 1985.

-
- [159] S. C. Tan, K. M. Ting, and T. F. Liu, "Fast anomaly detection for streaming data," in *Twenty-Second International Joint Conference on Artificial Intelligence, IJCAI*, 2011, pp. 1511–1516.
- [160] X. Tan, J. Yang, and S. Rahardja, "Sparse random projection isolation forest for outlier detection," *Pattern Recognition Letters*, vol. 163, pp. 65–73, 2022.
- [161] R. Thorndike, "Who belongs in the family?" *Psychometrika*, vol. 18, no. 4, pp. 267–276, 1953.
- [162] K. M. Ting, G.-T. Zhou, F. T. Liu, and J. S. C. Tan, "Mass estimation and its applications," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2010, pp. 989–998.
- [163] K. Ting, S. Tan, and F. Liu, "Mass: A new ranking measure for anomaly detection," *Gippsland School of Information Technology, Monash University*, 2009.
- [164] M. Tokovarov and P. Karczmarek, "A probabilistic generalization of isolation forest," *Information Sciences*, vol. 584, pp. 433–449, 2022.
- [165] E. Trauwaert, "On the meaning of Dunn's partition coefficient for fuzzy clusters," *Fuzzy Sets and Systems*, vol. 25, no. 2, pp. 217–242, 1988.
- [166] A. Tsopelakos and G. Fellouris, "Sequential anomaly detection under sampling constraints," *IEEE Transactions on Information Theory*, vol. 69, no. 12, pp. 8126–8146, 2023.
- [167] L. Utkin, A. Ageev, A. Konstantinov, and V. Muliukha, "Improved anomaly detection by using the attention-based isolation forest," *Algorithms*, vol. 16, no. 1, 2022, Art. no. 19.
- [168] N. L. Vora, K. Gilmore, A. Brandt, C. Gustafson, N. Strande, L. Ramkissoon, E. Hardisty, A. K. M. Foreman, K. Wilhelmsen, P. Owen, et al., "An approach to integrating exome sequencing for fetal structural anomalies into clinical practice," *Genetics in Medicine*, vol. 22, no. 5, pp. 954–961, 2020.
- [169] C. Wang, J. Liu, K. Wu, and Z. Wu, "Solving multitask optimization problems with adaptive knowledge transfer via anomaly detection," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 2, pp. 304–318, 2021.
- [170] D. Wang, H. Liu, and Y. Li, "Intelligent weight generation algorithm based on binary isolation tree," *Engineering Applications of Artificial Intelligence*, vol. 109, 2022, Art. no. 104604.
- [171] R. Wang, F. Nie, Z. Wang, F. He, and X. Li, "Multiple features and isolation forest-based fast anomaly detector for hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 9, pp. 6664–6676, 2020.

