University of Verona

Department of

Computer Science

Graduate School of

Natural Sciences and Engineering

Doctoral Program in

Computer Science

Cycle: XXXIV, 2018

Advanced Random Forest Approaches for Outlier Detection

S.S.D. ING-INF/05

Author:
Antonella Mensi

Supervisor:
Prof. Manuele Bicego

Advanced Random Forest Approaches for Outlier Detection
Antonella Mensi
Ph.D. Thesis
Verona, Italy, 08/06/2022

Abstract

Outlier Detection (OD) is a Pattern Recognition task which consists of finding those patterns in a set of data which are likely to have been generated by a different mechanism than the one underlying the rest of the data. The importance of OD is visible in everyday life. Indeed, fast, and accurate detection of outliers is crucial: for example, in the electrocardiogram of a patient, an abnormality in the heart rhythm can cause severe health problems. Due to the high number of fields in which OD is needed, several approaches have been designed. Among them, Random Forest-based techniques have raised great interest in the research community: a Random Forest (RF) is an ensemble of Decision Trees where each tree is diverse and independent. They are characterized by a high degree of flexibility, robustness, and high generalization capabilities. Even though originally designed for classification and regression, in the latest years, due to their success, there has been an increased development of RF-based approaches for other learning tasks, including OD. The forerunner of several RF methods for OD is Isolation Forest (iForest), a technique which main principle is isolation, i.e. the separation of each object from the rest of the data. Since outliers are different from the rest of the data and thus easier to separate, we can easily identify them as those objects isolated after few splits in the tree. iForests have been employed in a great variety of application fields, showing excellent performances [41, 47].

This thesis is inserted into the above scenario: even if some extensions of basic RF-based approaches for OD have been proposed, their potentialities have not been fully exploited and there is large room for improvements. In this thesis, we introduce some advanced RF-based techniques for OD, investigating both methodological issues and alternative uses of these flexible approaches. In detail, we moved along four research directions. The starting point of the first one is the absence of RF methods for OD able to work with non-vectorial data: here we propose ProxIForest, an approach which works with all types of data for which a distance measure can be defined, thus including non-vectorial data as well. Indeed, for the latter, many powerful distances have been proposed. The second direction focuses on how to measure the outlierness degree of an object in an RF, i.e. the anomaly score, since most extensions of iForest concern only the tree building procedure. In detail, we propose two novel classes of methods: the first class exploits the information contained within a tree. The second one focuses on the ensemble aspect of RFs: the aggregation of the anomaly scores extracted from each tree is crucial to correctly identify outliers. As to the third research direction we took a different perspective exploiting the fact that each tree in a forest is a space partitioner encoding relations, i.e. distances, between objects. Whereas this aspect has been widely researched in the clustering field, it has never been investigated for OD: we extract from an iForest a distance measure and input it to an outlier detector. As last research direction, we designed a new variant of iForest to characterize

multiple sclerosis given a brain connectivity network: we cast the problem as an OD task, by making an analogy between disconnected brain regions, the hallmark of the disease, and outliers. All proposals have been thoroughly empirically validated on either classical or ad hoc datasets: we performed several analyses, including comparisons to state-of-the-art approaches and statistical tests.

This thesis proves the suitability of RF-based approaches for OD from different perspectives: not only they can be successfully used for the task, but we can also use them to extract distances or features. Further, by contributing to this field, this thesis proves that there are still many aspects requiring further investigation.

Sommario

L'Outlier Detection (OD) è un task di Pattern Recognition che consiste nell'identificare in un insieme di oggetti quei pattern che è probabile che siano stati generati da un meccanismo diverso da quello sottostante il resto dei dati. L'importanza dell'OD è percepibile nella vita di tutti i giorni. Infatti, un veloce e accurato rilevamento degli outlier è cruciale: ad esempio, se consideriamo l'elettrocardiogramma di un paziente, un'anormalità nel ritmo cardiaco può causare gravi problemi di salute. Dato l'alto numero di campi in cui vi è bisogno dell'OD, sono stati inventati molti approcci. Tra di essi nella comunità hanno suscitato grande interesse le tecniche basate sulle Random Forest (RF): una Random Forest è un insieme, detto ensemble, di alberi di decisione dove ogni albero è diverso e indipendente dagli altri. Queste metodologie sono altamente flessibili, robuste e caratterizzate da ottime capacità di generalizzazione. Nonostante siano state introdotte per risolvere problemi di classificazione e regressione, negli ultimi anni, grazie al loro successo, si è rilevato un aumento di approcci basati sulle RF anche in altri campi, tra cui l'Outlier Detection. Il capostipite di molti approcci basati su RF per l'OD è l'Isolation Forest (iForest), una tecnica la cui caratteristica principale è l'isolamento, cioè la separazione di ogni oggetto dal resto dei dati. Siccome gli outlier sono differenti dagli altri oggetti e quindi più facili da separare, possiamo identificarli facilmente, infatti vengono isolati all'interno di un albero dopo pochi split. Le iForest sono state impiegate in una grande varietà di campi applicativi e in generale si sono dimostrate altamente performanti [41, 47].

Questa tesi si inserisce nello scenario appena descritto: nonostante siano state proposte delle estensioni di approcci basilari basati sulle RF per l'OD, le potenzialità di questi non sono state completamente sfruttate e vi è ampio margine di miglioramento. In questa tesi introduciamo delle tecniche avanzate per l'OD basate sulle RF, studiando sia problemi metodologici che usi alternativi di questi approcci così flessibili. In particolare, ci siamo mossi lungo quattro direzioni di ricerca. Il punto di partenza della prima è l'assenza di metodi basati su RF per l'OD in grado di lavorare con dati non vettoriali: in questa tesi proponiamo ProxIForest, un approccio che lavora con tutti quei dati per cui una misura di distanza può essere definita, includendo quindi anche i dati non vettoriali. Infatti, per questi ultimi, in letteratura sono presenti molte misure di distanza significative. La seconda direzione si focalizza invece su come misurare il grado di outlierness di un oggetto in una RF, i.e. anomaly score, in quanto la maggior parte delle estensioni delle iForest si concentra solo sulla procedura di costruzione dell'albero. In dettaglio, proponiamo due nuove classi di metodi: la prima sfrutta l'informazione contenuta all'interno di ogni albero. La seconda si focalizza sul concetto di ensemble proprio delle RF: l'aggregazione degli anomaly score estratti da ogni albero è cruciale per una corretta identificazione degli outlier. Per quanto riguarda la terza direzione di ricerca, abbiamo assunto una prospettiva differente sfruttando il fatto che

ogni albero in una foresta partiziona lo spazio codificando quindi le relazioni, i.e. distanze, tra gli oggetti. Mentre questo aspetto è stato largamente studiato nel campo del clustering, non vi sono ricerche a riguardo in quello dell'outlier detection: noi proponiamo di estrarre da un'iForest una misura di distanza, il cui risultato viene dato in ingresso a un detector di outlier. Come ultima direzione di ricerca abbiamo ideato una nuova variante delle iForest per caratterizzare la sclerosi multipla data una rete cerebrale di connettività: consideriamo il problema come un task di outlier detection instaurando un parallelismo tra le regioni cerebrali disconnesse, il tratto distintivo della malattia, e gli outlier. Tutte le proposte fatte in questa tesi sono state scrupolosamente validate dal punto di vista empirico sia usando dataset classici che ad hoc; abbiamo eseguito molte analisi, tra cui confronti con lo stato dell'arte e test statistici.

Questa tesi dimostra l'idoneità di usare approcci basati sulle RF nel contesto dell'OD sotto diverse prospettive: non solo possono essere usati con successo per risolvere il problema, ma possiamo anche utilizzarli per estrarre distanze o features. Inoltre, contribuendo a questo campo di ricerca, questa tesi dimostra che ci sono ancora tanti aspetti che richiedono ulteriore approfondimento.

Contents

Chapter 1

Introduction

Humans have cognitive and neural abilities that allow them to easily, quickly and almost unknowingly perform recognition tasks, such as recognizing someone they have already seen, correctly identifying a misshaped fruit or recognizing letters independently of the handwriting. For machines, it is much harder to perform such tasks: they have to learn how to recognize something. In other words, machines have to replicate the process behind the human reasoning. Since the second half of the twentieth century, with the exponentially increasing importance of machines and automation in everyday life, automated recognition tasks have become fundamental. An intuitive example is the identification of a defected object in a production line: to meet the increasing customer demand, faster recognition techniques than those performed by humans were needed.

All these recognition activities performed by a machine go under the name of (automated) *Pattern Recognition* (PR). One of the formal definitions of Pattern Recognition is given by Theodoridis et al. [147]: PR is the scientific discipline aimed at the "classification of objects into a number of categories". The objects to be categorized are commonly known as *patterns*; a pattern, according to Watanabe [156], can be formally defined as "the opposite of chaos; it is an entity, vaguely defined, that could be given a name". In other words, a pattern is anything that can be categorized, from a simple image representing an animal to a network encoding protein-protein interactions.

To solve a PR task, four steps must be carried out [74]. The first step usually consists of collecting the data, i.e. the set of patterns, via a sensor, which can be for example a camera, a software, or even a person collecting medical data; then the data must be processed in order to be readable by a machine. The second step consists of extracting a relevant representation for the data, which is usually problem-dependent; the representation can be: a set of features, which can be numerical, nominal, binary, etc. –this is known as vectorial representation; or it can be more structured, like a graph, a sequence, a string, an image, etc. The following step of a PR system consists of building a model using a part of the data. This step is commonly known as the training phase. The last step of a PR pipeline is the decision-making one, i.e. the testing phase: given the decision rule of the trained model and a pattern, different from those used during training, we assign a category, also known as *label*, to the pattern.

Depending on the availability of labels when building the model, i.e. the third step, PR methodologies can be divided into three categories: supervised,

semi-supervised and unsupervised techniques [67]. Supervised techniques employ the labels of objects when building the model whereas unsupervised ones do not, usually because labels are unavailable –the unavailability can be due either to labels being non-collectable in adequate times or unknowable. Semi-supervised techniques instead are hybrid: not all labels need to be known. The availability of labels is also important in evaluating the goodness of a model, i.e. how well it fits the data under analysis. In detail, if the true labels are known the performance of the model can be easily measured via a direct comparison between the labels predicted by the model and the true ones. Instead, when no labels are available, the goodness of a model is evaluated using more complex measures which are often dependent on the type of task that is being performed.

Another common way to divide PR techniques, which is highly linked to the previous categorization, is based on the final Pattern Recognition task. A first example is classification, in which a pattern is assigned to a previously defined category, or class, after using a supervised or semi-supervised methodology. Classification can be binary, i.e. there are only two categories between which to choose, or multiclass, which deals with the presence of more than two classes. Regression is akin to classification, but the output is continuous, in other words there is not a fixed number of categories. Another learning task of great relevance is clustering, which consists of partitioning objects into an unknown number of groups, i.e. clusters, in an unsupervised manner, such that each of the obtained groups contains similar objects while being different from the remaining clusters.

Even though the learning tasks we briefly described are all of great interest and importance, in this thesis we focus on another crucial task called outlier detection, which is the task of finding abnormal objects in a set of data. Outlier detection is fundamental in a wide variety of fields, and the detection is often required to be highly accurate and fast to prevent outliers from causing any further damage. Indeed, outliers, if undetected, can cause problems of a different nature.

To better understand the importance of such task, let us make a practical example. Outlier detection is often used to identify in a brain Magnetic Resonance (MR) image abnormal masses [46, 69]. In detail, we report an image directly from [69] in Figure 1.1: two different types of the same MR image are represented, where the abnormal mass, representing a chronic brain infarct, is surrounded by a red frame. These masses are rather heterogeneous in terms of morphology, location, cause and other factors, and therefore it is more than suitable to treat them as outliers. Since the cause behind these masses is usually related to the presence of a neurodegenerative disease, or at least to some cognitive defect, it is of the uttermost importance to detect them quickly and accurately. Outlier detection is highly relevant in many other different fields, a non-exhaustive list is: network traffic analysis to find whether there are intruders [54, 164]; finance to detect anomalous bank account movements [78]; urban traffic analysis to find in real time alternative routes when an accident,

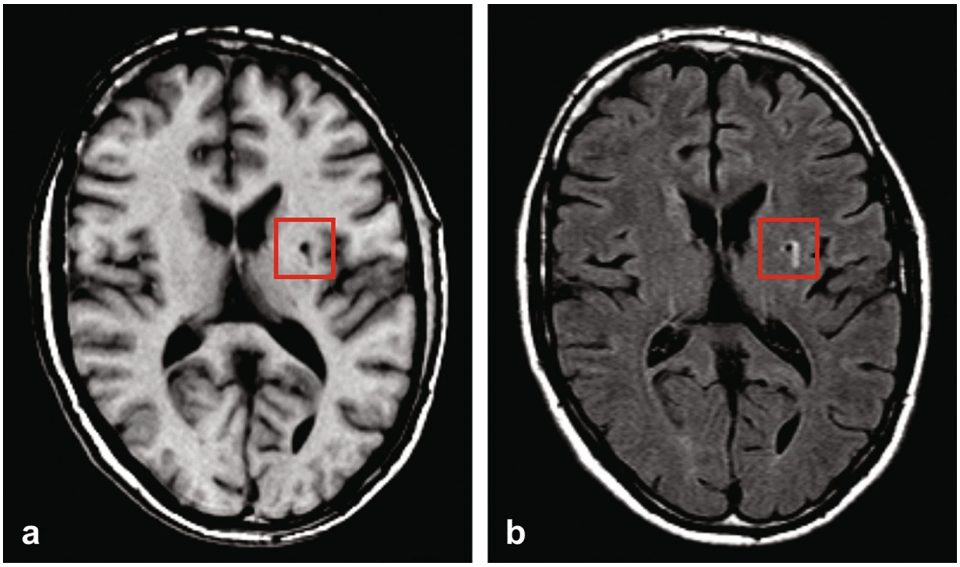


Figure 1.1: *"Example transversal image slice of the T1-weighted (a) and T2-FLAIR (b) acquisitions. A brain infarct can be observed in the right hemisphere next to the basal ganglia, in the red square, as the hypointense region in the T1-weighted image (a) and the hypointense region with hyperintense ring in the T2-FLAIR image (b)."* by van Hespen et al. [69] (`https://doi.org/10.1038/s41598-021-87013-4`) used and licensed under CC BY 4.0 (`https://creativecommons.org/licenses/by/4.0/`)

.

an outlier, occurs [39]; image processing to detect anomalous images or parts of them, e.g. defects in images representing fabric samples [11]; text data analysis to discover for example the presence of fake news within a social platform [122] and Internet of Things, e.g. to timely detect and replace faulty sensors [75].

Formally an outlier is "an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism" as defined in [68], where the other observations, the normal data, are also known as inliers. The difference between outliers, which are usually few patterns in a dataset, and the inliers can be of different nature and can be measured in several ways. For example, if the probability density function of the normal data is known, we can measure the probability of each object to belong to the described distribution: if it is low then the object is more likely to be an outlier. Another way to detect outliers is by computing some distance-based or density-based measures: many types of outliers tend to be either distant from inliers or in lower density areas of the space. To face outlier detection, several methodologies, based on different principles, have been developed during the years. These methodologies can be either supervised, semi-supervised or unsupervised, with the latter being the most common due to the difficulty in labelling outliers. There are many surveys covering different aspects of outlier detection, each giving a different taxonomy of the methodologies [28, 154, 168] –we will describe these aspects in detail in Chapter 2.

A particular type of methodologies for outlier detection, based on Random Forests (RF), has begun to raise particular interest over the last few years. Random Forests [19] are a successful ensemble methodology for classification and regression which base components are randomized Decision Trees (DT) [17]. A DT is a model which output prediction depends on the set of internal nodes that each object traverses: in detail, each internal node is defined by a *test* to which the object has to answer. Assuming each internal node is split into two children, the answer to the test will make the object either follow the left or right edge, i.e. it will determine the next test. Typically, to build such tree structure, a set of labelled examples is used: in each node we define a test according to an optimization procedure and as result objects are partitioned into two disjoint subsets. The building procedure stops when a stopping criterion, e.g. there are too few objects in a node, is met. Even though DTs are easily interpretable, they can be computationally expensive and lack generalization capabilities, i.e. they may be incapable of correctly categorizing new unknown patterns. These limitations led to the development of RFs for classification and regression. In an RF, several trees are trained independently and randomization is introduced at different levels of the tree building procedure; for example, each tree is built on a subset of the objects. Given a built forest, the final output is obtained by aggregating the predictions obtained by traversing each tree. RFs are much more robust, faster to train and with much higher generalization capabilities than Decision Trees. Further, randomization makes RFs highly scalable with respect to both the number of objects and the number of features in the dataset. All these favourable characteristics make Random Forests a highly suitable choice for classification and regression.

Indeed, the success of RFs for classification and regression has been pervasive, leading to an increasing interest in studying RFs also for solving other recognition tasks, such as outlier detection. In detail, as to RF-based techniques for outlier detection, there exist two categories: i) methodologies which first create an artificial class of outliers and then employ a standard RF for classification [37, 142]; ii) techniques which design an unsupervised RF able to deal with unbalanced datasets –even with datasets composed of objects coming only from one class, the inlier one. The innate limitation of the first type of approaches stands in the outlier sampling process: not only it may be computationally complex, but also inadequate to accurately represent outliers. Instead, this limitation is absent in the latter type of methods: no assumption is made or needed on the labels of the objects. The cornerstone of such techniques is Isolation Forest [96, 97]. Isolation Forest, iForest for short, solves outlier detection by isolating each object from the rest of the data, independently of the class the object belongs to. Since outliers are usually few and different from the inlier distribution they tend to be easier to separate, i.e. isolate, and therefore they should have a higher isolation capability.
iForests implement this principle in two steps. First, the tree building procedure is completely random, inspired by [53]: in a node data are partitioned

by choosing randomly both the feature along which to split the data and the cut-off value in the range of that feature. As to outliers, due to their nature, it is likely that on some features they will differ greatly from inliers: this leads to a higher probability of choosing a cut-off value that will separate them from the rest of the data in the first few splits of the tree. The isolation capability of an object is retrieved during the second step, the traversal stage: since an outlier is likely to be separated after few splits, its probability of being an outlier will be higher as the depth of the leaf it ends up into decreases, i.e. the traversed path is shorter. Therefore, Isolation Forests define an anomaly score, i.e. a score higher for outliers, inversely proportional to the depth of the reached leaf. The advantages of employing iForest are not only linked to it being an RF-based technique, but also to its peculiar nature. For example, some of the most important characteristics that make iForest an efficient and sound outlier detector are its completely unsupervised nature and its completely random tree building procedure. Further, other than being methodologically sound, iForests are also easy to use and to interpret. In detail, there exists a default parametrization that can be used in any context with good results, and the anomaly score of an object output by an iForest, is highly interpretable since, as previously mentioned, its value is proportional to the probability of the object being an outlier. These observations are also confirmed in practice, since iForest has shown to be one of the state-of-the-art outlier detectors [41, 47] and it has been used in many different applications, such as [3, 87, 90, 95, 102, 145, 155, 160].

Various efforts have been made to further improve iForest: several extensions have been proposed, such as [24, 57, 63, 65, 80, 81, 83, 98, 144, 165]. Most of these extensions are aimed at improving the building procedure, i.e. the implementation of the isolation principle, whereas a minority of them try to exploit in a better way the information embedded in the tree structure to compute the isolation capability of the objects, i.e. the scoring function. In addition to these extensions, there are still several aspects that have not yet been adequately researched. An example is that all RF-based techniques for outlier detection work with vectors and cannot manage non-vectorial data: it can be a big limitation considering the high amount of non-vectorial problems in which outlier detection is needed, such as [6, 46, 69, 76, 100, 140].

The success of isolation-based techniques, both iForest and its extensions, the related open challenges that are yet to be faced, along with the importance of outlier detection, led us to choose isolation-based outlier detection as the main focus of this thesis.

1.1 Contributions

This thesis is inserted into the described scenario, aimed at improving isolation-based outlier detection, by focusing on unresearched or partially researched

aspects, and at widening its scope of application. In detail, this thesis moves along four research directions, both methodological and applicative ones.

The starting point of the first research direction is the observation that all RF methods for outlier detection work with vectorial data and are incapable of managing non-vectorial data, which are however pervasively present in outlier detection. A common solution to deal with the latter type of data is to extract high-level features. However, this procedure is not suitable since it often leads to information loss concerning the structure of the data and the relations between the objects. An alternative and much suitable procedure, consists of computing pairwise distances between non-vectorial objects, given that a meaningful distance measure has been used, since it does not lead to any information loss. Given the rationale above, we propose an RF-based outlier detector called Proximity Isolation Forest (ProxIForest) which works with all types of data for which a distance measure can be defined, thus including non-vectorial data, for which many powerful distances have been proposed in the literature. In detail, ProxIForest exploits the pairwise relations between the objects, which is a natural way to implement the principle of isolation: in many scenarios, outliers are expected to be *distant* from the rest of the data.

The second and third research directions both concern the testing stage of Isolation Forest, but in very different fashions.
The second direction focuses on the computation of novel anomaly scores in alternative to the original one proposed in [96, 97]. Indeed, most extensions of iForest mainly concern the tree building procedure and do not focus on the testing phase. However, a forest contains a lot of information that can be used to better characterize the outliers. Concerning this aspect, we propose two different contributions, which aim is to provide alternative and more refined scoring functions for isolation-based methods. The first contribution measures the outlierness of an object by weighting the path, i.e. the nodes, it traverses in each tree, i.e. in addition to the depth of the reached leaf, it uses additional information. The rationale behind it is that each tree in a forest contains different types of information, and depending on the adopted information criterion, different nodes can have a different importance. Our second contribution, instead, is based on two rather different concepts. The first, is that we can interpret the anomaly score from a probabilistic perspective. The second concept is linked to the ensemble nature of RFs: indeed, one of the core concepts in ensemble theory is how to get the final scoring function, i.e. how to combine at forest level the tree scores. Exploiting these two concepts, our second contribution consists of a novel function to aggregate the anomaly scores at forest level, less restrictive than the original one.
As to the third research direction, we took a different perspective: a tree in a forest is primarily a space partitioner encoding relations, i.e. distances, between objects. In other words, we can extract informative distances from RFs: actually, these distances have already been proved to be quite successful in clustering [15, 142, 149, 167]. Even though there exists a great variety of

distance-based outlier detectors, due to the nature of outliers which are often quite distant from the rest of the data, this interesting aspect has never been investigated in the outlier detection field. Therefore, our proposal aims at filling this gap: in detail, we extract an informative distance measure from an Isolation Forest and then use the obtained distance matrix as input to an outlier detector. Indeed, by exploiting the unsupervised nature of Isolation Forest and its isolation principle, the extracted pairwise distances should be capable of discerning outliers from inliers.

All our methodological contributions are supported by a thorough experimental evaluation. In detail, we tested each proposal on a large pool of datasets; we analyzed several parametrizations, and we compared our proposals not only to iForest, but, if feasible, to other isolation-based techniques or other state-of-the-art outlier detectors.

The last research direction we investigated in this thesis makes a truly innovative application: we specifically propose a novel RF-based approach for characterizing multiple sclerosis (MS) given a brain connectivity network. A brain connectivity network encodes the connectivity strength between each pair of a given set of brain regions. Its importance is related to the fact that one of the hallmarks of MS are the disconnections between brain regions. Indeed, the rationale behind our approach is to derive an informative representation that can help in identifying these disconnected regions. To this aim, in our approach, we first cast the problem as an outlier detection task, by making an analogy between disconnected brain regions and outliers. Then we propose a novel model, called MS-ProxIF: it extends Proximity Isolation Forests to adequately deal with the context. We chose ProxIForest as starting point since we can make an analogy between a connectivity network, which is non-vectorial in nature, and similarity: indeed, the higher the connectivity strength between two brain regions, the more *similar* they are. After building an MS-ProxIF model, we use it as a feature extractor to characterize the different brain regions based on their disconnection degree.
The methodology is thoroughly tested via classification of a set of patients, provided by the Mount Sinai Ichan School of Medicine of New York (US), in collaboration with another group within our department and with the University of Genova and La Sapienza University of Rome.

1.2 Organization of the Thesis

The thesis can be divided into seven chapters, including the current one.
Chapter 2 concerns all background-related concepts, which are divided into three main sections. The first section covers the most relevant aspects of outlier detection; the second one describes Decision Trees and Random Forests, and the latter section presents the Isolation Forest methodology in detail. Each of the following four chapters, Chapter 3, 4, 5 and 6, is dedicated to one of the research directions we investigated in this thesis. In detail, Chapter 3

thoroughly presents Proximity Isolation Forest, i.e. the RF approach able to manage non-vectorial data. Chapter 4 presents enhanced anomaly scores, whereas in Chapter 5 we extract RF-distances from an iForest and use them as input to a distance-based outlier detector. Chapter 6 presents MS-ProxIF, a model defined to characterize Multiple Sclerosis connectivity networks. In the last chapter, Chapter 7, we make thorough conclusions on the proposed work and discuss some interesting future ideas linked to this research field.

1.3 Publications

Several parts of this thesis have either been submitted to or published in conference proceedings or international journals. Indeed, part of Chapter 3 has been published in the proceedings of the International Conference of Pattern Recognition (ICPR) [109] whereas the complete proposal has been submitted to Pattern Recognition (PR). Some ideas of Chapter 4 have been published in the proceedings of the International Conference on Image Analysis and Processing (ICIAP) [107] while the comprehensive methodology has been published in PR [108]. As to Chapter 5, it has led to two publications: the basic methodology has been published in the proceedings of the IAPR Joint International Workshops on Statistical techniques in Pattern Recognition and Structural and Syntactic Pattern Recognition (S+SSPR) [111], whereas a more complete version has been published in ICIAP 2021 [110]. Finally, a part of Chapter 6 has been accepted to the Computational Intelligence Methods for Bioinformatics and Biostatistics (CIBB) conference, and following the invitation of the conference organizers, we submitted a slightly extended version to Lecture Notes in Bioinformatics (LNBI). A much more comprehensive work, which includes ongoing experiments on a novel cohort of subjects, is going to be submitted in the near future to a journal.

Chapter 2

Background

This chapter describes the background of this thesis. In particular, the first section is dedicated to a thorough presentation of the outlier detection problem. In detail, first we define what outlier detection is and give a general overview; then we illustrate the main aspects impacting an outlier detection model. Subsequently, we briefly present the main types of outlier detection techniques, including ensemble-based methodologies, to which Random Forest-based techniques belong. Finally, we briefly discuss how to evaluate the output of an outlier detector and which is the most suitable option. In the second section, we briefly present Decision Trees and Random Forests, in order to fully understand Isolation Forests, which we thoroughly describe in Section 2.3. In detail, in this last section, other than describing how the original method implements isolation both in the training and testing phase, we also describe some applications and several extensions. Lastly, we point out the advantages and disadvantages of the methodology and highlight the open research directions, representing the starting point of this thesis.

2.1 Outlier Detection

Outlier detection consists of detecting apparently abnormal objects among a set of patterns. The underlying concept is vague and complex: several formal definitions are present in literature [154]; also, several names have been given to outliers, such as intrusions or anomalies. Nevertheless, there seems to be a definition by Hawkins [68] able to capture the whole nature of outlier detection: "an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism". This definition includes two concepts: the dissimilarity from other objects and the mechanism that caused outliers. This division, as stated in [168], is emphasized by [10] which denotes the former as *discriminant observations* and the second ones as *contaminants*. On the contrary, the definition in [68] is able to include them both in one comprehensive definition. An example of a dataset perturbed by outliers is depicted in Figure 2.1: there is a set of objects normally distributed depicted as blue circles. In addition, there are two patterns, depicted as red squares, which are very far from the rest of the data and do not seem to follow the same distribution: these two objects are outliers.

Another defining characteristic, which can also be observed in Figure 2.1, is that outliers are usually very few with respect to normal data, making it unfeasible to consider them as a class like inliers are. Further, outliers are

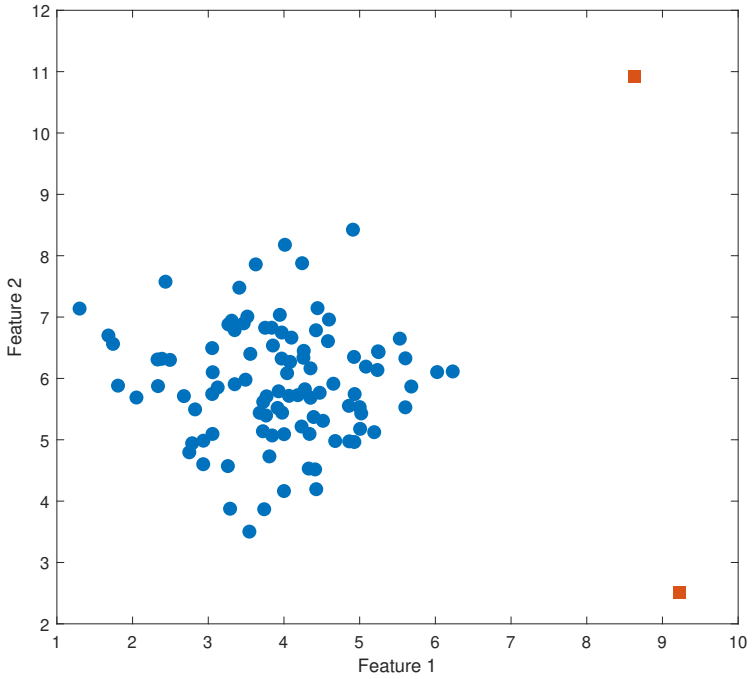


Figure 2.1: Example of a dataset which objects, depicted as circles, follow a bivariate normal distribution. In the plot, there are also two outliers, depicted as red squares.

often different from one another, making the modelling of these objects as a unique class even harder and highly unsuitable. Therefore, identifying outliers is not easy, and the complexity of this task is further increased if we consider the nature of these outliers. Indeed, there exist several mechanisms which can generate outliers, and these mechanisms may require a different approach to correctly identify them. Therefore, it would be of crucial importance if possible, before deciding which approach to adopt, to uncover the causes behind the presence of outliers. Examples of such causes are: human errors, instrument errors, general malfunctions, malevolent actions and environmental changes among many others [154].

The importance of outlier detection is highly linked to two aspects, the first being the pervasiveness of this task. To exemplify this, we describe in detail some main fields of application of outlier detection:

- *Network intrusion detection*: a network is a collection of devices that share information. A classical example is the internet, which use has increased exponentially in recent years [164]. In this scenario, a pattern is a connection, i.e. a set of packets exchanged by a client and a server [54]. Often, there are unexpected exchanges of information that try to mimic normal connections: these are called intrusions or attacks. These are outliers since: i) they are few with respect to the massive amount of information flowing in a network; ii) they cannot be predicted since, with respect to the past, they constantly change in order not to be detected.

The difficulty of detecting these attacks is reflected in their importance: they must be detected promptly and accurately due to their malevolent nature. In detail, these attacks aim at causing damage to the system, e.g. steal confidential data from one of the devices in the network. An example is [54]: they use a modular ensemble technique to detect attacks in different protocols and services of the network and to overcome the high false alarm rate of intrusion detection systems.

- *Healthcare*: in healthcare, there are several fields in which outlier detection is needed [44, 46, 69, 159]. Data may come in very different forms depending on the application, and so do outliers, therefore we focus on a few illustrative examples. As illustrated in Chapter 1 detecting outliers is necessary when analyzing brain MRIs. In this context, patterns are patches of an image and outliers are abnormal patches corresponding to a malignancy, e.g. tumour [46] or brain infarct [69]. Nevertheless, due to the heterogeneity of these masses, i.e. appearance and location, they are often difficult to detect.
  Another example is outlier detection for cancer-related gene expression [44, 159]. In this scenario, the pattern is the expression of a gene, derived for example from microarray data [159]. Outliers are those genes which are either up-regulated or down-regulated in a small sample of diseased subjects with respect to either a healthy cohort or to the rest of the diseased population. In detail, in [159] they identify some melanoma antigen genes as outliers: these genes are up-regulated in a small percentage of subjects which suffer of a more aggressive form of gastric cancer. In both examples, the malignant nature of the outliers is the reason why it is so important to be accurate in the detection; further, by identifying them we can help clinicians decide the most appropriate follow-up for the patient.

- *Image processing*: outlier detection is important both in the analysis of static images and in that of dynamic images, i.e. sequence of images often corresponding to a video. A pattern can be either an entire image or a patch of it: the definition of what is an outlier instead depends on the specific application. For example, concerning static images, outlier detection is used in the fashion industry [11]: each image represents a piece of fabric and outliers are defects in the texture, which are locally searched. Obviously, detecting outliers is not trivial since defects can be very small and of very different nature; nevertheless it is crucial: it can replace manual inspection, thus allowing to save money and fasten the production line. An example of a piece of defected fabric is pictured in Figure 2.2: there is a small hole, which we surrounded by a red frame[1].

[1]The original image can be found at `https://www.kaggle.com/rmshashi/fabric-defect-dataset`.

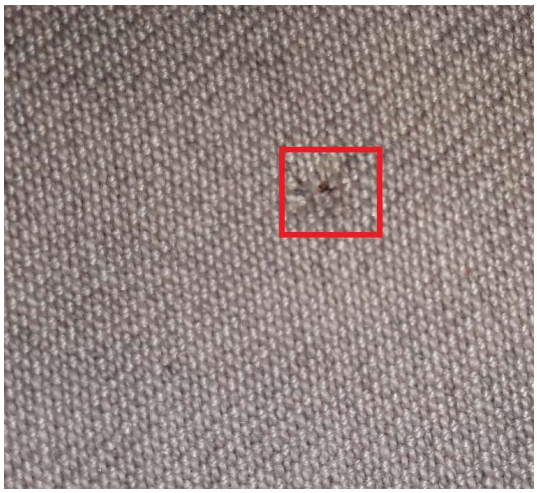


Figure 2.2: Example of a patch of defected fabric.

Outlier detection, in the context of dynamic images, is widely used, for example for video surveillance. An example is [4] in which they aim at detecting changes in visibility and in the camera view. These events are rare and very different with respect to what the camera usually records, in other words they are outliers. Their fast identification is crucial since often behind these events there are burglars tampering the camera in order not to be seen: by identifying the tampering we can prevent the theft to happen.

- *Text data*: this field has gained importance in the latest years due to exponential increase of internet users and in particular in the usage of social networks [79, 122]. There are different reasons why outlier detection may be useful in analyzing text data, and therefore the definition of outlier depends on the application. A first example concerns detecting fake news within a newspaper website, blog or social media [122]. Fake news can be correctly detected as outliers if we analyze the text from a linguistic perspective, i.e. if we extract features such as N-grams and punctuation. Nevertheless, the detection is far from easy: if we look at the content, fake news tend to contain true information as well. Identifying them is crucial to prevent false information to cause pointless alarmism, and, more in general, to spread. Another example of text-based outlier detection concerns blogs: when an internet user inputs a query on a search engine, not all top-ranked results are legitimate or of quality. Indeed, spam blogs [166] are those blogs which content is not up to the expectation of their rank, due to their deliberate misuse of promoting tools. Spam blogs can be considered as outliers with respect to the other top-ranked quality blogs returned by a user search. The identification of spam blogs is not trivial, and it often relies on analyses that span over a prolonged period of time. At the same time, it is crucial

to give more credits to legitimate blogs and saving network resources in order to improve the user experience.

- *Internet of Things (IoT)*: IoT are small devices composed of sensors that capture specific information about the physical world [75]. This information is usually transmitted to other devices thanks to the IoT ability to connect to a wireless network. An example of an IoT device is the fitness tracker, which often comes in the shape of a wristwatch. This tracker has usually some motion sensor to collect, for example, the number of steps made in a day and the most recent models also have an optical sensor to detect the heart-rate. The number of IoT devices is increasing exponentially: they differ in purpose and in the type of collected data, thus making outlier detection application-dependent in most cases. Nevertheless, in [75] an attempt to find faulty sensors is done independently of the domain. The core concept is that some IoT devices, such as those for the collection of environmental data, have several identical sensors that are close to each other. Therefore, these sensors are expected to detect similar information in each time unit. In this case the data to analyze are those collected by all sensors and outliers are those data which differ greatly in one sensor when compared to the other sensors. Obviously, detection is not easy because the variations may be small and go undetected; however, it is important to identify them promptly to adequately replace the sensor and prevent any further damage.

- *Fraud detection*: in banks, insurance agencies and other commercial organizations, a huge amount of delicate operations, in terms of data and money, takes place every day. These data can be polluted by criminal activities: unauthorized operations which aim is to steal information or money, from the attacked organization. This type of operation, analogously to what is done by network intruders, usually tries to mimic the real one, thus making more difficult the detection. An example is linked to the e-commerce explosion of the latest years, which makes it easier for thieves to steal credit cards' information [129]. Unauthorized transactions are the outliers to detect; however, an expert thief tends to mimic normal transactions or to hide their fraudulent behaviour by, for example, making small operations in between of those of the credit card's owner. It is clear that in this field, outlier detectors must be very accurate to limit these criminal activities as soon and as much as possible.

The other and main factor that makes outlier detection so important is the *use of outliers* once they have been identified. Indeed, depending on the application, we may do one or more of the following:

- Infer relevant information from the outlier(s). An example is the analysis of demographic data concerning different towns, where one of them is

characterized by much better health-related parameters. In other words, with respect to these features, this town is an outlier: by studying it we would maybe be able to improve the health situation also in other towns.

- Prevent critical situations, which extreme case may result in saving lives. An example is the analysis of electrocardiogram (ECG) sequences: if a heart attack is promptly detected, the patient's life is saved.

- Clean data and improve their quality via removing outliers [22]. In these cases, outliers need to be removed to have an unbiased and accurate set of data.

The importance of outlier detection is reflected in the number of years this topic has been researched, and subsequently in the large number of existing techniques. Indeed, the problem of outlier detection has been studied for two centuries, with its initial roots in the statistical field: outliers are objects which probability to belong to the distribution under analysis is really low. Last century, the communities of Pattern Recognition, Machine Learning and Data Mining recognized the importance of outlier detection and started to thoroughly investigate this field. In detail, a lot of techniques have been developed: some are very simple, such as those based on the estimation of the Nearest Neighbor, whereas others are much more complex, e.g. autoencoders. Several surveys are present, most of them focusing only on some relevant aspects, leaving out other important notions –which is understandable due to the pervasiveness of the field. However, there exist two surveys which try to focus on the general view and by considering both of them, we can describe all the important aspects of the problem [28, 154]. The first, by Chandola et al. [28], is one of the most extensive surveys which does not only describe outlier techniques but also the fundamental features characterizing an outlier detection task. The other survey, by Wang et al. [154], is more recent and focuses on different aspects, i.e. the nature of outliers, the evaluation protocol and a different categorization of the methodologies with respect to [28].

In this section, we have briefly introduced outlier detection from a general perspective, highlighting its crucial importance. In the following part of the section, we describe in detail the main characteristics to consider when facing an outlier detection problem.

2.1.1 Main Aspects

There are several aspects of an outlier detection problem to consider. In the following, we describe in detail some of the most important ones.

- **Nature of input data**: not all data are equals, on the contrary they can be very different. A dataset can be composed of objects which are either vectors of features or that have a non-vectorial representation, such as images. The former can be univariate or multivariate, i.e. have

one or more features characterizing them; features can be of the same or different types, such as binary, categorical, continuous, etc. As to non-vectorial objects, they are usually made of several linked components, i.e. in an image different patches are in a spatial relationship one with the other. Other examples of non-vectorial data, which we can also denote as *structured* objects, are sequences, strings and graphs. In general, it is more complex to deal with structured data. Different types of data require different approaches to identify outliers; for example, when detecting outliers in images, the spatial relationship is fundamental.

- **Availability of labels**: in some contexts it may be difficult to collect labels, or at least a portion of them. This is especially true for outliers: either they are unknown and therefore unclassifiable or they mimic the behaviour of inliers, and we risk assigning them a wrong label. Other than being impossible to obtain them, collecting labels could also take too long in terms of time. Therefore, depending on this aspect of the availability of labels, the adopted methodology can be: supervised, semi-supervised or unsupervised. Supervised methodologies can use all data and their labels to build the model; semi-supervised techniques can manage problems where only some objects are labelled –these usually being inliers. Unsupervised techniques do not need any label to build the model. The latter category is the one which makes the most sense for outlier detection, and it is also the most used, since outliers are unpredictable and a new mechanism generating them can emerge any time. In other words, a model trained with the available outliers may not be able to identify the new ones correctly.

- **Type of outliers**: Chandola [28] distinguishes outliers in three categories, which we describe in the following.

  - A *point outlier* is an object that is an outlier independently of whether or not there are other outliers close by. In other words, a point outlier can be correctly identified even if all other outliers were to be removed. A clarifying example is the one depicted in Figure 2.3: we measure the levels of blood sugar after fasting in a set of apparently healthy subjects. By analyzing them we discover that there is one subject, depicted as a red cross, having much higher levels than the rest of the population, i.e. they suffer of diabetes. This subject is therefore an outlier, regardless of whether there are other subjects with the same issue. Most outliers belong to this category.
  - *Collective outliers* are objects that if individually considered are inliers, but when taken as a set they are outliers. This type of outlier is typical in sequential data, where often a single value does not have a particular meaning, while a sequence may do. An example

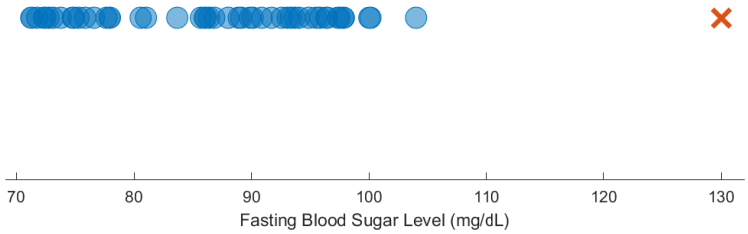


Figure 2.3: Measurements of fasting blood sugar levels on 50 subjects.

is the analysis of an ECG of a patient: a heart defect is characterized by a particular sequence of heartbeats and not by a single measurement. Indeed, in Figure 2.4 we have two different ECGs[2]: the one above describes a normal sinus rhythm whereas it is evident that the ECG depicted in the plot below presents some problems. Indeed, the signal is almost constant across several subsequent time instants, differently from the other ECG sequence. If we focused only on the single values, we would not detect the abnormal sequence highlighted in red in the bottom plot.

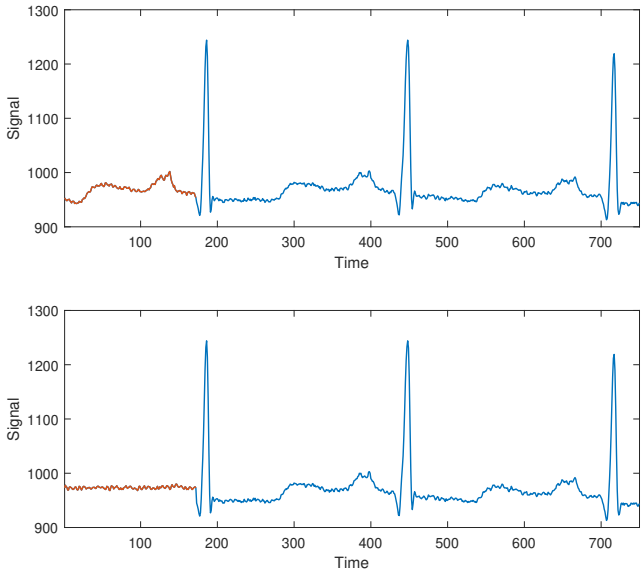


Figure 2.4: Two examples of ECG signals: above the signal is measured on a healthy subject, below a heart defect is detected at the beginning of the sequence.

– *Contextual outliers* this last category is not mutually exclusive from the other two: both point and collective outliers can be contextual. Contextual outliers are those objects which are outliers in a particular context, but they are not in the general world. An example is depicted in Figure 2.5: in the plot above, we depict the blood

[2]Dataset available at `https://data.mendeley.com/datasets/7dybx7wyfn/3` [125].

sugar levels of a healthy cohort. There is a subject, depicted by a red cross, with a very high level of blood sugar: it is an outlier since the population is supposed to be healthy. In the bottom plot of Figure 2.5 the same measurement, depicted again by a red cross, is considered an inlier since the underlying population is composed by diabetic subjects.