- [172] A. Wilbik and J. M. Keller, “A fuzzy measure similarity between sets of linguistic summaries,” *IEEE Transactions on Fuzzy Systems*, vol. 21, no. 1, pp. 183–189, 2012.
- [173] G. Williams, R. Baxter, H. He, S. Hawkins, and L. Gu, “A comparative study of RNN for outlier detection in data mining,” in *2002 IEEE International Conference on Data Mining*, IEEE, 2002, pp. 709–712.
- [174] K.-L. Wu, “Analysis of parameter selections for fuzzy c-means,” *Pattern Recognition*, vol. 45, no. 1, pp. 407–415, 2012.
- [175] K.-L. Wu and M.-S. Yang, “A cluster validity index for fuzzy clustering,” *Pattern Recognition Letters*, vol. 26, no. 9, pp. 1275–1291, 2005.
- [176] H. Xiang, H. Hu, and X. Zhang, “DeepiForest: A deep anomaly detection framework with hashing based isolation forest,” in *2022 IEEE International Conference on Data Mining*, IEEE, 2022, pp. 1251–1256.
- [177] H. Xiang, Z. Salcic, W. Dou, X. Xu, L. Qi, and X. Zhang, “OPHiForest: Order preserving hashing based isolation forest for robust and scalable anomaly detection,” in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, ACM, 2020, pp. 1655–1664.
- [178] H. Xiang, J. Wang, K. Ramamohanarao, Z. Salcic, W. Dou, and X. Zhang, “Isolation forest based anomaly detection framework on non-IID data,” *IEEE Intelligent Systems*, vol. 36, no. 3, pp. 31–40, 2021.
- [179] D. Xu, Y. Wang, Y. Meng, and Z. Zhang, “An improved data anomaly detection method based on isolation forest,” in *2017 10th International Symposium on Computational Intelligence and Design*, IEEE, 2017, pp. 287–291.
- [180] H. Xu, G. Pang, Y. Wang, and Y. Wang, “Deep isolation forest for anomaly detection,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 12, pp. 12 591–12 604, 2023.
- [181] K. Yamanishi, J.-I. Takeuchi, G. Williams, and P. Milne, “On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms,” in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2000, pp. 320–324.
- [182] T. Xu-Yan, S. Palaiahnakote, W. Chen, K. Cheng, and B. Du, “A novel autoencoder for structural anomalies detection in river tunnel operation,” *Expert Systems with Applications*, vol. 244, 2024, Art. no. 122906.
- [183] Q. Yang, J. Singh, and J. Lee, “Isolation-based feature selection for unsupervised outlier detection,” in *Annual Conference of the PHM Society*, vol. 11, PHMS, 2019, pp. 1–8.

-
- [184] X. Yang, Y. Zhuang, M. Shi, X. Cao, D. Chen, and Y. Tang, "SPiForest: An anomaly detecting algorithm using space partition constructed by probability density-based inverse sampling," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2022.
- [185] Y. Yang, X. Yang, M. Heidari, M. A. Khan, G. Srivastava, M. Khosravi, and L. Qi, "ASTREAM: Data-stream-driven scalable anomaly detection with accuracy guarantee in IIoT environment," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 5, pp. 3007–3016, 2023.
- [186] C. Yao, X. Ma, B. Chen, X. Zhao, and G. Bai, "Distribution forest: An anomaly detection method based on isolation forest," in *International Symposium on Advanced Parallel Processing Technologies*, Springer, 2019, pp. 135–147.
- [187] C. Yin, S. Zhang, J. Wang, and N. N. Xiong, "Anomaly detection based on convolutional recurrent autoencoder for iot time series," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 1, pp. 112–122, 2020.
- [188] J. Yu, Y. Lee, K. C. Yow, M. Jeon, and W. Pedrycz, "Abnormal event detection and localization via adversarial event prediction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 8, pp. 3572–3586, 2021.
- [189] S. Yuan, L. Shi, B. Yao, Y. Zhai, F. Li, and Y. Du, "Detection of the new class of hypersonic targets under emerging hyperspectral sample streams: An unsupervised isolation forest solution," *Remote Sensing*, vol. 14, no. 20, 2022, Art. no. 5191.
- [190] Y. Yuan, Y. Li, et al., "A modified hybrid method based on PSO, GA, and k-means for network anomaly detection," *Mathematical Problems in Engineering*, vol. 2022, 2022.
- [191] Z. Yuan, X. Zhang, and S. Feng, "Hybrid data-driven outlier detection based on neighborhood information entropy and its developmental measures," *Expert Systems with Applications*, vol. 112, pp. 243–257, 2018.
- [192] L. Zhang and L. Liu, "Data anomaly detection based on isolation forest algorithm," in *2022 International Conference on Computation, Big-Data and Engineering*, IEEE, 2022, pp. 87–89.
- [193] N. Zhang, Y. Liu, L. Xu, P. Lin, H. Zhao, and M. Chang, "Magnetic anomaly detection method based on feature fusion and isolation forest algorithm," *IEEE Access*, vol. 10, pp. 84 444–84 457, 2022.
- [194] X. Zhang, Y. Yao, C. Lv, and T. Wang, "Anomaly credit data detection based on enhanced isolation forest," *The International Journal of Advanced Manufacturing Technology*, vol. 122, no. 1, pp. 185–192, 2022.