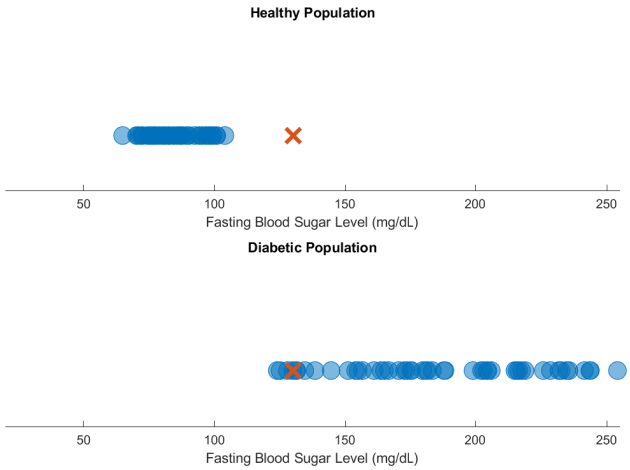


Figure 2.5: Measurements of fasting blood sugar levels on 50 subjects. In the plot above, the sampled population is healthy, in the bottom one the population is diabetic.

Other aspects that may create difficulties when solving an outlier detection problem include: the computational complexity of the chosen methodology, the size of the dataset and the number of features. Indeed, as the number of objects increases also the number of outliers does: this results also in an increased number of *different* outliers and therefore makes the detection more complex. A common problem in all PR tasks is that having a too high number of features, relatively to the size of the dataset, can be burdening, leading to the curse of dimensionality. Indeed, as the number of features increases at a higher trend than the number of objects does, data become much more sparse and the resulting model is overfitted, i.e. it has poor generalization capabilities.

2.1.2 Taxonomy of Outlier Detection Methodologies

There is an abounding number of outlier detection techniques which can be organized in several ways. Indeed, even though we adopt a specific categorization, as noted by [168], often these methods stay on a continuum where all methodologies are connected by the more or less loose probabilistic interpretation of the anomaly score each of them returns. In this thesis, we adopt the perspective of [154], one of the most recent surveys on outlier detection.

In [154] outlier detectors are partitioned into: *clustering-based* methods, *statistical* approaches, *learning-based* techniques, *distance-based* and *density-based* methods and finally *ensemble* techniques. In the following, we briefly present each category of methods.

- *Statistical techniques*: they are the oldest methodologies for outlier detection. They simply detect as outliers those objects which do not fit the distribution followed by inliers. Statistical methods can be divided into parametric and non-parametric. The former requires making assumptions on the shape of the distribution followed by the inlier data: indeed, the core of these methods consists of learning the parameters of the distribution. Instead, non-parametric methods do not make any assumption on the shape of the distribution. In both cases, the final step consists in assessing how well each object fits to the learnt model: a low probability of belonging to the distribution indicates a higher probability of being an outlier. An example of a simple parametric technique is the Gaussian Mixture Model [106]: Maximum Likelihood estimation is used to learn the mean and variance –or covariance if data are multivariate– of the Gaussian distributions making up the data; and then a simple statistical measure is used to compute whether the object adequately fits the learnt model. Statistical methods are easy, they allow a fast detection, and they are mathematically robust. Nevertheless, parametric methods are difficult to use since often the distribution is unknown and in general there is no prior knowledge about the shape of the data. In addition, there are many methods which work only with univariate data or which are unable to correlate multiple dimensions, which is fundamental to accurately detect outliers.

- *Clustering-based techniques*: in this class, classical clustering algorithms are used to detect outliers. Obtained the clusters, outliers can be identified as those objects which: i) are too dissimilar from the cluster they belong to; ii) form an unlikely, i.e. badly shaped, cluster with other outliers. The latter usually happens when there are enough objects not properly fitting the other clusters. There are a lot of techniques for clustering that are used to detect outliers: hierarchical, partitional, grid-based and density-based ones. An example of clustering technique often used for anomaly detection is the *K-Means* [163]. The advantages of using a clustering technique are: i) the unsupervised nature of clustering is the most suitable for outlier detection; ii) partitional clustering techniques are simple and scalable; iii) hierarchical techniques are robust and can deal with different types of data. One of the main disadvantages of employing these techniques consists of most of them not returning an outlier score, which usually conveys a lot of information. To overcome this, the distance of the outlier to the cluster center can be used –if the outliers do not form a cluster on their own. Another limitation is linked to the setting of K, the number of clusters, which, if not adequately

set, can have a negative impact on the parametrization. For example, if the number of clusters is set too low with respect to how data naturally group, outliers may result not too far away from the assigned cluster, which is likely to contain multiple groups of inliers. Finally, most methods are unfeasible to be used with highly dimensional datasets, for example because often clusters compute geometric distances between objects and geometric distances computed across many dimensions may not be significant.

- *Learning-based techniques*: this category comprises all techniques based on active learning, subspace learning, graph-based learning and deep learning.
  - *Active learning*: these semi-supervised techniques are commonly used in those cases where it would take too long to label the entire dataset. After training the model on unlabelled data, the learning process consists of asking the user or the chosen information source to label specifically chosen patterns. The active learning algorithms use these labels to re-train the model and improve it. Several outlier detection methods have been proposed that use active learning. For example, in [60] they use SVDD and the active learning step consists of choosing carefully the objects to be labelled: they propose a combined criterion able to discover novel clusters of anomalies lying close to the boundary of the model.
  - *Subspace learning*: it includes all those techniques which project data into subspaces to find the most relevant dimensions. In outlier detection, this allows to easily deal with highly dimensional data. For example, in [45] they project objects onto sparse subspaces made up of few features; outliers tend to have a lower density and therefore to be found in such subspaces.
  - *Graph-based learning*: in this pervasive field, objects are components of a graph connected with one another. A common way to detect outliers using such techniques is by performing random walks on the graph representation of the data [99].
  - *Deep learning*: there are many techniques aimed at detecting outliers. An example are autoencoders [135] which in an unsupervised way label as outliers those objects which reconstruction error is high.

  Each type of learning-based category has its own advantages and disadvantages [154]: for example, active learning is fast at detecting outliers, while deep learning techniques are able to deal with large-scale data. In general, though, both subspace and deep learning methods can be computationally burdening.

- *Distance-based and density-based approaches*: these approaches use the concepts of density and distance, which are highly interconnected; for example, a typical way to estimate the density in a region of the space is via the use of distances. There exists a numerous amount of outlier detection techniques based on either one of these two concepts: indeed, often outliers are located in low-density regions in the space or/and are far away from the inlier distribution. These techniques include both local and global methodologies. An example of a local density-based methodology is the well-known Local Outlier Factor (LOF) [21] in which an outlier is an object which neighborhood density is much lower than the density of the neighborhood of its neighbors. An example of a distance-based outlier detector is KNNd [146]: an object is more likely to be an outlier if the distance to its K^{th} neighbor is much higher than the distance of the latter to its own K^{th} neighbor. These techniques are cornerstones for outlier detection, and they are non-parametric and simple; in addition, distance-based methods are often easy to interpret. However, they can be expensive from a computational point of view and in case of high-dimensional data, the commonly used geometric distances may not adequately represent the relations between the objects.

- *Ensemble-based methods*: in this class, the core principle is the aggregation of the results coming from several models. Then the aggregated scores can be either used as input to another final model, or as the final result. Ensemble methods are more accurate than single classifiers, they are robust, and can easily manage high-dimensional data [134]. We can divide this category into two subfields.

    1. *Different learning algorithms are used as models.* As to outlier detection, an example is the feature bagging-based approach proposed in [92] which combines the results of different outlier detectors trained with different subsets of features, to deal with high-dimensional data. This class of methods has not received much attention in outlier detection. Some reasons, as stated in [154], may be linked to overfitting and scarce interpretability.
    2. *Several instances of the same learning algorithm are aggregated together*: the instances of the algorithm differ from one another because each one is trained on a subsample of the available data or/and on a subsample of features. This category has received much more attention than the one based on different learning algorithms. An example are ensemble learners for Intrusion Detection Systems [54, 55] where each classifier is trained on a subset of predetermined features. To this category belong also Random Forest-based approaches for outlier detection [96, 142], which have been shown to be very successful [41, 47]: we will focus on these methodologies in the remainder of the thesis.

All the methodologies listed above have both strength points and limitations: in this thesis we focus on the latter introduced techniques, Random Forests, which we describe in detail in Section 2.2.

2.1.3 Evaluation Protocol

After building an outlier detector, it is important to evaluate its performances, i.e. to assess its generalization capabilities on a testing dataset.

Most outlier detectors output for each object an *anomaly score*, which reflects the probability of the object being an outlier. Since different methods measure the anomaly score in different ways, the performance measure should be independent of it. In other words, a good performance measure should allow comparing different techniques. Performance measures that allow such comparison are usually based on the ranking of the objects: objects are ranked from the highest to the lowest anomaly score, i.e. at the top of the ranking there will be objects more likely to be outliers. For outlier detection the choice is among three performance measures, as described in [25]: precision, average precision and the Area under the ROC curve (AUC). All these measures have another common characteristic: to compute them, we need to know the true labels.

In detail, *precision*, also known as precision at n, where n is the number of target candidate outliers, computes how many true outliers are present among the top n ranked objects. Setting n may be troublesome: even if the true number of outliers is known, if n is too small or too large relatively to the size of the entire dataset, it can lead to deceptive results.

To avoid setting n, we can compute the *average precision*, which aggregates the precision at n where n spans over a large range of values. Both precision and average precision, in order to compare different datasets and methods with different outlier ratios, need to be adjusted for chance [25], such that the returned index measures the closeness of the original precision to the expected value. In reality, these two measures have another issue: they are unable to correctly overcome an innate limitation of outlier detection, which is the high imbalance between inliers and outliers, with the latter being very few.

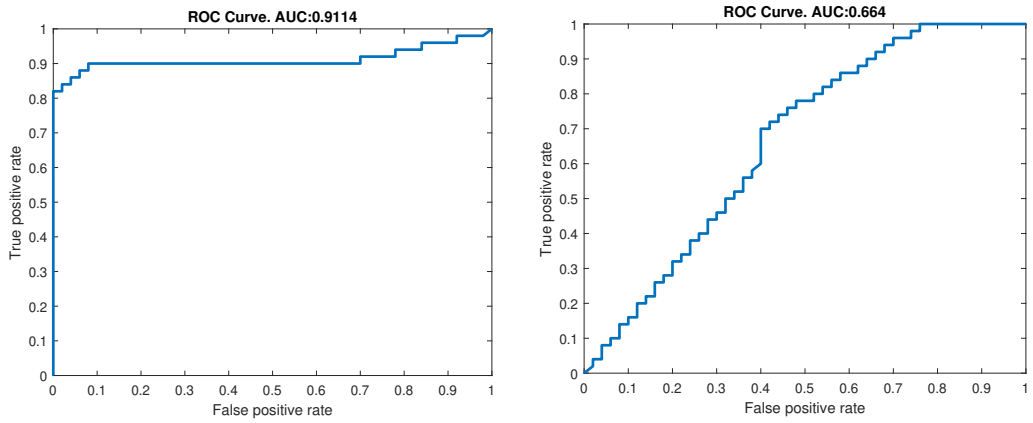
The last measure is the area under the Receiver Operating Characteristic (ROC) curve (AUC). AUC is the most popular performance measure for outlier detection: not only it overcomes the imbalance of the classes, but it does not need to be adjusted for chance thanks to its innate probabilistic meaning. For a better understanding of the AUC measure, let us define the following notions:

- We can decide which label to assign to an object x based on the anomaly score $as(x)$ and a threshold θ on such score: x is an outlier if $as(x) \geq \theta$ whereas it is an inlier if $as(x) < \theta$.

- True Positive (TP): it is the number of outliers that have been correctly identified by the algorithm.

- False Positive (FP): it is the number of inliers that have been erroneously classified as outliers.
- True Negative (TN): it is the number of inliers that have been correctly identified by the algorithm.
- False Negative (FN): it is the number of outliers that have been erroneously classified as inliers.
- True Positive Rate (TPR): also known as sensitivity, it is the ratio of outliers which have been correctly identified, formally: $TPR = \frac{TP}{TP+FN}$.
- False Positive Rate (FPR): it is the ratio of inliers which have been misclassified, i.e. $FPR = \frac{FP}{FP+TN}$.

The first step for computing the AUC consists of plotting the ROC curve. To do that we rank the scores and for each unique value we find the labels, compute FPR and TPR, and plot the FPR against the TPR. From the ROC curve, we can compute the AUC value. AUC ranges in $[0, 1]$ with a value of 1 meaning that there are no FPs, as depicted by the ROC curve in Figure 2.6 (a); instead, as the AUC value gets closer to 0.5 the more likely the classification is performed randomly, as depicted in Figure 2.6 (b).



(a) (b)

Figure 2.6: Examples of ROC curves.

The above measures are fair indicators to understand the goodness of a model. Nevertheless, it is not possible to infer that one outlier detector is better than another simply because it reaches higher performances, especially if the differences are small. To confirm whether the observed difference is true, i.e. it is not due to some random phenomenon, we should perform a statistical test. In short, a statistical test allows to further validate the used model: that is the reason why they are so widespread both in Pattern Recognition and Machine Learning [35, 38].

There exists a wide variety of statistical tests which all rely on a common procedure. In detail, a generic statistical test starts from an assumption called *null hypothesis* which usually states that the compared models are equal, for example in terms of mean. Then a statistical procedure is carried out, which is test-specific, and returns a *p-value*. The *p-value* is a statistical measure denoting the probability of the difference between the models being due to a random phenomenon. By comparing it to a threshold α, also called significance level, which is usually set to either $\alpha = 0.01$ or $\alpha = 0.05$, we can conclude whether the null hypothesis is accepted or rejected. The difference between the compared models is not statistically significant, i.e. the null hypothesis is accepted, when $p - value > \alpha$. Instead, when $p - value \leq \alpha$, we can reject such null hypothesis and confirm that the observed difference is statistically significant.

Statistical tests can be distinguished into two categories: parametric and non-parametric tests. The first category assumes that the shape of the distribution of the evaluated data, in our case one of the performances measures listed above, is known (in many cases it is assumed to be normal). This assumption is rather strong and almost impossible to satisfy in this field: the number of used datasets to evaluate is too low. The latter category of tests is safer since it makes no assumption related to the data distribution. Following this reasoning, in this thesis we perform only non-parametric tests.

In addition to this distinction, the suitability of a test depends also on other factors, such as: the dependence or independence of the models, the number of compared models, the number of datasets on which they are compared and in some cases also on the number of the experiments performed on one dataset. In some scenarios, more than one test may be suitable –often because there is no agreement in the literature about which procedure to follow. Restricting the focus on non-parametric tests only, we can distinguish between the following scenarios:

- *Pairwise Comparison*: this is the most common situation. Two models are compared on one or several datasets, i.e. observations. In the non-parametric case the choices are two [157]: the Whitney-Mann U test, which assumes that the compared models are independent, and the Wilcoxon signed-rank test, which assumes that they are dependent. According to [35], when comparing two classifiers across one or more datasets, the most suitable choice is the *Wilcoxon signed-rank* test: indeed we are comparing how two models perform on the same set of observations. Therefore, in this thesis, we follow the suggestion of [35] and adopt the Wilcoxon signed-rank test: the difference between the two models are computed for each observation. Then the computed differences are ranked based only on their magnitude and split into two groups: one containing the observations for which the first model is better than the second one and the other group containing the converse. The differences are summed up and the minimum between the two is taken, and used as input to compute a z statistic where the other parameter is N,

the number of observations, i.e. datasets. This test is able to handle well the presence of unlikely observations.

- *Global comparison*: with this term, we indicate a comparison of more than two models across one or more datasets. Even though in [35] they suggest the adoption of the rank-based Friedman test [52] followed by a post-hoc Nemenyi test only when multiple datasets are available, in [124] they propose the use of such test also when evaluating several classifiers on one dataset only –even though the analysis is less thorough than [35]. The Friedman test allows comparing at once more than two models in order to assess whether there is a global significant difference among them; to do so, it computes the ranking of the models for each dataset. If the Friedman tests rejects the null hypothesis, i.e. there is a global statistically significant difference among the models, the following step can be performed. The second step consists of a post-hoc Nemenyi test which assesses which pairs of techniques are significantly different by computing the difference of the ranks; if said difference is greater than a critical value, then the two techniques are statistically different. The results of each test can be depicted via a critical difference (CD) diagram [35]: each model is represented via its rank on a single line, with the best rank represented on the right. Whenever two (or more) models are comparable, i.e. there is no significant difference, they are connected by a red line.

- *One versus Others comparison*: this type of testing procedure involves several pairwise comparisons performed independently one of the other. Even though in [35] they discourage to perform this as a set of pairwise comparisons, if we can assume that indeed the various hypotheses are independent, the usual literature approach can be adopted: each comparison is carried out via a pairwise test, in our case the Wilcoxon signed-rank test, corrected by Bonferroni. The *Bonferroni correction* restricts the interval of rejection by a factor equal to the number of evaluated hypotheses. In other words, if we have H distinct hypotheses, and $\alpha = 0.05$ we define the corrected α' as $\alpha' = \frac{\alpha}{H}$.

2.2 Random Forests

The study of Random Forests for outlier detection has received a lot of attention during the latest years, leading to the distinction of two categories. The first are techniques which by creating an artificial class of outliers [37, 142] solve the task in a supervised way by using standard Random Forest for binary classification; nevertheless these techniques are rather complex due to the sampling stage which can be expensive and inadequate. The other category, which is more recent, consists of Random Forest-based approaches that are completely unsupervised and built for the sole purpose of detecting outliers.

These methodologies achieve this by focusing not on the discrimination between the two classes, but rather on separating each instance from the rest of the objects. Outliers are easier to separate due to their nature, i.e. they tend to be separated earlier in the tree building process. This principle is called *isolation* and the cornerstone and first methodology based on it is called Isolation Forest [96, 97], which is thoroughly described in Section 2.3.

Before delving into how Isolation Forests work, in this section we briefly present Random Forests, on which Isolation Forests are based. In detail, we first describe the classifiers composing an RF, Decision Trees, and then we present the ensemble methodology.

2.2.1 Decision Trees

Decision Trees (DT) for classification and regression were firstly introduced by Breiman in 1984 [17]. In detail, they introduced a framework called *Classification and Regression Trees*, commonly known as *CART*. A CART is a binary tree structure that an object traverses based on the answers to a sequence of tests; at the end of the traversal, i.e. when the leaf is reached, the tree assigns to the object an output. In other words, the aim of CART is to classify or regress unknown objects.

A very simple example of a CART is depicted in Figure 2.7: it consists of classifying incoming objects either as cats or dogs. The first question is related to the length of the animal: obviously we imagine that dogs tend to be longer, nevertheless there are some cats' species which exemplars are very long. Therefore, both children need further splitting: both questions are very specific and lead to the creation of two leaf nodes. Each leaf is labelled with the most probable class.

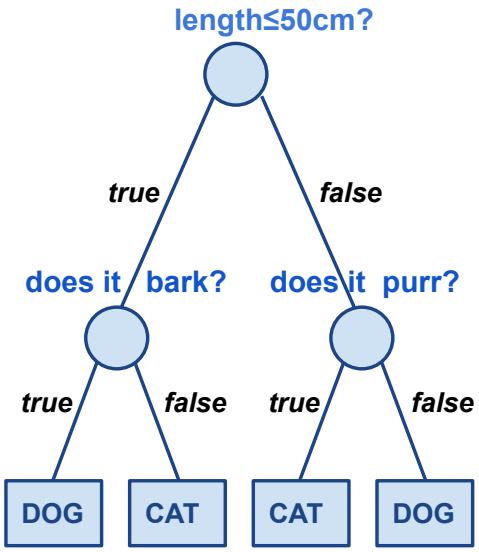


Figure 2.7: Example of a CART to classify cats and dogs.

More formally, a CART algorithm can be divided into two phases: i) training, i.e. building the tree on a set of objects derived from the problem, following the so-called *learning from examples paradigm*; ii) testing, i.e. assigning the output to novel incoming objects based on their tree traversal.

Training: to build a CART t on a set of objects $\mathcal{O}$, we employ a recursive procedure: at each node n, data are partitioned into two disjoint subsets according to a rule inferred from an optimization step. Each of the two subsets represents one child of n: n_L is the left child and n_R is the right one. If a node n cannot be partitioned, for example because it contains only one object, then it is labelled as a *leaf*. In other words, the splitting procedure is repeated until a stopping criterion is met. To build a CART, we have to take several decisions: *i)* how to partition the data in each node; *ii)* what is the best way to perform such partition; and *iii)* how to decide whether a node must be split or be labelled as a leaf. In the following, we describe in detail each of these aspects. For the sake of simplicity, we assume we are working with numerical data on a classification task[3]. In detail, the training set $\mathcal{O}$ is defined as follows: $\mathcal{O} = \{\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_N\}$ where N is the number of objects in the dataset and each object is represented as a vector of f features: $\mathbf{x}_i = [x_{i1}, x_{i2}, \dots, x_{if}]$ where $i = 1 \dots N$. In other words, x_{ij} is the value of the j^{th} feature of vector $\mathbf{x}_i$. We denote as $\mathbf{x}$ a generic object in its vectorial form.

- *i)* The set of objects in node n, $\mathcal{O}_n$, must be partitioned into two disjoint subsets: each subset represents a child of n. To split n we have to evaluate, as defined by Breiman [17], a "condition on the coordinates of x". In other words, a split is determined by a test on a feature –or combination thereof– of the input data. The simplest and most classical type of test is defined by one feature j and a threshold θ on the value of the feature –which is also known as the cut-point. Formally, given a test for a node n defined by (j, θ) , for each object $\mathbf{x}_i$ in n, we evaluate the following:

  $$x_{ij} \leq \theta$$

  if it passes, i.e. the inequality is true, then $\mathbf{x}_i$ will traverse the left branch and reach the left child of n, n_L. Otherwise, i.e. if $x_{ij} > \theta$, $\mathbf{x}_i$ will go to the right one, n_R. That is n_L will contain all objects in n which along the evaluated feature have a value below the threshold. Instead, n_R will contain those objects which value along feature j is above the threshold. Please note that the classical procedure to select the possible candidates for θ is the following: i) sort the objects $\mathbf{x} \in \mathcal{O}_n$ based on their value along the j^{th} feature; ii) remove duplicates; iii) compute the mean along the j^{th} feature of each pair of subsequent objects; iv) choose θ from the set of means.

[3]Please note that except for some adjustments related to aspects ii) and the testing phase, analogous reasoning can be made for a regression task.

The type of test we presented is really simple and straightforward, nevertheless it can be limiting if the discrimination between two classes depends on the correlation of two or more features. In such cases, to improve the approach, linear or boolean combination of features may be used.

- *ii)* There are several ways in which a node n could be partitioned since there exist many choices for both the feature j, in most cases, and the threshold θ. In other words, each possible pair (j, θ) determines a different partitioning of the node n into n_L and n_R, which we denote as $n_L^{(j,\theta)}$ and $n_R^{(j,\theta)}$. To choose the test that determines how n is partitioned, we must evaluate the goodness of each candidate pair (j, θ) in terms of purity of the obtained child nodes. The term purity refers to the labels of the objects: indeed being the final aim to discriminate the different classes, as we go deeper into the tree structure, nodes should contain an increasing percentage of one of the classes. The ideal tree structure would have leaves completely pure, i.e. which contain objects coming from only one class. Summarizing, when choosing the feature j and the threshold θ, we would like to obtain two child nodes as pure as possible. Given a candidate pair (j, θ), we measure the improvement in terms of purity of $n_L^{(j,\theta)}$ and $n_R^{(j,\theta)}$ with respect to the parent node n. In particular, we measure the decrease in impurity between n and its putative children, also known as *misclassification cost*:

$$\Delta I(n, n_L^{(j,\theta)}, n_R^{(j,\theta)}) = I(n) - p_L^{(j,\theta)} I(n_L^{(j,\theta)}) - p_R^{(j,\theta)} I(n_R^{(j,\theta)}) \tag{2.1}$$

where $I()$ is the chosen impurity function and $p_L^{(j,\theta)}$ is defined as $p_L^{(j,\theta)} = \frac{|n_L^{(j,\theta)}|}{|n|}$ and analogously $p_R^{(j,\theta)}$. In other words, they are the proportion of objects in n that end up respectively in $n_L^{(j,\theta)}$ and $n_R^{(j,\theta)}$. By maximizing the impurity decrease, we find the best pair $(\hat{j}, \hat{\theta})$ defining the test of node n. Formally:

$$(\hat{j}, \hat{\theta}) = \underset{(j,\theta)}{\operatorname{argmax}} \quad \Delta I(n, n_L^{(j,\theta)}, n_R^{(j,\theta)}) = I(n) - p_L^{(j,\theta)} I(n_L^{(j,\theta)}) - p_R^{(j,\theta)} I(n_R^{(j,\theta)}) \tag{2.2}$$

and since the term $I(n)$ is a constant with respect to node n, i.e. it is independent of the way n is going to be split, we can perform the optimization by minimizing:

$$(\hat{j}, \hat{\theta}) = \underset{(j,\theta)}{\operatorname{argmin}} \quad I(n_L^{(j,\theta)}, n_R^{(j,\theta)}) = p_L^{(j,\theta)} I(n_L^{(j,\theta)}) + p_R^{(j,\theta)} I(n_R^{(j,\theta)}). \tag{2.3}$$

The function $I()$ is measured using only the labels of the objects of the node under analysis and in literature the most used functions are [67]:

1. Shannon entropy: $-\sum_k P(\omega_k) \log_2 P(\omega_k)$.

2. Gini : $\sum_{k \neq l} P(\omega_l)P(\omega_k)$.
3. Misclassification impurity: $1 - \max_k P(\omega_k)$.

where ω_k is the k^{th} class and $P(\omega_k)$ is the probability of class ω_k estimated as the frequency: $P(\omega_k) = \frac{|n_k|}{|n|}$ i.e. the proportion of objects belonging to class ω_k within n. All these measures reach their maximum value when the classes are equally frequent, i.e. maximum impurity, and minimum when only one class is present in a node. Please note that in general it has been shown that the choice of the impurity function does not have a great impact on the final performances: in other words, in the majority of situations these functions are interchangeable.

- *iii)* In a CART, we have internal nodes and leaf nodes: the latter are created when a stopping criterion is satisfied. There exist several typical choices for the stopping criterion, and more than one can be adopted. For example, a node n is labelled as leaf when its size, i.e. the number of objects it contains, is under a predefined threshold. Alternatively, another choice consists in stopping the tree from growing when the decrease in impurity is very small independently of how n is partitioned, i.e. independently of the test defined on n. Nevertheless, since it is not easy to set the threshold in the first criterion and the minimum impurity difference in the second one, another approach is usually adopted. The approach, called *pruning* [17, 114], consists of growing the tree to its maximum depth, i.e. leaves contain either one object or identical ones, and then remove nodes in a bottom-up fashion to maintain only those nodes that are relevant and informative for the task.

Testing: as we briefly mentioned at the beginning of this section, the testing phase consists of making an object traverse a CART by answering the questions defining each split, starting from the root until a leaf is reached. Depending on the leaf it ends up into, we make a prediction of its output. More formally $\mathbf{z}$ enters the root node and given the test defined by (j, θ), if $z_j \leq \theta$ then it traverses the left edge and ends up into n_L otherwise $\mathbf{z}$ traverses the right edge and ends up into n_R. The traversal continues in the same way until a leaf is reached and a class is assigned to $\mathbf{z}$ according to the adopted decision rule.

Indeed, the latter establishes how a leaf predicts the output of objects ending up there. As to classification, the most adopted technique consists of assigning the class which frequency, observed during the tree building procedure, is the highest. Nevertheless, in some contexts it may be more impacting to obtain FP other than FN, and thus we have to use some cost function which allows preventing an unwanted classification. Let us clarify this concept with an example: we build a CART for evaluating whether a subject has lung cancer or not; it is better to misclassify a healthy subject as diseased, than the converse.

There are several advantages in employing a tree structured classifier [17]:

- It is highly interpretable. Due to the transparency of each step in the tree building procedure, i.e. each test is defined as a threshold on a feature, it is possible to analyze which are the most relevant features (or a combination of features) for discriminating objects with different outputs. This characteristic is of the uttermost importance, especially in recent years where successful models are in most cases black-boxes.

- It is flexible. In detail, there are several ways in which we can build different CARTs from the same set of objects. An example is by changing the type of test or one of the other core aspects of the training procedure. Further, a CART can deal with any type of data by adapting the definition of test defining a node.

- The decision rules stored in the leaves are simple: data can be efficiently classified. At the same time, the decision rule also provides the probability of misclassifying an object.

- Splitting a node is less burdening from a computational point of view than splitting its ancestors: a fewer number of tests, i.e. pairs of feature and threshold (j, θ), needs to be evaluated due to the reduced number of objects with respect to the ancestor nodes.

- To build an optimal tree structure, we do not need to standardize the data, i.e. monotone data transformations do not change the tree structure. This holds if data are made of individually ordered features.

- If the set of objects used to build the tree contains few patterns which label is wrong (this can happen for example when a dataset is labelled manually) the impact on the tree of these objects is going to be low. The limited impact is due to the fact that in a tree all objects are equally important.

Nevertheless, CART can present also some limitations:

- A CART model may be overfitted, i.e. it is unable to correctly classify and/or regress unseen data. This happens if all possible tests are evaluated at each step and no pruning is performed.

- Since the optimization process is independent for each node, it leads to locally optimized tree structures and not to globally optimized ones. This can lead to tree structures which do not describe well data as a whole.

- If we do not use heuristics, as stressed in [67], building the tree can be computationally expensive. Indeed, it is highly dependent on the size of the dataset and on the number of features: at each node data need to be sorted, and several tests need to be evaluated. Longer training procedures are often linked to an increased number of features –which

is the same in all nodes of the tree, contrary to the number of objects which decreases.

2.2.2 Random Forests

Random Forests [19] try to overcome the limitations of a single DT by building ensembles of tree-structured classifiers. In detail, ensembles, by building several base-classifiers which results are aggregated together, are known to be more accurate and with higher generalization capabilities than a single trained model [134]. Indeed, if for a given object one of the models is unable to correctly predict its output, this can be compensated by the other models. Further, each base-learner is trained independently of the others, allowing to capture different aspects of the data.

Breiman formally introduced Random Forests (RF) in 2001 [19] as a general framework that included all previously proposed methods consisting of tree ensembles [5, 18, 20, 38, 70]. Breiman highlighted one common characteristic of these ensemble approaches: each Decision Tree t is characterized by a random vector Θ_t, which is independently and identically distributed (i.i.d.) with respect to the other random vectors Θ_k, where $k \neq t$. In other words, Θ_t represents the randomness injected in the tree t. For example Θ_t can represent:

- A subsample of the training set. The subsample may be drawn with replacement, i.e. an object can appear more than once in the same tree –this technique is known as bagging–, or without replacement, in other words an object can appear at most once in a tree.

- A subsample of the tests to evaluate in a node. In other words, at each node instead of evaluating all possible tests, i.e. pairs of the type (j, θ), we can either: i) evaluate all tests relative to a subset of features; ii) evaluate a random subset of tests.

Given these concepts, we can define a RF as an ensemble classifier $\mathcal{F}$ made of T trees, where each base learner $t = 1 \dots T$ is trained independently using the training set $\mathcal{O}$ and the random vector Θ_t. The trees are then aggregated together at testing level: the output, assigned by $\mathcal{F}$ to an object $\mathbf{z}$ traversing each tree of the forest, is a combination of the outputs returned by each base learner t. An example of such function for RF for classification, is majority voting: the class assigned by the forest $\mathcal{F}$ is the class assigned by the majority of the trees [88].

Since trees are diverse and built independently and thanks to the randomness present within each tree, with RF we can obtain an accurate and robust classifier, with high generalization capabilities. Indeed, the peculiarity of Random Forests is that even if trees are continuously added to the pool of classifiers composing the ensemble, a plateau in accuracy is reached and no overfitting is detected [19].

In all these years, RF for classification and regression have been successfully employed and studied. This thorough research has also led to very peculiar

techniques such as Extremely Randomized Trees (ERT) [53]. This methodology drives randomness to the extreme: each ERT is built by evaluating only one test for each feature, i.e. the cut-off is randomly chosen, and in its even more extreme form, the test is defined completely at random, i.e. both the feature and the threshold are randomly picked.

Considering all the positive features characterizing an RF and their wide success in classification and regression, it would be interesting to focus on other learning tasks. Indeed, in the latest years novel RF-based approaches have been proposed for: survival analysis [73], multiple instance learning [93], clustering [14, 115, 142, 143, 167], multi-label classification [77] and, among others, the already mentioned outlier detection.

2.3 Isolation Forest

Isolation Forest (iForest) is one of the most successful methodologies for outlier detection, as shown in [41, 47]. iForest was first proposed in [96] and the extended version was published in [97]: it is an ensemble RF-based technique where each base-learner is an Isolation Tree (iTree). As mentioned in the first section of this chapter, it is the first approach to contrast the other current of RF-based techniques for outlier detection by proposing a completely unsupervised methodology.

Indeed, the other techniques, such as [37, 142], faced outlier detection in a supervised manner aiming at discriminating the inlier class from outliers. To do that, they artificially sample the class of outliers in order to employ the standard RF for binary classification, leading to computationally expensive methods due to the sampling step. Further, by sampling from the space described by inliers, the obtained representation of the outlier class is often inadequate since we cannot predict how outliers behave, i.e. how they are distributed. In detail, the nature of outliers depends on the problem and for the same problem we may have different types of outliers, e.g. as single points or as clusters.

iForest instead aims at separating each object from the rest of the data independently of the class it belongs to –therefore the method can be used even if only data from one class are available. Indeed, the main principle on which iForest is based is *isolation*: by isolating each object from the rest it is easy to detect outliers since, being few and different from the inlier distribution, they are easier to separate. In the context of Decision Trees, it means that it takes a smaller number of splits to isolate outliers than it does to separate inliers. In other words, outliers have a higher isolation capability. In the next subsections, we describe in detail how iForest works, i.e. how it encodes the isolation principle respectively in the training and in the testing phase.

2.3.1 Training Stage

An Isolation Forest $\mathcal{F}$ is composed by T iTrees which are inspired by *totally randomized trees*, one of the *ExtraTrees* variants proposed in [53].

The *ExtraTree* is a particular way to build the tree with respect to the standard learning algorithm used for RF for classification. The main difference with respect to the methods illustrated in [19] is that the *ExtraTree* algorithm introduces a higher degree of randomness in the training procedure. Indeed, given f the number of features used to build the tree, while in [19] the randomization at node level is given by the selection of a subset of features along which to split, the *ExtraTree* algorithm selects one random cut-value for each of the chosen features, i.e. it evaluates exactly f tests in each internal node. The choice of the best test is carried out in the standard manner, by optimizing an impurity function based on the class labels, such as the Gini index [56]. Lastly, differently from [19] in which a tree is built using a subsample of objects, in [53] the whole training set is used, to make the classifier more unbiased. In [53] they propose another learning algorithm called *totally randomized tree* which is an alternative version of the *ExtraTree* where there is no optimization procedure, i.e. each node is partitioned according to a test defined completely at random. In detail, first we randomly select a feature, and then we randomly pick a cut-point in the domain, relative to the node to be split, of the chosen feature. Even with such a high degree of randomness, both learning algorithms proposed by [53] lead to accurate models. This is interesting since the optimization aimed at discriminating between the two classes is minimal, if not absent.

Algorithm 2.1 iForest

1: **procedure** iForest($\mathcal{O}$,T,S,D)
2: **Input:** $\mathcal{O}$ is the training set, T is the number of trees, S is the size of the training set of each iTree, D is the maximum depth.
3: **Output**: $\mathcal{F}$ is the iForest, i.e. an ensemble of T iTrees
4: $\mathcal{F} = \{\}$
5: $N \leftarrow$ number of objects in $\mathcal{O}$
6: **if** $S > N$ **then** $S \leftarrow N$
7: **for** $i = 1 \dots T$ **do**
8: $\mathcal{O}^i \leftarrow$ Subsample S objects from $\mathcal{O}$ without replacement.
9: $t \leftarrow []$ ▷ Initialize an empty tree t
10: $t.depth \leftarrow 0$ ▷ Set the depth of the root
11: $t \leftarrow iTree(\mathcal{O}^i, D, t)$
12: $\mathcal{F} \leftarrow \mathcal{F} \bigcup t$
return $\mathcal{F}$

iForest implements isolation using *totally randomized trees*, and it is a very suitable choice, one of the reasons being that we are in a completely unsupervised scenario. Algorithm 2.1 shows how to build an iForest $\mathcal{F}$: each iTree t is built independently of the other trees on a subsample of size S of the training

Algorithm 2.2 iTree

procedure iTree($\mathcal{O}, D, t$)
  Input: $\mathcal{O}$ is the set of training objects $\mathbf{x}$, D is the maximum depth and t is the tree structure.
  Output: an iTree t.
  $t.S \leftarrow$ number of objects in $\mathcal{O}$
  $f \leftarrow$ number of features in $\mathcal{O}$
  if $t.depth = D \vee t.S = 1$ **then** ▷ Create a leaf
    $t.left \leftarrow []$, $t.right \leftarrow []$, $t.feat \leftarrow []$, $t.theta \leftarrow []$
    return t
  else ▷ Split the current node
    $j \leftarrow$ Randomly pick in $1 \dots f$
    $\theta \leftarrow$ Randomly pick in $[\min_{i=1\dots t.S} x_{ij}, \max_{i=1\dots t.S} x_{ij}]$
    $\mathcal{O}_L \leftarrow \{\mathbf{x}_i | i = 1 \dots t.S \wedge x_{ij} \leq \theta\}$
    $\mathcal{O}_R \leftarrow \{\mathbf{x}_i | i = 1 \dots t.S \wedge x_{ij} > \theta\}$
    $t.feat \leftarrow j$
    $t.\theta \leftarrow \theta$
    $t.left.depth \leftarrow t.depth + 1$
    $t.left \leftarrow$ iTree$(\mathcal{O}_L, D, t.left)$
    $t.right.depth \leftarrow t.depth + 1$
    $t.right \leftarrow$ iTree$(\mathcal{O}_R, D, t.right)$

set $\mathcal{O}$ drawn without replacement. In other words, an object cannot be used more than once in the construction of a certain t.

Algorithm 2.2 shows how to build each t. Each tree is recursively built by choosing how to split a node n in a completely random way. As described for the *ExtraTree* algorithm, both the feature and the cut-value are chosen randomly. In detail, the latter is picked at random in the range of values of the feature along which to cut, and not among a predefined finite set of values. This procedure is repeated until a stopping criterion is met: if the depth of the current node is greater than a pre-established maximum depth D, then it is labelled as a leaf.

Hypothetically, this learning algorithm should lead to a tree structure able to isolate earlier outliers than inliers. Indeed, due to the nature of outliers and thanks to the randomness of the test determining the split, there is a higher probability to pick a cut-value that will separate the outlier from the rest of the data. In other words, during the first few splits of an iTree, there is an increased probability that the isolated objects are outliers. We can assess this by observing Figure 2.8. This figure depicts a bivariate normal distribution which has been contaminated by the presence of two outliers. We show, as a shaded area, the possible splits that could be performed by an iTree along Feature 1 that would lead to the separation of the two outliers from the rest of the data.

In Figure 2.9 we have the same distribution of Figure 2.8. We also represent

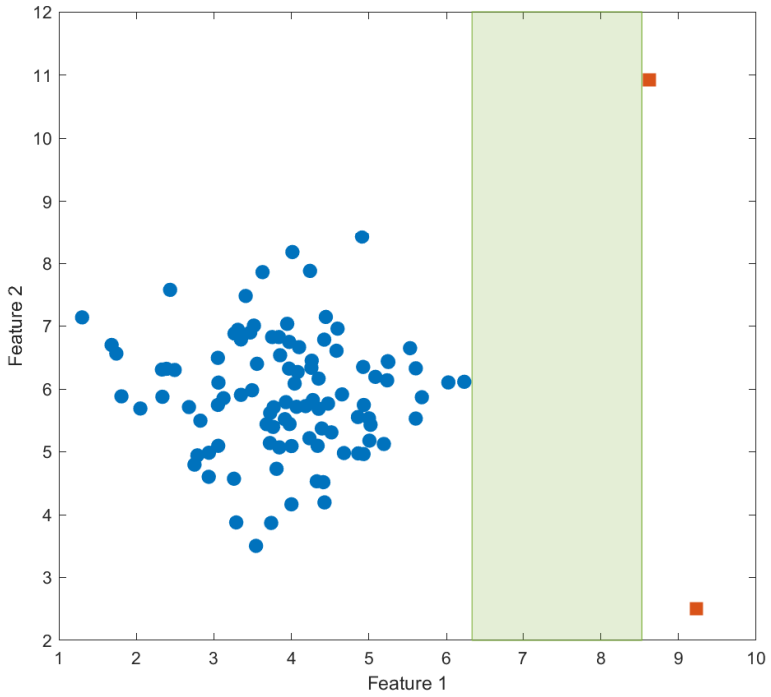


Figure 2.8: Depiction of the possible splits along Feature 1 that would separate outliers from inliers.

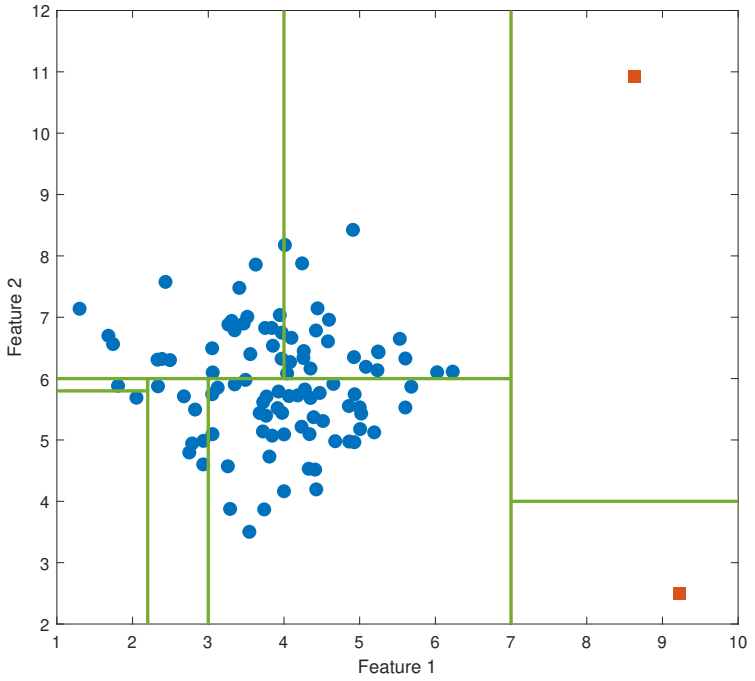


Figure 2.9: Example of how an Isolation Tree works, represented via random splits (not all points are isolated).

a hypothetical iTree that isolates these objects: each split is represented by a line parallel to one of the axes –each representing one feature. We can observe that this iTree is good, i.e. it is able to quickly isolate outliers. Indeed, we can observe that both outliers are isolated in 2 splits; we also show the isolation of a marginal inlier, which instead takes 5 splits.

2.3.2 Testing Stage

At testing level, given a built iForest $\mathcal{F}$ and a new unknown object $\mathbf{x}$, the aim is to isolate $\mathbf{x}$ in $\mathcal{F}$ to estimate its probability of being an outlier. To do that, as described by Algorithm 2.3, $\mathbf{x}$ must traverse each tree from the root to the leaf: in each internal node n, $\mathbf{x}$ goes to n_L if $x_j \leq \theta$ is true, otherwise it goes to n_R. This procedure is repeated until a leaf is reached. The second step consists of retrieving the isolation capability of $\mathbf{x}$: considering the assumption that outliers tend to traverse a shorter path before being isolated, in [96, 97] the *anomaly score* is defined as a function of the depth of the reached leaf in each tree.

Algorithm 2.3 iForest Testing

procedure AnomalyScore($\mathcal{F}, c, \mathbf{x}$)
  Input: $\mathcal{F}$ is a trained iForest, $\mathbf{c}$ is a vector of size S containing the normalization factor defined in Eq. (2.5), $\mathbf{x}$ is a testing object.
  Output: s is the anomaly score of $\mathbf{x}$.
  $\mathbf{h} \leftarrow$ Vector of zeros of length T ▷ h contains the depth of the reached leaf
  for $i = 1 \dots T$ **do**
    $t \leftarrow \mathcal{F}(i)$ ▷ Start the traversal from the root
    while $t.left \neq [] \vee t.right \neq []$ **do**
      if $x_{t.feat} \leq t.\theta$ **then**
        $t \leftarrow t.left$
      else ▷ $x_{t.feat} > t.\theta$
        $t \leftarrow t.right$
    $h_i \leftarrow t.depth + c_{t.S}$
  $Eh \leftarrow \frac{\sum_{i=1}^{T} h_i}{T}$
  return $s \leftarrow 2^{-\frac{Eh}{c_S}}$

Before formally presenting the anomaly score, we define some useful notation: i) $l_t(\mathbf{x})$ is the leaf node reached by $\mathbf{x}$ in t; ii) $h_t(\mathbf{x}) = h_t(l_t(\mathbf{x}))$ is the depth of $l_t(\mathbf{x})$; iii) $\mathcal{P}_t(\mathbf{x}) = \{n_1, n_2, \dots n_{h_t(x)}\}$ is the path of $\mathbf{x}$ in t, i.e. the set of nodes traversed by $\mathbf{x}$ in t from root to $l_t(\mathbf{x})$. Formally, the anomaly score of $\mathbf{x}$ in $\mathcal{F}$ is defined as:

$$s(\mathbf{x}) = 2^{-\frac{E(h_t(\mathbf{x}))}{c(S)}} \tag{2.4}$$

where $c(S)$ is a normalization factor that allows making a fair comparison of forests of different sizes, defined for a generic integer n as:

$$c(n) = \begin{cases} 2H(n-1) - 2(n-1)/N & \text{if } n > 2 \\ 1 & \text{if } n = 2 \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

where $H(n)$ is the n^{th} harmonic number. This formula was initially designed by [127] to estimate the average path length of unsuccessful searches in Binary Search Trees. In [96, 97] they show how the latter tree structure and an iTree are equivalent. Therefore $c(n)$, in the context of iTrees, can be interpreted as the average depth of a tree built with n samples, thus justifying its normalization role.

Finally, the term $E(h_t(\mathbf{x}))$ in Eq. (2.4) computes the average height as:

$$E(h_t(\mathbf{x})) = \frac{\sum_{t \in \mathcal{F}} h_t(\mathbf{x}) + c(|l_t(\mathbf{x})|)}{|\mathcal{F}|} \tag{2.6}$$

where $c(|l_t(\mathbf{x})|)$ is the same normalization factor defined above, and it estimates the average depth of the subtree rooted in $l_t(\mathbf{x})$.

Since it is likely that with few splits, outliers get isolated, the anomaly score defined in Eq. (2.4) is an excellent measure to characterize outliers and distinguish them from the remainder of the data. Indeed, outliers will have a higher anomaly score due to their higher probability of traversing a shorter path. The anomaly score ranges in $(0, 1]$: a value ≥ 0.5 is usually an indicator that the object is an outlier [96, 97]. Please note that it is unlikely for an object to have the maximum score: it happens only when all t in $\mathcal{F}$ are composed by one node only (the root, which has depth 0). Indeed, the final score is averaged across all trees to have a more robust estimate of the probability of an object being an outlier. As to the minimum, it cannot happen that $s(\mathbf{x}) = 0$ since the scoring function is exponential.

Just to provide an example of how an iForest works, given the data distribution shown in Figure 2.8, we make each object traverse a given forest $\mathcal{F}$, trained using only the inlier objects. In Figure 2.10 we plot the same distribution and colour the objects based on the anomaly score: it can be observed that points that belong to the inlier distribution but are more marginal tend to have a higher score, but more importantly that the highest anomaly scores are assigned to the two outliers.

2.3.3 Applications and Extensions

As briefly mentioned in Chapter 1, Isolation Forest is one of the state-of-the-art methodologies for outlier detection [41, 47]. Indeed, it has been widely used from an applicative point of view, and it has been thoroughly studied and extended.

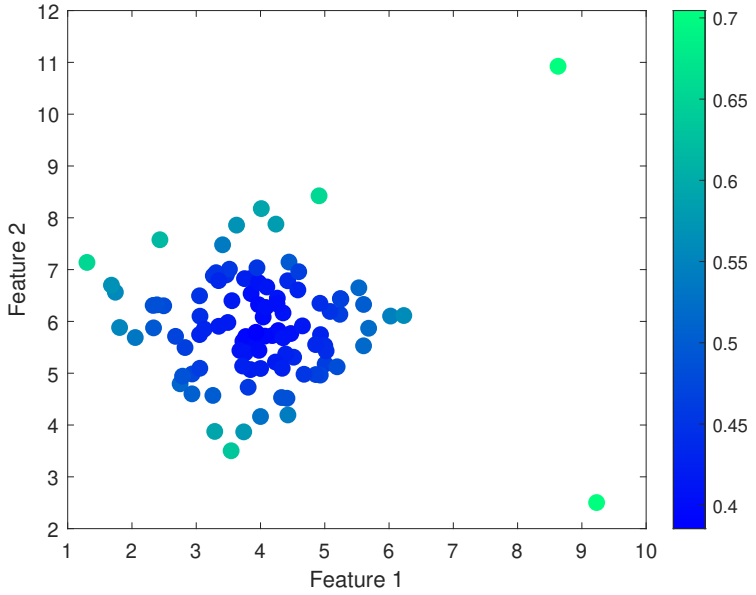


Figure 2.10: Distribution of the anomaly scores computed via an Isolation Forest.

Applications

In this section, we make few examples to show the flexibility and pervasiveness of the methodology. In detail:

- The first example concerns the field of smart production monitoring [145]. In this work they adopt Isolation Forest to monitor plasma etching, and they adapt it to the on-line nature of the context. To do that, the authors define the model off-line and update it as new objects are added to the dataset; a threshold is defined to generate warnings in real-time. Whenever a warning shows up, i.e. an outlier is identified, an on-line search is launched to find what generated the outlier. The methodology is compared to standard and other state-of-the-art techniques, with iForest reaching the best results.

- In [87] an analysis of the energy consumed by institutional buildings certified by the Leadership in Energy and Environmental Design (LEED) program is performed using iForest. The aim is to understand whether the certification, which is given prior to the end of the construction of the building, is valid in terms of energy efficiency also after the building process is ended. The methodology works as follows: energy consumption data are clustered in various groups, then iForest is used to identify outliers. Outliers are for example data related to a particular building which consumes great amounts of energy even though according to the LEED certification it should not. The methodology confirms that the tracking of the energy consumption post construction is necessary, otherwise the efficiency may not be as high as it is assumed.

- An example of the usefulness of the iForest in the medical field is made by [90]. Indeed, iForests differently from other methods are reliable and quick: these are important and necessary requirements in the healthcare field. In this study, iForest is used to detect changes in vital signs in the elderly population which could hinder a sudden decrease in their health status. In detail, data are clustered using K-means and then iForest is used to detect the anomalies. A lower error rate is shown for this hybrid approach –its usefulness is also shown on some benchmark datasets.

These examples are just to show how this methodology can be employed easily in different fields. Indeed, there are many more works that use iForest, and it would not be of use to cite them all. Nevertheless, it must be noted that iForest are used in a variety of fields, such as: weather predictions [95] and in general in environment-related researches [160]; cybersecurity [3]; load forecasting [102]; image processing [155] among many others.

Extensions

In addition to be applied in its original formulation, Isolation Forest has been thoroughly extended. We may divide the extension into *training extensions*, which propose alternative ways to build the trees able to encode isolation, and into *testing extensions*, which consist of different definitions of the anomaly score. Often though, the latter are found combined with the first. In this section, we present several of such extensions, trying to highlight their core concepts.

In the following, we present those works which propose *only* a training extension, i.e. they use the anomaly score defined in Eq. 2.4.

- In [65] a generalization of iForest, called Extended Isolation Forest, is proposed. The assumption is that splitting data along one feature may not be informative enough. Therefore, the test on each node is defined by a random hyperplane. The hyperplane can be defined using from one to all features. If one feature is used, the methodology is equivalent to the original formulation of iForest.

- In [57] they propose an optimized training alternative that encodes the isolation principle. The aim is to split each node such that one child will be of maximum volume and will contain the fewest objects, whereas the other child will be of minimum volume and will consists of the highest number of objects. Ideally these nodes should contain only outliers and inliers respectively. To obtain such partitioning, they propose an optimization function based on a local estimation of the number of outliers and an estimation of the node volume.

- The methodology in [144] modifies the iTree structure to deal with datasets for which no outliers are available during the training phase. Detection

is improved by splitting nodes outside the range of the chosen feature and by establishing a further stopping criterion based on the node size.

Below we describe those studies which propose a novel anomaly score, even if jointly with an alternative training procedure:

- In [98] the authors of iForest propose SCiForest, which aims at identifying not only single outliers but also those that come in small clusters. Indeed, clusters are difficult to identify when using standard iForest. The proposed training procedure is similar to that of [65]. In detail, SCiForest proposes a novel learning algorithm which consists of performing non-parallel splits, i.e. based on tests defined on multiple features, in an optimized way. The optimization function is unsupervised, and it measures the decrease in standard deviation of the child nodes created by the evaluated test with respect to the parent node. SCiForest also slightly redefines the scoring function defined in [96]. Indeed, since the main aim is to detect clusters of outliers, to account for unpredicted point outliers, for each internal node n they define an *acceptable range* on the value of the feature j used to split n[4]. Indeed, if an object along the j^{th} feature has a value outside that range, i.e. it is very different from the training data, it is more likely to be an outlier. Therefore, when measuring the path length, whenever an object $\mathbf{x}$ has to traverse n and the value of its j^{th} feature is outside the acceptable range, the edge exiting from n is removed from the total path length. The rest of the procedure is identical to that of [96, 97].

- The work of Guha et al. [63] instead slightly modifies the original training procedure by assigning a weight to each feature. The weight is directly proportional to the width of the range of the feature: a small range means that outliers on that feature have similar values to inliers, in other words, the feature could hinder the identification of outliers. These weights are used when choosing the feature along which to split: it is more probable to choose a feature with a high weight, i.e. a feature on which outliers and inliers are more likely to differ. In addition, since [63] is aimed at dealing with streaming data, a novel anomaly score is proposed, which takes into account how the complexity of the model, i.e. the tree, changes if the leaf where the object ends up into were to be removed.

- Karczmarkek et al. propose several extensions of iForest [80, 81, 82, 83].

  - In [80] they focus *only* on the testing phase. They define for an object $\mathbf{x}$ a membership value in $[0, 1]$ for each traversed node in a tree t. This membership value is based on the average distance to the center of the node computed during the training stage. The

[4]Please note that for simplifying the explanation we assume that data are univariate and therefore the test to split a node is defined on the only available feature, i.e. it is not a hyperplane.

membership score of $\mathbf{x}$ in t is obtained by summing the membership values for all nodes in the path of $\mathbf{x}$ of each node. By computing the average across all t in $\mathcal{F}$, the score at forest level is obtained, which is inversely proportional to the probability of $\mathbf{x}$ being an outlier.

- In [81] the authors focus on the training phase by proposing a $K-ary$ tree, i.e. a tree where each node has at most K children. To partition each node, after randomly choosing the feature along which to split, a K-means clustering on the values of that feature is carried out for all possible values of $k \in [1, K]$. Then the optimal k, and therefore partitioning of the node, is chosen according to the elbow rule. The scoring function is the one defined in [80].
- The contributions of both [81] and [80] are at the core of [82]: the main difference consists of the use of the K-medoids algorithm instead of the K-means.
- In [83] three alternative ways to implement isolation are proposed. The first is analogous to [81] but the Fuzzy-C means algorithm is used instead of K-means. In each node, the dispersion of each cluster is measured and then it is used in the membership score computation as follows: given an object $\mathbf{x}$ traversing a node n, its membership value is increased by the dispersion of the cluster to which $\mathbf{x}$ belongs to. To obtain the membership score at tree level, the contributions of each node are summed and then averaged to obtain the membership score at forest level. The second strategy proposed in [83] differs from the first only in the number of clusters, which is fixed to 2. The third proposal defines a more refined membership score for the second strategy: given an object $\mathbf{x}$, for each traversed node n, both the dispersion of the cluster to which $\mathbf{x}$ belongs in n and the distance to the cluster center are used.

- In [165] a general framework is proposed, called LSHiForest. This methodology generalizes the iForest for any type of data for which a Locality-Sensitive Hashing (LSH) family [72] is defined. A LSH family puts in the same hash bucket similar objects with a higher probability than it puts dissimilar objects together –where the dissimilar and similar terms are relative to a threshold. There have been defined different types of families for different distance measures. In [165] they start from the assumption that iForests are implicitly based on distances and neighborhoods and that at the same time they do not use well such information. In LSHiForest neighborhood-related information is one of the core principles: data are split into K children according to a hashing function. Two objects which answer in the same way to such function, i.e. which are similar, end up in the same child. As to the testing phase, they propose a normalized anomaly score where the normalization factor is relative to each tree; further, they combine the tree scores using the arithmetic mean.

- A very different way of implementing isolation is proposed in [24] where $K - ary$ trees are built. After fixing K, each node n is partitioned into K children by choosing K objects in n as prototypes. In detail, the similarity between each object in n and each prototype is computed: the objects are assigned to the child node which representative prototype is the closest. For the similarity computation, a specific predefined kernel function must be used. If in a node there are $k < K$ objects, the number of children is set to k. As to the testing phase, the anomaly score is inversely proportional to the size of the reached leaf, i.e. the number of training objects that have ended up in the leaf.

In Chapter 4 of this thesis, we used some of the presented methods [24, 65, 98] to further support the suitability of our contribution and to compare to other works in the field of isolation-based outlier detection.

2.3.4 Advantages and Limitations

Isolation Forest presents several advantages, also with respect to other techniques. One of the most important is the innate flexibility of this technique: as shown by the great variety of extensions, the isolation principle can be implemented in many different ways. Secondly, it is innately unsupervised, which is the most suitable scenario for outlier detection. This is an important difference with respect to other RF-based outlier detectors, which sample artificially outliers and use them within a supervised technique. Another relevant advantage is that the isolation principle seems to work very well [97] leading to generally better performances than distance and density-based outlier detectors. In addition, in [41, 47] it has been shown to be comparable to sophisticated state-of-the-art detectors: this observation is even more relevant considering that iForests are composed of completely random trees. In addition, some advantages of iForest are linked to them being: i) ensemble classifiers; ii) RF-based techniques. As to the former, they tend to have higher generalization capabilities and are more robust than single classifiers do [88]. As to the latter, an important advantage is the scalability of the method, both as to the size of the dataset and as to the number of features. Another advantage worth mentioning is that iForests are fast: to obtain an accurate model a small number of trees is enough, and these trees usually need to be trained using very few objects (often each tree is trained using only *256* objects). In addition, a trained iForest does not require much space in memory and also the testing phase is rather quick.

Even though iForests are very advantageous from several perspectives, as just illustrated, there is large room for improvements. For example, Isolation Forests are not able to deal with non-vectorial data, i.e. they work only with vectorial data. This limitation is of crucial importance due to the many outlier detection problems which are non-vectorial in nature [6, 46, 69, 76, 100, 140] that would benefit of an RF-based approach. In Chapter 3 we overcome this limitation by proposing a general methodology which can deal with all types

of data for which a distance measure is defined. Further, the distance measure does not have to satisfy any specific mathematical requirement. Thus, the proposed approach can manage many different types of non-vectorial data without any further adjustment. Please note that this is the first approach designed for non-vectorial data, aside from [165] in which although the distance measure must satisfy specific requirements.

Another aspect which needs further research is the testing phase, i.e. how to compute the scoring function. In detail, most extensions of iForest propose an alternative implementation of the isolation principle, i.e. how to build the tree, and there is a lack of researches which main focus is on the testing phase. Nevertheless, an RF model is very rich in information which can be used to better characterize the outliers both at tree and at forest level. We study this aspect thoroughly in Chapters 4 and 5 where the latter is a very peculiar alternative to the standard testing phase.

Lastly, there are a lot of application fields which could benefit of iForests, either in their original formulation, or extended and adapted for a specific context. We investigate along the latter research direction in Chapter 6 where we propose a novel RF-based approach for characterizing multiple sclerosis.

Chapter 3

Proximity Isolation Forests

In this chapter, we propose Proximity Isolation Forest, an advanced RF approach for outlier detection that works with pairwise distances. In other words, it can manage non-vectorial representations. First we present the motivations and the intuition behind the contribution, and then we thoroughly describe the proposed approach, with a particular focus on the different training strategies.

3.1 Introduction

In Chapter 1 we briefly mentioned that data can be represented in several ways. An important categorization divides data into:

- **Vectorial**: these data are represented via a set of features. These features have a straightforward mathematical interpretation, since they can be represented in a euclidean space.
- **Non-vectorial**: this category comprises all data which structure is more complex. Examples of non-vectorial representations are: images, sequences, graphs, strings, etc.

The most common approach to manage non-vectorial data consists of extracting high-level features since the majority of PR techniques manage only vectors whereas dealing with other types of representation is rather complex. Nevertheless, the extraction of meaningful features from non-vectorial data is not straightforward and can lead to a severe information loss, i.e. we lose information about the true structure of the data. Further, the relation between different patterns may not be encoded correctly due to these non-representative features. Let us clarify with an example: a 2D image is made up of several pixels, each surrounded by other pixels. If we vectorize an image, we lose the information concerning the position of each pixel and subsequently the relation between different pixels. Therefore, the most reasonable choice is to deal with non-vectorial objects without modifying their nature.

A good solution to lower the innate complexity of vectorial data while maintaining their nature is to reason in terms of distances between the patterns. Reasoning in terms of distances not only is methodologically sound, since we do not lose any information related to the structure of the data, but also easy to interpret. Indeed, also from a human perspective, it is straightforward to think about an object in terms of how similar (or dissimilar) it is to another object. However, if asked to describe why these two objects are similar, i.e.

which are the features, the answer is usually more complex and not always correct. We can see this from another perspective: when dealing with a recognition task we ought to find classes which are composed by *similar* objects, i.e. objects which distance is low. Therefore, we can say that the notion of distance is beyond the notion of class and even that of features, since features are needed to discriminate between different classes. Please note that often we use the terms similarity and dissimilarity/distance interchangeably. Nevertheless, it is more intuitive to reason in terms of the latter, since they focus on the differences between the objects.

Given a set of data described with the set of all pairwise distances and a recognition task to carry out, we can use different methods:

- *Methods which directly use the distances to solve the recognition task.* Considering a classification task, the most classical example is the K-Nearest Neighbor [32]: for an object x, we find the K closest objects. Then we assign to x the most frequent label among the K neighbors. Also, several clustering techniques directly exploit distances to build the clusters [51, 153].

- *Methods which define an embedding of the dissimilarity in a vectorial space.* These methods embed the dissimilarity in a vectorial space while trying to preserve the pairwise distances. An example is to embed distances in a pseudo-euclidean space [58, 66], i.e. a space made of two orthogonal subspaces of dimensions p and q which allow dealing with negative distances. The strong constraint is that the input distance measure to embed must be a metric. A metric is a distance measure which satisfies several mathematical constraints that are seldom satisfied by distances defined for non-vectorial objects. Nevertheless, when the constraint is satisfied, we can apply these techniques to use all classical PR approaches for vectorial data.

- *Methods which define a dissimilarity-based vectorial representation.* These techniques deal with all types of distance measures by directly embedding the distance matrix into a vectorial space, in which each object is represented by a vector of distance values to all other objects. In order to reduce the dimensionality of such space, only few objects are selected as prototypes, i.e. features. In this way, we obtain a compact and discriminative representation space [121]. These methods as well allow employing all techniques for vectorial data.

Please note that the techniques that directly use the distance matrix for the recognition task do not apply any modification to the representation. This can be very beneficial when we do not want to perform any transformation or selection procedure.

Most Random Forest-based approaches work with vectorial representations. Nevertheless, in the latest years it has been realized that working with distances

is of the uttermost importance and some recent approaches for classification and regression investigate in this research direction [9, 42, 64, 101, 136, 161]. However, there have been no similar proposals in the outlier detection context even though there are several problems which deal with non-vectorial representations: finding abnormal sequences in an ECG [100], detecting small masses in a brain image [46], or detecting traffic-related anomalies where data are represented by graphs [140] among others [6, 76]. While there exist several distance-based methodologies that can deal with such data –even though some may be restricted to metrics– and even though some recent RF approaches for classification and regression have been designed, using RF to deal with non-vectorial data in the outlier detection context has not been investigated. Indeed, to our knowledge only [165] proposes a general framework that deals with several types of distances, as briefly described in Chapter 2. This technique builds K-ary trees and while being very versatile it is restricted only to those measures which belong to an LSH family.

In this chapter we insert ourselves into the above scenario by proposing **Proximity Isolation Forest** (ProxIForest), an extension of Isolation Forest, that works only with distance measures, which do not have to be metrics. After introducing the model, we focus on describing the traversal modalities of a tree and the training procedures. As to the latter, we design nine different ways to perform the training step: two of them are random, inspired by Isolation Forest, while the remaining seven make an informed decision, trying to capture the difference between the inlier distribution and the outliers. In detail, three of them try to discern outliers from the rest via the evaluation of the scatter within a node, i.e. the sparseness of the distance values, the rationale being that outliers increase the scatter of a set of data. The remaining four criteria try to isolate outliers by capturing the difference between the two child nodes in terms of Hausdorff distance [71] and via an estimation of the Rényi divergence [118, 120, 131] respectively. The rationale is that two nodes are expected to be the most different –according to the Hausdorff distance or the Rényi divergence– when one of the two children isolates outliers.

The aim of ProxIForest is thus to identify outliers independently of their original representation, given that a meaningful distance measure exists between the objects. In other words, ProxIForest is applicable in all those scenarios in which distances are one of the best tools to identify outliers, i.e. whenever it can be assumed that outliers lie far away from the remainder of the objects and/or whenever data are non-vectorial. A practical example of the applicability of the proposed method, is the detection of heart defects via the comparison of ECG sequences. The first step is to choose an appropriate distance measure for sequences such that, if there is an anomalous instance indicating an abnormality in the ECG, it will be highly dissimilar from the rest of the sequences. This difference can then be easily captured by an adequately trained ProxIForest, which will identify the anomalous ECG sequence as an outlier.

To assess the suitability and robustness of ProxIForest, we carried out a

thorough experimental evaluation: all designed criteria have been tested on 10 non-vectorial datasets employing a great variety of parametrization settings. Further, we make a comparison with state-of-the-art density and distance-based methodologies, which assess that using ProxIF is beneficial when working with non-vectorial representations. Part of the ProxIF methodology was published in the proceedings of the International Conference on Pattern Recognition [109] and the complete approach has been submitted to Pattern Recognition.

The rest of the Chapter is organized as follows: in Section 3.2 we describe in detail the existing DT and RF methodologies that reason in terms of distances between objects to solve classification and regression tasks. In Section 3.3 we thoroughly present how to build a Proximity Isolation Forest starting from the description of its base components, Proximity Isolation Trees. In Section 3.4 we make a thorough experimental evaluation. Lastly, in Section 3.5 we make a conclusive discussion highlighting the advantages and disadvantages of the proposed technique and discussing some ideas for future research.

3.2 Background

In this section, we briefly describe some DT and RF approaches for classification and regression that work with distances [9, 42, 64, 84, 101, 136, 161]. First we briefly present methodologies that deal with a specific distance measure, e.g. Dynamic Time Warping, Hausdorff distance, etc., and then we describe techniques which can work with any distance measure.

In detail, as to the first type of approaches, there have been designed several DT/RF techniques for classification of time series data [9, 42, 84, 161, 162]. Before describing each approach in detail, let us highlight their common notions:

- The aim is to classify the pattern x. Each pattern x, also called time series, is composed by several time sequences. Each time sequence is a set of chronologically ordered values.

- A distance measure is chosen to evaluate the dissimilarity between two time series on the i^{th} time sequence.

- The test on a node n is defined by (P, θ, i) or (P_L, P_R, i) where P, P_L and P_R are time series in n, i is a time sequence and θ is a threshold on a distance value. When the test is defined by (P, θ, i), a pattern x will go to the left child of n if $d(x^i, P^i) \leq \theta$, i.e. the distance between the two time series along the selected time sequence is lower than θ. Otherwise, it will go to the right child. An analogous reasoning holds if the test is defined by (P_L, P_R, i).

- Each candidate test is evaluated using a classical impurity function: entropy, gain ratio or Gini index.

The peculiarities of each approach are described in the following:

- In [161] they propose two ways to learn the tree: either the test of a node is defined by (P, i, θ) or it is defined by (P_L, P_R, θ). The distance used for the evaluation is a classical distance measure for time series data: the Dynamic Time Warping (DTW) Distance. In each node, all possible tests are evaluated using the gain ratio as impurity function.

- The core principle of [42] is the design of a novel distance measure used for the evaluation of the tests in a node. In detail, they define a measure combining value-based measures, e.g. the DTW distance, with behavioural-based measures, e.g. a temporal correlation coefficient. The latter is needed because it takes into account the fact that two time sequences may be similar in terms of their growth rate. The obtained distance measure has several parameters to be set, and in [42] they claim that they should be adaptively set for each node. In other words, these parameters represent additional elements to be fixed, when deciding how to split a node, i.e. when defining the test. Another relevant characteristic of this method is that it aims at finding the most discriminating subsequences, called shapelets, to use for the definition of the test in a node. In other words, a test may not be defined on a time sequence but on one of its subsequences. The Gini index is used as impurity function.

- Another DT-based methodology is [9]: actually this approach manages different types of data, i.e. not only a pattern can be a time series but also a pattern which attributes are vectors. Therefore, the learning algorithm is slightly different from what we described: after choosing an attribute for the distance evaluation, the distance is measured either using the Euclidean distance or the DTW distance depending on the attribute. Then objects in a node are partitioned into two clusters using K-means. From the resulting clusters, the candidate patterns P_L and P_R defining the test of such putative split are selected as: the centers of the clusters if the Euclidean distance was used, or as the object closest to all other objects in each cluster if the distance was measured using DTW. The procedure is repeated 1000 times for each candidate attribute. The Gini index is used as impurity function.

- In [84] an RF-based approach is proposed. The methodology exploits the concept of shapelet, analogously to [42]. A Random Shapelet Forest is made of several Random Shapelet Trees [162]. Each tree is built on a subset of randomly sampled time series. In a node only a subset of candidate tests is evaluated: each test is defined not only by (P, i, θ), all randomly picked, but also by the length l of the subsequence of P^i to use as shapelet. Then the length-normalized Euclidean distance is computed between the shapelet and each subsequence of length l of the time sequence i in each time series in n. The best split is found by optimizing an entropy-based impurity function.

While these Decision Trees methodologies rely on a specific distance measure, recently RF-based techniques that can work with several types of distances have been proposed. We describe these approaches in the following:

- *Proximity Forests* [101] is an RF-based methodology for classifying time series that can work with several distance measures. Analogously to the previous works, each split is defined by a pair of objects (P_L, P_R), i.e. prototypes, and a distance measure. Nevertheless, the procedure to choose such pair is rather different from before: r pairs of objects are randomly sampled with the only constraint that they must belong to different classes. Also, the distance measure is randomly picked in a set of 11 distances. Having defined these elements, each candidate pair generates two different nodes, n_L and n_R, analogously as what done by [161]: each object is assigned to the node which representative prototype is closer. Each test is evaluated in terms of the Gini index. Please note that even though the authors focus on time series data, the methodology can be generalized to work with any distance measure without any further adjustment.

- Another methodology which works in a similar fashion is *Similarity Forest* [136]. This technique can work with any data type: the initial assumption of this technique is that objects can be theoretically embedded in a multidimensional space, but the related representation is unknown. Each split is defined by a randomly sampled pair of objects (P_L, P_R); the sampling procedure can be constrained such that they belong to different classes. When evaluating how to split a node n, r such pairs are sampled. In practice, to partition the data in n given a pair of prototypes, the procedure works as follows:

  1. Draw a line from P_L to P_R.
  2. On this line, project all other objects x in n: this allows to know all possible splitting points.
  3. Choose the best split according to the Gini criterion. [56], analogously as what Proximity Forest does.

  This process is unfeasible in practice, since the representation of the objects is unknown. Nevertheless, they show that the projection of x on the line from P_L to P_R corresponds to $s(x, P_R) - s(x, P_L)$ where $s()$ is a similarity measure. Therefore, to split n, for each x in n this estimation of the projection is computed, then the obtained values are sorted and lastly all possible splitting points are evaluated. Similarity Forests do not require for all similarities to be known: indeed, they can be computed only when needed. Similarity Forest can handle well missing similarities and multidimensional data.

- *Comparison-based Random Forests* [64] analogously to Similarity Forests is independent of the type of data, and it does not require for all pairwise

similarities to be known. The main difference with respect to previous approaches is that it works only with the following information: given a triplet of objects (x, y, z) we only know whether $d(x, y) \leq d(x, z)$ is true or not, but we do not know either the true representation of the data or the distance values. The authors design two different versions of the methodology: one for classification and the other for regression. A Comparison-based Random Tree for classification operates analogously to the already described techniques. Each split of a node n is characterized by two prototype objects P_L and P_R, ideally one per class, chosen among the objects in n; likewise the other techniques an object x goes to n_L if closer to P_L and analogously it holds for n_R. Differently from the previous techniques, there is no optimization concerning the choice of the prototypes. The regression version works differently only with respect to the selection of the pivots: they are chosen randomly without looking at the labels, i.e. in an unsupervised way.

These techniques have been shown to be successful but are devoted only to classification and regression, and to our knowledge no methodology exists for outlier detection that works in a similar fashion. In other words, there exists no Random Forest-based outlier detection approach able to deal with all types of non-vectorial data. To this aim, the following section describes thoroughly our contribution: Proximity Isolation Forests.

3.3 Proximity Isolation Forests

The core principle of the proposed approach is that in input we only have pairwise distances. The contents are organized as follows: first we describe thoroughly how to build a Proximity Isolation Tree, the base classifier of Proximity Isolation Forests; then we describe the ensemble, and finally we present the testing function, i.e. the computation of the anomaly score.

3.3.1 Proximity Isolation Trees

Before describing the methodology, let us define (or recall) the notation:

- $\mathcal{F}$ is a Proximity Isolation Forest of $T = |\mathcal{F}|$ Proximity Isolation Trees. Each tree is denoted as t.
- $\mathcal{O}$ is the training set used to build $\mathcal{F}$ and has $|\mathcal{O}|$ objects.
- $\mathcal{O}^t$ is the training set used to build t and has $S = |\mathcal{O}^t|$ objects.
- $\mathbf{D}$ is the distance matrix containing all pairwise distances $d(i, j)$ between each pair (i, j) of objects in $\mathcal{O}$.
- $\mathbf{D}^t$ is the distance matrix computed on $\mathcal{O}^t$.

- Each node in t is denoted as n and if it is not a leaf it has two children: n_L the left one, and n_R, the right one.
- The size of a node n is $|n|$, and it corresponds to the number of training objects used to build t that have reached node n.
- $\mathbf{D}_n^t$ is the distance matrix containing the pairwise distances between all objects in n in a tree t.

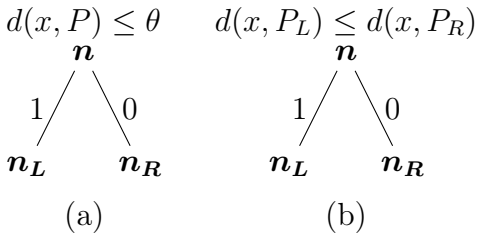
A Proximity Isolation Tree t, or *ProxIT*, is an unsupervised top-down recursively built Decision Tree defined on $\mathcal{O}^t$, more precisely on the respective distance matrix $\mathbf{D}^t$.

To define a ProxIT we first have to define how an object x traverses it. Generally, an object x traverses a Decision Tree starting from the root and following a path determined by the answer to specific questions, until a leaf is reached. As to the traversal modalities in a ProxIT, we define two:

1. In the first one for an internal node n, we have one prototype P and a threshold θ. If the distance of an object x to the candidate prototype P is smaller or equal than θ, then x will end up in the left child node, otherwise in the right one. This is summarized in Figure 3.1 (a).
2. Instead, in the second modality for the same node n we have two prototypes P_L and P_R. If the object x is closer to the former than it will end up into the left node n_L, otherwise into the right one, n_R. See Figure 3.1 (b).

The traversal modality is fixed for all nodes in a ProxIT and for all ProxITs in a ProxIForest.

Figure 3.1: Traversal modalities in a ProxIT: (a) one prototype, (b) two prototypes.



Having defined how objects traverse a ProxIT, which tell us indirectly how objects are partitioned, we can describe the tree building procedure. The building procedure of a ProxIT t is recursive: each node n is split into two children until a stopping criterion is met. When that happens, n becomes a leaf node. Differently from methods described in Chapter 2, as input data we do not have vectors whereas a distance matrix containing all pairwise distances.

Having pairwise distances as input leads to a novel training procedure. In detail, we designed nine training strategies or criteria, which we can categorize

in different ways depending on the adopted perspective. Indeed, if we consider the traversal modalities previously defined we can divide these criteria into two classes:

- **One prototype (1P) criteria**: we define 4 training strategies where the test of a node n is defined by one prototype object P, chosen among the objects in n, and a threshold θ on the distance value. These criteria partition the objects such that the child nodes are created as follows:

  $$n_L = \{x | x \in n \wedge d(x, P) \leq \theta\}$$

  and

  $$n_R = \{x | x \in n \wedge d(x, P) > \theta\}.$$

  In other words, objects are assigned to the left or right child depending on whether their distance to the prototype is smaller or greater than θ. Lastly, each possible pair (P, θ) splits the node n differently into two child nodes n_L and n_R.

- **Two Prototypes (2P) criteria:** we design 5 training strategies where a node n is characterized by two prototype objects P_L and P_R, representing respectively n_L and n_R. These training strategies partition the objects such that n_L and n_R are respectively created as follows:

  $$n_L = \{x | x \in n \wedge d(x, P_L) \leq d(x, P_R)\}$$

  and

  $$n_R = \{x | x \in n \wedge d(x, P_L) > d(x, P_R)\},$$

  i.e. each object is assigned to the child whose representative prototype is closer. Lastly each possible pair (P_L, P_R) determines a different split of node n, i.e. two different child nodes n_L and n_R[1].

In the following, we present in detail each training criterion, grouped based on their core principle. In detail, we distinguish them into:

1. *Random Criteria.*
2. *Scatter-based Criteria.*
3. *Separation-based Criteria.*
4. *Information Theoretic Criteria.*

Aside to the first group of training algorithms, as the name suggests, all remaining criteria are based on the optimization of an impurity-like function. Before delving into the definition of each strategy, let us clarify some concepts common to all *optimized* training schemes:

[1]Please note that in order not to burden the notation, and differently from Chapter 2, we do not indicate in the notation of the child nodes n_L and n_R, the pair (P_L, P_R) (or (P, θ)) that led to their creation, i.e. we do not write $n_L^{P_L, P_R}$ but simply n_L.

- For a node n we choose to evaluate only r among all the possible pairs (P, θ) (or (P_L, P_R)), since it would not be feasible in computing terms –in detail that would take $\mathcal{O}(|n|^2)$. In addition, we observed that the performance tends to reach a pivot after a certain amount of pairs is evaluated, thus supporting our choice.

- If a node n has a number of objects $|n| < r$, then we evaluate all possible pairs.

- If the strategy is $1P$, each of the r pairs of prototype and threshold (P, θ) is chosen as follows: randomly pick $P \in n$ and randomly pick $\theta \in \{d(x, P) | x \in n\}$, where the latter is the set of distance values of the objects in n to P.

- If the criteria is $2P$, each of the r pairs of prototypes (P_L, P_R) is chosen as follows: pick randomly the two prototypes such that $P_L \neq P_R$.

Random Criteria

We design two random training strategies, inspired by the success of the original Isolation Forests in which isolation is implemented by choosing completely at random both the feature and the cut-value along which to split a node n.

1. **R-1P**: In this training strategy, we *randomly* choose one prototype $\hat{P}$ among the objects contained in node n. The threshold $\hat{\theta}$ is subsequently picked, randomly as well, in the range of distances to $\hat{P}$:

$$[\min_{x \in n} d(x, \hat{P}), \max_{x \in n} d(x, \hat{P})].$$

   The last step consists in creating the child nodes n_L and n_R as previously described: the left node will contain the objects closer to the prototype and the right one those that are more dissimilar.

2. **R-2P**: This criterion *randomly* pick two different objects among the ones in n as prototypes $\hat{P}_L$ and $\hat{P}_R$. By splitting the node each object is assigned to the child whose representative prototype is closer.

Scatter-based Criteria

In detail, the intuition behind the following three criteria is that in a data distribution composed of inliers and contaminated by outliers, the variance is very different from the variance of the same distribution where outliers are absent. Therefore, when a split of a node in a tree is able to isolate outliers, we expect a drastic reduction in variance of both the left and the right child. In order to embed this feature in a training criterion, we must formally define some fundamental concepts:

(i) Breiman [17] defined the *misclassification cost* as a measure to find the test of a node n which reduces the most the misclassification rate. Such measure can be computed using an impurity function $I()$. The definition of the misclassification cost of splitting n and the related optimization functions have been already thoroughly defined in Eqs. (2.1), (2.2), (2.3) of Chapter 2. Briefly, we aim at optimizing a function that encodes how good are the child nodes n_L and n_R determined by the test under evaluation. The goodness, for classification and regression, is measured in terms of *purity* and exploits labels. In this context, we are in an unsupervised scenario, and we aim at isolating outliers earlier than inliers.

(ii) Since our input data are pairwise distances, we cannot measure data variance, for which a vectorial representation is needed. Instead, we compute the *scatter*, a measure closely related to the variance, which is the sparseness of the distance values [148]. In detail, we provide two different definitions of scatter, which we call *ScatterD* and *ScatterP*. Consider a matrix $\mathbf{D}$ of size $N \times N$ containing pairwise distances, we define *ScatterD* as the average dispersion across all distances, i.e. first for each object in $\mathbf{D}$ we compute the average distance with respect to all other objects, and then we average the obtained values. Formally:

$$S_D(\mathbf{D}) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} d(i,j). \tag{3.1}$$

ScatterP is defined on a prototype object P and we formally define it as:

$$S_P(\mathbf{D}, P) = \frac{1}{N} \sum_{i=1}^{N} d(i,P). \tag{3.2}$$

In other words, it is the average dispersion of the distance of each object to the prototype P, i.e. for each object we consider only one distance value.

Having defined these concepts, we can employ the scatter as impurity function I. More specifically, the scatter is measured within a node n and therefore with respect to the related distance matrix. Additionally, if the second definition is employed, it can be computed with respect to a prototype, which is either the prototype that led to the creation of the node or the prototype defining how the node is going to be split. In detail, we define two functions to compute the impurity with respect to a specific choice of (P, θ) (or (P_L, P_R)), which defines a specific split, i.e. a triplet of nodes (n, n_L, n_R). These functions will then be optimized to find the best possible pair of prototype and threshold $(\hat{P}, \hat{\theta})$ in case of 1P strategies, or left and right prototype $(\hat{P}_L, \hat{P}_R)$ in case of 2P training criteria. The first function employs *ScatterD* as impurity function:

$$\Delta I_{S_D}(n, n_L, n_R) = S_D(\mathbf{D}_n) - p_L S_D(\mathbf{D}_{n_L}) - p_R S_D(\mathbf{D}_{n_R}) \tag{3.3}$$

where $\mathbf{D}_n$, $\mathbf{D}_{n_L}$ and $\mathbf{D}_{n_R}$ are the distance matrices defined over the objects of n,n_L and n_R respectively. The function measures the difference in scatter between a node and its children: we expect the scatter to decrease as outliers gets isolated and therefore the difference to increase. Thus, as to the optimization step, this function has to be maximized over all possible pairs, but since the scatter in n is independent of how the node itself is going to be split, we can minimize it by rewriting the function as:

$$I_{S_D}(n_L, n_R) = p_L S_D(\mathbf{D}_{n_L}) + p_R S_D(\mathbf{D}_{n_R}). \tag{3.4}$$

The second function employs S_P as an impurity function, and it is defined as:

$$\begin{aligned}\Delta I_{S_P}(n, n_L, n_R, P_L, P_R) = \frac{1}{2}(S_P(\mathbf{D}_n, P_L) + S_P(\mathbf{D}_n, P_R)) \\ -p_L S_P(\mathbf{D}_{n_L}, P_L) - p_R S_P(\mathbf{D}_{n_R}, P_R).\end{aligned} \tag{3.5}$$

Analogously to the first function, it measures the difference in terms of scatter between the node n and its putative children. In this case the node n must be taken into consideration in the computation since *ScatterP* depends on the selected prototypes, and the r candidate partitions of node n are linked to r different pairs (P_L, P_R).

By employing these functions, we can define three training criteria.

3. **O-1PS$_D$** This criterion evaluates each pair of prototype and threshold using Eq. (3.4) and the best pair $(\hat{P}, \hat{\theta})$ is selected by optimizing the following:

$$(\hat{P}, \hat{\theta}) = \underset{(P,\theta)}{\operatorname{argmin}} I_{S_D}(n_L, n_R). \tag{3.6}$$

   In other words, O-1PS$_D$ aims at finding the pair that minimizes the scatter of the two distance matrices related to the child nodes, $\mathbf{D}^t_{n_L}$ and $\mathbf{D}^t_{n_R}$.

4. **O-2PS$_D$**: The evaluation of each pair is done using Eq. (3.4) and the best pair of prototypes $(\hat{P}_L, \hat{P}_R)$ is:

$$(\hat{P}_L, \hat{P}_R) = \underset{(P_L,P_R)}{\operatorname{argmin}} I_{S_D}(n_L, n_R). \tag{3.7}$$

   Analogously to O-1PS$_D$, this training strategy aims at identifying the best pair $(\hat{P}_L, \hat{P}_R)$ minimizing the total scatter within the child nodes.

5. **O-2PS$_P$**: Using this criterion, the evaluation of each pair is done using Eq. (3.5) and the best pair of prototypes $(\hat{P}_L, \hat{P}_R)$ is found by:

$$(\hat{P}_L, \hat{P}_R) = \underset{(P_L,P_R)}{\operatorname{argmax}} \Delta I_{S_P}(n, n_L, n_R, P_L, P_R). \tag{3.8}$$

   For this criterion the extent of change of dispersion is computed considering only the distances to the evaluated prototypes. As explained

before, since the scatter in n is not constant, and thus being different for each pair (P_L, P_R), it must be considered in the optimization function.

We do not define the strategy *O-1PS*$_P$, that is the one that employs S_P and is defined by only one prototype P, since it is not meaningful. Indeed, when a node n is split, each object in n ends up in either the left or the right child of n. Specifically, only one between n_L and n_R will contain P, and therefore if $P \in n_L$ we can compute the term $S_P(\mathbf{D}_{n_L}, P)$ and vice-versa if $P \in n_R$. Nevertheless, computing the other term, i.e. $S_P(\mathbf{D}_{n_R \cup P}, P)$, is not meaningful since P is not contained in $\mathbf{D}_{n_R}$.

Algorithms 3.1 and 3.2 summarize the procedure to choose respectively the best pair $(\hat{P}, \hat{\theta})$ and $(\hat{P}_L, \hat{P}_R)$ for a node n. Please note that these Algorithms are valid also for the other training strategies that we have yet to define, since *EvaluatePair* is a generic function working for all optimized strategies.