- [195] X. Zhang and H. Gan, “An enhanced anomalies detection method based on isolation forest and fuzzy set,” in *2022 IEEE 8th International Conference on Cloud Computing and Intelligent Systems*, IEEE, 2022, pp. 432–436.
- [196] X. Zhang, J. Shi, X. Huang, F. Xiao, M. Yang, J. Huang, X. Yin, A. S. Usmani, and G. Chen, “Towards deep probabilistic graph neural network for natural gas leak detection and localization without labeled anomaly data,” *Expert Systems with Applications*, vol. 231, 2023, Art. no. 120542.
- [197] A. Zimek, M. Gaudet, R. J. Campello, and J. Sander, “Subsampling for efficient and effective unsupervised outlier detection ensembles,” in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2013, pp. 428–436.
- [198] Z. Zou, Y. Xie, K. Huang, G. Xu, D. Feng, and D. Long, “A docker container anomaly monitoring system based on optimized isolation forest,” *IEEE Transactions on Cloud Computing*, vol. 10, no. 1, pp. 134–145, 2022.

List of Figures

2.1	Example of a minimal spanning tree.	23
2.2	Example of clustering using a minimal spanning tree.	25
2.3	Visualization of the Elbow method for the k -Means algorithm.	35
2.4	Visualization of fuzzy partition coefficient analysis for the FCM algorithm. . . .	36
2.5	Diagram of the Takagi–Sugeno fuzzy inference model.	37
3.1	Example of the Isolation Forest algorithm operation.	43
4.1	General scheme of the dimensionality reduction algorithm.	49
4.2	Visualization of clustering analysis using the k -Means algorithm.	50
4.3	Visualization of clustering analysis using the FCM algorithm.	53
4.4	Example of split value determination using MST-based clustering.	58
4.5	Example of element isolation using the MSTCIF method.	59
4.6	Example of isolation tree formation using the MSTIF algorithm.	65
4.7	Example of element evaluation parameter determination for a single isolation tree using the MSTIF method.	67
4.8	Fuzzy set membership functions for linguistic variables.	72
4.9	Output value plot of the proposed TS model depending on the values of linguistic variables.	75
5.1	Heatmap presenting the difference in the average ROC AUC scores.	83
5.2	Heatmap presenting the difference in average PR AUC scores.	83
5.3	Heatmap presenting the difference in the median ROC AUC scores.	84
5.4	Heatmap presenting the difference in median PR AUC scores.	84
5.5	Heatmap presenting the differences in the number of better and worse ROC AUC scores.	85
5.6	Heatmap presenting the differences in the number of better and worse PR AUC scores.	85
5.7	Chart presenting the number of highest ROC AUC scores.	86
5.8	Chart presenting the number of highest PR AUC scores.	86

5.9	ROC curves for the Annthyroid dataset.	92
5.10	Heatmap presenting the difference in the average accuracy scores.	108
5.11	Heatmap presenting the difference in average precision scores.	108
5.12	Heatmap presenting the difference in the average recall scores.	109
5.13	Heatmap presenting the difference in the mean specificity scores.	109
5.14	Heatmap presenting the difference in the average false alarm ratio scores. . . .	110
5.15	Heatmap presenting the difference in average F1 scores.	110
5.16	Heatmap presenting the difference in median accuracy scores.	111
5.17	Heatmap presenting the difference in the median precision scores.	111
5.18	Heatmap presenting the difference in the median recall scores.	112
5.19	Heatmap presenting the difference in the median specificity scores.	112
5.20	Heatmap presenting the difference in median false alarm ratio scores.	113
5.21	Heatmap presenting the difference in the median F1 scores.	113
5.22	Heatmap presenting the difference in the number of better and worse accuracy scores.	114
5.23	Heatmap presenting the differences in the number of better and worse precision scores.	114
5.24	Heatmap presenting differences in the number of better and worse recall scores.	115
5.25	Heatmap presenting the difference in the number of better and worse specificity scores.	115
5.26	Heatmap presenting the difference in the number of better and worse false alarm ratio scores.	116
5.27	Heatmap presenting the differences in the number of better and worse F1 scores.	116
5.28	Chart presenting the number of the highest accuracy scores.	117
5.29	Chart presenting the number of highest precision scores.	117
5.30	Chart presenting the number of highest recall scores.	118
5.31	Chart presenting the number of highest specificity scores.	118
5.32	Chart presenting the number of the lowest false alarm ratio scores.	119
5.33	Chart presenting the number of highest F1 scores.	119
5.34	Plots of the properties of the proposed methods for different hyperparameter settings. Part I.	137
5.35	Plots of the properties of the proposed methods for different hyperparameter settings. Part II.	138
5.36	Plots of the properties of the proposed methods for different hyperparameter settings. Part III.	139
5.37	Normalized density plots of the score function values for the considered methods. Part I.	142