Separation-based Criteria

The core idea of the following two criteria is that a good pair (P, θ) (or (P_L, P_R)) generates two children which are well separated, i.e. ideally one of the two children contains only outliers.

To measure the separation, we use the Hausdorff distance, which is used to measure the degree of resemblance between two sets, such as two images [71]. To compute such value, the Hausdorff distance makes use of pairwise distances between a generic a and b, where a belongs to the first set A and b to the other set B. In addition, it does not make any assumption on the distance function, making its use feasible also when dealing with non-metric distance measures. In detail, in [71] they define the *directed* Hausdorff Distance from a set of objects $A = \{a_1, a_2, ..., a_N\}$ to a set of objects $B = \{b_1, b_2, ..., b_M\}$ as:

$$HD(A, B) = \max_{a \in A} \min_{b \in B} d(a, b) \tag{3.9}$$

where $d(a, b)$ is the pairwise distance between a and b. In other words, it measures how distant is the farthest nearest neighbor in B considering all objects in A: if the value is small it means that all objects of A are rather close to those of B, making the sets similar.

This concept is easily transferable to our problem: the two sets are the putative child nodes n_L, n_R determined by either (P, θ) or (P_L, P_R). In other words, $A = n_L$ and $B = n_R$ and therefore $HD(n_L, n_R)$ measures how much and how well the two nodes are separated from n_L point of view. To make a fair measurement, we need to compute the degree of separation also from n_R point of view. Lastly, we can average the obtained values to obtain an average separation degree of the two nodes:

$$HDA(n_L, n_R) = \frac{1}{2} \left(\max_{l \in n_L} \min_{r \in n_R} d(l, r) + \max_{r \in n_R} \min_{l \in n_L} d(r, l) \right). \tag{3.10}$$

Starting from Equation (3.10), we can define two novel training strategies:

Algorithm 3.1 ProxIT, OnePrototype

1: **procedure** OnePrototypeLearning($\mathbf{D}, criterion, r$)
2: **Input: D** is a distance matrix, *criterion* contains information related to the training strategy and r is the number of splits to evaluate.
3: **Output**: $(\hat{P}, \hat{\theta})$ are respectively the selected prototype and related threshold. Left (right) is a vector containing the indices of the objects in **D** that go to n_L (n_R).
4:
5: $S \leftarrow$ number of objects in **D**
6: **if** criterion.name=R-1P **then**
7: $\hat{P} \leftarrow$Randomly pick one value in $1 \dots S$
8: $\hat{\theta} \leftarrow$ Randomly pick one value in $[\min_{i=1\dots S \wedge i \neq \hat{P}} d(i, \hat{P}), \max_{i=1\dots S \wedge i \neq P} d(i, \hat{P})]$
9: $left \leftarrow \{i | i = 1 \dots S \wedge d(i, \hat{P}) \leq \hat{\theta}\}$
10: $right \leftarrow \{i | i = 1 \dots S \wedge d(i, \hat{P}) > \hat{\theta}\}$
11: **else** ▷ Optimized criteria require the evaluation of r pairs.
12: $(\mathbf{P}, \boldsymbol{\theta}) \leftarrow$ Pick randomly $k = 1 \dots r$ different pairs (P_k, θ_k)
13: where P_k in $1 \dots S$ and $\theta_k = d(P_k, j)$ where $j = 1 \dots S \wedge j \neq P_k$
14: **for** k=1... r **do**
15: $left_{tmp} \leftarrow \{i | i = 1 \dots S \wedge d(i, P_k) \leq \theta_k\}$
16: $right_{tmp} \leftarrow \{i | i = 1 \dots S \wedge d(i, P_k) > \theta_k\}$
17: $I_{tmp} \leftarrow EvaluatePair(\mathbf{D}, left_{tmp}, right_{tmp}, criterion)$
18: **if** criterion.name=O-1PS_D **then**
19: $I_{tmp} \leftarrow -I_{tmp}$
20: **if** $I_{tmp} > I$ **then**
21: $\hat{P} \leftarrow P_k$, $\hat{\theta} \leftarrow \theta_k$
22: $left \leftarrow left_{tmp}$, $right \leftarrow right_{tmp}$
23: **return** $\hat{P}, \hat{\theta}, left, right$
24:
25: **EvaluatePair* returns the value of the function to optimize, which depends on *criterion*. Please note that Line 20 is needed since **O-1P**S_D is a minimization problem whereas all other optimized criteria are maximization ones.

6. **O-1PH**: The evaluation of each pair is done using Eq. (3.10) and the best pair of prototype and threshold $(\hat{P}, \hat{\theta})$ is found as:

$$(\hat{P}, \hat{\theta}) = \underset{(P,\theta)}{\operatorname{argmax}} HDA(n_L, n_R). \tag{3.11}$$

The optimization is encoded via a maximization problem: we aim at finding the pair which defined split creates maximally separated children.

Algorithm 3.2 ProxIT, TwoPrototypes

1: **procedure** TwoPrototypesLearning($\mathbf{D}, criterion, r$)
2: **Input:** $\mathbf{D}$ is a distance matrix, *criterion* contains information related to the training strategy and r is the number of splits to evaluate.
3: **Output**: $(\hat{P}_L, \hat{P}_R)$ are respectively the left and right selected prototypes. Left (right) is a vector containing the indices of the objects in $\mathbf{D}$ that go to n_L (n_R).
4:
5: $S \leftarrow$ number of objects in $\mathbf{D}$
6: **if** criterion.name=R-2P **then**
7: $\hat{P}_L \leftarrow$Randomly pick one value in $1 \dots S$
8: $\hat{P}_R \leftarrow$Randomly pick one value in $1 \dots S \setminus \hat{P}_L$
9: $left \leftarrow \{i | i = 1 \dots N \wedge d(i, P_L) \leq d(i, P_R)\}$
10: $right \leftarrow \{i | i = 1 \dots N \wedge d(i, P_L) > d(i, P_R)\}$
11: **else** ▷ Optimized criteria require the evaluation of r pairs.
12: $(\mathbf{P_L}, \mathbf{P_R}) \leftarrow$ Pick randomly $k = 1 \dots r$ different pairs (P_{Lk}, P_{Rk}) where
13: P_{Lk} in $1 \dots S$ and P_{Rk} in $1 \dots S \wedge P_{Lk} \neq P_{Rk}$
14: **for** $k = 1 \dots r$ **do**
15: $left_{tmp} \leftarrow \{i | i = 1 \dots S \wedge d(i, P_{Lk}) \leq d(i, P_{Rk})\}$
16: $right_{tmp} \leftarrow \{i | i = 1 \dots S \wedge d(i, P_{Lk}) > d(i, P_{Rk})\}$
17: $I_{tmp} \leftarrow EvaluatePair(\mathbf{D}, left_{tmp}, right_{tmp}, criterion.name)*$
18: **if** criterion.name=O-2PS_D **then**
19: $I_{tmp} \leftarrow -I_{tmp}$
20: **if** $I_{tmp} > I$ **then**
21: $\hat{P}_L \leftarrow P_{Lk}$, $\hat{P}_R \leftarrow P_{Rk}$
22: $left \leftarrow left_{tmp}$, $right \leftarrow right_{tmp}$
23: **return** $P_L, P_R, left, right$
24:
25: **EvaluatePair* returns the value of the function to optimize, which depends on *criterion*. Please note that at Line 23 the $-I_{tmp}$ is needed since **O-2P**S_D is a minimization problem whereas all other optimized criteria are maximization ones.

7. **O-2PH**: This criterion evaluates each pair of prototypes (P_L, P_R) using Eq. (3.10) and the best pair is:

$$(\hat{P}_L, \hat{P}_R) = \underset{(P_L, P_R)}{\operatorname{argmax}} HDA(n_L, n_R). \tag{3.12}$$

This training strategy aims at finding the pair of prototypes $(\hat{P}_L, \hat{P}_R)$ which leads to the highest achievable separation of the child nodes.

Information Theoretic Criteria

Since many years, information theoretic concepts have been widely studied and

employed in the field of Pattern Recognition [49]. Further, very often RF for classification and regression use as impurity function an information theoretic measure [17, 19]. The most classical example is the Shannon entropy, which is the amount of information contained in a set of data. In detail, as mentioned in Chapter 2, this is done by considering the labels of the objects as the data to analyze, and therefore the measured entropy, is the entropy of the classes.

Information theoretic concepts may be quite useful also in the contexts of outlier detection and specifically for isolation-based RF approaches; therefore, we decided to investigate their use within a training strategy, inspired by the work done in [13]. In detail, in [13] in the context of RF for clustering, they design a training strategy based on the Rényi divergence. This function measures how different are the child nodes n_L and n_R in terms of information, and therefore how much *information* we gain by splitting n [49]. Therefore, a higher difference means a better partitioning of the nodes. This is meaningful also in the context of isolation-based outlier detection since two child nodes n_L and n_R contain the highest amount of different information, ideally when one of the two children isolates an outlier. Based on this idea, we define two novel training criteria. In the following, we describe some concepts necessary to understand how we derived the optimization function based on the estimation of the Rényi divergence, prior to defining the training criteria.

- In general, the term *entropy* can have different (but highly related) meanings depending on the context. In information theory, hence the name, when measuring the entropy of a process we are measuring the amount of information that it conveys [33]. Here we are interested in Rényi's entropy, a generalization of several entropy measures that have been defined in the literature, including the classic Shannon entropy [139]. Formally, we define the Rényi's entropy of order α for a probability density function f measured on a random variable z as [49, 131]:

$$H_\alpha(f) = -\frac{1}{1-\alpha}\log_2 \int_z f^\alpha(z)dz \tag{3.13}$$

  where $\log_2$ indicates that we measure the information in bits and $\alpha = [0,1) \cup (1,\infty)$. Please note that as α changes also $H_\alpha(f)$ does, and it assumes different meanings. In particular, please note that for $\alpha \to 1$ the Rényi's entropy converges to the Shannon entropy, the most spread and used entropy measure.

- Given this notion, we can define the Rényi divergence, which measures how much a probability density function g diverges from a probability density function f in terms of different information. Formally, we define the Rényi divergence of order α of g from f as:

$$RD_\alpha(f||g) = \frac{1}{\alpha-1}\log_2 \int g(z)\left(\frac{f(z)}{g(z)}\right)^\alpha dz. \tag{3.14}$$

Please note that this measure is not symmetric, i.e. the divergence of f from g is not necessarily equal to the divergence of g from f. In addition, if $\alpha \to 1$ the Rényi's divergence converges to the Kullerback-Leier divergence, which is measured using the Shannon entropy.

- A K-Nearest Neighbor graph (KNNG) is built on a set of objects $A = \{a_1, \ldots, a_N\}$ and it encodes the relation of each $a_i \in A$ with its first K-Nearest Neighbors (KNNs). In detail, each object is represented by a node, and we have a directed edge (a_i, a_j) where $i \neq j$ if a_j is among the KNNs of a_i. In other words, the edge is present if $d(a_i, a_j) < d_K(a_i)$ where with the term $d_K(a_i)$ we indicate the distance value of the KNN of a_i.

Since the computation of information theoretic measures is usually unfeasible in practice, estimations are used. We recall that we want to estimate the divergence between two distributions using the Rényi divergence as defined in Eq. (3.14). However, a more classical measure could be estimated, the Kullerback-Leier (KL) divergence: nevertheless, its estimation is rather complex and often leads to non robust estimates [120]. Instead, it is easier to estimate the Rényi divergence, thus supporting our choice. On a rather different note, but still related to the topic, recently in [105] they have shown the superiority of Rényi with respect to Shannon entropy in measuring the impurity of a node of a DT, which can be considered as another supporting factor.

In detail, quite recently non-parametric *bypass* estimators of Rényi's entropy have been designed [120]: a bypass estimator estimates the entropy directly from the samples instead of using the probability density function. The estimator used in [120] is based on the KNN graph built in the following way: the Euclidean distance is computed between each pair of objects of both distributions and then the K-Nearest Neighbors are identified. In [118] the authors propose a similar estimator also for Rényi's divergence which main feature is that it can provide reliable estimates independently of the used distance measure, which does not have to be a metric. Given two sets of objects $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_M\}$ the estimation procedure of the Rényi divergence of A from B is done as follows [118]:

1. Build the KNNG of B in the joint set $C = A \cup B$. The first step consists of finding the KNNs of each object $b \in B$ within the set C. Then represent each sample $c \in C$ as a node and create a directed edge (c_i, c_j) where $c_i \neq c_j \wedge c_i \in B \wedge c_j \in C$. In other words, we are interested only in the neighbors of B.

2. Given the KNNG, define N_i as the number of objects in A that are KNNs of $b_i \in B$. Analogously, define M_i as the number of objects in B that are neighbors of $b_i \in B$. Lastly, define $\eta = \frac{M}{N}$.

3. The estimation of the Rényi divergence of order α of set B from set A is computed as follows:

$$RD(A,B) = \frac{1}{\alpha - 1} \log \left[\frac{\eta^\alpha}{M} \sum_{i=1}^{M} (\frac{N_i}{M_i + 1})^\alpha \right]. \tag{3.15}$$

In other words, the Rényi divergence can be estimated by using only the information related to the K-Nearest Neighbors of $b \in B$ computed in the joint set of objects of A and B –for further details, see [118].

Analogously to the Hausdorff-based criteria, we can exploit this concept in our framework by considering the two sets as the two child nodes, n_L and n_R. Since the measure is not symmetric, to measure the amount of different information provided by the pair (P, θ) (or (P_L, P_R)) that led to the creation of n_L and n_R, we have to estimate the divergence using the KNNG of both children. Formally, we define as RDA the average estimation of the Rényi's divergence between two nodes as:

$$RDA(n_L, n_R) = \frac{1}{2}(RD(n_L, n_R) + RD(n_R, n_L)). \tag{3.16}$$

We aim at maximizing this function since the best pair (P, θ) (or (P_L, P_R)) is the one generating nodes which divergence is maximum, i.e. which are well separated and convey different information.

From this dissertation we can derive the last two optimized training strategies:

8. **O-1PRD**: The evaluation of each pair is done using Eq. (3.16) and the best pair of prototype and threshold $(\hat{P}, \hat{\theta})$ is the one solving the following optimization function:

$$(\hat{P}, \hat{\theta}) = \underset{(P,\theta)}{\operatorname{argmax}} RDA(n_L, n_R). \tag{3.17}$$

   This criterion aims at finding the pair $(\hat{P}, \hat{\theta})$ which leads to the highest achievable amount of different information in the two child nodes.

9. **O-2PRD**: Using this criterion, each pair of prototypes is evaluated using Eq. (3.16) and the best pair $(\hat{P}_L, \hat{P}_R)$ is found by:

$$(\hat{P}_L, \hat{P}_R) = \underset{(P_L,P_R)}{\operatorname{argmax}} RDA(n_L, n_R). \tag{3.18}$$

   This training strategy aims at finding the best pair of prototypes $(\hat{P}_L, \hat{P}_R)$ which leads to two nodes conveying very different information.

Please note that both the separation-based and information theoretic criteria have been already successfully used in the clustering field [13].

Up to this point, we have defined and explained several training strategies: nevertheless, this is just one step of the tree building procedure. In Algorithm 3.3 we describe all the steps to build a ProxIT t which input is a distance matrix $\mathbf{D}^t$ of size $S \times S$:

1. A root node is created which contains all S objects.
2. The root is split into two children according to the training criterion.
3. A node n is recursively split according to the training strategy until a stopping criterion is met[2]. When a stopping criterion is met, we create a leaf node.
4. When there is no node that can be split, the tree building procedure ends.

As to Step 3, i.e. the stopping criterion, we defined three. Specifically n is labelled as leaf when: (i) $|n| = 1$, i.e. there is only one object, (ii) the maximum depth D has been reached, (iii) each pair of objects in n is described by the same set of distance values to all objects in n, i.e. $\forall(x, y) \wedge x \neq y$ we have that $\mathbf{D}_n(x, \cdot) = \mathbf{D}_n(y, \cdot)$.

3.3.2 Proximity Isolation Forests

A ProxIForest (ProxIF in short) $\mathcal{F}$ is an ensemble of ProxITs t. As shown in Algorithm 3.4, each ProxIT t of $\mathcal{F}$ is built by randomly subsampling the input dataset, without replacing the objects that have already been employed, i.e. an object can be represented at maximum once in a tree. This randomization step ensures diversity among the trees. Each tree is built independently using the function *ProxIT* presented in Algorithm 3.3.

Anomaly Score Computation Given a built ProxIF $\mathcal{F}$ we can retrieve the anomaly score for any object x by making the object x traverse each ProxIT t. The anomaly score of x, $s(x)$, is computed according to Eq. 2.4 defined in Chapter 2. We briefly recall the idea behind it: an outlier tends to be isolated earlier in a tree, i.e. it likely ends up in a leaf at a small depth. Therefore, the anomaly score of x in t is a function inversely proportional to the depth of the reached leaf, such that a higher score is assigned to outliers. The aggregation at forest level is performed via a function of the averaged tree scores.

The main difference with respect to the standard iForest is that to traverse a node n, we have to follow one of the two traversal modalities defined at the beginning of this section, depending on whether the training strategy used to build t is $1P$ or $2P$. The testing phase of a ProxIF is summarized in Algorithm 3.5.

[2]Please note that the stopping criterion can be satisfied even in the root node, i.e. the ProxIT will be made of one node only.

Algorithm 3.3 ProxIT

procedure PROXIT($\mathbf{D}, D, t, Prototype, r$)
    Input: $\mathbf{D}$ is a distance matrix, D is the maximum depth, t is the tree structure, *criterion* contains information related to the training strategy, r is the number of pairs to evaluate in each node.
    Output: a ProxIT t.

    $t.S \leftarrow$ number of objects in $\mathbf{D}$
    if $t.S = 1 \vee t.depth = D \vee equal(\mathbf{D})$ **then** ▷ Create a leaf
        $t.left \leftarrow []$, $t.right \leftarrow []$,
        return t
    else ▷ Split the current node
        if $criterion.name \in \{R-1P, R-2P\} \wedge r > 1$ **then**
            $r \leftarrow 1$
        if $r > \frac{t.S^2 - t.S}{S}$ **then** ▷ r is bigger than the number of eligible pairs
            $r = \frac{t.S^2 - t.S}{S}$
        if $criterion.prototype = 1P$ **then** ▷ One Prototype Learning
            $\hat{P}, \hat{\theta}, left, right* \leftarrow OnePrototypeLearning(\mathbf{D}, criterion, r)$
            $t.P \leftarrow \hat{P}$, $t.\theta \leftarrow \hat{\theta}$
        else ▷ Two Prototypes Learning
            $\hat{P}_L, \hat{P}_R, left, right \leftarrow TwoPrototypesLearning(\mathbf{D}, criterion, r)$
            $t.P_L \leftarrow \hat{P}_L$, $t.P_R \leftarrow \hat{P}_R$
        $\mathbf{D}_L \leftarrow \mathbf{D}(left, left)$
        $\mathbf{D}_R \leftarrow \mathbf{D}(right, right)$
        $t.left.depth \leftarrow t.depth + 1$
        $t.left \leftarrow$ ProxIT($\mathbf{D}_L, D, t.left, criterion, r$)
        $t.right.depth \leftarrow t.depth + 1$
        $t.right \leftarrow$ ProxIT($\mathbf{D}_R, D, t.right, criterion, r$)

*left (right) is a vector containing the indices of the objects in $\mathbf{D}$ that go to the left (right) child.
equal($\mathbf{D}$) returns true if all objects in $\mathbf{D}$ are described by the same set of distance values.

Algorithm 3.4 ProxIF

procedure PROXIF($\mathbf{D}$,T,S,D,*criterion*)
  Input: $\mathbf{D}$ is a distance matrix, T is the number of trees, S is the size of the training set of each iTree, D is the maximum depth and *criterion* is a variable containing information related to the training strategy.
  Output: $\mathcal{F}$ is the iForest, i.e. an ensemble of T ProxITs.
  $\mathcal{F} = \{\}$
  $N \leftarrow$ number of objects in $\mathbf{D}$
  if $S > N$ **then** $S \leftarrow N$
  for $i = 1 \dots T$ **do**
    $objs \leftarrow$ Subsample S objects from $1 \dots N$ without replacement.
    $\mathbf{D}^i \leftarrow \mathbf{D}(objs, objs)$
    $t \leftarrow []$ ▷ Initialize an empty tree t
    $t.objs \leftarrow objs$ ▷ Save the indexes of the S objects used to build t
    $t.depth \leftarrow 0$ ▷ Set the depth of the root
    $t \leftarrow ProxIT(\mathbf{D}^i, D, t, criterion)$
    $\mathcal{F} \leftarrow \mathcal{F} \bigcup t$
  return $\mathcal{F}$

Algorithm 3.5 ProxIF Testing

procedure ANOMALYSCORE($\mathcal{F}, c, \mathbf{d}$,prototype)
  Input: $\mathcal{F}$ is a trained ProxIF, c is a vector of size S containing the normalization factor defined in Eq. (2.5), $\mathbf{d}$ is the testing object represented via its distances to $\mathcal{O}$ and prototype is {1P,2P}.
  Output: s is the anomaly score of $\mathbf{d}$.
  $\mathbf{h} \leftarrow$ Vector of zeros of length T ▷ Depth of the reached leaf
  for $i = 1 \dots T$ **do**
    $t \leftarrow \mathcal{F}(i)$ ▷ Start the traversal from the root
    $\mathbf{d}_{tmp} \leftarrow \mathbf{d}(t.objs)$ ▷ It contains the distances only to the training objects of t
    while $t.left \neq [] \vee t.right \neq []$ **do**
      if prototype=1P **then** ▷ One prototype training strategy
        if $\mathbf{d}_{tmp}(t.P) \leq t.\theta$ **then**
          $t \leftarrow t.left$
        else ▷ $\mathbf{d}_{tmp}(t.P) > t.\theta$
          $t \leftarrow t.right$
      else ▷ Two prototypes training strategy
        if $\mathbf{d}_{tmp}(t.P_L) \leq \mathbf{d}_{tmp}(t.P_R)$ **then**
          $t \leftarrow t.left$
        else ▷ $\mathbf{d}_{tmp}(t.P_L) > \mathbf{d}_{tmp}(t.P_R)$
          $t \leftarrow t.right$
    $h_i \leftarrow t.depth + c_{t.S}$
  $Eh \leftarrow \frac{\sum_{i=1}^{T} h_i}{T}$
  return $s \leftarrow 2^{-\frac{Eh}{c_S}}$

3.4 Experimental Evaluation

In this section, we present the empirical evaluation of the proposed methodology, ProxIForest. In detail, after describing the datasets and some experimental details in Subsection 3.4.1 we make three different analyses. In Subsection 3.4.2 we present several analyses aimed at assessing the robustness and suitability of the proposed methodology. Then in Subsection 3.4.3 we make a comparison to some state-of-the-art and/or recent techniques. Lastly, in Subsection 3.4.4 we make some considerations on the computational complexity of ProxIF.

3.4.1 Experimental Details

We performed the experiments on ten datasets, all included in the *prdisdata* MATLAB package available at `http://prtools.tudelft.nl/Guide/37Pages/distools.html`. Each of these datasets is a distance matrix, i.e. it encodes distances between all pairs of objects. In other words, the real representation is not known –or at least given.

All datasets are in origin classification problems, which is usual for outlier detection, since it is rather difficult to collect outliers. The procedure we carried out to transform them in outlier detection tasks is rather straightforward and consists of selecting the class with the highest within-scatter –computed on the distance matrix– as the outlier class. We assign the remainder of the data to the inlier class. The applied transformation does not depend on the class prior, and therefore it may happen that the chosen outlier class is numerous. Even though it is an extreme case, and it may not reflect the real situation, it is more than acceptable for several reasons. First, the putative outliers are very dissimilar from one another, since they belong to the class with the highest within-scatter, and thus they are suitable outlier candidates. Another reason is that it is not unusual to have a dataset with a high outlier percentage, indeed datasets born for the task are rare [97]. Lastly, as explained in [25], the percentage of outliers only influences the overall performance of a dataset. Indeed, the outlier percentage does not have any impact in the comparison of several methodologies to discover which, among them, is the most suitable for the problem at hand, thus justifying our pool of datasets.

Table 3.1 describes the main characteristics of the ten datasets, in detail for each dataset we report the size (N), the percentage of outliers ($O\%$) and the distance measure. The used datasets span a wide range of cases: they widely vary in size going from 213 objects up to 10992; in the percentage of outliers, which range is [3.92%-54.74%]; and in the used distance measure, which is never the same. Please note that for several datasets, multiple versions are available; the used version can be inferred from the details provided in the last column of Table 3.1. Two versions of the same dataset may be obtained by employing different distance measures or by extracting a different initial representation, e.g. for Chickenpieces we can approximate the contour of a

two-dimensional blob as a vector in different ways by varying, for example, the final length of the vector.

Table 3.1: Overview of the 10 datasets used for the experimental evaluation.

Dataset	N	% O	Distance Type
BrainMRI	124	51.61%	Euclidean distance between histograms of normalized intensities representing the left amygdala of a subject.
ChickenPieces	446	13.68%	Weighted edit distance between contours of 2D blobs (Norm 5, Cost 45).
CoversSongs	205	8.29%	Dynamic Programming Local Alignment dissimilarity between musical fragments.
DelftGestures	1500	5%	Dynamic Time Warping distances between hand gestures.
DelftPedestrians	689	3.92%	Cloud distances between pedestrians, cars and other objects.
Flowcyto	612	54.74%	L1 distances between flow cytometry histograms (tube 3).
Pendigits	10992	9.60%	Weighted edit distance between handwritten digits.
Protein	213	14.08%	Evolutionary distances between protein sequences.
WoodyPlants	791	7.96%	Shape dissimilarities between plant leaves.
Zongker	2000	10%	Dissimilarities between handwritten digits based on deformable template matching.

After making datasets suitable for outlier detection, several and different experiments were performed by varying the value of several parameters:

1. Training strategies: **R-1P**, **R-2P**, **O-1PS**$_D$, **O-2PS**$_D$, **O-2PS**$_P$, **O-1PH**, **O-2PH**, **O-1PRD** and **O-2PRD**. *9 options*, all described in detail in Section 3.3.

2. Size of the forest, i.e. number of ProxITs, T: 50, 100, 200, 500. *4 options.*

3. Number of objects used to build each tree in a forest S: 64, 128, 256, 512. *4 options.* For those datasets for which the size of the training set is smaller than S we carry out the experiment using all available samples, i.e. no subsampling is performed. An example is a dataset where the training set has size $|\mathcal{O}| = 100$: in this case for $S = 64$ the experiments will be carried out with subsampling, for $S = 128$ the whole training set will be used, while for $S = 256$ we simply report the results obtained for $S = 128$ since the mechanism is identical.

4. Maximum depth: $D : S - 1, \log_2(S)$. *2 options.* The standard choice is to use $\log_2(S)$, which works very well. Nevertheless, in [108] it has been shown that using the maximum possible depth can be fruitful for some datasets and/or in some particular cases.

As to r, the number of evaluated pairs (P, θ) or (P_L, P_R) in each node n, was set to maximum 20. This choice follows an initial evaluation, not shown here, which led to this value being a good compromise. For the experiments carried out using the information theoretic criteria, i.e. *O-1PRD* or *O-2PRD*, we set $K = \sqrt{|n|}$ where $|n|$ is the number of objects in node n and $\alpha = 0.9999$. The former is a standard choice for KNN-based approaches, whereas the latter is often used to make information theoretic measures based on Rényi converge to Shannon based ones [16].

Each combination of parameters has been iterated 10 times, leading to a total –maximum– number of experiments equal to 16000. In detail, each iteration is generated by randomly splitting the dataset in two halves: one for the training set and the other for the testing set. The training set has been constrained to contain a maximum of 5% of outliers. Given a dataset the iterations, i.e. the training and testing objects, are the same for all parametrizations. To measure the performances of the experiments, we adopted the Area Under the ROC Curve (AUC) since, as thoroughly explained in Chapter 2, it is one of the most used performance measures in the outlier detection field.

3.4.2 Experimental Analyses

In this subsection, we make several analyses, each focusing on a different aspect of the proposed approach. Indeed, since we performed a large amount of experiments we try to highlight only the most relevant aspects, whereas the detailed results are reported in Appendix A.

The first three analyses aim at finding the best setting of the following parameters: the size of the tree training set S, the number of trees T in a forest and the maximum depth D to which a tree can be grown. In detail, each analysis is made by two parts: we first analyze how the performances in terms of AUC vary as the value of the parameter changes; then we perform a statistical analysis to assess whether different values lead to statistically different performances. As to the latter point, we employ for the analyses of S and T the Friedman test followed by a post-hoc Nemenyi test; instead as to D we use a Wilcoxon signed-rank test since we have to make a pairwise comparison –we made these choices in accordance to what we explained in Chapter 2. The fourth and last analysis focuses on the training schemes. In detail, we try to understand which strategy is the most suitable in a specific scenario by analyzing their general behaviour across different datasets. Further, we make a global statistical analysis to assess which criteria are comparable and which is the most suitable choice. For all statistical tests, the significance level α has been set to $\alpha = 0.05$.

Impact of S

To properly analyze the behaviour of S we select the following AUC results: given a value of S, dataset and training strategy, we choose the values of T and D for which the performance is the best. In Figures 3.2 and 3.3 we analyze the behaviour of the training sample size S on each strategy.

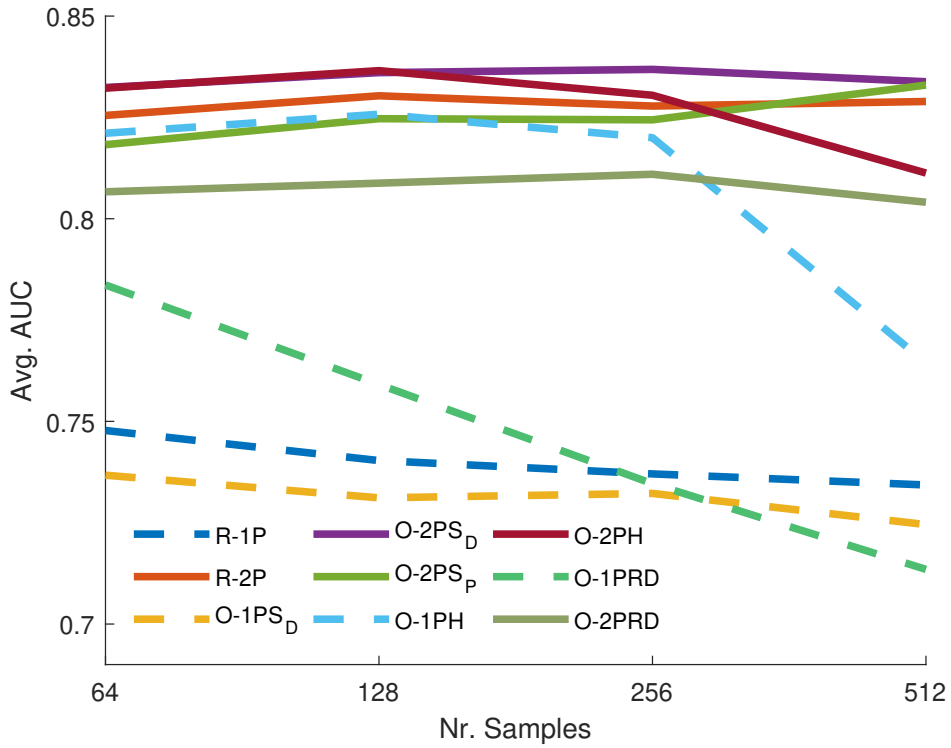


Figure 3.2: Behaviour of the average AUC of the 9 training strategies when changing the training sample size S.

In detail, in Figure 3.2 we plot the averaged AUC values across the datasets and the iterations of the selected parametrization. From Figure 3.2 we can observe that in general performances are robust across several values of S, i.e. if $S \geq 64 \wedge S \leq 256$, with an overall tendency to reach the highest performances for $S = 128$. On the contrary, if we set $S = 512$, i.e. the highest value, we can observe a decline which in some cases it is quite astonishing. As to the training strategy, the best performing and most robust to different values of S is **O-2PS**$_D$. It is also interesting to observe that *O-1PH* shows competitive results when compared to $2P$ training strategies, differently from the other $1P$-based criteria which are globally less performing. For complete results on each dataset, please refer to Table A.1 in Appendix A.

To better understand whether the criteria work significantly better with a certain value of S, we perform a non-parametric statistical analysis which consists of a Friedman Test followed by a post-hoc Nemenyi test [35]. Figure 3.3 depicts the CD diagrams assessing whether, for each training strategy, there

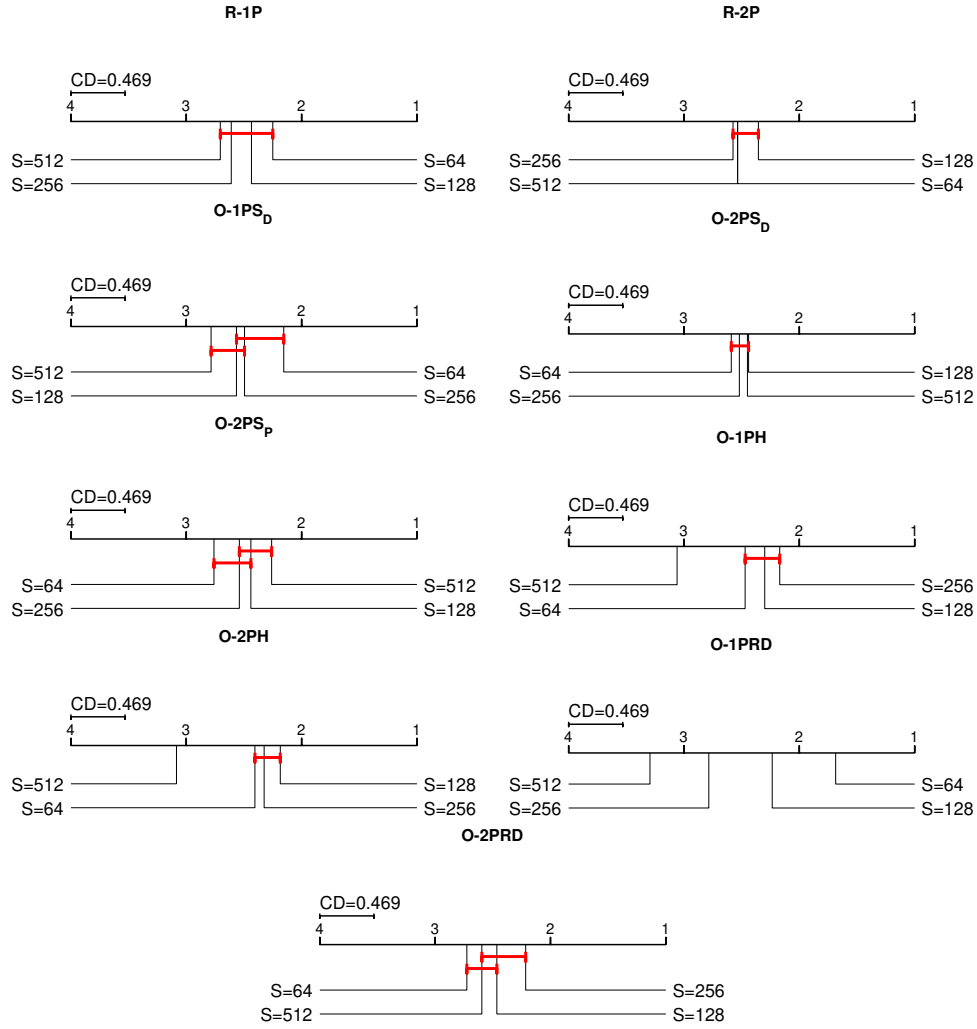


Figure 3.3: CD diagram for each training strategy, comparing the different options for the size of the training set of each tree S.

is a value of S which is significantly better than the rest. We briefly recall that a red line connects two methods if there is no statistically significant difference between them. In general, we can observe that we reach the best performances, as observed also from Figure 3.2, with either $S = 64$ or $S = 128$. In addition, aside from one training strategy ($O - 1PRD$), the best ranked value of S is always comparable to at least the second one in the ranking. Further, equivalent performances are observed for $S = 64, 128$ and $S = 256, 512$, i.e. we can distinguish between *small* and *big* trees. These observations are crucial: not only small trees are quick to train, but they are also very informative and diverse, allowing to reach very good performances. Overall the best choice seems to be $\mathbf{S = 128}$ since it is either: first or second in the ranking, or comparable to the first ranked.

Impact of T

This analysis is aimed at investigating how performances behave as the num-

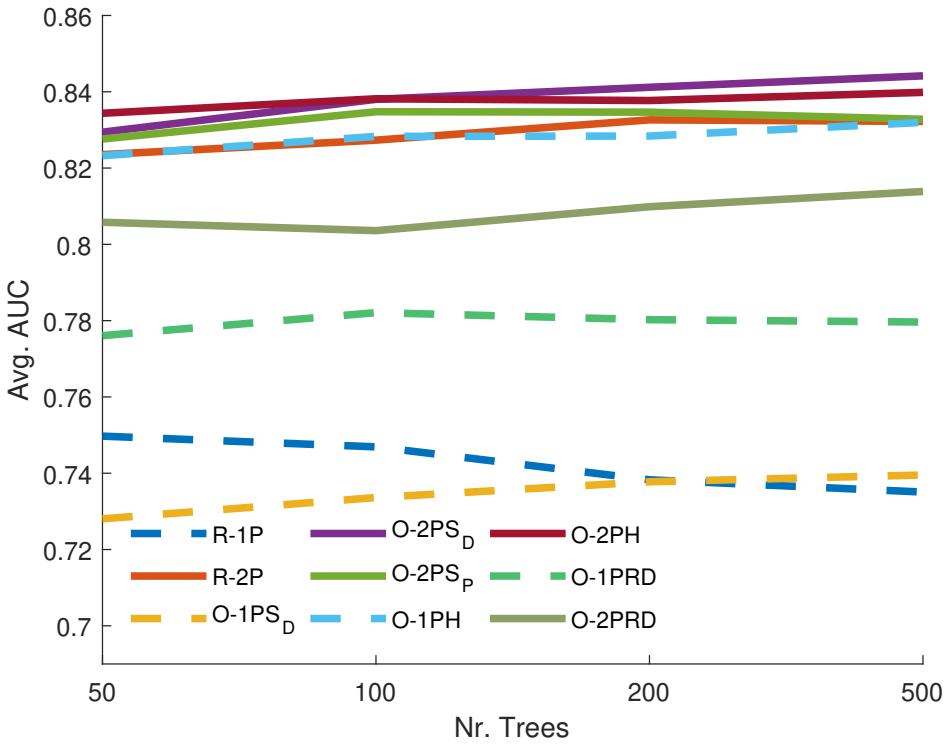


Figure 3.4: Behaviour of the average AUC of the 9 training strategies when changing the forest size T.

ber of trees T changes, and which is the global best choice for the proposed methodology. In Figure 3.4 we analyze the behaviour of T across all training criteria. The plotted AUC values are derived in the same way they were derived for S. A general observation is that the value of T does not seem to have a huge impact on the performances. In detail, we detect two different behaviours: i) some criteria show a slow but steady improvement as T increases; ii) others tend to reach a pivot around $T = 100$. As to the best performing criteria, analogously to what we inferred from Figure 3.2, they are **O-2PS$_D$** and **O-2PH**. For complete results on each dataset, please refer to Table A.2 in Appendix A.

Also in this case, we performed a Friedman test followed by a post-hoc Nemenyi test: we depict the related CD diagram in Figure 3.5. For most criteria, 6 out of 9, changing T does not have a statistically significant impact on T. As to the other three criteria, we can observe that: for *R-1P* it is significantly better to use $T = 50, 100$ while for *O-2PS$_D$* and *O-2PS$_P$* it is significantly worse to use $T = 50$. Putting together the results of Figures 3.4 and 3.5 we can conclude that using **T=100** may be the wisest option: it is

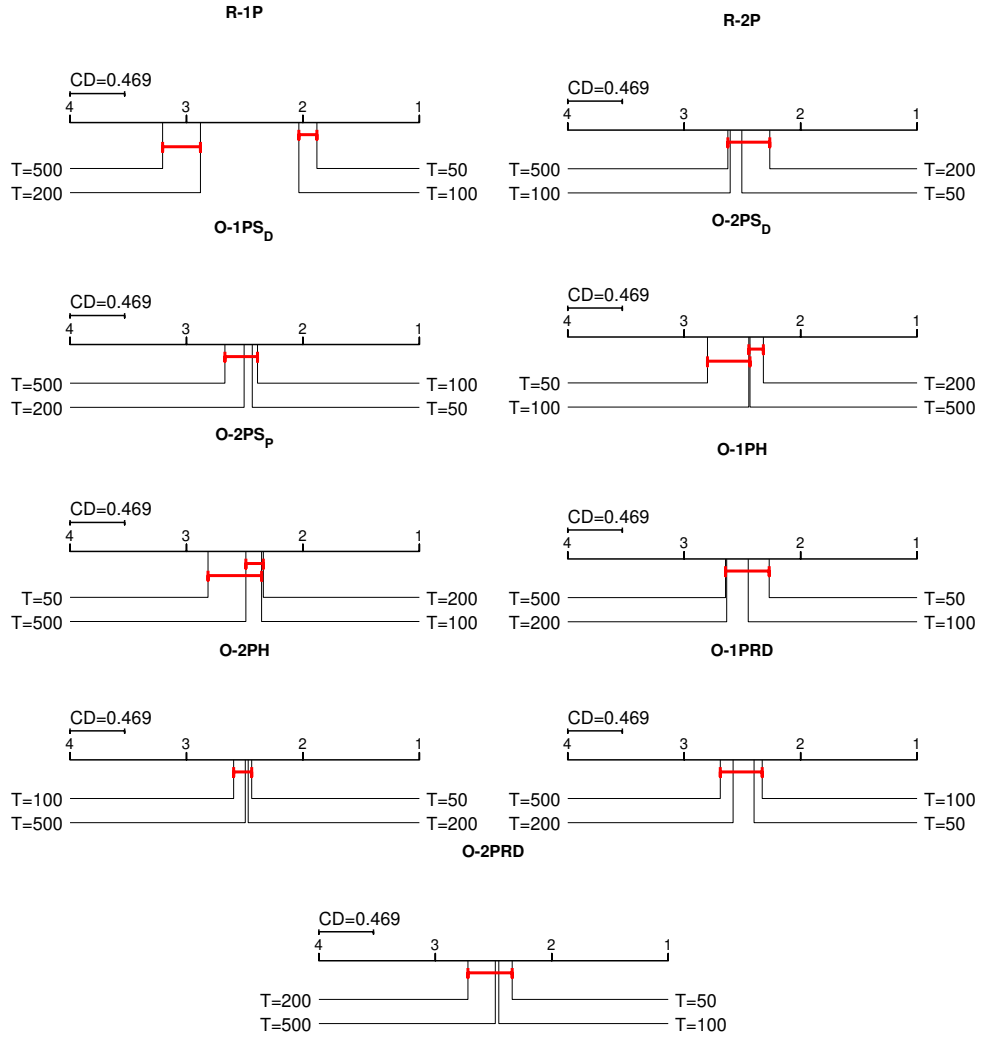


Figure 3.5: CD diagram for each training strategy comparing the different options for forest size T.

always comparable to the first ranked value of T, which is not true for the other values, and in many cases it is one of the best performing options.

Impact of D

The last parameter we focus on is D: the maximum depth allowed for each ProxIT in $\mathcal{F}$. In detail, we studied two values of D, both dependent on S. We analyze the behaviour of D differently from how we did for S and T: indeed, we do not graphically visualize the results since it would be difficult to reason on them. Instead, we analyze the results via Table 3.2: given a value of D and a training strategy, we report the averaged AUC across the datasets and the iterations of a specific parametrization. The latter has been chosen analogously to what was done for S and T. In Table 3.2 we highlight in **bold** the best value of D for each training criterion. In general, we can observe that the variations

in performance are very small. Nevertheless, there is an interesting aspect to highlight: training strategies with one prototype show better performances as D increases, and the contrary holds for criteria using two prototypes. As to the best performing strategy, results are consistent with the previous analyses, with **O-2PS**$_D$ being the best choice. For complete results on each dataset, please refer to Table A.3 in the Appendix A.