5.38	Normalized density plots of the score function values for the considered methods. Part II.	143
5.39	Normalized density plots of the score function values for the considered methods. Part III.	144
5.40	Normalized density plots of the score function values for the considered methods. Part IV.	145
5.41	Normalized density plots of the score function values for the considered methods. Part V.	146
5.42	Normalized density plots of the score function values for the considered methods. Part VI.	147
5.43	Normalized density plots of the score function values for the considered methods. Part VII.	148
6.1	Graphical presentation of the generated anomaly detection datasets.	154
6.2	Heatmaps of the score function values for the Circles dataset using a forest composed of a single isolation tree.	156
6.3	Heatmaps of the score function values for the Rectangle dataset using a forest composed of a single isolation tree.	157
6.4	Heatmaps of the score function values for the Target dataset using a forest composed of a single isolation tree.	158
6.5	Heatmaps of the score function values for the Spiral dataset using a forest composed of a single isolation tree.	159
6.6	Heatmaps of the score function values for the Circles dataset using a forest of one hundred isolation trees.	160
6.7	Heatmaps of the score function values for the Rectangle dataset using a forest of one hundred isolation trees.	161
6.8	Heatmaps of the score function values for the Target dataset using a forest of one hundred isolation trees.	162
6.9	Heatmaps of the score function values for the Spiral dataset using a forest of one hundred isolation trees.	163
6.10	Graphical representation of the anomaly detection process for the Circles dataset.	166
6.11	Graphical representation of the anomaly detection process for the Rectangle dataset.	167
6.12	Graphical representation of the anomaly detection process for the Target dataset.	168
6.13	Graphical representation of the anomaly detection process for the Spiral dataset.	169

List of Tables

5.1	Characteristics of the anomaly detection datasets.	77
5.2	ROC AUC values for the new methods.	79
5.3	ROC AUC values for the reference methods.	80
5.4	PR AUC values for the new methods.	81
5.5	PR AUC values for the reference methods.	82
5.6	Wilcoxon test p-values for the ROC AUC measure for compared algorithm pairs. Part I.	87
5.7	Wilcoxon test p-values for the ROC AUC measure for compared algorithm pairs. Part II.	87
5.8	Wilcoxon test p-values for the PR AUC measure for compared algorithm pairs. Part I.	88
5.9	Wilcoxon test p-values for the PR AUC measure for compared algorithm pairs. Part II.	88
5.10	Detection threshold values for the new methods.	93
5.11	Detection threshold values for the reference methods.	94
5.12	Accuracy values for the new methods.	96
5.13	Accuracy values for the reference methods.	97
5.14	Precision values for the new methods.	98
5.15	Precision values for the reference methods.	99
5.16	Recall values for the new methods.	100
5.17	Recall values for the reference methods.	101
5.18	Specificity values for the new methods.	102
5.19	Specificity values for the reference methods.	103
5.20	False alarm ratio values for the new methods.	104
5.21	False alarm ratio values for the reference methods.	105
5.22	F1 score values for the new methods.	106
5.23	F1 score values for the reference methods.	107
5.24	Wilcoxon test p-values for accuracy measure for compared algorithm pairs. Part I.	120
5.25	Wilcoxon test p-values for accuracy measure for compared algorithm pairs. Part II.	120

5.26	Wilcoxon test p-values for the precision measure for compared algorithm pairs.	
	Part I.	121
5.27	Wilcoxon test p-values for the precision measure for compared algorithm pairs.	
	Part II.	121
5.28	Wilcoxon test p-values for the recall measure for compared algorithm pairs. Part I.	122
5.29	Wilcoxon test p-values for the recall measure for compared algorithm pairs. Part II.	122
5.30	Wilcoxon test p-values for the specificity measure for compared algorithm pairs.	
	Part I.	123
5.31	Wilcoxon test p-values for the specificity measure for compared algorithm pairs.	
	Part II.	123
5.32	Wilcoxon test p-values for the false alarm ratio for compared algorithm pairs.	
	Part I.	124
5.33	Wilcoxon test p-values for the false alarm ratio for compared algorithm pairs.	
	Part II.	124
5.34	Wilcoxon test p-values for the F1 score for compared algorithm pairs. Part I. . .	125
5.35	Wilcoxon test p-values for the F1 score for compared algorithm pairs. Part II. .	125
5.36	Execution times of the training phase for new methods.	131
5.37	Execution times of the training phase for reference methods.	132
5.38	Execution times of the evaluation phase for new methods.	134
5.39	Execution times of the evaluation phase for reference methods.	135
5.40	Intersection area values of the density distributions of normal and anomalous elements for the new methods.	149
5.41	Intersection area values of the density distributions of normal and anomalous elements for the reference methods.	150
6.1	Characteristics of the generated anomaly detection datasets.	153

List of Algorithms

2.1	Standardization of attribute values.	21
2.2	Standardization of attribute values for a new record	21
2.3	Tree merging algorithm	23
2.4	Kruskal's algorithm for finding the minimal spanning tree	24
2.5	Root node finding algorithm	24
2.6	Selection of the split value for two clusters by optimized clustering using the minimal spanning tree	28
2.7	<i>k</i> -Means	29
2.8	FCM – calculation of the objective function value	30
2.9	FCM – general scheme	31
2.10	FCM – updating the fuzzy membership matrix	31
2.11	FCM – calculation of distances between elements and cluster centers	32
2.12	FCM – calculation of cluster centers	32
2.13	Elbow rule for determining the number of clusters in the <i>k</i> -Means algorithm . .	34
2.14	Fuzzy partition coefficient analysis for determining the number of clusters in the FCM algorithm	35
3.1	Isolation Forest – single tree formation phase	41
3.2	Isolation Forest – forest formation phase	41
3.3	Isolation Forest – calculation of the path length of an element in the isolation tree	42
4.1	Dimensionality reduction using <i>k</i> -Means algorithm – training phase	52
4.2	Dimensionality reduction using <i>k</i> -Means algorithm – data transformation phase	52
4.3	Dimensionality reduction using the FCM algorithm – training phase	54
4.4	Dimensionality reduction using the FCM algorithm – data transformation phase	55
4.5	Minimal Spanning Tree Clustering for Isolation Forest – single tree formation phase	57
4.6	Minimal Spanning Tree Clustering for Isolation Forest – evaluation phase . . .	61
4.7	Minimal Spanning Tree Clustering for Isolation Forest – calculation of the path length in the tree and distance from the split point	62
4.8	Minimal Spanning Tree-Based Isolation Forest – single tree formation phase . .	66

4.9	Minimal Spanning Tree-Based Isolation Forest – evaluation phase	68
4.10	Minimal Spanning Tree-Based Isolation Forest – calculation of the path length and distance to the nearest element in the isolation tree	68
4.11	Minimal Spanning Tree-Based Isolation Forest with Evaluation Function Built on the Fuzzy Rules in Takagi-Sugeno Model – calculation of the evaluation value for the fuzzy inference model	74
4.12	Minimal Spanning Tree-Based Isolation Forest with Evaluation Function Built on the Fuzzy Rules in Takagi-Sugeno Model – evaluation phase	74