As anticipated in the preamble of this section, since D can take only two values, performing a Wilcoxon signed-rank test is a more appropriate choice with respect to Friedman. In detail, we carry out a test for each training strategy comparing the two values of D. In Table 3.3 we report for each strategy the result of the test in terms of p-value, which is highlighted in **bold** if lower than the significance level α. In addition, we report for each value of D the average rank, computed as follows: i) given a dataset we sort the two values of D in terms of AUC, the best one will have rank 1; ii) we repeat this for all datasets and average the obtained ranks. The results presented in Table 3.3

Table 3.2: Behaviour of the average AUC of the 9 training strategies when changing the maximum depth D.

Dataset	$D = \log_2(S)$	$D = S - 1$
R-1P	0.7665	**0.7671**
R-2P	**0.8477**	0.8459
O-1PS$_D$	0.7692	**0.7705**
O-2PS$_D$	**0.8549**	0.8514
O-2PS$_P$	**0.8480**	0.8465
O-1PH	0.8323	**0.8398**
O-2PH	**0.8509**	0.8480
O-1PRD	0.7893	**0.7948**
O-2PRD	**0.8280**	0.8229

Table 3.3: Evaluation of the values of D in terms of mean ranks and p-value obtained from a Wilcoxon signed-rank test.

	Mean Rank		
Dataset	$D = \log_2(S)$	$D = S - 1$	**p-value**
R-1P	1.6	1.4	0.322
R-2P	1.4	1.6	0.375
O-1PS$_D$	1.9	1.1	**0.020**
O-2PS$_D$	1.1	1.9	**0.004**
O-2PS$_P$	1.3	1.7	0.322
O-1PH	1.6	1.4	0.105
O-2PH	1.2	1.8	0.105
O-1PR	1.7	1.3	0.084
O-2PR	1.2	1.8	0.084

confirm that for most training schemes, the choice of D is not statistically relevant in terms of final performances. In detail, setting D is crucial, i.e. a

statistically significant difference is observed, only for $O{-}1PS_D$ and $O{-}2PS_D$– indeed this is visible also from the ranks which are quite different. In general, both choices of D lead to very good results, and neither of them severely outperforms the other. Therefore, we adopt $\mathbf{D} = \log_{\mathbf{2}}(\mathbf{S})$ since $D = S - 1$ is more costly from a computational point of view.

Analysis of the Training Criteria

Table 3.4: Average AUC across all experiments that use the same training strategy.

Dataset	R-1P	R-2P	O-1PS$_D$	O-2PS$_D$	O-2PS$_P$	O-1PH	O-2PH	O-1PRD	O-2PRD
BrainMRI	0.6839	0.6231	**0.7645***	0.6060	0.6434	0.6883	0.6168	0.7434	0.5767
Chickenpieces	0.8145	0.8268	0.7968	0.8301	0.8251	0.8368	**0.8627***	0.7919	0.8126
CoversSongs	0.9898	0.9893	0.9867	0.9897	0.9895	0.9874	0.9896	0.9871	**0.9901**
DelftGestures	0.8261	0.9620	0.7613	0.9423	**0.9625**	0.9082	0.9171	0.8496	0.8838
DelftPedestrians	0.7222	0.7485	0.7227	0.7730	0.7215	0.7593	**0.7839**	0.7683	0.7818
Flowcyto	0.5663	**0.7310***	0.5630	0.7151	0.6947	0.6221	0.6568	0.5209	0.6028
Pendigits	0.6675	0.6903	0.6553	0.7463	0.7289	0.7414	0.7765	0.5947	**0.7852***
Protein	**0.9901***	0.9818	0.9801	0.9758	0.9713	0.9863	0.9806	0.9849	0.9778
Woodyplants	0.4392	0.9062	0.4281	0.9135	**0.9143**	0.7161	0.7843	0.4986	0.7221
Zongker	0.5414	0.7381	0.5487	0.7748	0.7318	0.7122	**0.8400***	0.6130	0.8372

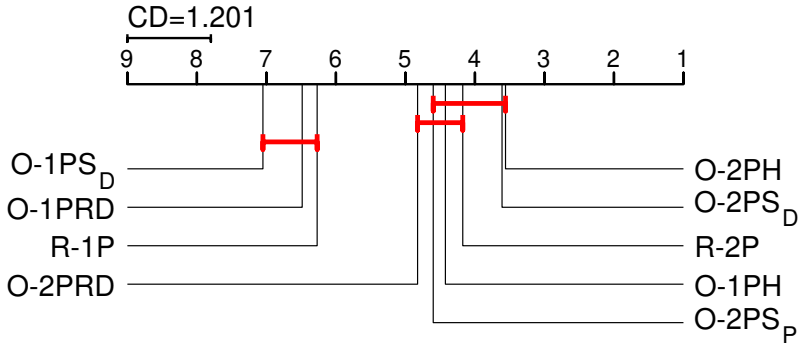


Table 3.5: CD diagram depicting a global comparison of the training strategies.

As briefly explained in the preamble, this analysis focuses on the training strategies. In detail, the aim is to understand in which cases it is beneficial to train a ProxIF with one training strategy (or a set of them) with respect to another. In Table 3.4 we report for each dataset the following results: given a training strategy, we compute the average across all experiments that employ such training strategy[3]. In the table, for each dataset, we highlight in **bold** the best training strategy. Additionally, we assess whether the best strategy for each dataset is significantly better than the second best: to do that, we use a Wilcoxon signed-rank test with $\alpha = 0.05$. If the best strategy, i.e. the one in bold, is significantly better than the second one in the ranking, we put a *

[3]Please note that the results for the criteria using one prototype are different from those published in [109]. This is due to an implementation difference: to choose the threshold, the distance from the prototype itself is left out from the range, something noticeable also in the pseudocodes of the algorithms.

next to it. By observing the results in bold in Table 3.4, we can group most of the datasets by training strategy based on their outlier percentage ($O\%$):

- $O\% < 10\%$: for CoversSongs, DelftGestures, DelftPedestrians, Pendigits and WoodyPlants, the best strategy is either **O-2PS**$_P$ or **O-2PRD**. Both are optimized criteria that use two prototypes.
- $O\% \geq 10\% \wedge O\% \geq 14\%$: the best strategy is **O-2PH**. The dataset that falls into this category are Zongker and Chickenpieces.
- $O\% \geq 14\% \wedge < 50\%$: for this type of datasets, the best choice is to use **R-1P**. Please note that only Protein falls into this category, and therefore the observation must be strengthened by evaluating the methodology on other datasets which fall into this category.
- $O\% \geq 50\%$: this class of datasets is very peculiar, since most objects are outliers. The criterion to use is either **O-1PS**$_D$ or **R-2P**.

Please note that there is only one dataset not fitting these observations: DelftPedestrians. However, if we consider its best strategy, *O-2PH*, we can observe that it is statistically comparable to the second ranked one, which is *O-2PRD*, in agreement with the considerations we have just made.

To assess which strategy leads globally to the best results and whether there is some significant difference in employing one criterion with respect to another, we performed a Friedman test followed by a Nemenyi post-hoc test. The test was performed on a fixed parametrization: $S = 128$, $T = 100$, $D = \log_2(S)$ which parameters have been set following the conclusions of the previous analyses. The CD diagram in Figure 3.5 shows that the best ranked strategy is comparable to almost all other criteria with two prototypes, including the random one. In addition, also confirming previous observations, the strategy *O-1PH* is significantly better than the other criteria with one prototype, and it is comparable to some $2P$ criteria. We can conclude that even though this test does not highlight *one* best criterion, **O-2PH** and **O-2PS**$_D$ are both a very suitable choice, also in light of the previous analyses –however please recall that this last analysis is done on a specific parametrization and by varying the latter, statistically significant differences further supporting our choice may be found.

The analyses in this section have proven that even though some parameters, i.e. S, need to be adequately set in order not to have significantly worse performances, overall the results are very good. Indeed, only for some sporadic parametrizations on few datasets, the performances of ProxIF are not satisfactory. This reasoning also holds for the training strategies: even though many $1P$ criteria may be negatively affected by some parametrizations, they show very bad results only on some datasets. In conclusion, the methodology seems very robust: we can reach very good performances by training small ($\mathbf{S = 128}, \mathbf{D = \log_2(S)}$) and few ($\mathbf{T = 100}$) trees. Further, we have proven

that even though random is a very good choice, specifically designed criteria for outlier detection can be beneficial (such as **O-2PH** and **O-2PS**$_D$).

3.4.3 Comparison to State-of-the-art

In this subsection we compare the proposed technique, ProxIF, to some state-of-the-art density or distance-based outlier detectors: *NNd*, *KNNd*, *KNNd-Av*, *LOF*, *LOF-Range*, *K-Centers* and *RDOF*. We have chosen this pool of techniques because they all rely on distances, and they are either very simple golden standard approaches or they are quite recent techniques which have shown promising results. In the following, we briefly summarize the core idea of each of these techniques:

- *NNd* [146] is an adaptation of the classification algorithm Nearest Neighbor. The idea is to compute a ratio of nearest neighbor distances to assess the probability of an object being an outlier. In detail, given an unseen object x and a set $\mathcal{O}$ of training objects, it compares the distance of x to its nearest neighbor in $\mathcal{O}$ with the distance between the latter and its nearest neighbor in $\mathcal{O}$. If the value of said ratio is much greater than 1 that it is likely that x is an outlier, since it is quite distant from the rest of the data.

- *KNNd* is the generalization of the above to any $K > 1$, i.e. the NN^{th} neighbor is replaced with the K^{th} [146]. *KNNd-av* is a slight variation where the average of the distance from the first K-neighbors is computed; if the value is close to 1 it means that the object under analysis has a similar density to that of its first K neighbors.

- The Local Outlier Factor methodology, known as *LOF* [21], estimates the K-Neighborhood density of an object, i.e. the density of the sphere containing its first K neighbors, via distances. The main concept is to assign to each testing object an anomaly score, which encodes the probability of the object being an outlier. If the density of the neighborhood of the object's neighbor is much denser than that of the object under analysis, then it is likely that the latter is an outlier. A variation of *LOF* is *LOF-Range* [21]: for an object, different neighborhoods are evaluated, i.e. K changes, and the maximum score is taken. Please note that in Table 3.6 we abbreviate LOF-Range to *LOFR*.

- A rather different technique is *K-Centers* which is a clustering method that partitions the objects into K clusters and assigns to each object a score proportional to the distance between the object and the cluster center to which the object belongs. Outliers are objects which are very far from the cluster they belong to, in other words they probably do not belong to the same distribution [146]. The main limitation of the approach is linked to the mathematical properties of the distance measure,

which must be symmetric. Please note that in Table 3.6 we abbreviate K-Centers to *KCent.*.

- Lastly, we compare to a recent methodology called *Relative Density-Based Outlier Detection Algorithm* [117] which computes a Relative Density-Based Outlier Factor, RDOF in short. This methodology, similarly to LOF, looks at the neighborhood of an object x but with an additional constraint: given a maximum size K of the neighborhood, it estimates the density of a relative neighborhood, i.e. a neighborhood composed of only the objects in the K-neighborhood that also have x in their K-neighborhood. The bigger the relative neighborhood and the smaller its radius, the more likely the object is an outlier.

As to NN, there are no parameters to set. Instead, as to the methodologies based on KNN and those on LOF the size of the neighborhood K must be set; after a considerable amount of tests which we do not report, we set the same $K = 5$ for all datasets which resulted in a global good choice; for *LOF-Range* $K = [2, 10]$. With respect to *K-Centers* we set the number of clusters to $K = 3$. Lastly, as to RDOF, we followed the approach adopted by the original paper, which estimates the natural neighborhood size K for a dataset.

Table 3.6: Comparison with state-of-the-art techniques. As to ProxIF we report two results: the one obtained with the guideline parametrization, and between parenthesis, the best result for each dataset.

Dataset	NNd	KNNd	KNNd-Av	LOF	LOFR	KCent.	RDOF	ProxIF
BrainMRI	0.5295	0.5082	0.6705	0.6033	**0.7082**	0.5541	0.6803	0.6377(*0.7738*)
Chickenpieces	0.4618	0.4617	0.4246	0.4334	0.4594	NaN	0.7748	**0.9039**(*0.9048*)
CoversSongs	0.7455	0.9070	0.9313	0.9629	0.9681	0.9630	0.9597	**0.9883**(*0.9922*)
DelftGestures	0.4192	0.4399	0.3881	0.4054	0.4415	0.6432	0.8222	**0.9154**(*0.9758*)
DelftPedestrians	0.5243	0.5674	0.5343	0.5474	0.5814	0.6563	0.7100	**0.7917**(*0.8168*)
Flowcyto	0.4980	0.4476	0.4616	0.4444	0.4695	0.6293	**0.7061**	0.6769(*0.7369*)
Pendigits	0.5046	0.4895	0.4972	0.4794	0.4617	0.5598	0.6222	**0.7895**(*0.7954*)
Protein	0.4134	0.8202	0.7976	0.9000	0.8733	0.8610	0.9614	**0.9800**(*0.9995*)
Woodyplants	0.4507	0.3899	0.3827	0.3559	0.3730	0.7139	**0.8123**	0.7963(*0.9300*)
Zongker	0.5662	0.4756	0.4224	0.5211	0.7287	0.7519	0.7159	**0.8462**(*0.8512*)

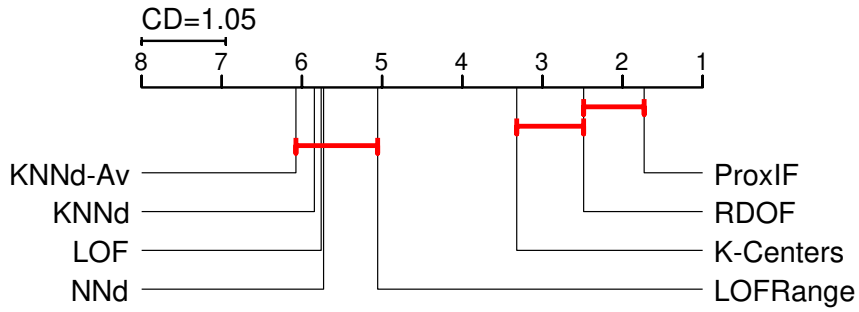


Table 3.7: CD diagram comparing state-of-the-art techniques.

In Table 3.6 we report the AUC averaged across 10 iterations for all methods. For ProxIF the results are obtained using the guideline parametrization

obtained via the analyses in Section 3.4.2: $\mathbf{T = 100}$, $\mathbf{d = \log_2(S)}$, $\mathbf{S = 128}$, **Criterion=O-2PH**. In addition, for the sake of completeness, we report between parenthesis for each dataset the best result achievable with a ProxIF, independently of the parametrization –averaged across the iterations. Further, we highlight in **bold** the best outlier detector for each dataset.

From Table 3.6 we can clearly infer that the proposed technique is very robust, leading to very good results on all datasets. In detail, ProxIF is the best solution for 7 datasets out of 10 and if we tune the parameters based on the input problem, ProxIF would be the best solution for all of them. Nevertheless, it must be highlighted that the performances of the guideline parametrization are very similar to those of the best parametrization for each dataset, confirming the suitability and robustness of our proposal. To further validate the results, we performed a Friedman test followed by a post-hoc Nemenyi test, comparing all the above methods to the guideline parametrization of ProxIF. The CD diagram in Figure 3.7 shows that ProxIF is indeed the first ranked, and it is comparable only to RDOF, a very recent technique. Nevertheless, we must point out that the K in RDOF is tuned for each dataset, which may be the reason why RDOF and ProxIF are statistically comparable and why RDOF reaches on few datasets higher performances than the proposed approach.

Summarizing, we can conclude that ProxIForest is very competitive and robust, often leading to the highest AUC performances. In addition, if we tuned the parameters of ProxIF independently for each dataset, we would probably detect an overall superiority.

3.4.4 Considerations on Complexity

In this section we make some considerations on the time and space complexity of the proposed methodology, and we assess them from an experimental point of view. Please note that our aim is just to give an intuition on the computational characteristics of the method and not to make a classical complexity analysis.

The time complexity of building a ProxIT t depends on the training strategy. Given a node n that we want to split, we can make the following considerations:

- To partition n, independently of the training strategy, we have to sort the objects in the node.
- Using *R-1P, R-2P* does not require any additional operation other than the sorting.
- If the training strategy is optimized, we have to repeat the procedure r times.
- Using *O-1PS*$_D$, *O-2PS*$_D$, *O-2PS*$_P$ requires making an additional operation: computing the mean across either all values in $\mathbf{D}$ or, if *O-2PS*$_P$ is used, on a subset of values.

- When using *O-1PH, O-2PH* for each object in n_L, we have to find the closest object in n_R and vice-versa.

- The training strategies *O-1PRD, O-2PRD* require several operations. First, we have to sort the entire distance matrix to search for the K-Nearest Neighbors. Then for each object, we have to make several comparisons to identify which neighbors belong to n_L and which to n_R.

From these considerations it seems that random criteria are the less costly, followed by scatter-based criteria, separation-based criteria and lastly, the most costly seem to be the training strategies based on Rényi. Please recall that these considerations only refer to the split of a node n, whereas the total training time also depends on the number of times the recursion procedure is repeated, i.e. the number of nodes. Nevertheless, the latter is usually small since we have proved that to obtain a good model, a ProxIT usually requires no more than 128 samples to be built with, independently of the size of the dataset.

Since the testing phase is identical to that of the standard iForest, also the time complexity does not change. As to space complexity, assuming that we do not compute distances during the training phase –which can lead to further overheads in terms of time– the whole distance matrix must be available at all times independently of the used training strategy.

Table 3.8: Average training time in seconds of a ProxIF using different training criteria.

Dataset	**R-2P**	**O-2P**S_D	**O-2P**S_P	**O-2PH**	**O-2PRD**
BrainMRI	0.684s	0.798s	0.780s	0.757s	0.865s
Chickenpieces	1.393s	3.603s	2.922s	3.544s	6.214s
CoversSongs	1.127s	1.245s	1.233s	1.290s	1.448s
DelftGestures	1.458s	3.370s	2.526s	3.585s	6.297s
DelftPedestrians	1.347s	3.491s	2.290s	3.189s	4.564s
Flowcyto	1.422s	3.737s	2.579s	3.458s	5.427s
Pendigits	1.562s	3.837s	2.796s	4.114s	6.825s
Protein	1.071s	1.256s	1.236s	1.241s	1.382s
Woodyplants	1.478s	3.721s	3.378s	3.938s	7.125s
Zongker	1.562s	3.608s	2.617s	3.652s	5.369s
Average	1.310s	2.867s	2.236s	2.877s	4.552s

Considering the above observations, we compare the running time of training a ProxIF with different training criteria. We analyze the training time of a ProxIF when built with one of the following training strategies: *R-2P*, *O-2PS*$_D$, *O-2PS*$_P$, *O-2PH*, and *O-2PRD*. We leave out $1P$ training strategies, since the considerations we made are independent of the number of prototypes. The rest of the parameters are set according to the guideline parametrization, i.e. $T = 100$, $S = 128$, $D = \log_2(S)$. In Table 3.8 we report the training time in seconds averaged across 10 iterations of the training phase.

From the table, we can confirm the truthfulness of the considerations we made above. It is evident that the fastest way to build a tree is by choosing randomly the split whereas the methodology that takes the longest, due to the many operations required for the divergence estimation, is *O-2PRD*, the criterion based on Rényi. We can also observe a difference in running times between *O-2PS*$_D$ and *O-2PS*$_P$: indeed, the former has to compute the mean of up to S^2 values, whereas the latter only up to S. Overall the variations are very small –due to the very small trees– thus supporting the use of optimized criteria, instead of the random ones, since they lead to much better performances. A final note: there may be some relevant differences in running times on the same training strategy when using different datasets. This phenomenon is simply due to some ProxITrees being smaller than the rest; indeed, some datasets have a training set of size smaller than S, e.g. BrainMRI training set is made of 62 objects.

3.5 Conclusions and Future Work

In this chapter we presented a novel outlier detection methodology based on Random Forests, called Proximity Isolation Forest, in short ProxIF, that can work with any type of data for which a distance measure can be defined. This methodology is thus able to work with both vectorial and non-vectorial data, the latter being of particular relevance since representing these objects via distances is a natural and meaningful representation. We designed several criteria to implement the isolation principle: from random, which exploit the distance that outliers have from inliers, to optimized ones, which are based on other principles that naturally allow capturing the separation of outliers from the rest of the data. The robustness of this novel contribution is tested by a large number of experiments performed on ten non-vectorial datasets. A thorough analysis has been made, showing that except very few cases the methodology is robust, and the variations are small. Further, via several analyses, we identified a guideline parametrization that can be used for any dataset with good results. The superiority and goodness of the proposed approach is further shown when compared to other outlier detectors that work with distances.

Other than being a highly suitable and robust outlier detector, ProxIF is also characterized by an easiness of use. In detail, other than the first step, which consists of the choice of a meaningful distance measure for the dataset at hand –and it is therefore highly dependent on the context–, the other phases are rather straightforward. Indeed, as already said, we have provided a default parametrization of ProxIF to use in any scenario, the only exception being the selection of the training criterion whenever the user has some prior knowledge on the outlier percentage, for which we provide additional indications to follow. The last point making ProxIF easy to use in any context, is linked to the interpretability of the anomaly score, inherited from iForest: the output is a score proportional to the probability of the object being an outlier.

As to future research, an interesting idea would be to employ ProxIF to detect outliers on some real-life dataset, e.g. ECG data and see how it performs compared to outlier detectors for time series data, which is a very researched field of study. Another relevant application of ProxIF would be to use it as a feature extractor starting from some distance-based data: we do this in Chapter 6 where we extract from a brain connectivity network a vector encoding the outlierness degree of each ROI. The obtained representation is used to identify MS patients in a pool of subjects. Concerning methodological improvements, it would be interesting to design a training strategy able to capture different aspects characterizing an outlier. An example could be the combination of a separation-based criterion and one based on the concept of scatter. Some additional details on these ideas are provided in Chapter 7.

Chapter 4

Enhanced Anomaly Scores

In this chapter, we propose two novel classes of anomaly scores. The first consists of weighting the path traversed by an object by exploiting the great amount of information present in a tree. The second one starts from a probabilistic reasoning and proposes a novel way to aggregate the tree scores at forest level. First, we present the motivations and the intuitions behind this research direction, and then we thoroughly describe both contributions.

4.1 Introduction

As stated in Chapter 2, in an Isolation Forest the isolation capability is encoded by a depth-based anomaly score, which is higher for outliers since they usually have a higher isolation capability than inliers do. In detail, given an unknown testing object and an Isolation Forest, we make the object traverse each Isolation Tree and retrieve the depth of the reached leaf. The procedure is repeated for all trees and the average depth is computed. The final anomaly score is inversely proportional to the average reached depth.

As thoroughly described in Chapter 2, there exist several works that extend iForests, and some of them also propose an extension to the testing phase [24, 63, 80, 98, 165] –although often tied to an alternative training phase and thus tuned for the different tree structure. Indeed, each of these extensions defines a novel anomaly score based on specific assumptions, different from the other works. In the following, we briefly recall the main idea behind each work –see Chapter 2 for a complete description:

- [63] defines a score able to manage streaming data. The core of the novel scoring function is to measure the difference in complexity between the tree built with the testing object and the same tree if the object were to be removed.
- In [24] the anomaly score is defined as the proportion of training objects that end up in the leaf reached by the testing object.
- In [165] they propose a modification of the tree scores via a normalization factor that is specific for each tree. In addition, they briefly mention that they aggregate the scores using the arithmetic mean.
- In [80, 83] the authors introduce two membership scoring functions. The first is based on the average distance to the center of the node computed during the training stage. The latter combines this information

with another notion that is although strictly related to the alternative tree building procedure that they propose, as thoroughly described in Chapter 2.

- In [98] the scoring function is akin to [96, 97] but it may lead to different scores. Indeed, when an object is traversing a tree, nodes may be removed from its path length. The removal happens when the object is outside the range of the split of the node, defined at training time. In other words, it means that the object is so diverse that its probability of being an outlier increases, and thus the node should not be considered in the total path length.

In spite of the numerous testing alternatives, there is still room for improvements. Indeed, all the above works lack in proposing a global approach that can work with all isolation-based methodologies (aside few exceptions) since they focus on few specific aspects.

In this chapter, we present an innovative proposal that aims at refining the standard anomaly score while maintaining the original concept [96, 97]. In detail, we make two contributions.
The first contribution starts from the information used to compute the anomaly score in each tree: the depth of the reached leaf. Using the depth as the only element to capture outlierness may be a bit myopic: indeed, each Isolation Tree contains many different types of interesting information, including information that can help in better capturing the nature of outliers. Examples are: the proportion of training objects that passes from a node or which part of the vectorial space the latter defines, etc. Thus, our proposal consists of several scoring functions which weigh the path, i.e. each traversed node, using different information.
The second contribution starts from the three following arguments:

1. We can interpret a tree, or better each leaf, from a probabilistic perspective. We make a starting assumption based on the analogy between the probability of a leaf in a dyadic tree [33] and the anomaly score of one Isolation Tree.

2. A forest is an ensemble classifier which base components are the trees, therefore it seems reasonable to exploit all the huge amount of research done in the field [88, 134],

3. The original anomaly score is highly impacted by bad trees, i.e. trees which are made of *unlucky* randomly generated splits that conceal the true isolation capability of the objects, i.e. the feature or/and the cut-point along which to split are bad in terms of isolation. In other words, the obtained anomaly scores from these trees are misleading.

Our proposal focuses on overcoming the latter limitation: our reasoning is based on a probabilistic interpretation of the trees and further supported by

findings on the ensemble learning field. Specifically, we design a novel class of anomaly scores that, via a different aggregation function of the tree scores, allows obtaining more reliable estimates of the anomaly scores at forest level.

Both contributions are therefore highly suitable in all those scenarios in which the standard anomaly score is not refined enough, e.g. the anomaly score of an object is too similar to that of another object which instead has clearly a different probability of being an outlier. Let us clear this concept with an example: we have a set of withdrawals from a bank account and to each of them have assigned an anomaly score after an iTree traversal, and two of them have an equally high anomaly score. In detail, one identifies a great amount of money withdrawn without permission, whereas the other is an authorized withdrawal which amount is slightly higher than usual. Since the depth is not enough to identify this difference, we could take into account, for example, the degree of similarity, measured via a comparison of the traversed paths, between the object that gets isolated and all the other objects in the tree. Indeed, we expect the unauthorized withdrawal to be highly dissimilar from the majority of the other withdrawals; whereas this does not hold for the other withdrawal mistakenly identified as anomalous. In other words, by combining this path-based similarity information with the path length, we can obtain a higher anomaly score for the unauthorized withdrawal, i.e. the true anomaly.

The suitability and the potential robustness of the technique is tested via a thorough experimental evaluation on 16 different datasets. The obtained results are highly promising, also when compared to refined extensions of the Isolation Forest. The evaluation consists of several analyses, comparing our contributions to the original anomaly score and also combining the two proposals; for all these analyses we make a statistical assessment. The first contribution has been partially published in the proceedings of the International Conference on Image Analysis and Processing [107] whereas the entire research work has been published in Pattern Recognition [108].

The rest of the chapter is organized as follows: in Section 4.2 we present the first contribution, whereas the second one is described in Section 4.3. For both contributions we make a step-by-step description, highlighting the starting point of their definition and making, when appropriate, also practical examples. In Section 4.4 we make a thorough experimental evaluation, which analyses the proposed methodology from several perspectives. Lastly, in Section 4.5 we make some conclusions and briefly describe some future research ideas.

4.2 Path-Weighted Scores

The core of this contribution consists of employing additional information to enrich the anomaly score to try and overcome its intrinsic limitations. In detail, the original formula of the anomaly score takes into account only the

depth of the traversed path: there is no distinction between traversed nodes, in other words they are considered equally important. On the contrary, it seems reasonable to take into consideration each node of the traversed path: indeed, different nodes contain different sets of objects which can be more or less descriptive for the task. Therefore, we propose a novel class of anomaly scores which considers the differences between nodes in the traversed path. In detail, we design five *path-weighted anomaly scores*: they all exploit information contained in the nodes, i.e. related to the tree building procedure and to the tree structure itself. These scores encode the information as a weight assigned to each node traversed by the testing object.

In particular, the core of our reasoning is based on a novel definition of the function $h_t(x)$ which computes the path length of an object x in a tree t, i.e. it is the depth of the leaf reached by x. Let us recall, for the sake of clarity, the formula of the anomaly score defined in Chapter 2:

$$s(x) = 2^{-\frac{E(h_t(x))}{c(S)}}. \tag{4.1}$$

The extended form of $h_t(x)$ is the following:

$$h_t(x) = \sum_{n \in \mathcal{P}_t(x)} 1 \tag{4.2}$$

where we recall that $\mathcal{P}_t(x)$ is the set of nodes traversed by x in t, i.e. the traversed path. Since all nodes count equally in the path, they give the same contribution of 1 to the total path length.

Our proposal is to define $h_t^w(x)$:

$$h_t^w(x) = \sum_{n \in \mathcal{P}_t(x)} w_{tn} \tag{4.3}$$

where w_{tn} is the weight assigned to node n in tree t. In other words, we define a weighted version of the path length: each node in the path contributes to the total path weight proportionally to a predefined weight function. The latter should encode relevant information for capturing outliers. It is clear that Eq. (4.3) is a generalization of 4.2: indeed if $w_{tn} = 1 \quad \forall t, n$, we obtain the original formulation of the anomaly score, i.e. $h_t^w(x) = h_t(x)$.

Using $h_t^w(x)$, we can define a novel class of anomaly scores as:

$$s^w(x) = 2^{-\frac{E(h_t^w(x))}{c(S)}} \tag{4.4}$$

where we embed the weighted path function defined in Eq. (4.3). Evidently, the original anomaly score defined in Eq. (2.4) is a special case of $s^w(x)$: indeed, when $h_t^w(x) = h_t(x)$ the two functions can be used interchangeably.

Summarizing, the proposed approach can be divided into the following steps:

1. Compute the weights w_{tn}, i.e. which information we want to use to weight the nodes and subsequently to which nodes we want to give more importance to.

2. Embed w_{tn} into Eq. (4.3). In this way, we retrieve the weighted path traversed by an object x in the tree t.

3. Compute the path-weighted anomaly $s^w(x)$ by embedding Eq. (4.3) into Eq. (4.4). This function performs the aggregation at forest level of the tree scores.

Clearly, prior to Step 1 we must define the function to weight the nodes. That is, we must define the *importance* of a node, and there are numerous ways to do that. We give five different definitions for w_{tn}, and we thoroughly describe each of them in the following.

Variant 1 – Neighborhood: The starting point behind the definition of the first variant is the notion of neighborhood given by [167]. Consider an object x that traverses a node n in a tree t, its neighborhood in relation to n consists of all the training objects passing from said node, in other words it is the set of training objects that node n contains. We can generalize the concept of neighborhood by defining the neighborhood of a node n, removing the connection to the object x. Formally, we define the neighborhood of n related to a tree t as the set of training objects in n, i.e.

$$N_{tn} = \{o | o \in \mathcal{O}^t \wedge o \in n\}$$

where we recall $\mathcal{O}^t$ is the training set used to build t. We can relate the size of N_{tn} to the importance that n has in t: given a set of training objects, a node with a small neighborhood describes them in a better way than a node with a larger neighborhood does. Indeed, a node n that contains few objects is likely to encompass a small and thus more refined portion of the space, defined by all previous splits that led to the generation of n. An opposite reasoning can be made for nodes containing many objects. Summarizing, a node with a small neighborhood has a higher relevance in a tree.

Given a generic decision tree, the neighborhood size usually decreases in a balanced way from the root, which contains the entire training set, to the leaves, which tend to contain very few objects –if not only one. If the tree structure is an Isolation Tree, the interpretation slightly changes: nodes at small depths may have neighborhoods smaller than expected, and this happens when an outlier gets isolated. Indeed, the path leading to the isolation of an outlier is composed by ancestor nodes which are quite big but at the same time which size tends to decrease faster than that of nodes leading to the isolation of inliers. Considering both these observations, we can confirm that a smaller node tends to be more relevant within the path of an object, therefore we should attribute a bigger weight to nodes with a small neighborhood. Formally,

we define the weight w_{tn}^N of a node n in a tree t as:

$$w_{tn}^N = \frac{1}{|N_{tn}|} \tag{4.5}$$

where $|N_{tn}|$ indicates the size of the neighborhood, i.e. the number of objects that node n contains. Having defined w_{tn}^N, we can embed it into Eq. (4.3) and then into Eq. (4.4), obtaining the path-weighted anomaly score, which we denote by $V1$.

Variant 2 – Proxy: The second variant that we propose defines a weight based on some concepts proposed in [57], a training extension of Isolation Forest that we briefly described in Chapter 2. Here we describe it more thoroughly for the sake of comprehension of our proposal. The core concept of [57] is that the isolation-based training strategy is optimized via the minimization of a proxy function, called one-class proxy, aimed at capturing the information loss caused by the split defined by (j, θ) where j is the feature along which to split and θ a threshold in its domain. In detail, a good pair (j, θ) should generate: a child node of minimum volume containing many objects, and its sibling of maximum volume containing few objects. Respectively, the two child nodes should contain only inliers and only outliers, i.e. to be as pure as possible.

The absence of labels is handled by the one-class proxy by defining: i) a function estimating the number of outliers; ii) a volume measure. In detail:

- The distribution of outliers is uniform within a node n and not within the whole tree, i.e. the number of outliers is defined in an adaptive way. In detail, given a node n and being NI_n the number of inliers within it, the number of outliers NO_n is defined as $NO_n = NI_n\gamma$ where γ is a fixed constant.

- To evaluate each pair (j, θ), i.e. measure the information loss caused by the split, we compute the ratio λ_n of the volume of a node n to the volume of its parent $parent(n)$. The volume is measured using the Lebesgue measure $Leb()$. So in detail we have: $\lambda_n = \frac{Leb(n)}{Leb(parent(n))}$.

Given these basic notions –for further mathematical details, see [57]– the one-class proxy is defined as:

$$proxy(n) = \frac{NI_{n_L}\gamma NI_n\lambda_L}{NI_{n_L} + \gamma NI_n\lambda_L} + \frac{NI_{n_R}\gamma NI_n\lambda_R}{NI_{n_R} + \gamma NI_n\lambda_R} \tag{4.6}$$

where we recall n_L and n_R are respectively the left and right child of n. γ is usually set to 1. The proxy thus measures how good is the pair (j, θ) at splitting node n: the lower its value, the better the split, i.e. the better the isolation process. On the contrary, if the proxy of the node has a high value, it means that the pair (j, θ) is not able to separate well the objects contained in the node.

We therefore assign a weight to a node proportional to its capability of separating each object from the rest, in particular the outliers. This capability is measured using Eq. (4.6). Starting from these concepts, we define in a tree t and a node n belonging to t, the weight w_{tn}^{P} as:

$$w_{tn}^{P} = \frac{1}{proxy_t(n)} \tag{4.7}$$

where $proxy_t(n)$ is the proxy measured in the node n of tree t. By embedding w_{tn}^{P} into Eq. (4.3) and then the resulting function into Eq. (4.4) we obtain the related path-weighted anomaly score, which we define as $V2$.

Variant 3 – Proxy-Neighborhood: The third weight we have designed is a combination of the first two proposed variants. In other words, we define as w_{tn}^{PN} the weight that in a node considers both its neighborhood size and how well it was split.
Formally, given a node n in a tree t, w_{tn}^{PN} is:

$$w_{tn}^{PN} = \frac{1}{proxy_t(n)|N_{tn}|}. \tag{4.8}$$

The weight of a node gets higher as the product of the neighborhood size and the proxy gets smaller: the highest value is reached when both are very low. Analogously to the other two variants, after embedding the weight in the anomaly score computation formulas, we obtain $V3$, the path-weighted anomaly score that employs w_{tn}^{PN}.

We further define other two-weighted variants, both based on the same initial observation: there are some outliers which predicted isolation capability is lower than expected, i.e. the number of splits leading to their isolation is higher than hypothesized. This phenomenon is likely to happen when: i) an outlier is not very distant from the inlier distribution; ii) there are several outliers which are both distant from each other and from the normal data. Indeed, outliers close to the normal distribution tend to be masked by the presence of inliers. Instead, in the second scenario, in which hypothetically these outliers should have the same isolation capability, it is unfeasible, within one tree, to isolate them all using the same number of splits. In other words, some outliers will be penalized.

Let us clarify our initial observation: Figure 4.1 depicts a practical example of how the standard anomaly score works. The example has been generated by performing manual splits, highlighted in the figures, which simulate an Isolation Tree. Figure 4.1 is composed of four plots, each dedicated to the isolation of one pattern:

- The highlighted object in Figure 4.1 (a) is an outlier which gets isolated after 1 split, i.e. its standard anomaly score $s(x)$ is 0.5.

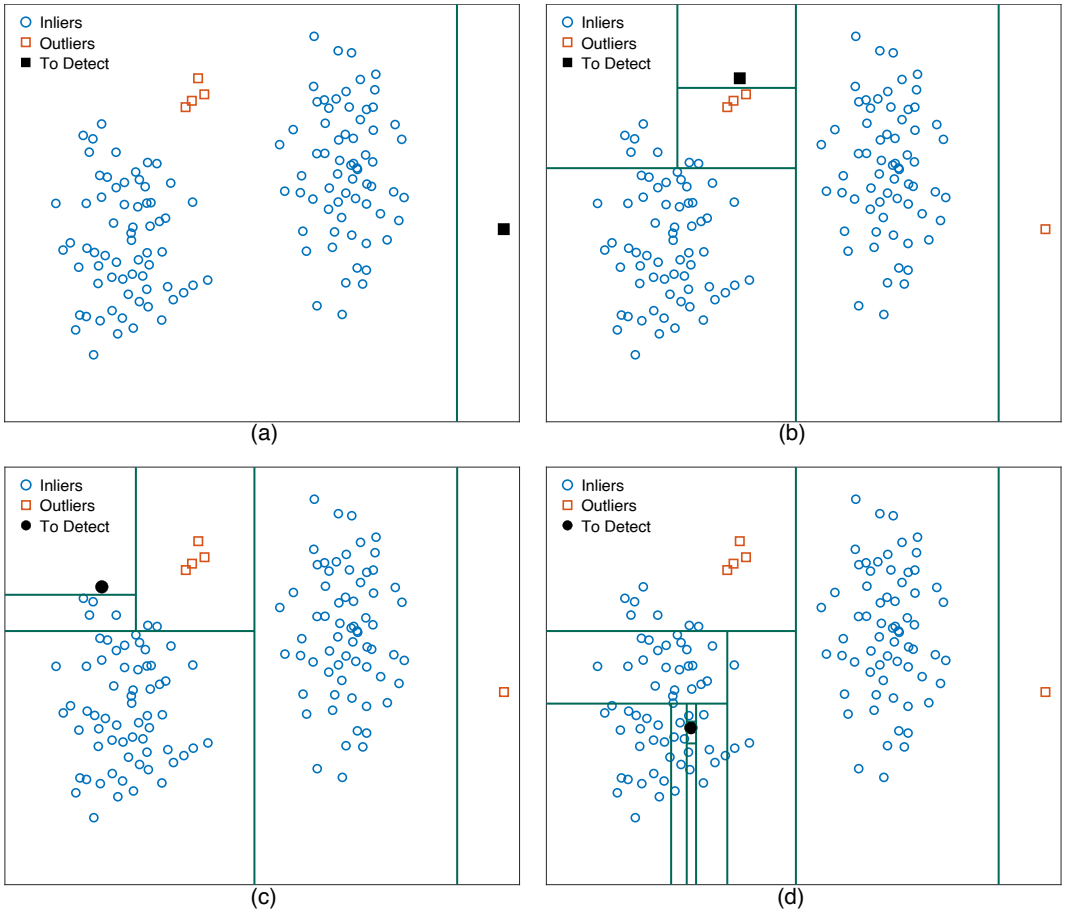


Figure 4.1: Each plot represents the isolation of one object via an Isolation Tree. The objects are respectively: (a), (b) outliers and (c), (d) inliers.

- The object in Figure 4.1 (b) is an outlier closer to the normal distribution. It gets separated from the rest after 5 splits, i.e. $s(x) = 0.0625$.
- The highlighted object in Figure 4.1 (c) is a marginal inlier which score is $s(x) = 0.0625$, i.e. it takes 5 splits to isolate it.
- Lastly, Figure 4.1 (d) represents the isolation of an inlier deep within the normal distribution. Indeed, it gets isolated after 10 splits, i.e. $s(x) = 0.00098$.

The first observation is that the isolation capability of outliers which are far from the inlier distribution is high, as expected. There is also a relevant difference in terms of anomaly score between Figure 4.1 (a) and Figures 4.1 (c) and (d), i.e. the ones isolating inliers. The other important observation is rather different: it is impossible, in terms of anomaly score, to distinguish the two cases in Figure 4.1 (b) and (c), which however represent two completely different objects: (b) is an outlier close to the normal distribution and (c) is a marginal inlier. Therefore, it is crucial to highlight their difference.

To face that, we make two considerations: i) due to the isolation principle encoded in each tree, it is more likely that leaves deep in a tree contain inliers and those higher in the tree outliers; ii) an innate characteristic of trees is that nodes higher in a tree consist of many objects whereas deeper nodes contain fewer objects. Then we can reasonably connect these two observations and infer that leaves high in a tree, i.e. which depth is small, are likely to have ancestor nodes composed of many objects. On the other hand, inliers often end up in leaves very deep in a tree and therefore, the last steps before these objects get isolated are characterized by nodes containing few objects –this is reasonable since the inliers belong to a precise data distribution.

Thus, we aim at defining a weight for the anomaly score that takes into account both the number of objects in each of the traversed nodes and the maximum depth at which each object is encountered. This information is actually the depth of the *Lowest Common Ancestor* (LCA) of the testing object and each training object encountered in the path from the root to the leaf by the testing object itself. Taking these assumptions as true, for an unseen object x we define a weight with respect to a tree t and a training object y employed in the tree building procedure as $w_t^{LCA}(x, y)$:

$$w_t^{LCA}(x, y) = dp(x) - dp(LCA_t(x, y)) \tag{4.9}$$

where $dp(LCA_t(x, y))$ is the depth of the LCA node of x and y in t and $dp(x)$ is the depth of the leaf $l_t(x)$ reached by the testing object in t, i.e. $dp(x) = dp(l_t(x))$. The latter accounts for the fact that the importance of the depth of the LCA depends on the depth of the testing point itself –note that the weight can be computed only after x has traversed the tree, since we need to know the leaf it ends up into. In other words, the weight is small when the two objects share a great proportion of the path of x, i.e. $dp(LCA_t(x, y))$ is close to the depth of the leaf reached by x; the weight instead increases when the converse happens. Please note that if x is an outlier it is likely that for many training objects y, $LCA_t(x, y) = root$ thus assigning to x the biggest possible weight in concordance with our assumption–where the term *big* is relative to the depth of the leaf reached by x. Based on $w_t^{LCA}(x, y)$, we propose two different variants of w_{tn}:

Variant 4 – LCA of the Neighborhood: To define w_{tn}, we must extend the definition of $w_t^{LCA}(x, y)$ to all objects y used to build the tree t that have been used to create nodes traversed by x. In other words, $w_{\mathcal{P}_t(x)}^{LCA}$ is the total weight of the path traversed by x and we define it as:

$$w_{\mathcal{P}_t(x)}^{LCA} = \frac{\sum_{y \in \mathcal{O}^t} w_t^{LCA}(x, y)}{S} \tag{4.10}$$

where we recall that $\mathcal{O}^t$ is the training set employed to build t and S is the size of $\mathcal{O}^t$. Since all training objects are encountered in the path of x, each $y \in \mathcal{O}^t$ must be taken into account in the computation of the path-weight –this is due

to the fact that all objects of the training set start from a common node, the root, from which the traversal procedure for all testing objects begins. The weight is then averaged across the encountered training objects. The weight $w^{LCA}_{\mathcal{P}_t(x)}$ can be formulated in terms of w_{tn} but it is dependent also on the testing object x. Indeed, given a node n that belongs to a tree t, we define w^{LCA}_{tn} as:

$$w^{LCA}_{tn} = \frac{\sum_{y \in N_{tn}} \mathbb{1}(dp(n) = dp(LCA_t(x,y)))w^{LCA}_t(x,y)}{\sum_{y \in N_{tn}} \mathbb{1}(dp(n) = dp(LCA_t(x,y)))} \tag{4.11}$$

where $dp(n)$ is the depth of node n in tree t and $\mathbb{1}()$ is an indicator function. In other words, we weigh a node n in a tree t with respect to a testing object x employing only those training objects y for which $LCA_t(x,y) = n$. In detail, the function has value 1 only once for a specific pair (x,y) in a tree t: when the node n that is being weighted has the same depth of the LCA of x and y. This event occurs when n and the LCA are the same node since in the path of x there exists one and only one node for which the depth is $dp(n)$. The indicator function is necessary to leave out from the computation, the relation between x and y in each node n' that is different from the LCA and traversed by both objects. Then the weight is averaged, but only across the samples in n which have been given a non-zero weight. When embedding w^{LCA}_{tn} in Eq. (4.4), we obtain the path-weighted variant $V4$.

Variant 5 – LCA of the Neighborhood (Anomaly Score): The fifth and last weight is slightly different from the latter: the starting assumption is the same, but we redefine the weight between x and y, $w^{LCA}_t(x,y)$ as $w^{LCA\_AS}_t(x,y)$:

$$w^{LCA\_AS}_t(x,y) = 2^{-w^{LCA}_t(x,y)}. \tag{4.12}$$

With respect to $w^{LCA}_t(x,y)$, $w^{LCA\_AS}_t(x,y)$ has an inverse meaning, i.e. a smaller weight is assigned to objects which are very similar, i.e. for which the depth of $LCA_t(x,y)$ is close to that of x.

Following the novel definition of the weight between x and y, we must make a novel formulation of the path-weight: $w^{LCA\_AS}_{\mathcal{P}_t(x)}$:

$$w^{LCA\_AS}_{\mathcal{P}_t(x)} = \frac{\sum_{y \in \mathcal{O}^t} w^{LCA\_AS}_t(x,y)}{S}. \tag{4.13}$$

The last step consists of redefining we redefine the weight assigned to each node n in t relative to the testing object x as $w^{LCA\_AS}_{tn}$:

$$w^{LCA\_AS}_{tn} = \frac{\sum_{y \in N_{tn}} \mathbb{1}(dp(n) = dp(LCA_t(x,y)))w^{LCA\_AS}_t(x,y)}{\sum_{y \in N_{tn}} \mathbb{1}(dp(n) = dp(LCA_t(x,y)))}. \tag{4.14}$$

The essential difference with respect to the fourth variant, i.e. Eq. (4.11), consists of assigning an anomaly score in the range $[0,1]$ as weight to each node present in the path. Namely, the weight $w^{LCA\_AS}_{tn}$ describes how much n

contributes to the outlierness of x in the tree t.

With respect to the other four variants, the last criterion is a stand-alone: the node weight is embedded into Eq. (4.3), analogously to the other weights, however it is not feasible to embed it into Eq. (4.4). Therefore, we define a novel aggregation function to combine the tree scores at forest level, to obtain the path-weighted anomaly score, which we call $V5$:

$$V5(x) = \frac{\sum_{t \in \mathcal{F}} h_t^w(x)}{|\mathcal{F}|} \tag{4.15}$$

where $h_t^w(x)$ takes $w_{tn}^{LCA\_AS}$ as weight for each node. The interpretation of this anomaly score, $V5$, is rather straightforward since it is solely the average of the tree scores.

Summarizing, the latter variants, based on the LCA concept, are able to capture outliers closer to the inlier distribution by assigning them a greater anomaly score. In this way, the probability of misclassifying this type of outliers as inliers decreases. Another relevant observation is that by employing $V4$ or $V5$ also the anomaly score of inliers will be likely to increase, but since their starting value, i.e. the unweighted anomaly score, is rather low, the impact of employing this additional information is feebler. In practice if we compute $V5$ for the objects depicted in the previous example, i.e. in Figure 4.1, we get the following path-weighted anomaly scores: (a) 0.5, (b) 0.1118, (c) 0.1099 and (d) 0.0226 –analogously it works for $V4$. We can make several observations. The first is that there is only one correspondence between $s(x)$ and $V5$ and that is for score (a); the reason behind this equivalence stands in the LCA being the root, which has depth 0 for all training points except for one, which generates the leaf into which x ends up into. The second observation is that all remaining anomaly scores have a greater value with respect to $s(x)$: in detail the marginal inlier and the outlier closer to the inlier distribution, Figures 4.1 (c) and (b) respectively, have now a different anomaly score, with the former having the lower one, as expected from the assumptions on which $V4$ and $V5$ are based on. Therefore, with $V5$ we are able to distinguish the situation in Figure 4.1 (b) from the one in Figure 4.1 (c) via their weighted anomaly scores. Lastly, we can observe that even if the anomaly score of the inner inlier, depicted in Figure 4.1 (d), increases, it is still much lower than the other scores. This example hints that the use of the depth of the LCA as a way to enrich the anomaly score is beneficial at tree level, and subsequently it can be advantageous also when combining the tree scores at forest level.

To sum up, we make a contribution that enriches the anomaly score by weighting each node in the path: to score an object x that traverses a tree t, we should look at the entire structure of the tree and not only at the leaf it ends up into. In other words, we should highlight the relation x has with the training set of t.

Please note that by comparing the proposed approach to other testing

extensions, we can clearly deduce that [80] is a special case of our framework and that in [98] the starting intuition is similar to ours. In detail, in [98] the authors employ path-related information, but they exploit it in an opposite way: they remove nodes from the path which can be informative, rather than exploiting the information within them.

4.3 Probability-Based Aggregation Function

The other contribution, related to the computation of the anomaly score in an isolation-based tree structure, focuses on the score at forest level. In other words, we focus on the aggregation function used to combine the scores retrieved from each tree in the forest. The starting points of our rational are two: i) each isolation-based tree structure can be looked at from a probabilistic perspective; ii) aggregating the tree scores at forest level is analogous to combining the scores of several classifiers of an ensemble, since forests are ensemble classifiers themselves.

As to the former aspect, there are two different probabilistic interpretations of a tree [33, 62]:

1. *Probabilistic tree*: a probabilistic tree [62] is a structure in which each node represents an event. More in detail, each node conveys a probability. Given a node A and its parent node B, the probability associated to the event $E(A)$ represented by A is $p(E(A) \cap E(B))$, which, if the events are independent, becomes $p(E(A))p(E(B))$. In case of decision trees, the event represented in a node A is the set of tests that have led to its creation. The probability of said event is linked to the number of objects that have thus reached A, i.e. $\frac{1}{|A|}$ where $|A|$ stands for the number of objects in A.

2. *Dyadic tree*: a dyadic tree is a tree which encodes a dyadic probability distribution. The latter is defined in [33] as "a probability distribution is called D-adic if each of the probabilities is equal to D^{-n} for some n". If the dyadic tree is binary it means that we have $D = 2$, and thus that to each node A is associated the probability $p(A) = 2^{-n_A}$ where n_A represents the depth of the node A. It can also be written as $p(A) = 1/2^{n_A}$.

The latter interpretation is rather interesting, since it is strictly related to Eq. (2.4), which computes the anomaly score in a forest. In detail, if we had only one tree t the formula of the anomaly score becomes[1]:

$$s_t(x) = 2^{-h_t(x)}. \tag{4.16}$$

From Eq. (4.16) it is evident that the anomaly score at tree level is equivalent to the dyadic tree-based formulation $p(l_t(x))$ where $l_t(x)$ is the leaf reached

[1]To simplify the notation we assume tree t to be completely grown so that we can omit the normalization term $c(|l_t(x)|)$.

by x, i.e. the probability of the object is the probability of the associated leaf. Following this interpretation, it seems legitimate to enrich the anomaly score by assuming a probabilistic perspective, i.e. we can interpret $s_t(x)$ as the probability of x being an outlier.

The probabilistic meaning of the score at forest level is more intricate. Eq. (2.4) can be broken down into the following formula:

$$2^{-\frac{\sum_{t\in\mathcal{F}} h_t(x)}{|\mathcal{F}|}} = 2^{-\frac{h_1(x)+\ldots+h_{|\mathcal{F}|}(x)}{|\mathcal{F}|}} = 2^{-\frac{h_1(x)}{|\mathcal{F}|}} \cdot \ldots \cdot 2^{-\frac{h_{|\mathcal{F}|}(x)}{|\mathcal{F}|}}. \tag{4.17}$$

Maintaining a probabilistic perspective, we can view Eq. (2.4) as the product of the tree probabilities. The latter implies that we are computing a joint probability of independent events: each tree, or better each leaf reached by x, is an event, and it is independent of the other reached leaves, i.e. trees. This aggregation function is highly reasonable since trees are built independently, i.e. the only direct tie between them is the training set from which the set of objects used to build each tree is sampled. In practical terms, if a forest $\mathcal{F} = \{t_1, t_2\}$ is given, t_1 is independent of t_2 and vice-versa; the anomaly score of an object x in $\mathcal{F}$ encodes the probability of x being an outlier in t_1 *and* in t_2 at the same time. This reasoning can be generalized to any number of trees. Nevertheless, in some cases combining the tree scores via their multiplication can lead to unwanted consequences in terms of anomaly scores. For example, assume that we have an iForest and a new unseen object arrives, which we know to be an outlier; we make the novel object go down each tree in the forest, and we observe that one of them is not capable of isolating the point in a few steps. This *bad tree* implies that the score for the outlier will have a lower value than the expected one due to the multiplication effect (an example of a *bad tree* is Figure 4.2 (b) which we will shortly describe in detail).

To get a less limiting anomaly score, we make a different starting assumption: trees are replicas of the same event. In this scenario the anomaly score at forest level is the expected value of the probability distribution; each value of the latter represents a different leaf, i.e. a different replica, reached by the object x. In other words, the probability of x being an outlier in a forest is the average probability of x being an outlier in each tree composing the forest. Following this line of thinking, we define the novel anomaly score $p(x)$ of an object x in a forest $\mathcal{F}$ as:

$$p(x) = \frac{2^{-h_1(x)}}{|\mathcal{F}|} + \ldots + \frac{2^{-h_{|\mathcal{F}|}(x)}}{|\mathcal{F}|}. \tag{4.18}$$

We can rewrite $p(x)$ in a shorter form as:

$$p(x) = \frac{\sum_{t\in\mathcal{F}} 2^{-h_t(x)}}{|\mathcal{F}|}. \tag{4.19}$$

This novel formulation of how to aggregate the tree scores, allows to obtain a

score at forest level which is less restrictive than the original formulation. In other words, $p(x)$ is capable of alleviating the effect that the presence of badly built trees, i.e. unable to isolate correctly objects, has on the final anomaly score.

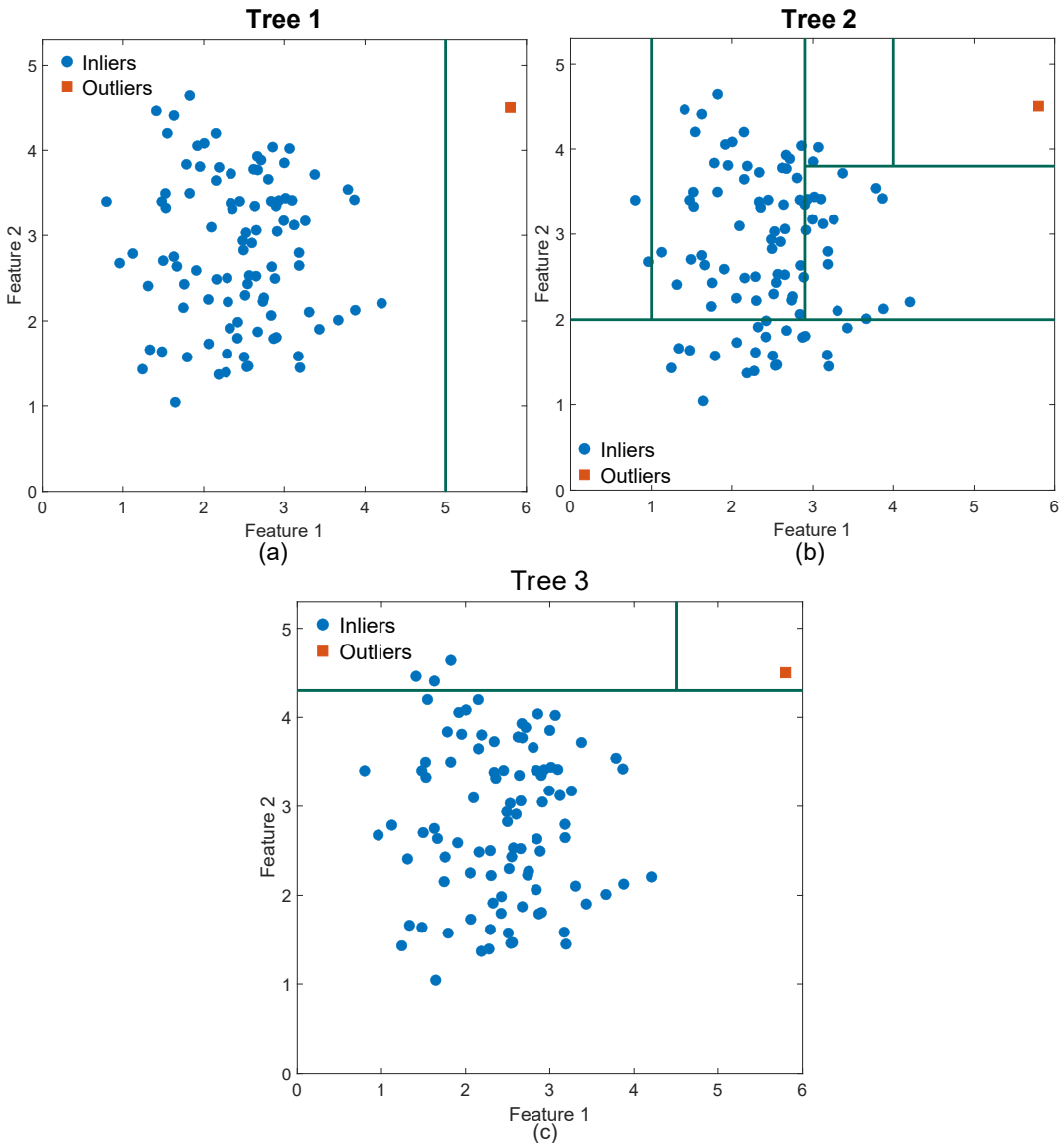


Figure 4.2: The three plots represent a data distribution contaminated by an outlier. In each figure, the outlier is isolated via random splits of an Isolation Tree.

For a better understanding of the drawbacks of using $s(x)$ and the advantages of employing $p(x)$, we can refer to Figure 4.2. Figures 4.2 (a), (b) and (c) represent a normal distribution which objects are depicted as circles; there is also an outlier, i.e. an object that does not belong to the inlier distribution –indeed it is very far away from the remainder of data– which is represented via a square. Each figure represents three possible (and hypothetical) Isolation Trees, in which we depict a series of axis-parallel splits performed to isolate the

outlier. In Tree 1 the outlier gets separated from the rest after only one split, while in Tree 2 and 3 respectively it takes five and two splits to isolate it. If we consider the scores at tree level, we would obtain the following: $s_1(x) = 0.5$, $s_2(x) = 0.03125$ and $s_3(x) = 0.25$. It seems evident that Tree 2 is *inaccurate* since it is not capable to detect the outlier, i.e. it isolates the point in many steps. If we aggregate the scores using $s(x)$, the original function, we get $s(x) = 0.1575$ which is rather low due to the multiplication effect, as we briefly mentioned before, which excessively penalizes badly built trees. Indeed, even though 2 out of 3 Isolation Trees are very good, the anomaly score at forest level is rather low because of the impact of Tree 2. This effect is mitigated if we compute the mean of the anomaly scores at tree level. Indeed, if we aggregate using the novel aggregation function, i.e. we compute $p(x)$, we obtain $p(x) = 0.2604$ which is almost two times $s(x)$. In other words, the probability of x of being an outlier when evaluated with $p(x)$ is higher.

The proposal of this novel aggregation function is supported also by the findings in the ensemble classifiers field [134], to which Isolation Forest and all Random Forest-based techniques belong to. Indeed, we can interpret the considerations made above on the differences and analogies of the two different aggregation fields from an ensemble point of view. In detail, the original anomaly score, i.e. Eq. (2.4) is a variation of the product rule, i.e. the scores of the various classifiers composing the ensemble are multiplied. Whereas our proposal, defined in Eq. (4.19), is another well-known rule, the sum rule, i.e. scores are summed and usually divided by the number of classifiers. A lot of efforts have been dedicated to the understanding of which combiner function is the best choice for an ensemble classifier [88], i.e. which is more relevant and important to obtain a reliable estimate of the ensemble scores. In [88] the superiority of the sum rule is stated: the product rule becomes unreliable with the presence of a single bad score, leading to a reduced overall performance of the ensemble technique.

4.4 Experimental Evaluation

This section presents and discusses the experiments performed to assess the suitability and robustness of the proposed methodology. In detail, the section is organized in six subsections, each focusing on a different aspect of the experimental evaluation. In Subsection 4.4.1 we thoroughly describe the used datasets and the related experimental details. In the second subsection, Subsection 4.4.2, we focus on the first contribution by comparing the path-weighted variants to $s(x)$, the original anomaly score: we aim at assessing whether weighting the path leads to any improvement. In Subsection 4.4.3 we focus on the second contribution: we compare the two combining criteria $s(x)$ and $p(x)$ in order to confirm that a less stringent aggregation function is beneficial. Then Subsection 4.4.4 is dedicated to the analysis of the combination of the two contributions to assess whether further increases in performances

can be reached. i.e. we weight the tree scores and aggregate them at forest level by employing the novel combiner function $p(x)$. Lastly, Subsection 4.4.5 compares the best proposed contribution with some Isolation Forest extensions, while Subsection 4.4.6 analyzes the run-time of the different weighted variants.

4.4.1 Experimental Details

To assess its robustness, the proposed methodology has been tested on 16 datasets. In detail, 12 of them were chosen since largely used in the outlier detection literature [57, 97, 98]; whereas the remaining 4 were taken from [25] since they are built specifically for the task of outlier detection. Indeed, [25] illustrates some guidelines on how to build a *benchmark* dataset for outlier detection and proposes several variants of each dataset. The variants of a dataset differ based on whether: presence of duplicates, data normalization and the outlier percentage among other factors.

The pool of 12 datasets belongs to the UCI-ML repository [43] and they result from the union of the training and testing partitions available in the repository. Most of these problems are in origin classification problems: to adapt them to the outlier detection field, we processed them following the indications of [57]. This procedure consists mainly in deleting categorical features and dividing the classes into two: outliers and inliers, where the former is often the less numerous one. As to the other 4 datasets, *Cardiocotography, Hepatitis, PageBlocks* and *Stamps*, taken from [25], we followed the same processing procedure: we removed duplicates, normalized the dataset, and we chose to keep all the outliers, i.e. the outlier percentage is the maximum possible. Please note that the normalization step is different from [25] in which they perform min-max scaling; instead we perform z-score standardization, analogously to [57], for all datasets[2].

We present the main characteristics of each dataset in Table 4.1: name, number of objects, number of features and outlier percentage. The datasets cover a wide range of situations: they differ in size (the smallest having 351 objects and the biggest 567498), in the dimensionality (features vary in the range $[3, 164]$) and in the number of outliers (from a very small percentage, i.e. 0.03% up to almost half the dataset, i.e. 45.8%). Concerning the latter assertion, it may seem counter-intuitive to have datasets with such a high percentage, but analogously as what stated in Chapter 3, it is not uncommon in the outlier detection field to have such type of datasets. Indeed, among other motives, as explained in [25] the percentage of outliers only influences the absolute performance whereas it does not have any impact when comparing approaches. Therefore, our choice of datasets is completely reasonable. Analogously to Chapter 3, we measure the accuracy of the proposed approach using

[2]Indeed, we assessed from an empirical point of view that z-score standardization is more beneficial for all techniques involved in the experimental evaluation.

the AUC, a typical choice for outlier detection [25, 41, 59, 96] as illustrated in Chapter 2.

Table 4.1: Overview of the 16 datasets used for the experimental evaluation.

Datasets	Nr. of Objects	Nr. of Features	Outlier %
Adult	48842	6	23.93%
Annthyroid	7200	6	7.42%
Arrhythmia	452	164	45.80%
Cardiotocography	2126	21	22.15%
ForestCover	286048	10	0.96%
Hepatitis	80	19	16.25%
Http	567498	3	0.39%
Ionosphere	351	32	35.90%
PageBlocks	5473	10	10.23%
Pendigits	10992	16	10.41%
Pima	768	8	34.90%
Shuttle	49097	9	7.15%
Smtp	95156	3	0.03%
Spambase	4601	57	39.40%
Stamps	340	9	9.12%
Wilt	4839	5	5.39%

We carried out a copious amount of experiments by varying several parameters of the Isolation Forest:

1. Size of the forest, i.e. number of Isolation Trees, T: 50, 100, 200, 500. *4 options.*

2. Number of objects used to build each tree in a forest S: 64, 128, 256, 512 and 1024. *5 options.* Analogously to the experiments in Chapter 3, for those datasets for which the size of the training set is smaller than S we carry out the experiment using all available samples, i.e. no subsampling is performed.

3. Maximum depth: $D : S - 1, \log_2(S)$.

4. Number of features used to build each tree in a forest $F : f, \frac{f}{2}$ where f is the number of features of the dataset under analysis.

In addition, for each dataset we performed the same experiment 10 times, which difference stands in the training and testing sets. In other words, for each iteration of an experiment we randomly split in half the dataset, using one half for the training procedure and the other half for the testing one. The only constraint to be respected is that the training set must be composed of only inliers, as done in [57]. Please note that given a dataset, the 10 partitions of training and testing sets are identical across all parametrizations.

4.4.2 Comparison Between s(x) and Path-Weighted Anomaly Scores

The first analysis compares the path-weighted anomaly scores $V1 - V5$ as defined in Section 4.2 with $s(x)$ defined in Eq. (2.4).

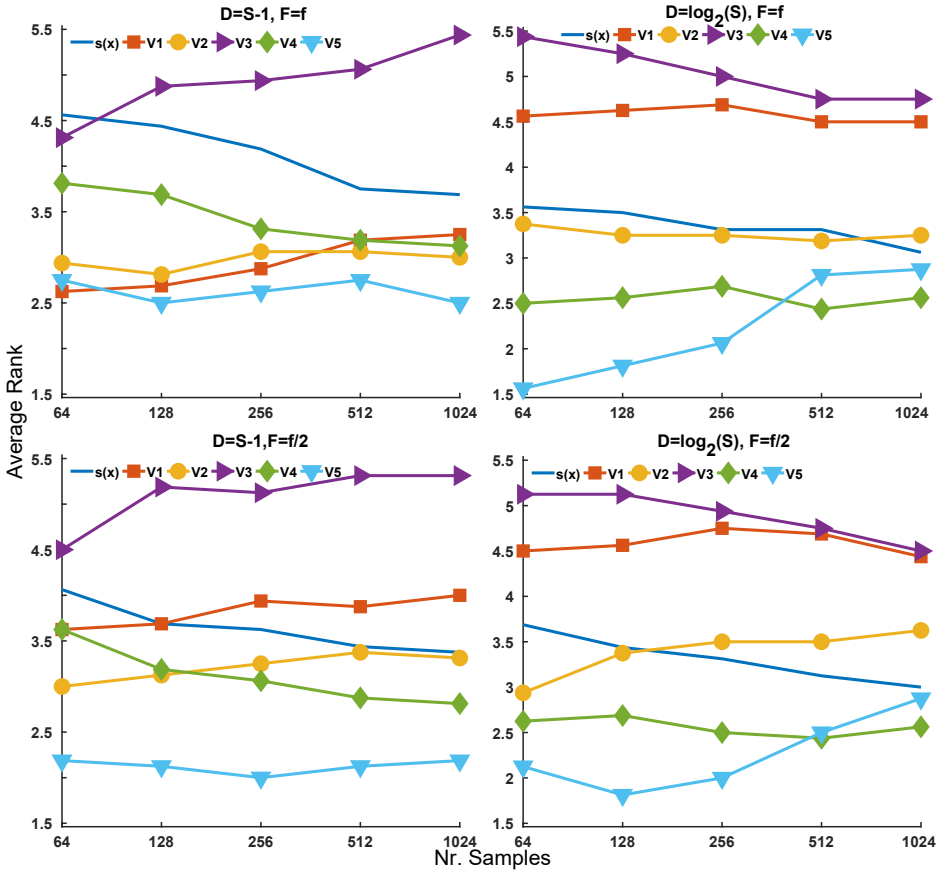


Figure 4.3: Comparison in terms of mean rank between $s(x)$ and the weighted variants when varying the training sample size S.

Figure 4.3 is composed of four plots, each one evaluating how the behaviour of $s(x)$ and of each path-weighted variant changes as S does. The four plots differ on the analyzed parametrization: the parameters encoding the depth and the number of features, D and F, are fixed for each plot –by combining all possible values that the two parameters can assume we have 4 possible plots. In detail, in each plot, fixed a scoring function, we depict its mean rank, which is obtained by sorting the AUC of all techniques for each dataset and by averaging across the other non-fixed parameters, i.e. the forest size T and the iterations. Please recall that a smaller rank indicates a better performance: for example, if 10 variants are compared and one of them has rank 3 it means that on average it performs better than 7 variants and worse than 2 of them. Figure 4.4 represents the same type of analysis in which the behaviour of the AUC is analyzed as the number of trees composing a forest, T, varies.

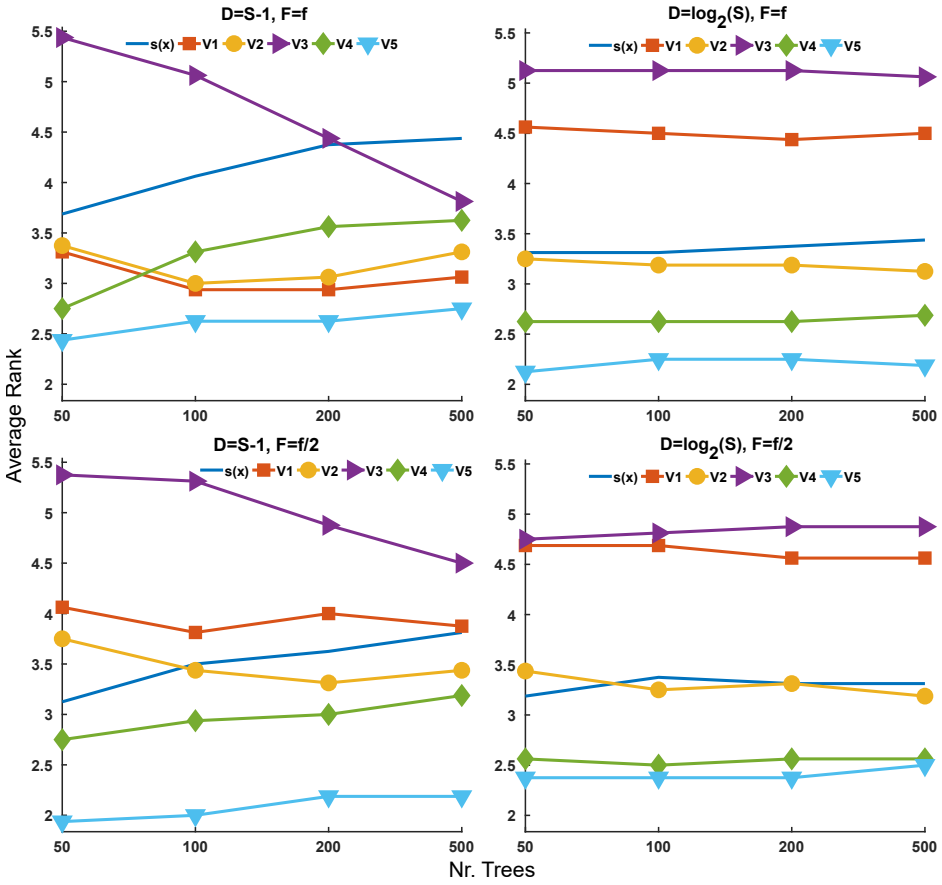


Figure 4.4: Comparison in terms of mean rank between $s(x)$ and the weighted variants when varying the forest size T.

An observation that can be made in both Figures 4.3 and 4.4, is that most path-weighted variants perform well, often outranking $s(x)$ independently of the analyzed parameters. In detail, the above affirmation is true in all cases for the score variants based on the LCA, **V4** and **V5**, and in particular the latter seems to be the best choice in all experiments. Even though it is not easy to generally establish whether it is better to use the path-weighted variant based on the one-class proxy, $V2$, with respect to $s(x)$, the former outranks $s(x)$ on more than half the examined parametrizations. Instead the neighborhood-based variants, $V1$ and $V3$, are almost in all cases outranked by the unweighted variant $s(x)$. Further, we can observe that in Figure 4.3 the rank of $s(x)$ increases as the size of the training set of each tree does; this observation is reasonable: with fewer data the relevance of employing additional information increases. Nevertheless, we can also notice that the improvement of $s(x)$ is not big enough to make it the best score –or the second-best choice– in any case. Instead, in Figure 4.4 the rank of each path-weighted scoring function does not vary much across different parametrizations, except for $V3$. Indeed, the rank of $V3$ drastically changes when using a particular parametrization, i.e. when $D = S - 1, F = f$, probably because it exploits a greater amount of

information. Also, the rank of $s(x)$ does not vary much as T does. In general, it should be observed that $s(x)$ is always outranked by at least one variant.

To grasp the behaviour of the proposed variants on each dataset, in Table 4.2 we present the results on each dataset when training the iForest with the default parametrization: $S = 256$, $T = 100$, $F = f$, $D = \log_2(S)$ [96, 97] (see Appendix B for the results on each dataset when using the same parametrization except for $D = S - 1$). In detail, we report the average AUC across the 10 iterations. We also assess whether each path-weighted variant is significantly different from $s(x)$: to recover this information, we performed a Wilcoxon signed-rank test with significance level $\alpha = 0.05$ followed by a Bonferroni correction: we put a * next to each path-weighted variant whenever there is a statistically significant difference from $s(x)$. Furthermore, we mark in **bold** the best overall variant for each dataset.

Table 4.2: Comparison between $s(x)$ and path-weighted scores when employing the standard iForest parametrization.

Dataset	s(x)	V1	V2	V3	V4	V5
Adult	**0.657**	0.655*	0.656	0.655*	0.655	**0.657**
Annthyroid	0.915	0.906*	0.910	0.907*	0.922*	**0.942***
Arrhythmia	0.758	0.753*	0.756	0.753*	0.759	**0.772***
Cardiotocography	**0.755**	0.752	0.743*	0.751	0.744*	0.743
ForestCover	0.841	0.816*	0.824*	0.817*	0.853*	**0.927***
Hepatitis	0.701	0.688	0.709	0.676*	0.711	**0.745***
Http	0.991	0.990	**0.994***	0.987*	**0.994***	**0.994**
Ionosphere	0.905	0.889*	0.909	0.888*	0.921*	**0.943***
PageBlocks	0.802	0.787*	0.792*	0.779*	0.827*	**0.862***
Pendigits	0.799	0.807	0.837*	0.804	0.836*	**0.852***
Pima	0.733	0.737*	**0.738***	0.737*	0.727*	0.714*
Shuttle	0.996	0.995*	0.996*	0.995*	0.997*	**0.998***
Smtp	0.908	0.916*	**0.924***	0.912*	0.918*	0.920*
Spambase	0.844	0.838*	0.839*	0.839*	0.843	**0.845**
Stamps	0.957	0.956	0.956	0.956	**0.958**	0.951
Wilt	0.474	0.468	0.474	0.469*	0.503*	**0.577***

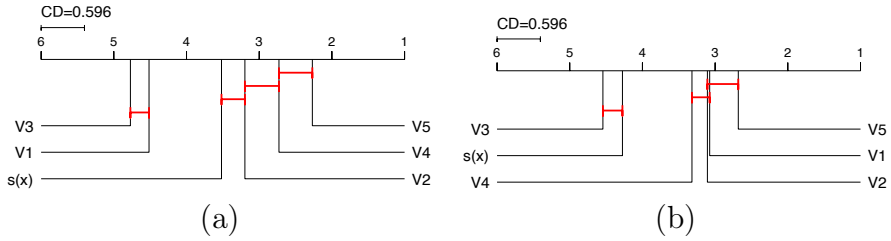


Figure 4.5: CD diagram comparing the 6 scoring functions when employing the standard parametrization with the exception of the depth set to (a) $D = \log_2(S)$ and (b) $D = S - 1$.

From Table 4.2 we can infer that there is at least one path-weighted variant significantly outperforming $s(x)$ on 12 out of 16 datasets; with **V5** being the best performing variant, reinforcing the observations made on Figures 4.3 and 4.4. In terms of performance, $V4$ closely follows $V5$. Further, as expected from the analyses on Figures 4.3 and 4.4, $V1$ and $V3$, perform rather poorly compared to the other variants. Lastly, there is no dataset for which $s(x)$ is the best significant choice.

We also globally compare all the proposed weighted variants and $s(x)$ via a non-parametric statistical procedure made of a Friedman test followed by a post-hoc Nemenyi test. To depict the results of such statistical analysis, we employ the CD diagram [35]. For the explanation of this procedure and the motivation behind its adoption, please refer to Chapter 2. We perform such tests using as input the results of Table 4.2 and we visualize the resulting CD diagram in Figure 4.5 (a). In Figure 4.5 (b) we show the results of the tests when the depth parameter is $D = S - 1$ (refer to Appendix B for the related table). We have set the significance level to $\alpha = 0.05$. The CD diagram in Figure 4.5 (a) confirms that whereas $V1$ and $V3$ are the worst performing variants, not even comparable to $s(x)$ by which they are outranked, **V5** is the highest ranked one, comparable only to $V4$, confirming the robustness and persistence of the improvement. Slightly different observations can be made from the CD diagram depicted in 4.5 (b): $s(x)$ is one of the worst variants, and it is comparable to $V3$, and even though **V5** is again the best methodology, it is comparable to a higher number of path-weighted variants with respect to the CD diagram in Figure 4.5 (a).

We can therefore conclude that in general weighting the path is convenient and advantageous, especially if we grow the tree to its maximum depth. In detail, the scoring function achieving the highest performances is **V5**, independently of D, immediately followed by $V4$.

A final note must be made concerning Table 4.2 and in particular for HTTP and Shuttle, two datasets on which all methodologies work well and for which in some cases the difference between two techniques in terms of averaged AUC is very small but statistically significant. We inspected this strange behaviour, discovering that, on these datasets, the AUC of each technique shows very small variations across the different repetitions of the experiment. In other words, even though the difference in terms of AUC is small it is always present: therefore the Wilcoxon signed-rank test, being based on rankings, detects a statistically significant difference. This effect is also observable in Table 4.4.

4.4.3 Comparison of s(x) with p(x)

In this second subsection we make a comparative analysis between $s(x)$ and $p(x)$ which are respectively the classical anomaly score defined in Eq. (2.4) and the novel anomaly score that we propose in Eq. (4.19). The comparison is carried out between the unweighted scoring functions, i.e. we do not compute the weights of the traversed paths.

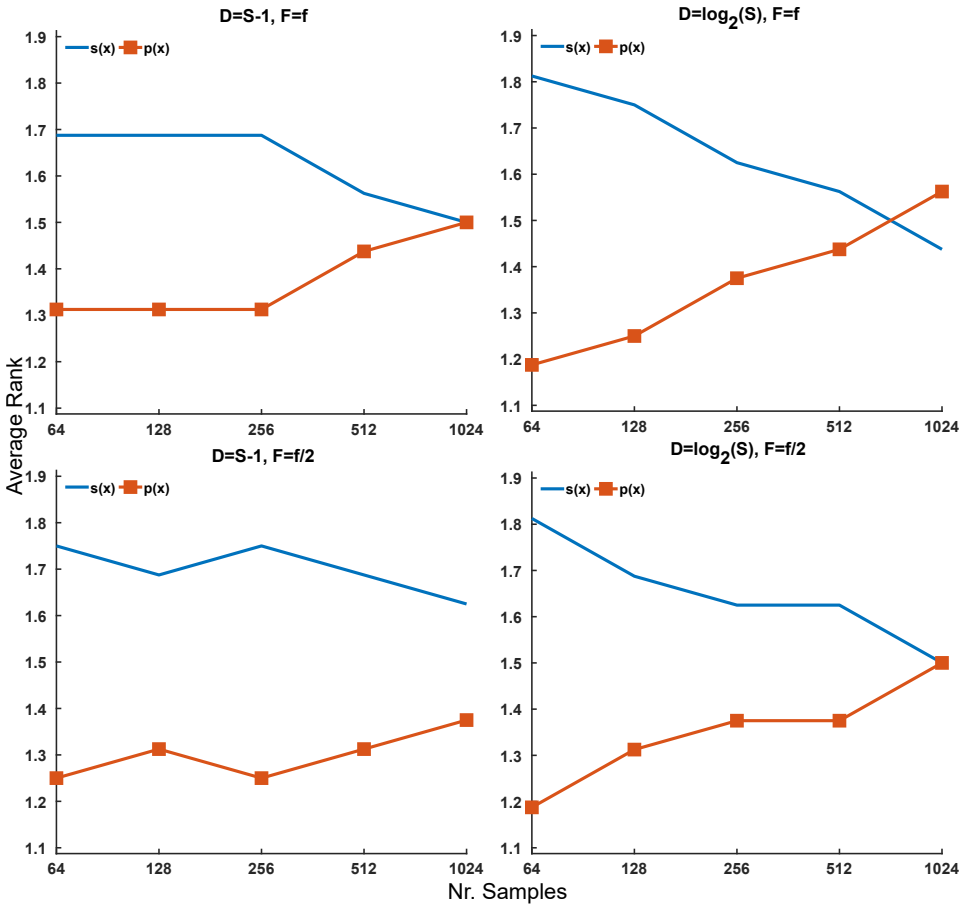


Figure 4.6: Comparison between s(x) and p(x) in terms of the mean rank when varying the training sample size S.

Analogously to the previous analysis, to understand the general behaviour of the two combiner functions $s(x)$ and $p(x)$, we analyze their behaviour as the training sample size S of the trained iForest and its size T changes. These analyses are depicted respectively in Figures 4.6 and 4.7. The averaged rank is measured analogously to subsection 4.4.2. From Figure 4.6 we can infer that, independently of the employed parametrization, the refined proposed combiner function has a higher rank than $s(x)$ in all cases but three. Analogously to what observed in Figure 4.3, the rank of $s(x)$ increases as S does. We can make analogous considerations when analyzing the behaviour of $s(x)$ and $p(x)$ as T varies, as observable in Figure 4.7.

Further, in Table 4.3 we present the AUC results for the two unweighted anomaly scores for each dataset, when employing the default Isolation Forest parametrization: $S = 256, D = \log_2(S)$, $T = 100$, $F = f$ [96, 97]. The last row reports the mean rank. Analogously to subsection 4.4.2, to assess whether differences in performances are statistically significant, we performed a Wilcoxon signed-rank test with $\alpha = 0.05$. For each dataset, if the alternative hypothesis is accepted, we put in **bold** the best result. Observing Table 4.3

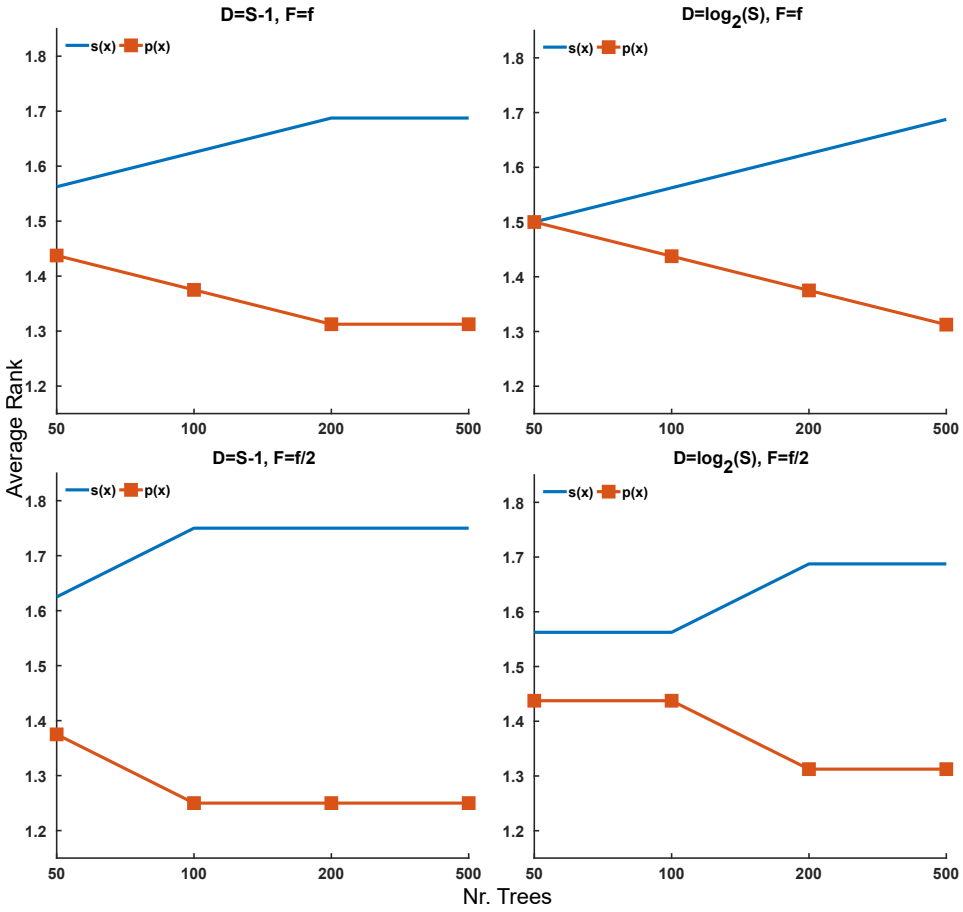


Figure 4.7: Comparison between s(x) and p(x) in terms of the mean rank when varying the forest size T.

we can infer that $p(x)$ is the best choice for most datasets, as confirmed by the mean rank. This consideration also validates the conclusions made on Figures 4.6 and 4.7.

Table 4.3: Comparison between s(x) and p(x) when using the standard iForest parametrization.

Dataset	**s(x)**	**p(x)**
Adult	0.657	0.656
Annthyroid	0.915	**0.927**
Arrhythmia	0.758	0.767
Cardiotocography	0.755	0.747
ForestCover	0.841	**0.925**
Hepatitis	0.701	**0.742**
Http	0.991	**0.993**
Ionosphere	0.905	**0.934**
PageBlocks	0.802	**0.843**
Pendigits	**0.799**	0.784
Pima	**0.733**	0.703
Shuttle	0.996	**0.997**
Smtp	0.908	0.910
Spambase	0.844	0.835
Stamps	**0.957**	0.949
Wilt	0.474	**0.531**
Ranks	1.62	**1.38**

4.4.4 Comparison Between p(x) and Path-Weighted Anomaly Scores

The first two subsections of the experimental evaluation have shown respectively the advantages of employing weights to enrich the anomaly score and the improvements that a less strict aggregation function, $p(x)$ can lead to. Considering that $p(x)$ and $s(x)$ differ only in the way they aggregate the tree scores, we can embed the weights defining $V1 - V4$ in the formulation of $p(x)$ (Eq. (4.19)) in the same way as done for $s(x)$. Therefore, this subsection is dedicated to an analysis aimed at understanding whether weighting the path is advantageous also when changing the aggregation rule, i.e. employing $p(x)$ instead of $s(x)$. For the sake of completeness we also include $V5$ (Eq. (4.15)) in the analysis, even though differently from $V1 - V4$ it has its own aggregation function –different from both $s(x)$ and $p(x)$. Therefore, its performance does not change with respect to the experiments presented in Subsection 4.4.2.

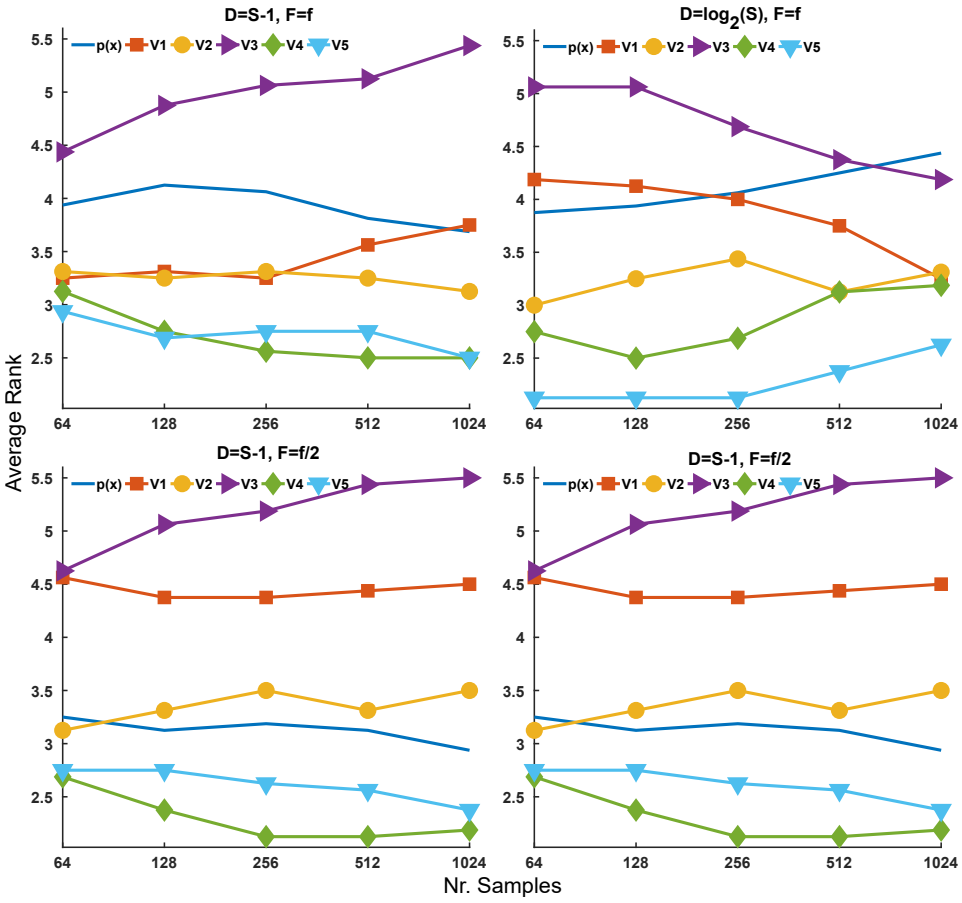


Figure 4.8: Comparison in terms of the mean rank between p(x) and the weighted variants when varying the sample size S.

In Figures 4.8 and 4.9 we analyze the behaviour of the scoring function as S and T respectively vary. The analysis is made in terms of the mean rank, and it is performed and pictured in the same way as done for Subsections 4.4.2

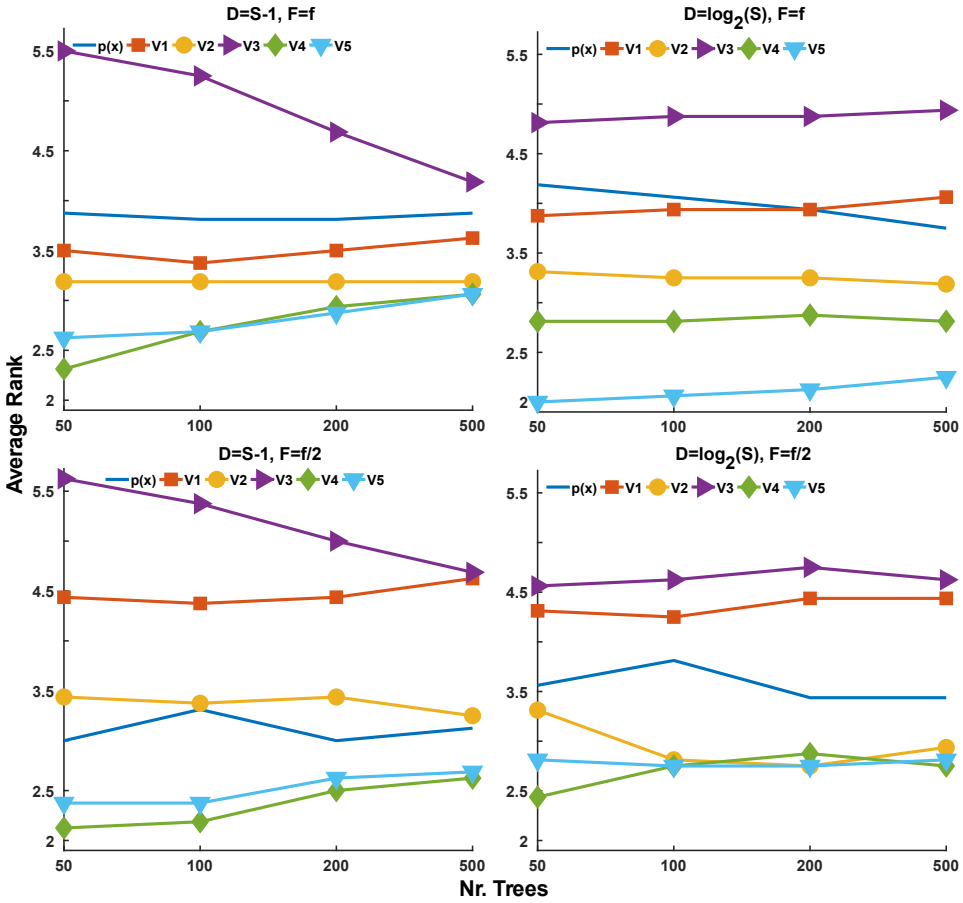


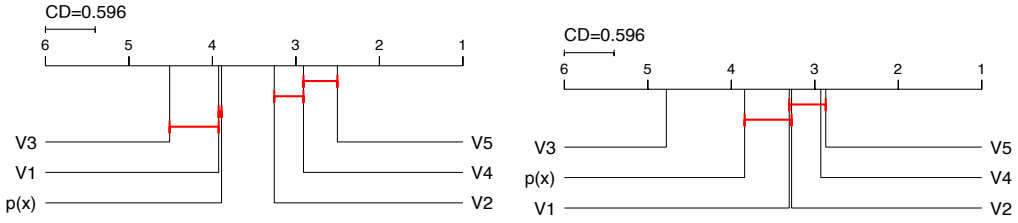
Figure 4.9: Comparison in terms of the mean rank between p(x) and the weighted variants when varying the forest size T.

and 4.4.3. From Figures 4.8 and 4.9 we can make similar observations to those made in Subsection 4.4.2. First, the LCA-based scores, **V4** and **V5**, always outperform the unweighted counterpart. Secondly, the neighborhood-based variants, $V1$ and $V3$, do not seem to lead to improvements. Nevertheless, they perform slightly better than when using $s(x)$ as combiner function. Lastly, from Figure 4.8, we can deduce that the rank of $p(x)$ gets higher as S increases (except one case). Instead, as observed from Figure 4.9, its rank does not have an overall explainable trend in terms of T: indeed the choice of T seems not to have an impact on the rank, except when $F = \frac{f}{2}$ for which $T = 100$ is a very bad setting.

We also analyze how the variants perform dataset-wise on the standard iForest parametrization, as done for the other two analogous analyses in Subsections 4.4.2 and 4.4.3. Table 4.4 contains the averaged AUC across 10 iterations (see Appendix B for the results when using the same parametrization except for $D = S - 1$). The statistical analysis, done via the Wilcoxon signed-rank test, and the visualization of the results is identical to that of Table 4.2. Analogously it holds for the global comparisons shown via the CD diagrams in Figures 4.10 (a) and (b).

Table 4.4: Comparison between $p(x)$ and path-weighted scores when employing the standard iForest parametrization.

Dataset	p(x)	V1	V2	V3	V4	V5
Adult	0.656	0.656	0.656	0.655	0.656	**0.657**
Annthyroid	0.927	0.922	0.928	0.921	0.941*	**0.942***
Arrhythmia	0.767	0.758	0.764	0.757	**0.772**	**0.772**
Cardiotocography	**0.747**	0.744	0.730	0.742	0.746	0.743
ForestCover	0.925	0.850*	0.876*	0.847*	**0.931**	0.927
Hepatitis	0.742	0.711*	0.732	0.697*	0.743	**0.745**
Http	0.993	0.995	**0.996***	0.994	0.994	0.994
Ionosphere	0.934	0.918*	0.938	0.917*	**0.943***	**0.943***
PageBlocks	0.843	0.830*	0.836	0.813*	0.857*	**0.862***
Pendigits	0.784	0.846*	**0.871***	0.84*	0.829*	0.852*
Pima	0.703	**0.729***	0.727*	0.728*	0.710	0.714*
Shuttle	0.997	0.997*	0.997	0.996*	**0.998***	**0.998***
Smtp	0.910	0.922*	**0.923***	0.918	0.918	0.920
Spambase	0.835	0.842	0.841	**0.845**	0.844	**0.845**
Stamps	0.949	**0.954**	0.951	0.953	0.951	0.951
Wilt	0.531	0.512	0.534	0.513	0.575*	**0.577***



(a) (b)

Figure 4.10: CD diagram comparing the 6 scoring functions when employing the standard parametrization with the exception of the depth set to (a) $D = \log_2(S)$ and (b) $D = S - 1$.

We can derive analogous considerations from Table 4.4 to those inferred from Table 4.2. In detail, all variants, except $V1$ and $V3$, perform reasonably well. In addition, using the novel aggregation function seems to lead to a general improvement. The results of the global statistical comparison are depicted in the CD diagram in Figure 4.10 (a), which confirms the observations made in Table 4.4. Indeed, several variants have comparable performances, with $V5$ being the best choice, whereas, as previously observed multiple times, $V1$ and $V3$ perform the worst while being comparable to the unweighted variant $p(x)$. Instead, Figure 4.10 (b), which depicts the CD diagram when $D = S - 1$ (see Appendix B for the related table), conclusions are slightly different: the worst performing variant, $V3$, is no longer comparable to $p(x)$.

4.4.5 Comparison with Extensions of Isolation Forest

In this subsection, we compare the proposed approach to three extensions of the Isolation Forest. Specifically, we considered *SciForest* (SCIF) [98] and two very recent approaches, namely *Extended Isolation Forest* (EIF) [65], and *Generalized Isolation Forest* (GIF) [24]. All three techniques have been thoroughly described in Chapter 2. We trained each technique using their default parametrization, as obtained from the original papers; since there were also parameters with no default value, we performed a preliminary analysis to set them properly. In detail, for EIF we set $T = 200$, $S = 256$, $D = \log_2(S)$ and the extension level to $f - 1$. For SCIF we set $T = 100$, $S = 256$, $D = \log_2(S)$ and $q = 2, \tau = 10$. For GIF, we set $T = 128$, $S = \max(N, 256)$ and $D = \log_2(S)$ where we recall that N is the size of the entire training set. As to the other parameters of GIF, analogously to what done in one of the analyses in [24], we set the kernel to RBF and $\tau = 0.1$; K, i.e. the number of children in which a node can be split, was set to 2, in order to have binary trees like the other methodologies; finally, the scaling value Ss (needed to compute the parameter of the RBF kernel σ) was set to $Ss = 0.75$ –aside from when $\sigma < 0.1$ for which we used a higher value of Ss. To compute the anomaly score in each of the compared methodologies we adopt the scoring function proposed or used by each paper: in [65] they employ $s(x)$, in [98] they propose a modified version of $s(x)$ that removes some of the traversed nodes in the path, as explained in Section 4.1 and in Chapter 2, whereas in [24] they use as score the ratio of training objects ending up in the leaf. As to the proposed approach, we employed our best variant **V5**, training the iForest with the standard parametrization.

Table 4.5 presents the average AUC across 10 iterations for all datasets. Analogously to the previous experimental analyses, we performed a Wilcoxon signed-rank test with $\alpha = 0.05$ adjusted with a Bonferroni correction, denoting with a * a statistically significant difference between the competitor and our approach. Further, we report in **bold** the best result. We globally compare the four methodologies by performing a Friedman test followed by a Nemenyi test with significance level set to $\alpha = 0.05$ –the corresponding CD diagram is shown in Figure 4.11. The analysis of the comparison gives undoubtedly interesting results, showing that standard Isolation Forests can be enriched and improved in two different ways: by using more sophisticated training strategies and by better exploiting the information contained in an Isolation Forest trained in the classical way during the testing phase. As a matter of fact the proposed approach compares very well with alternatives, being for many datasets the best choice; furhter from Figure 4.11 we can observe that it is significantly better than the other approaches[3].

[3]Please note that some of the results shown in Table 4.5 are different from those reported in [24], due to various reasons: we did not perform a specific tuning of the parameters, and we used a different version of the datasets, aside from ForestCover, to be coherent with the experiments in the previous section. In detail, we used a different normalization procedure and outlier percentage.

Table 4.5: Comparison between the proposed approach and other methodologies.

Dataset	Proposed Approach	Extended iForest	SciForest	Generalized iForest
Adult	**0.657**	0.639*	0.656	0.634*
Annthyroid	**0.942**	0.715*	0.932	0.679*
Arrhythmia	**0.772**	0.756	0.741*	0.498*
Cardiotocography	0.743	0.696*	**0.769**	0.693
ForestCover	0.927	0.887*	0.834*	**0.948**
Hepatitis	**0.745**	0.713*	0.678*	0.669*
Http	**0.994**	0.991*	0.990*	0.891*
Ionosphere	**0.943**	0.925*	0.889*	0.818*
PageBlocks	0.862	0.799*	**0.923***	0.662*
Pendigits	0.852	0.782*	0.817*	**0.913***
Pima	0.714	**0.73***	0.602*	0.663*
Shuttle	**0.998**	0.993*	**0.998**	0.789*
Smtp	**0.920**	0.899	0.917	0.749*
Spambase	**0.845**	0.604*	0.840	0.604*
Stamps	**0.951**	0.942*	0.945	0.864*
Wilt	**0.577**	0.361*	0.498*	0.355*

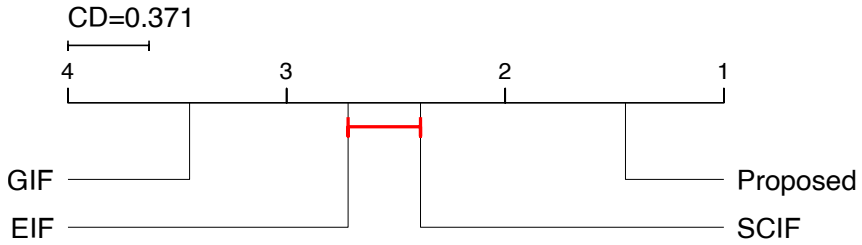


Figure 4.11: CD diagram comparing the proposed approach with extensions of iForest.

4.4.6 Considerations on Complexity

In this subsection, we make some remarks on the complexity of the proposed approach. In detail, we do not make a classical complexity analysis since we aim at providing only some qualitative and quantitative considerations on the computational overhead introduced by the proposed contributions.

First we must highlight that when using $p(x)$ as aggregation function there is no overhead with respect when employing $s(x)$; indeed the two scoring functions exploit the same information, but they combine it in a different way. As to the different path-weighted criteria, we can make the following considerations:

- To calculate $V1$ the only information that is needed is the size of each node, i.e. the number of training objects it contains; since this information is needed to compute the normalization factor of the unweighted anomaly score (see Eq. (2.6) in Chapter 2), the computation of $V1$ does

not cause overhead. Indeed, please note that this information can be easily stored during the training phase.

- As to $V2$ (and $V3$), we have to compute the proxy in each node. More in detail, the volume of each node must be computed: to do that each tree must be traversed again by the set of training objects used to build it. It is evident that this information can be stored as additional information during the training phase.

- To compute both $V4$ and $V5$ we have to calculate the LCA of the testing object and each training object, which require several traversals of each tree –nevertheless this operation can be optimized by exploiting the size of the neighborhood of each node. As to $V5$, there is also an additional operation to make for each weight in the path.

A quantitative illustration of these considerations is made by training an iForest employing the standard parametrization and by recording how much time (in milliseconds) it takes to score a testing object using both the unweighted and all the path-weighted scoring functions. We report in Table 4.6 the time in milliseconds for each variant averaged across all objects in the testing set. The results validate the qualitative considerations with $V5$ leading to the highest overhead; nevertheless, the increase in testing time is highly balanced by the very good performances of the variant in terms of AUC.

Table 4.6: Testing time in milliseconds of each scoring function using the standard parametrization.

Dataset	**s(x)**	**V1**	**V2**	**V3**	**V4**	**V5**
Adult	0.965ms	0.961ms	0.959ms	0.966ms	1.100ms	1.810ms
Annthyroid	0.978ms	0.988ms	1.010ms	0.993ms	1.150ms	1.860ms
Arrhythmia	0.838ms	0.836ms	1.070ms	1.080ms	1.280ms	1.810ms
Cardiotocography	0.956ms	0.959ms	1.020ms	1.020ms	1.250ms	2.160ms
ForestCover	0.990ms	1.020ms	0.994ms	0.990ms	1.130ms	2.090ms
Hepatitis	0.708ms	0.669ms	1.230ms	1.270ms	1.830ms	2.220ms
Http	1.000ms	1.030ms	0.992ms	0.995ms	1.100ms	2.120ms
Ionosphere	0.840ms	0.831ms	1.070ms	1.050ms	1.480ms	2.100ms
PageBlocks	0.894ms	0.898ms	0.921ms	0.911ms	1.050ms	1.780ms
Pendigits	0.982ms	0.971ms	0.991ms	1.000ms	1.180ms	2.490ms
Pima	0.988ms	0.977ms	1.160ms	1.130ms	1.520ms	2.420ms
Shuttle	0.957ms	0.972ms	0.981ms	0.976ms	1.100ms	2.000ms
Smtp	0.998ms	0.997ms	1.000ms	0.999ms	1.110ms	1.920ms
Spambase	1.120ms	0.976ms	1.000ms	0.992ms	1.290ms	1.800ms
Stamps	1.070ms	1.030ms	1.310ms	1.290ms	1.910ms	2.630ms
Wilt	0.964ms	1.010ms	1.030ms	1.030ms	1.530ms	2.440ms
Average	0.953ms	0.945ms	1.050ms	1.040ms	1.310ms	2.100ms

4.5 Conclusions and Future Work

In this chapter, we made two contributions to the field of isolation-based methodologies for outlier detection. In detail, we propose alternative testing phases to those used by Isolation Forests and most of its extensions. Our proposal focuses on the definition of several novel anomaly scores, and we can distinguish them into two separate contributions. The first contribution is aimed at improving the tree-scores by adding information that is innate in the tree structures via a weighting procedure of the traversed nodes. We propose five different variants. The second contribution consists of a novel aggregation function to combine the scores at forest level, supported by a probabilistic interpretation of the tree and by ensemble learning theory.

A thorough evaluation of the novel scoring functions has been made by testing the methodology on sixteen datasets, showing that most of the proposed variants are robust and well performing across several, if not all, parametrizations of the Isolation Forest. After identifying a candidate best variant, we also performed a comparison with other isolation-based methodologies, which have shown encouraging results –even though the compared techniques are really sophisticated in terms of the tree structure. Another relevant observation is that improvements are observed even by solely changing the aggregation scheme, i.e. using another unweighted variant.

We can also observe that the proposed scoring functions are easy to use and to interpret, independently of the domain of the problem at hand. In detail, the meaning behind each novel anomaly score is analogous to the standard one, i.e. given a testing object, a higher value indicates a higher probability of being an outlier. Further, the approach is easy to use: as to the scoring functions there are no parameters to set, and as to the training of the iForest, we can use the default parametrization which goodness has been proved throughout the experimental section. Lastly, if the user wants to adopt only one scoring function, we have thoroughly shown that LCA-based variants are the best choice, especially if combined –if feasible– with the novel aggregation function.

Some further aspects could be investigated in future researches. The first aspect to investigate would be to analyze more thoroughly the neighborhood-based variants to understand why they lack in terms of performances: maybe a starting point could be to verify whether assigning a lower weight to big nodes is the most informative choice for outlier detection. Another interesting idea would be to combine the proposed scoring functions with other isolation-based training procedures –such as those, but not exclusively, used in the comparative analysis. Nevertheless, some weights may require to be adapted, or further ones would need to be defined, since the underlying tree structures may be quite different from the standard Isolation Forest –even though we expect that the proposed variants should work relatively well since all methods are isolation-method. Lastly, another research direction to pursue is the study of other combiner rules to provide even more robust estimates of the anomaly scores at forest level. Some additional details on these ideas are provided in Chapter 7.

Chapter 5

iForest distances for Outlier Detection

In this chapter we propose an alternative way to carry out an outlier detection task: we extract pairwise distances from a trained Isolation Forest and then the resulting dissimilarity matrix is input to a distance-based outlier detector. First we present the intuition behind the contribution and then the proposed approach, with a particular focus on the used distance measures.

5.1 Introduction

A natural way to describe objects is via the relationship they have with other patterns. The relationship is usually encoded in terms of similarity (or dissimilarity), i.e. how similar (different) two objects are. Similarity is helpful not only to analyze pairwise relationships, but also to group objects and uncover the principles that tie them together: indeed in Pattern Recognition a class is a set of similar objects. Further, as thoroughly explained in Chapter 3, we can compute distances between objects of any type, including non-vectorial ones, prior to an adequate choice of the distance measure.

Indeed, dissimilarities can be computed via many different types of measures such as: geometric distance measures [27, 132], application-related measures [116], information theoretic measures [26, 94, 150], etc. Among them, geometric distances are the most used: they compute the dissimilarity between a pair of objects based on the position these objects have in the space. Geometric distances have also another fundamental property: they are metrics. A metric is a distance measure defined on a set of objects X where for any $x, y, z \in X$ the following properties hold [132]:

1. Non-negativeness: $d(x, y) \geq 0$.
2. Identity of Indiscernibles: $d(x, y) = 0 \iff x = y$.
3. Reflexivity: $d(x, y) = d(y, x)$.
4. Triangle inequality: $d(x, y) \leq d(x, z) + d(y, z)$.

where $d(x, y)$ is the pairwise distance between x and y. An example of geometric distance is the well-known *Euclidean distance*: the distance between two vectors $\mathbf{x}$ and $\mathbf{y}$ on one feature is the length of the line connecting the representation of that feature of the two objects in a Euclidean space. The concept can

be generalized to any number of features, each treated independently. Even though very intuitive from a mathematical perspective, the mathematical nature of geometric distances also hinders some problems [7] that can occur in some particular cases:

- Consider a dataset that is highly dimensional, i.e. each object is represented by many features, if we compute the Euclidean distance between two objects of such dataset it is likely that they will turn out to be at the same distance as many other pairs of objects. This phenomenon occurs when the objects are sparsely distributed, which is more likely to happen when features are many –compared to the size of the dataset. With objects being equidistant, it is difficult to establish the true nature of the relationship between a pair of objects.

- Another issue that may arise in some contexts is the unsuitability of using a geometric distance measure: satisfying mathematical constraints can be limiting and non-adequate to correctly represent the relation between two objects. In other words, the real meaning of their relationship can be lost. Let us clarify this concept with an example: we have two electrocardiograms sequences of different lengths, and we want to compute their distance. To use the Euclidean distance, which works only with vectors of equal length, we have to modify the sequences by adding artificial time points. This can obviously result in a higher or lower distance value than hypothesized. A solution is to choose a more adequate distance measure which takes into account the nature of the electrocardiograms, i.e. a sequence in time. For example, instead of the Euclidean distance, we could use Dynamic Time Warping (DTW) [116] a non-metric distance measure which manages sequences of different lengths –among having other nice properties that allow a correct characterization of the distance between the two sequences. Aside from DTW, in the literature there are plenty of distance measures which are non-metrics, many of which have been defined to manage non-vectorial representations.

- Another problem is that geometric distance measures tend to be sensitive to units and scale [7]: deciding which normalization and standardization procedure to carry out can be impacting on the final result.

- Lastly but not least importantly, these geometric distances do not consider how data are distributed, i.e. the context: the only factor taken into account to measure the distance between two objects, are the objects themselves.

Several studies [7, 8, 15, 48, 94, 142, 149, 151, 167] have focused on the above limitations and tried to propose a more inclusive definition of similarity. Some of these works make assumptions on the properties that these ideal measures should satisfy, other design new distance measures. The core concept of all these studies is that similarities should be *data-dependent*: a correct

similarity takes into account also the context. An example is to consider the density of the space where the objects are, either implicitly or explicitly: given two pairs of objects which are equally similar according to the Euclidean distance, if the first pair is surrounded by fewer objects than the latter, then the former pair of objects should have a higher similarity. In addition to data-dependence, some of the studies cited above [14, 94, 151, 167] make another assumption: to truly capture how similar two objects are, rather than simply computing a measure on the features, the similarity computation should account for the common characteristics and those that make the two objects different [151].

Many data-dependent similarity measures that have been proposed are defined on Random Forests [15, 48, 142, 149, 167]. The intuition behind using Random Forests is that they are a flexible data description tool. In detail, a tree in an RF is defined by several tests and each test describes a way to partition the data, usually based on how data are distributed. In other words, an RF implicitly encodes the relationship between two objects based on how they are partitioned by each tree, i.e. based on which nodes of the tree the two objects traverse. In detail, the general procedure to extract the distance between two objects in a tree consists of the following steps: i) make both objects traverse the tree and retrieve their paths; ii) compare their paths based on some function to extract their distance. In general, if the paths of the two objects are highly similar, it means that they are close in the space with respect to how the tree partitioned the input data.

These characteristics make an RF not only an excellent tool for classification or regression, but also a valid and flexible distance extractor tool. This is confirmed by the variety of successful clustering methodologies that compute the clusters starting from the pairwise distances extracted from an RF [15, 85, 142, 149, 167]: indeed, objects belonging to the same cluster are more similar, and they should traverse together an increasing amount of nodes proportional to their similarity. These measures, which we denote as *RF-distances*, have been proven to be more descriptive than standard geometric-based distances such as the Euclidean distance, and have been successfully applied in different domains [1, 61, 130, 141].

Since many outlier-based methodologies work with distances to detect outliers, it would be interesting to investigate the use of more informative measures than geometric distances, such as RF-distances. It is evident that we must use a tree structure suitable for an outlier detection task, i.e. it must be unsupervised and able to manage the presence of outliers: we can employ Isolation Trees [96, 97].

Therefore, in this chapter we propose an alternative use of the Isolation Forests, which can be interpreted as an alternative testing phase: rather than detecting outliers via the isolation-based anomaly score, we train an Isolation Forest to compute pairwise data-dependent dissimilarities between objects. The resulting distance matrix is then input to an outlier detector that works with distances. We investigated several dissimilarity measures [7, 15, 142, 149,

167], including very recent ones [15]. Our proposal, which aims at showing the suitability and informativeness of RF-distance measures, is innovative and to our knowledge there is no other study about RF-distances for outlier detection. The only exception is [149] which nevertheless focuses only on one RF-distance measure and on one outlier detector, whereas our study has a much larger scope, analyzing several RF-distances for various types of outlier detectors.

The proposed approach is suitable in all those scenarios in which outliers tend to be far away from the inlier distribution and standard geometric distances seem not to perform well, i.e. there is space for improvements. Let us consider the following example: we have a small set of healthy subjects described by thousands of genes. We aim at discovering whether there are some subjects suffering from a rare genetic disease, which can be identified by an unlikely expression pattern of an unknown subset of genes. To detect these putative diseased subjects, i.e. outliers, we can compare the genetic profiles via pairwise distances. However, as previously mentioned, geometric distances are not powerful in high-dimensional spaces. Instead, via using an RF-distance measure, extracted from an iForest, we are more likely to capture those subspaces, i.e. subset of genes, in which the anomalous behaviour is present.

The evaluation of the contribution is divided into two parts. The first part consists of a preliminary analysis, which compares the proposed framework to standard iForest on 10 outlier detection problems using the Local Outlier Factor methodology. Results were promising, confirming again the suitability of the Isolation Tree structure to capture outliers. The second analysis instead focuses on a different pool of datasets, a subset of those used in Chapter 4, and it is much more extended than the first one, focusing on more distances and outlier detectors. Results assess its robustness with respect to iForests, confirming the suitability of using RF-distances. In addition, we also compare the proposed framework to geometric distance measures which results were promising only in some cases, demonstrating that some further research must be done in the context of RF-distances for outlier detection. The methodology, comprehensive of the preliminary analysis, i.e. first batch of experiments, was published in Lecture Notes of Computer Science, as proceedings of the Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR) [111]. A more complete version of the proposed approach has been published in the proceedings of the International Conference on Image Analysis and Processing (ICIAP) [110].

This chapter is organized into several sections. Section 5.2 presents the proposed methodology, which is divided into three steps, each thoroughly described. In Section 5.3 we make a thorough experimental evaluation, which is divided into two pools of experiments, the latter being more extended than the first one, based on the promising results obtained in the first part. In Section 5.4 we make some conclusions and illustrate some future ideas.

5.2 Methodology

This section presents the proposed methodology: starting from a trained Isolation Forest, we extract for each object in the dataset its RF-distance to the remainder of the data and input the obtained distance matrix to an outlier detector.

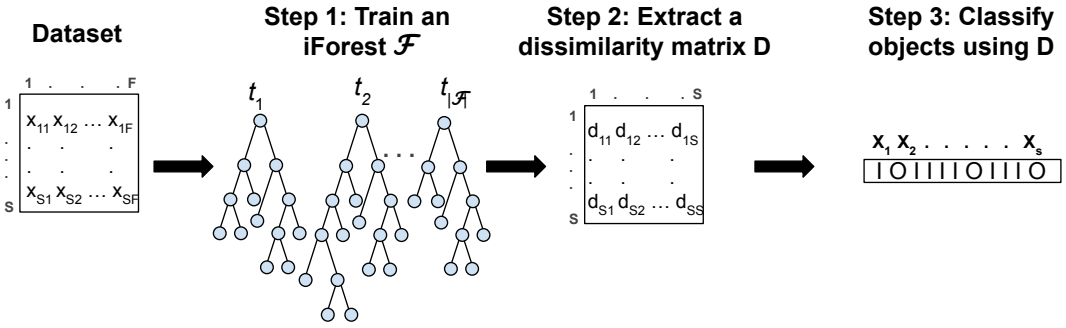


Figure 5.1: Pipeline of the proposed methodology.

The pipeline of the proposed technique is illustrated in Figure 5.1. The methodology is divided into three steps:

1. Train an Isolation Forest $\mathcal{F}$ on a vectorial dataset of size N.
2. From the trained $\mathcal{F}$ extract a dissimilarity matrix $\mathbf{D}$. The cell in the i^{th} row and j^{th} column contains the pairwise distance $d(\mathbf{x}, \mathbf{y})$ between $\mathbf{x}$ and $\mathbf{y}$.
3. Input $\mathbf{D}$ to any outlier detector that works with distances and then classify the objects of the dataset, as depicted in Step 3 of Figure 5.1 –where with I and O we respectively indicate inliers and outliers.

In the next sections, we describe each step in detail.

5.2.1 Step 1: Training of iForest

The first step consists of training an Isolation Forest. The choice of adopting iForest is based on the following reasoning: the RF model must be unsupervised since we do not have labels. Several unsupervised learning strategies for RF have been proposed in [16] aimed at extracting distances for clustering purposes: they show that building trees via completely random splits is beneficial, thus supporting our choice of adopting iForests.

We train an Isolation Forest $\mathcal{F}$ as thoroughly described in Chapter 2: recall that $\mathcal{F}$ is made up of T trees, each built independently on S objects sampled without replacement from the training set $\mathcal{O}$ of size N. Please recall that an iForest works only with vectorial representations. The building procedure of each tree is recursive: a node n is split, according to a test defined completely at random, and the process is repeated until a maximum depth D is reached,

i.e. the node n becomes a leaf. We recall the notation for the following elements within a tree t, which are needed for the sake of comprehension of the remainder of the proposal:

- *root* is the root node of the tree, which is composed by S objects.
- n is either an internal node of the tree, i.e. a node which has two children and is not the root, or a leaf node.
- $|n|$ is the size of a node, i.e. the number of objects it contains. Please note that each node n contains $< S$ objects.
- $dp()$ is the depth function which retrieves the depth of each node, where $dp(root) = 0$.

5.2.2 Step 2: Extracting the Distance Matrix D

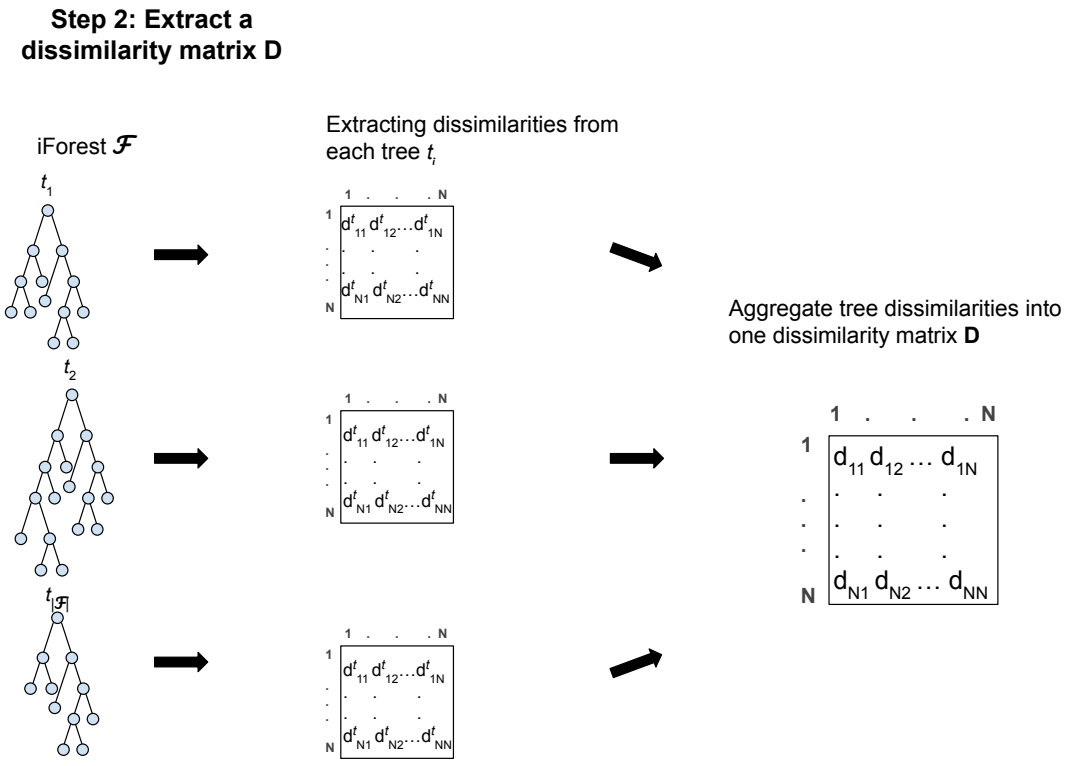


Figure 5.2: Step 2: Details on how to extract the dissimilarity matrix **D**.

The second step consists in the computation of the matrix **D**, as described in Figure 5.2: given $\mathcal{F}$ we make all pairs of objects in the dataset traverse a tree and compute their distance; the procedure is repeated for all trees; finally the tree distances are aggregated at forest level. We compute **D** using six different proposals of RF-distances [7, 8, 15, 142, 149, 167] which have shown to be successful in the clustering scenario [1, 15, 130, 141, 142, 149, 167].

Please note that these distances were never employed in the context of outlier detection aside from [149] in which, as mentioned in the previous section, only one distance measure is evaluated on one outlier detector.

Before presenting each RF-distance measure in detail, we define/recall some useful notation: please note that some terms have already been defined in the previous chapters, and we recall them below for the sake of completeness and comprehension.
Given a pair of objects $\mathbf{x}$ and $\mathbf{y}$ that traverse a tree t, we define:

- $l_t(\mathbf{x})$ as the leaf node reached by $\mathbf{x}$ and the depth of such node as $dp(\mathbf{x}) = dp(l_t(\mathbf{x}))$.
- $\mathcal{P}_t(\mathbf{x}) = \{n_1, n_2, \dots n_{dp(\mathbf{x})}\}$ is the path traversed by $\mathbf{x}$ in t, defined in terms of the set of traversed nodes. We exclude the root since it is the starting point for all objects, i.e. it is traversed by all of them. Given this assumption, note that $dp(\mathbf{x}) = |\mathcal{P}_t(\mathbf{x})|$.
- $LCA_t(\mathbf{x}, \mathbf{y})$ as the lowest common ancestor of $\mathbf{x}$ and $\mathbf{y}$ in t, i.e. the last node that $\mathbf{x}$ and $\mathbf{y}$ both traverse. In general, the LCA node is characterized by the split separating $\mathbf{x}$ from $\mathbf{y}$. Please note that $LCA_t(\mathbf{x}, \mathbf{y})$ can also coincide with the root node, i.e. $\mathbf{x}$ and $\mathbf{y}$ disagree on the first split; whereas the other extreme consists of $LCA_t(\mathbf{x}, \mathbf{y}) = l_t(\mathbf{x}) = l_t(\mathbf{y})$, i.e. $\mathbf{x}$ and $\mathbf{y}$ traverse the same path.
- $\lambda_t(\mathbf{x}, \mathbf{y}) = dp(LCA_t(\mathbf{x}, \mathbf{y}))$ is a shortcut notation for the depth of the LCA of $\mathbf{x}$ and $\mathbf{y}$ in tree t.
- $\mathcal{P}_t(\mathbf{x}, \mathbf{y}) = \{n_1, ..., LCA_t(\mathbf{x}, \mathbf{y})\}$ is the common path of objects $\mathbf{x}$ and $\mathbf{y}$, i.e. the subset of nodes traversed by both $\mathbf{x}$ and $\mathbf{y}$ in a given tree t. Note that $\lambda_t(\mathbf{x}, \mathbf{y}) = |\mathcal{P}_t(\mathbf{x}, \mathbf{y})|$.
- $c_{tx}(\mathbf{y}) = |\{n | n \in \mathcal{P}_t(\mathbf{x}) \wedge b(n, \mathbf{y}) = b(n, \mathbf{x})\}|$ where $b(n, \mathbf{x})$ is a function evaluating the answer of $\mathbf{x}$ (true or false) to the test defined in node n. In other words, it is the set of nodes in the path traversed by $\mathbf{x}$ on which $\mathbf{x}$ and $\mathbf{y}$ agree. Please note that if $\mathcal{P}_t(\mathbf{x}) = \mathcal{P}_t(\mathbf{y})$ then $c_{tx}(\mathbf{y}) = c_{ty}(\mathbf{x})$ and additionally $c_{tx}(\mathbf{y}) = dp(\mathbf{y}) = dp(\mathbf{x})$.

In the following, we introduce the six distance measures employed in our approach.

Distance 1: Shi

The first RF-distance measure we used was proposed by Breiman [17] in the context of decision trees, and it was used for the first time in the PAM algorithm in 1990 [85]. The computation of this measure at RF level was done for the first time only in 2005 for clustering purposes [142]. The principle behind this measure, which we denote as **Shi**, is that two objects $\mathbf{x}$ and $\mathbf{y}$ are similar if they end up in the same leaf in the tree, i.e. they traverse the same

path. This is a binary similarity measure, since objects can be either similar or dissimilar, the latter happening whenever $\mathbf{x}$ and $\mathbf{y}$ end up in different leaves. To strengthen this estimate, we can compute the similarity in all trees and aggregate it at forest level. Formally, given a forest $\mathcal{F}$ and two objects $\mathbf{x}$ and $\mathbf{y}$ the *SimShi* similarity measure is defined as:

$$SimShi(\mathbf{x}, \mathbf{y}) = \frac{\sum_{t \in \mathcal{F}} \mathbb{1}(l_t(\mathbf{x}) = l_t(\mathbf{y}))}{T} \tag{5.1}$$

where $\mathbb{1}$ is an indicator function returning 1 if there is a correspondence between the two leaves. Then it is transformed into a dissimilarity as follows:

$$Shi(\mathbf{x}, \mathbf{y}) = \sqrt{1 - SimShi(\mathbf{x}, \mathbf{y})}. \tag{5.2}$$

In other words, it is the square root of the number of trees in which $\mathbf{x}$ and $\mathbf{y}$ end up in different leaves.

This measure is rather approximate: if a tree is badly built, it is likely that it will not capture the existing similarity between two objects, thus lowering the similarity at forest level. Nevertheless, the core principle underlying the Shi distance is very robust, and it has been used successfully in a lot of applications [1, 61, 130, 141].

Distance 2: Zhu2

In [167] the starting point is to define an improved distance measure with respect to *Shi*. Indeed, the authors propose a generalization for *Shi* and define three different variants. Their aim is to use the extracted distances to build affinity graphs for spectral clustering. The first measure proposed in [167] is called *ClustRF-Bi* which corresponds to *Shi*. The other two measures are named *ClustRF-Strct-Unfm* and *ClustRF-Strct-Adpt*; for the sake of simplicity, we rename them as *SimZhu2* and *SimZhu3* respectively. *SimZhu2* starts from the assumption that also objects that do not end up in the same leaf may be similar: the degree of similarity is related to the length of the common path, i.e. the number of nodes that two objects traverse together before they go separate ways. In other words, the higher the number of nodes a pair of objects both traverse, the higher their similarity. Formally, we define $SimZhu2_t(\mathbf{x}, \mathbf{y})$ as

$$SimZhu2_t(\mathbf{x}, \mathbf{y}) = \frac{\lambda_t(\mathbf{x}, \mathbf{y})}{\max\{|\mathcal{P}_t(\mathbf{x})|, |\mathcal{P}_t(\mathbf{y})|\}}. \tag{5.3}$$

The denominator is a normalization factor needed to take into account that the importance of the LCA, as also explained in Chapter 4 in another scenario, depends on the length of the longest of the two paths. In other words, two objects $\mathbf{x}$ and $\mathbf{y}$ which LCA is at depth λ and $\max(dp(\mathbf{x}), dp(\mathbf{y})) = q$ are more similar than another pair of objects $\mathbf{m}$ and $\mathbf{n}$ which LCA is at the same depth λ but $\max(dp(\mathbf{m}), dp(\mathbf{n})) = r > q$, that is for which there is a higher number

of nodes to traverse, starting from the LCA, to reach the deepest of the leaves, i.e. $r > q \Rightarrow r - \lambda > q - \lambda$.

The measure at forest level is obtained by averaging $SimZhu2_t(\mathbf{x}, \mathbf{y})$ across all trees in the forest, as defined in the following:

$$SimZhu2(\mathbf{x}, \mathbf{y}) = \frac{\sum_{t \in \mathcal{F}} SimZhu2_t(\mathbf{x}, \mathbf{y})}{T}. \tag{5.4}$$

$SimZhu2(\mathbf{x}, \mathbf{y})$ can be easily turned into a dissimilarity measure, **Zhu2**, in the following way:

$$Zhu2(\mathbf{x}, \mathbf{y}) = 1 - SimZhu2(\mathbf{x}, \mathbf{y}). \tag{5.5}$$

Distance 3: Zhu3

The other measure proposed by [167], *SimZhu3* is a weighted version of *SimZhu2*. Similarly to what we did in Chapter 4, patterns that are found in the same node which is very deep in the tree are more alike than patterns that are together only in the first nodes of the tree, e.g. the root. Therefore, we can consider each node n to have a depth-based importance, which can be encoded as a weight. In detail, we define the weight of a node n to be $\frac{1}{|n|}$ which is inversely proportional to the node size –since smaller nodes have a greater relevance. Given this concept, the formal definition of the similarity measure $SimZhu3_t$ between $\mathbf{x}$ and $\mathbf{y}$ is the following:

$$SimZhu3_t(\mathbf{x}, \mathbf{y}) = \frac{\sum_{n \in \mathcal{P}_t(\mathbf{x},\mathbf{y})} \frac{1}{|n|}}{\sum_{n \in \mathcal{P}_t(\mathbf{b})} \frac{1}{|n|} + \frac{1}{|l_t(\mathbf{b})|}} \tag{5.6}$$

where $\mathbf{b} = \underset{\mathbf{i} \in \{\mathbf{x},\mathbf{y}\}}{\operatorname{argmax}} |\mathcal{P}_t(\mathbf{i})|$. The measure at forest level, $SimZhu3$ can be computed, analogously to the other variant, by averaging the single similarities obtained by traversing each tree:

$$SimZhu3(\mathbf{x}, \mathbf{y}) = \frac{\sum_{t \in \mathcal{F}} SimZhu3_t(\mathbf{x}, \mathbf{y})}{T}. \tag{5.7}$$

We can transform $SimZhu3$ into a distance measure, **Zhu3**, in the same way as done for $Zhu2$:

$$Zhu3(\mathbf{x}, \mathbf{y}) = 1 - SimZhu3(\mathbf{x}, \mathbf{y}). \tag{5.8}$$

Distance 4: RatioRF

In a recent work [15] we highlighted how all these measures are actually an implicit, but partial, implementation of the principles of similarity proposed by Tversky [151]. Tversky was one of the first to propose a novel model of similarity aimed at focusing on the human perception of similarity instead of on the geometric-related concept. In detail, his model, known as *Tversky*

ratio model of similarity, makes the following assumptions: i) objects should be represented only by those features necessary for estimating the pairwise similarity; ii) a feature is deemed as necessary if it is either an element of commonality between two objects or an element of difference, i.e. it describes only one object between the two. Indeed, such features are those needed for the similarity computation.

In [15] we propose an RF-based similarity measure, called **RatioRF**, that satisfies both assumptions of the Tversky's ratio model of similarity. This distance has shown its superiority to alternatives in the clustering scenario; here we also test it for outlier detection. In detail, in [15] we define the set of relevant features for an object $\mathbf{x}$ as the set of tests defining each node in its path $\mathcal{P}_t(\mathbf{x})$. Further, RatioRF takes into account both the features that are common and those that make up the difference between the two objects, via encoding the following notions in the similarity computation:

- Nodes that are in $\mathcal{P}_t(\mathbf{x}, \mathbf{y})$ are elements of commonality, i.e. the longer the common path the higher the similarity.
- Nodes that are in the remainder of the path traversed by $\mathbf{x}$, $\mathcal{P}_t(\mathbf{x}) - \mathcal{P}_t(\mathbf{x}, \mathbf{y})$, to which $\mathbf{x}$ and $\mathbf{y}$ answer in the same way, i.e. they would follow the same edge, are elements of commonality as well, accounting for a higher similarity.
- Nodes that are in the remainder of the path traversed by $\mathbf{y}$, $\mathcal{P}_t(\mathbf{y}) - \mathcal{P}_t(\mathbf{x}, \mathbf{y})$, to which $\mathbf{x}$ and $\mathbf{y}$ answer in the same way, i.e. they would follow the same edge, are elements of commonality as well, accounting for a higher similarity. This is analogous to the previous point.
- Nodes that are in the remainder of the path traversed by $\mathbf{x}$, $\mathcal{P}_t(\mathbf{x}) - \mathcal{P}_t(\mathbf{x}, \mathbf{y})$, to which $\mathbf{x}$ and $\mathbf{y}$ answer differently, i.e. they would follow different edges, are elements of difference, accounting for a higher dissimilarity. In other words, these nodes are relevant in describing *what is* $\mathbf{x}$ and in describing *what is not* $\mathbf{x}$.
- Nodes that are in the remainder of the path traversed by $\mathbf{y}$, $\mathcal{P}_t(\mathbf{y}) - \mathcal{P}_t(\mathbf{x}, \mathbf{y})$, to which $\mathbf{x}$ and $\mathbf{y}$ answer differently, i.e. they would follow different edges, are elements of difference, accounting for a higher dissimilarity. In other words, these nodes are relevant in describing *what is* $\mathbf{y}$ and in describing *what is not* $\mathbf{y}$. This is analogous to the previous point.

In other words, differently from the previously defined distances, i.e. Shi, Zhu2 and Zhu3, RatioRF takes into account also nodes that are not in the common path but that are used for describing what is $\mathbf{x}$ and what is $\mathbf{y}$, i.e. those nodes that are at least in one of the two paths. Formally, we define this similarity

measure within a tree t as follows[1]:

$$SimRatioDT_t(\mathbf{x}, \mathbf{y}) = \frac{c_{tx}(\mathbf{y}) + c_{ty}(\mathbf{x}) - \lambda_t(\mathbf{x}, \mathbf{y})}{dp(\mathbf{x}) + dp(\mathbf{y}) - \lambda_t(\mathbf{x}, \mathbf{y})}. \tag{5.9}$$

Analogously to the other RF-dissimilarity measures, when dealing with a forest we simply compute the average:

$$SimRatioRF(\mathbf{x}, \mathbf{y}) = \frac{\sum_{t \in \mathcal{F}} SimRatioDT_t(x, y)}{T}. \tag{5.10}$$

Then it is transformed into a distance by computing:

$$RatioRF(x, y) = \sqrt{1 - SimRatioRF(x, y)}. \tag{5.11}$$

In [15] we do not only thoroughly define the novel measure RatioRF, but we also make a qualitative comparison with other data-dependent measures [94, 167], highlighting the common points, the differences and the limitations with respect to our methodology. Further, the experimental evaluation is very thorough: the approach is tested on a great variety of clustering algorithms and datasets, and it is compared to other RF-distance measures. The obtained results in the clustering scenario are highly competitive and validate the theoretical assumptions about RatioRF being more accurate than other RF-distances.

Distance 5: Ting

The first four measures we described compute the similarity by counting the nodes that are relevant for its definition. In addition, they share another characteristic: independently of the leaf an object ends up into, the self-similarity is the same, i.e. it is independent of the region of the space defined by the leaf. There is another category of dissimilarities, not strictly related to RF, that overcomes this limitation, allowing an object to have a different self-similarity depending on where it ends up in the space. These measures, defined in [7, 8], are called mass-based data-dependent dissimilarities. The main principle behind both measures is that the similarity between two objects $\mathbf{x}$ and $\mathbf{y}$ can be estimated via the probability data mass of the minimum region enclosing both objects. In addition, since objects are usually defined by many characteristics, i.e. several dimensions, this probability is computed independently for each dimension and then the measure is aggregated across all of them. Formally in [8] they define the m_p-*dissimilarity* where $p \geq 1$ as:

$$m_p(\mathbf{x}, \mathbf{y}) = \left(\frac{1}{M} \sum_{i=1}^{M} \left(\frac{|R_i(\mathbf{x}, \mathbf{y})|}{N}\right)^p\right)^{\frac{1}{p}} \tag{5.12}$$

[1]Please note that we are reporting the definition according to the same notation used for defining the other measures, for a more general formulation refer to [15].

where $R_i(\mathbf{x}, \mathbf{y})$ is the minimum region along dimension i enclosing both $\mathbf{x}$ and $\mathbf{y}$. In other words, it is the region covering the range $[\min(x_i, y_i) \max(x_i, y_i)]$. The term $|R_i(\mathbf{x}, \mathbf{y})|$ indicates that the probability data mass of a region can be estimated by the number of objects enclosed in it. The term N is necessary for comparison purposes. The p factor is analogous to the one of the l_p norm, e.g. for the Euclidean distance $p = 2$.

This measure has also been employed in the context of RF in [149] leading to the Ting measure we used in our experiments. In detail, they implement [8] by making the following analogies: each dimension is a tree ($i = t$); the number of dimensions is the number of trees in the forest ($M = T$) and the number of objects is the size of the training set of each tree ($N = S$). Analogously, the minimum region enclosing $\mathbf{x}$ and $\mathbf{y}$ is defined by the last node that the two nodes traverse together, i.e. the LCA. Further, the size of the LCA, i.e. the region, is the number of objects it contains. Summarizing, Ting implements $m_p(\mathbf{x}, \mathbf{y})$ as follows (p has been set to 1):

$$Ting(\mathbf{x}, \mathbf{y}) = \frac{1}{T} \sum_{t=1}^{T} \frac{|LCA_t(\mathbf{x}, \mathbf{y})|}{S}. \tag{5.13}$$

Indeed, with this measure, if the minimum region enclosing $\mathbf{x}$ and $\mathbf{y}$, i.e. their LCA, is a very big node, their similarity is going to be small. This is in accordance to what stated by [167]: small nodes are more descriptive, and therefore objects within them are assumed to have a higher degree of similarity. Please note that the Ting self-similarity $Ting(\mathbf{x}, \mathbf{x})$ measured within a tree is the ratio of training objects ending up in the leaf reached by $\mathbf{x}$, i.e. $\frac{|l_t(\mathbf{x})|}{S}$, since the LCA of an object $\mathbf{x}$ with itself is indeed the reached leaf. From this observation, the characteristic of self-similarity being different for different objects is evident: objects which on average ends up in smaller leaves are more similar to themselves than objects which ends up in bigger ones.

Distance 6: Aryal

An extension of the m_p-dissimilarity for $p = 0$ has been proposed in [7] and leads to the following definition:

$$m_0(\mathbf{x}, \mathbf{y}) = \frac{1}{M} \prod_{i=1}^{M} \left(\frac{|R_i(\mathbf{x}, \mathbf{y})|}{N}\right). \tag{5.14}$$

We can observe that the only difference, although quite relevant, with respect to m_p is the fact that the contribution of each dimension is considered independent of the others –hence the product. In an analogous fashion to what is done in [149], in [15] we propose the implementation in the RF context, which is:

$$Aryal(\mathbf{x}, \mathbf{y}) = \frac{1}{T} \prod_{t=1}^{T} \frac{|LCA_t(\mathbf{x}, \mathbf{y})|}{S}. \tag{5.15}$$

In other words, the aggregation of the distance values retrieved from each tree is performed by multiplying the different contributions, which should be more effective than Ting when the forest is made of numerous trees [7]. Further, using the multiplication increases the impact of those trees in which the similarity between two objects is either very high or low with respect to its true value.

5.2.3 Step 3: Distance-based outlier detection.

The extracted distance matrix $\mathbf{D}$ contains the pairwise distances between each pair of objects in a dataset, which ensures the application of any outlier detection methodology that only needs distances as input. Since in the outlier detection field a usual assumption about outliers is that they tend to be distant or in low density areas with respect to inliers, a great variety of methodologies that work with distances have been designed. All these methodologies, some of which have been briefly illustrated in Chapter 2 and in Chapter 3, are usually employed with geometrical distances. In the following, we are going to describe more in detail the outlier detectors we employed in our proposal.

A group of very simple yet often performing methodologies is that of Nearest Neighbor-based techniques[2]. We decided to study four different variants:

- **KNNd** [146] evaluates the ratio of the distance between an object and its K^{th} nearest neighbor to the distance between the latter and its K^{th} nearest neighbor. If the former distance is much bigger than the latter, then the object under analysis has an increased probability of being an outlier. Formally, given a testing object $\mathbf{x}$, the anomaly score according to $KNNd$ is:

$$KNNd(\mathbf{x}) = \frac{d(\mathbf{x}, NN(\mathbf{x}, K))}{d(NN(\mathbf{x}, K), NN(NN(\mathbf{x}, K), K))} \tag{5.16}$$

  where $NN(\mathbf{x}, K)$ is a function returning the K^{th} nearest neighbor of $\mathbf{x}$ searched among the objects in the training set.

- **KNNDist** is simply the distance to the K^{th} nearest neighbor: it assumes that an outlier will be distant from its own K^{th} nearest neighbor independently of the density of the latter. Formally, given a testing object $\mathbf{x}$ the anomaly score according to $KNNDist$ is:

$$KNNDist(\mathbf{x}) = d(\mathbf{x}, NN(\mathbf{x}, K)). \tag{5.17}$$

- **KNNd-Av** [146] is analogous to KNNd but instead of considering just the K^{th} nearest neighbor it takes into account all of them via computing the average. Formally given a testing object $\mathbf{x}$ the anomaly score

[2]Please note that in [149] they proved the suitability of *Ting* when using the standard K-Nearest Neighbor.

according to $KNNd$ is:

$$KNNd - Av(\mathbf{x}) = \frac{1}{K}\sum_{k=1}^{K}\frac{d(\mathbf{x}, NN(\mathbf{x},k))}{d(NN(\mathbf{x},k), NN(NN(\mathbf{x},k),k))}. \quad (5.18)$$

- **KNNDist-Av** is the average distance of the first K^{th} neighbors. This measure can be more accurate than the KNNDist variant, i.e. the number of detected false negatives is likely to be lower. Formally, given a testing object $\mathbf{x}$ the anomaly score according to $KNNDist - Av$ is:

$$KNNDist - Av(\mathbf{x}) = \frac{1}{K}\sum_{k=1}^{K} d(\mathbf{x}, NN(\mathbf{x},k)). \quad (5.19)$$

The main drawback of all these methods is setting K: a too small K can lead to overfitting, i.e. there is the risk of having non-significative neighbors; whereas a too large K may simplify too much the model, e.g. all objects may have similar anomaly scores.

In our approach, we also employed more refined techniques which are based on the estimation of the relative density, such as the **Local Outlier Factor** (LOF) [21] and its several variations. LOF works by comparing the density of $\mathbf{x}$'s neighborhood to that of each of its neighbors. If there is at least one neighbor which has a much denser neighborhood, then the probability of $\mathbf{x}$ being an outlier increases, since the difference in density is a hint that $\mathbf{x}$ is distributed quite differently from at least one of its neighbors. To formally define the LOF score of an object, we have to clarify some other concepts:

- $K - distance(\mathbf{x})$ is the distance of $\mathbf{x}$ to its K^{th} nearest neighbor – computed from the set of unique distance values.
- $N_K(\mathbf{x})$ is the $K - neighborhood$, i.e. the set of neighbors of $\mathbf{x}$ which distance to the latter is less or equal than the $K - distance$. This is important because it can be bigger than K.
- $reach - dist_K(\mathbf{x},\mathbf{y}) = \max(K - distance(\mathbf{y}), d(\mathbf{x},\mathbf{y}))$ if the two objects are far away, then their reachability distance is simply their distance.
- $$lrd_K(\mathbf{x}) = \frac{1}{\frac{\sum_{\mathbf{y}\in N_K(\mathbf{x})} reach-dist_K(\mathbf{x},\mathbf{y})}{|N_K(\mathbf{x})|}}$$
  is the local reachability distance. This function looks at the whole neighborhood evaluating the reachability of each of its components.

Having defined these elements, the LOF score of an object is defined as:

$$LOF(x,K) = \frac{\sum_{\mathbf{y}\in N_K(\mathbf{x})} \frac{lrd_k(\mathbf{y})}{lrd_k(\mathbf{x})}}{|N_K(\mathbf{x})|}. \quad (5.20)$$

Even though this methodology works well, there is an evident drawback which consists of setting the size of the neighborhood K analogously to the NN-based techniques.

As to clustering-based techniques, one of the most used is the **K-centers**: after computing the clusters, the distance of each object to the center of the cluster to which it belongs to is computed. If an object is far away from said center then it is more likely to be an outlier. For $K - Centers$ as well, the main problem consists of setting K to an appropriate number of cluster. We define the outlier score of $\mathbf{x}$ according to K-centers as:

$$K - Centers(\mathbf{x}) = d(clust(\mathbf{x}), \mathbf{x}) \tag{5.21}$$

where $clust(\mathbf{x})$ is the center of the cluster that has been assigned to $\mathbf{x}$.

Finally, in our analysis we also include **ProxIF**, the approach we designed and described in Chapter 3.

5.3 Experimental Evaluation

This section is dedicated to the experimental evaluation of the methodology: first we describe the datasets and some experimental details, then we make two different sets of analyses on two different sets of datasets. The first set of analyses was originally proposed in [111] and extended in this thesis. It is aimed at preliminary assessing the general suitability of the methodology, focusing on three distance measures and only one outlier detector. The second set instead consists of a more complete set of analyses, focusing on a variety of aspects, including a comparison to geometrical distances; in addition several outlier detectors are used and all six distance measures presented in Section 5.2 are employed.

5.3.1 Datasets

We perform the evaluation of the methodology on 15 different UCI ML datasets[3]. Nine of these datasets have been previously used in Chapter 4: as previously described, they are datasets for classification, adequately transformed into outlier detection ones. The remaining 6 datasets as well, *BreastW, Glass, Letter, Musk, Satellite* and *Wbc*, have to be transformed into outlier detection tasks. For all of them, like the other 9, nominal attributes were removed. The subdivision of the classes follows the work of [96] for Satellite and Breastw, that of [86] for Glass and Wbc, that of [2] for Musk and lastly the work of [113] for Letter. As to the latter, the dataset was further modified according to [113]: starting from a dataset of 26 classes, normal objects are sampled from only 3 classes whereas outliers are sampled from the remaining ones. Additionally,

[3]Available at `https://archive.ics.uci.edu/ml/index.php`.

the number of features is doubled by random concatenation: each object, inlier or outlier, is concatenated randomly to an object of the inlier class [4].

Table 5.1: Overview of the 15 datasets used for the experimental evaluation.

Datasets	Nr. of Obj.	Nr. of Feat.	O%	Batch 1	Batch 2
Annthyroid	7200	6	7.42%		✓
Arrhythmia	452	164	45.80%	✓	✓
Breastw	683	9	34.99%	✓	
Cardiotocography	2126	21	22.15%		✓
Hepatitis	80	19	16.25%		✓
Glass	214	9	4.21%	✓	
Ionosphere	351	32	35.90%	✓	✓
Letter	1600	32	6.25%	✓	
Musk	3062	166	3.17%	✓	
Pima	768	8	34.90%	✓	✓
Spambase	4601	57	39.40%		✓
Stamps	340	9	9.12%		✓
Satellite	6435	36	31.64%	✓	
WBC	378	30	5.56 %	✓	
Wilt	4839	5	5.39%	✓	✓

In Table 5.1, we report the number of objects and features and the percentage of outliers for each dataset. Additionally, we report whether they were employed in the first batch of experiments, in the second one or in both of them. If we consider the whole set we can observe that the datasets cover a large range of cases, differing greatly in dimensionality (from 5 up to 166), in the outlier percentage O% (from 3.17% up to 45.80%) and in the number of samples (the smallest one having 80 objects whereas the biggest 7200). Please note that as to the latter characteristic the variation may seem small considering we are in the era of big data. However, since we are dealing with distances and running the experiments on a local computer, a computation on bigger datasets would be unfeasible in terms of time and space.

As done in other chapters, all experiments have been evaluated by measuring the accuracy in terms of AUC. As to other experimental details, since they differ for the two batches of experiments, we present them in the following sections.

5.3.2 Batch 1-Preliminary Analyses

In this section, we evaluate the performance of the experiments performed on the first batch of datasets, adequately marked in Table 5.1. Please recall that the majority of these results have been published in [111]: first we describe some experimental details; subsequently we analyze some parameters of the trained forest $\mathcal{F}$ to choose the best starting model; then we compare three

[4]Some datasets can be found already processed at `http://odds.cs.stonybrook.edu/`, while for others we follow the respective guidelines.

distance measures in terms of performance, and lastly we compare the proposed approach to the standard iForest.

Experimental Details

Two parameters where varied in this analysis:

- Number of trees T : $100, 150, 200$. *3 options.*
- Distance Measure: *Shi, Zhu2, Zhu3. 3 options.*

As to the outlier detector, we focused solely on LOF. After a preliminary analysis not shown here we set $K = 14$. Each experiment, i.e. parametrization setting, was iterated 20 times. The iterations have been created by splitting randomly the dataset in half, one half for the training set and the other for the testing set, with the only constraint related to the absence of outliers in the former.

iForest Parametrization

The preliminary evaluation on iForest aims at choosing the best, on average, starting model from which to extract the distance. Our reasoning focused solely on iForest, i.e. the analysis is independent of the distance measure: we train and test the iForest in the standard way, as described in Chapter 2. In detail, we decided to set S and D to their default values according to [96, 97] which are $S = 256$ and $D = \log_2(S)$. As to T, we analyzed its impact, since a higher number of trees –with respect to the standard $T = 100$– may have a beneficial effect on the distance in terms of robustness. Even though we are aware that the best setting for iForest-based outlier detection is not necessarily the best one for the extracted distances, it represents a good solution and compromise. Indeed, we do not have to compute the distances for all parametrizations.

Table 5.2: Analysis of the forest size in terms of AUC computed on iForest anomaly scores.

Dataset	T=100	T=150	T=200	T=400
Arrhythmia	0.7685	**0.7737**	0.7678	0.7708
Breastw	0.9943	0.9951	**0.9957**	0.9949
Glass	0.7249	0.7173	0.7403	**0.7409**
Ionosphere	0.8959	0.8940	**0.8962**	0.8881
Letter	0.6248	0.6365	**0.6408**	0.6390
Musk	0.9641	0.9827	0.9851	**0.9872**
Pima	0.7356	0.7363	0.7323	**0.7385**
Satellite	0.7925	**0.8150**	0.7932	0.7930
Wbc	0.9533	0.9517	**0.9595**	0.9588
Wilt	0.5192	0.5176	0.5154	**0.5273**

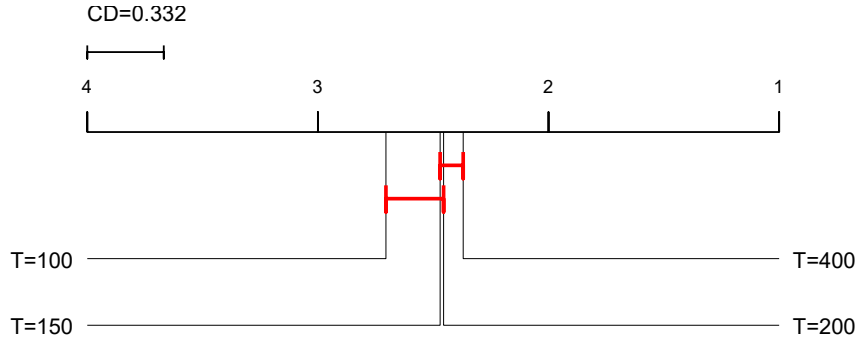


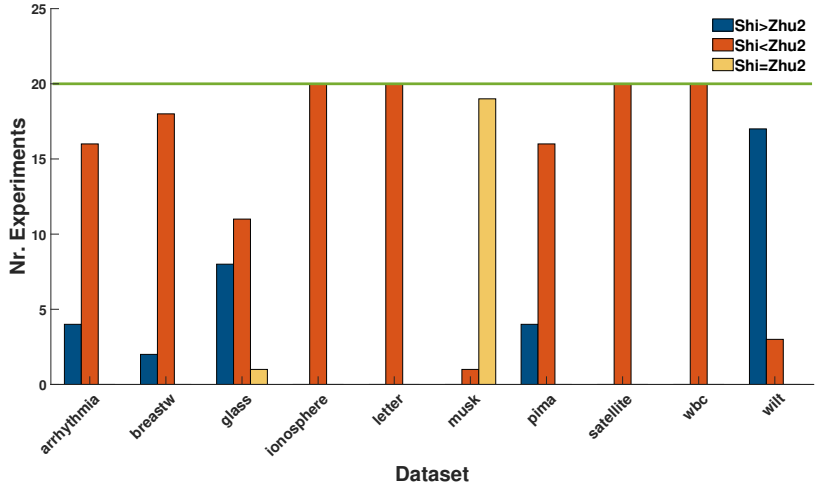
Figure 5.3: Comparison between the different forest sizes via a CD diagram.

In Table 5.2 we report for each dataset the AUC averaged across the 20 iterations for all forest sizes that we evaluated. We can observe that even though in general an increasing number of trees can lead to improvements, there are also datasets for which the trend is opposite; therefore we can conclude that there is not an apparent best choice for T. To statistically assess whether using a bigger size, which is more computationally expensive, is significantly better, we perform a non-parametric analysis similar to those done in Chapters 3 and 4 and thoroughly described in Chapter 2. In detail, we performed a Friedman test on the ten datasets, followed by a post-hoc Nemenyi test. The results of the tests are depicted via a CD diagram in Figure 5.3. We can observe that while the ranking confirms that a bigger T leads to better performances, we can also conclude that a plateau is reached for the third-ranked option, $T = 150$, which is comparable to both $T = 200$ and $T = 400$, which are respectively the second and first one in the ranking. Therefore, training a forest of size $T = 150$ seems to be the best choice in terms of performance and computational load.

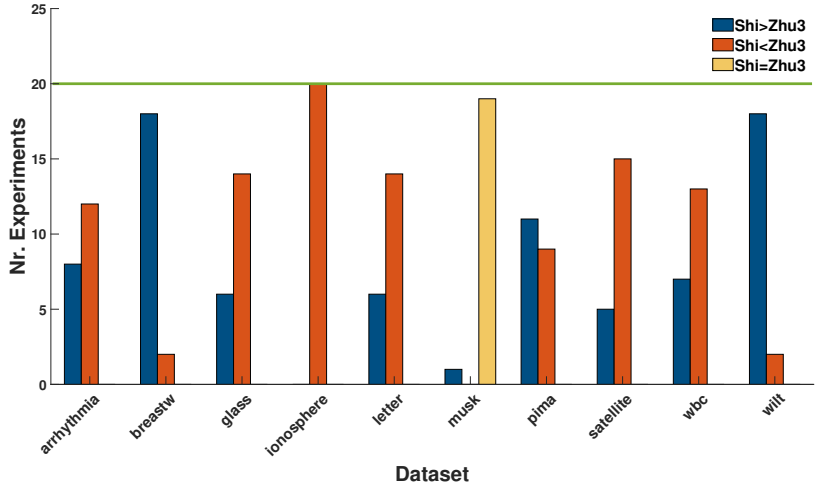
Comparison of Distance Measures

The second analysis focuses on three distance measures Zhu2, Zhu3 and Shi, as defined in Section 5.2. In detail, in Figure 5.4 (a), (b) and (c) we make a pairwise comparison between the distances to understand which choice is more suitable in terms of AUC. For each dataset there are 20 experiments to compare: starting from the same forest we extract 3 dissimilarity matrices, one per distance measure, and then classify the objects using the LOF with $K = 14$, as previously set. The results are represented as follows: for each dataset we count for how many experiments: the first named measure is better than the second (blue bar); the second is better than the first (orange bar) and for how many experiments the two distance measures perform the same (yellow bar). There is also a green line representing the upper limit for the height of the bars, i.e. the maximum number of experiments per dataset, which is 20, the number of iterations.

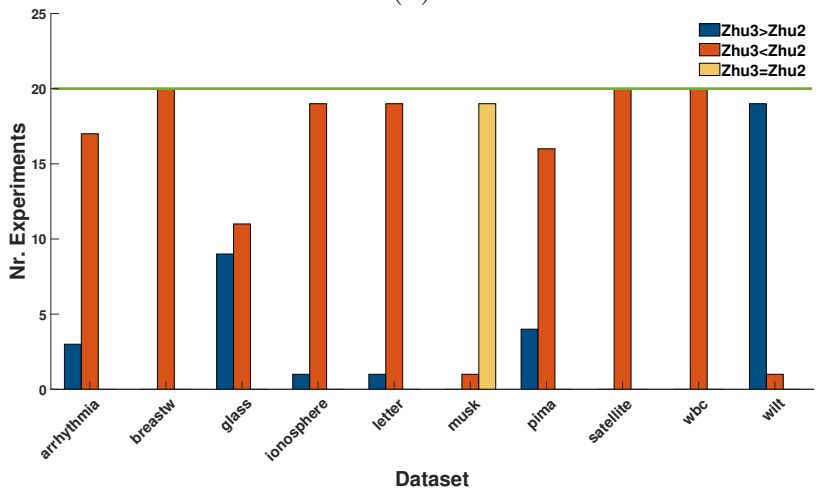
Figures 5.4 (a) and (c) respectively analyze the behaviour of *Zhu2* compared to *Shi* and *Zhu3*: the superiority of *Zhu2* is evident. Only for Wilt,



(a)



(b)



(c)

Figure 5.4: Comparison between the proposed distances. Respectively, each figure compares (a) Shi with Zhu2 (b) Shi with Zhu3 and (c) Zhu3 with Zhu2.

performances are better when using *Shi*: perhaps objects in Wilt are more diverse than they may apparently seem and only a strict distance measure such as *Shi* is able to capture that. As to Musk, performances are equal: it seems that both simpler and more complex distances are able to correctly encode the nature of the objects. From the comparison of *Zhu3* with *Shi*, depicted in Figure 5.4 (b), we can again infer the superiority of the former on the latter, but in a less marked fashion than in Figure 5.4 (a). Indeed, for Breastw, Pima and Wilt, which are all rather small, *Shi* is the better choice, probably due to the same reason observed in Figure 5.4 (a). For Musk, the independence of the performance of the distance is confirmed.

Comparison to iForest

Table 5.3: Comparison between iForest and the proposed method.

Dataset	iForest	Shi	Zhu2	Zhu3
Arrhythmia	0.7737	0.7722	**0.7766**	0.772
Breastw	**0.9951**	0.507*	0.5669*	0.4655*
Glass	**0.7173**	0.7042	0.7147	0.7133
Ionosphere	0.894	0.8597*	**0.9052**	0.8758
Letter	0.6365	0.8407*	**0.8646***	0.8433*
Musk	0.9827	**1.0000***	**1.0000***	**1.0000***
Pima	**0.7363**	0.6914*	0.6972*	0.6906*
Satellite	0.815	0.8015	**0.8375***	0.8047
Wbc	**0.9517**	0.8977*	0.9377*	0.9005*
Wilt	0.5176	**0.9015***	0.8796*	0.8956*

The last analysis is aimed at assessing whether the proposed pipeline, which can be interpreted as an alternative testing phase of the iForest, is indeed better than using the iForest in its standard formulation. We present the results in Table 5.3. For each dataset we report the mean across the 20 repetitions for each distance[5]. We highlight in **bold** the best result and in *italic* the best distance measure for each dataset –which may not correspond to the best result. We also performed a Wilcoxon signed-rank test corrected by Bonferroni to assess whether each of the RF-distance measures is statistically different from iForest, indicating with * if the null hypothesis is rejected.

From Table 5.3 we can observe that the proposed methodology is the best choice on 6 datasets, and on 4 of them it is significantly better than iForest. The best distance measure is *Zhu2* for 8 datasets, confirming what we concluded from the previous analysis. In detail, we can observe a remarkable improvement for Wilt and Letter with respect to using the standard iForest. The latter is the best significant choice for 3 datasets, nevertheless the improvement is remarkable only for Breastw. These results suggest that it is

[5]Please note that there is a slight difference with respect to the results published in [111] where we reported the median instead. Here, we report the mean to be consistent with all the other analyses in the chapter.

beneficial to employ distances extracted from a trained iForest $\mathcal{F}$ to detect outliers: this is particularly true if the dataset is big enough, i.e. if it has > 1000 objects.

5.3.3 Batch 2

Once we had a preliminary confirmation of the potentialities of the proposed approach, we decided to carry out a more thorough investigation on the suitability of the methodology. In detail, this section is dedicated to the experimental evaluation on the second batch of datasets (see the list in Table 5.1), aimed at thoroughly investigating six different distance measures in combination with several outlier detectors. Analogously to the first batch of experiments, we first present the experimental details. Then we make six analyses: the first aims at finding the best parametrization for the forest $\mathcal{F}$ for each distance; then we make an analysis on the classifiers and one on the distances; subsequently we compare the proposed methodology to the standard iForest and to the outlier detectors which input is the Euclidean distance matrix; lastly we make some qualitative and quantitative considerations on the computational burden of the proposed approach.

Experimental Details

Many experiments were performed by varying the value of several parameters:

- Number of trees T: $50, 100, 200$. *3 options.*
- Number of samples used to train each tree S: $64, 128, 256$. *3 options.*
- Maximum depth D each tree can reach: $\log_2(S), S - 1$. *2 options.*
- Distance measure: *Shi, Zhu2, Zhu3, RF-Ratio, Aryal, Ting. 6 options.*
- Outlier Detector: *KNNd*, *KNNDist*, *KNNd-Av*, *KNNDist-Av*, *K-Centers*, *LOF*, *ProxIF. 7 options.*

Differently from before, we decided to change the values that T can take: since datasets are of small size, we added $T = 50$. For the same reason, we decided to evaluate different values of S. As to the depth D, we decided to analyze also trees grown to their maximum depth: whereas some distances may suffer, others may benefit from the additional information. As to the distances and the outlier detectors, we analyzed all 6 measures on all 7 outlier detectors presented in Section 5.2. Each experiment, i.e. parametrization setting, was iterated 10 times. The iterations have been created by splitting randomly the dataset in half, one half for the training set and the other for the testing set, with the only constraint related to the absence of outliers in the former. In addition, given a dataset, the 10 partitions of training and testing sets are identical across all parametrizations.

iForest Parametrization

The first analysis aims at finding the best iForest parametrization for each distance measure. Indeed, different distance measures may benefit from a different parametrization setting. To find out the best values to assign to S and T we perform a non-parametric statistical analysis composed by a Friedman test followed by a post-hoc Nemenyi test, which results are visualized via a CD diagram. Instead, as to D, since we have to make a pairwise comparison, we carry out a Wilcoxon signed-rank test and visualize the results in terms of mean ranks and p-values. For all statistical analyses, we set the significance level to $\alpha = 0.05$. For each parameter, given a matrix $\mathbf{D}$ we classify the objects using $KNNd$ with $K = 1$, one of the simplest outlier detectors for which no parameter must be set. The analysis is performed independently for each distance measure.

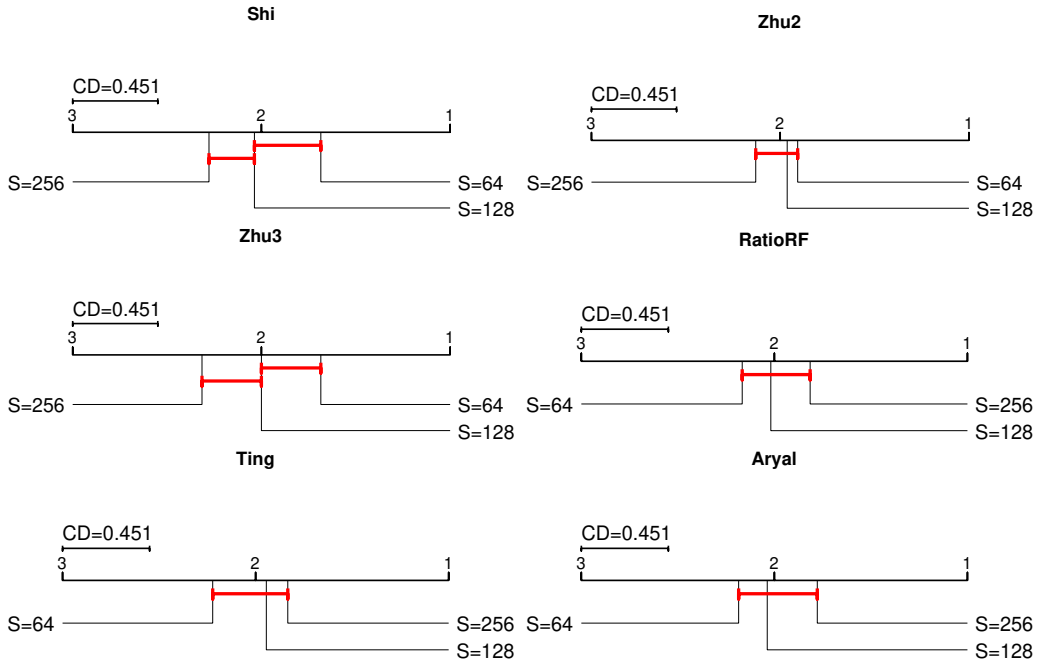


Figure 5.5: Comparison of S on each distance measure via a CD diagram.

The first analysis is aimed at finding the best value of S distance-wise: given a distance measure and a value of S, we compute for each dataset and for every parametrization the average AUC across the 10 iterations. By repeating this for each value of S we obtain the input for our statistical test. In Figure 5.5 we depict one CD diagram per distance measure comparing the different values of S. From the figure, we can make two observations. For *Shi, Zhu2* and *Zhu3* the best choice is $S = 64$: in detail for both *Shi* and *Zhu3* setting $S = 256$ would lead to significant worse results than employing either $S = 64$ or $S = 128$, which are comparable; whereas for *Zhu2* changing S does not have any statistically significant impact. The second observation is that for *RatioRF, Ting* and *Aryal* the best option is to set $S = 256$: nevertheless for

none of these three distances there is a statistically significant difference with the other values of S. From this analysis we can conclude that the overall best choice is **S=64** since it is either the first-ranked or comparable to it, in addition to being less expensive from a computational perspective.

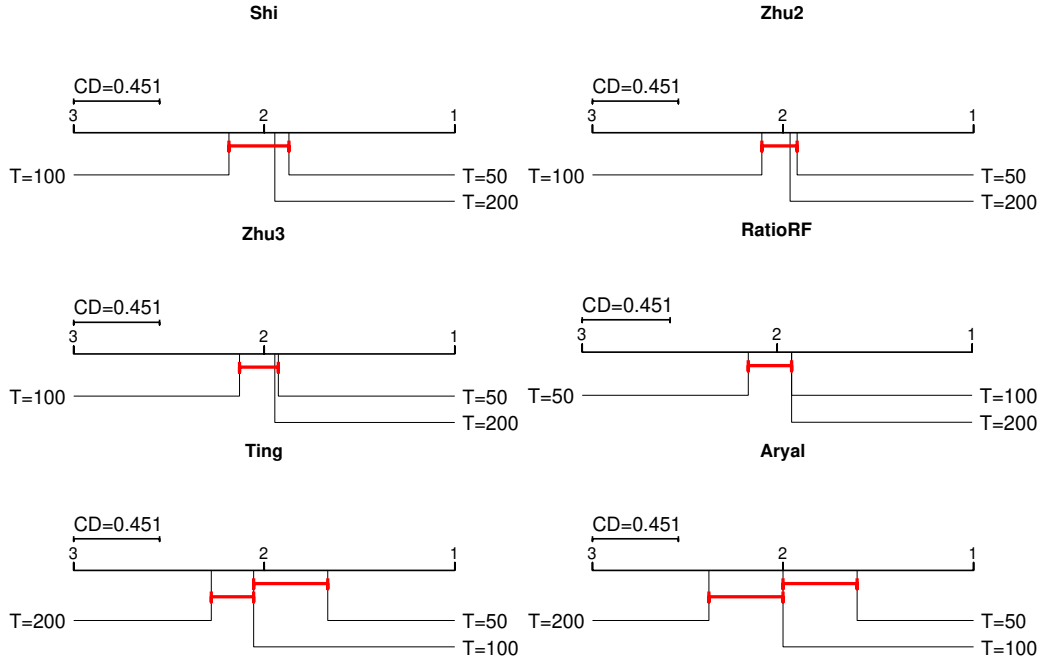


Figure 5.6: Comparison of T on each distance measure via a CD diagram.

In Figure 5.6 we depict the results of the statistical tests done to assess whether there is a value of T representing a significantly better choice for a specific distance measure. The averaged AUC values for the analysis are derived in the same way they were determined for S. Analogously to Figure 5.5 we can observe that T has a similar behaviour across several distance measures. Indeed, from the CD diagrams we can infer the following: when employing *Shi, Zhu2* or *Zhu3* the best choice is to use $T = 50$ even though it is statistically comparable to the other two values T can take; instead when using *RatioRF* the first-ranked value is $T = 100$ but also for this RF-distance measure setting T to another value would not lead to significantly worse results. Finally, for both *Ting* and *Aryal* we can observe a slightly different behaviour: whereas the best setting is again $T = 50$, using $T = 200$ is significantly worse than building a forest composed of fewer trees. Similarly to what we concluded for S, we set **T=50** independently of the extracted distance measure, since it is the only value of T which is either first in the ranking or comparable to the first-ranked value. In addition, from a computational point of view, training fewer trees takes less time.

The last analysis concerns the parameter D, i.e. the maximum depth reachable by each tree in a forest $\mathcal{F}$. The input of our analysis is derived in the same way as done for T and S, i.e. given a value of D and a distance measure

we compute the average AUC across the datasets, all parametrizations and the related iterations. Nevertheless, we carry out a different statistical analysis: since we are dealing with a pairwise comparison, as explained in Chapter 2, we perform a Wilcoxon signed-rank test to compare the two values of D for each distance measure. We depict the results of such analysis in Table 5.4: we report for each distance measure the mean rank of each value of D (analogously as what we visualize in a CD diagram) and the p-value output by the test. We highlight in **bold** the p-value if lower than the significance level α, i.e. the difference between $D = \log_2(S)$ and $D = S - 1$ is statistically significant. A first observation is that it is significantly better to extract *Shi, Zhu2* and

Table 5.4: Statistical analysis for D.

	Rank		
Distance	$D = \log_2(S)$	$D = S - 1$	**P-value**
Shi	1.28	1.72	**1.7e-05**
Zhu2	1.21	1.79	**2.76e-10**
Zhu3	1.26	1.74	**1.36e-07**
Ting	1.62	1.38	**0.0139**
RatioRF	1.52	1.48	0.701
Aryal	1.63	1.37	**0.000604**

Zhu3 from trees built until $\mathbf{D = \log_2(S)}$ rather than from trees built to the end. Indeed, the probability of two objects ending up in the same leaf (and in general of having a longer common path) decreases as D increases. In other words, if we extract *Shi* from a tree where $D = S - 1$ we may obtain a sparse similarity matrix: for several pairs of objects the detected similarity would be 0 across several trees, subsequently leading to $\mathbf{D}$ being sparse and probably less discriminative in the detection of outliers[6]. A second observation we can infer from Table 5.4 is that the performance of *RatioRF* is independent of the used D, in other words, it is very robust independently of the tree structure. As to *Ting* and *Aryal*, the two mass-based RF-distances, we observe an opposite behaviour with respect to Shi, Zhu2 and Zhu3: $D = \mathbf{S - 1}$ is ranked higher than $D = \log_2(S)$ and the difference is statistically significant. This may be due to the fact that when growing trees to their maximum depth, we have an increased variability in these types of distance matrix. In other words, in trees which stop sooner there may be several pairs of objects ending up in the same LCA, which therefore appear to be at the same distance even though their *true* distance may be different.

We can therefore conclude that independently of the distance measure the best choice is to train each tree with $\mathbf{S=64}$ and $\mathbf{T=50}$. As to the depth, we can clearly split the distance measures in two subsets: if they are not mass-based $D = \log_{\mathbf{2}}(\mathbf{S})$ whereas if they are mass-based, i.e. the distance measure

[6]Please note that we can extend this reasoning to *Zhu2* and *Zhu3*: the resulting matrix may not be sparse, but it may contain many low similar values thus impacting on the final outlier detection step.

is either *Aryal* or *Ting*, $D = \mathbf{S} - \mathbf{1}$. Lastly, this analysis confirms that the methodology is robust, i.e. for most distances results do not change in a statistically significant way if we set differently both S and T.

Comparison of Outlier Detectors

In this section we make a comparative analysis aimed at understanding whether, independently of the used distance, there is a best outlier detector to use in combination with RF-distance measures.

Before carrying out the comparative analysis, we have to adequately set the parameters of each outlier detector. In detail, after a thorough preliminary analysis (which we report in Section C.1 of Appendix C), we set them as follows:

- *KNNd*: $K = 8$.
- *KNNDist*: $K = 4$.
- *KNNd-Av*: $K = 16$.
- *KNNDist-Av*: $K = 8$.
- *K-Centers*: $K = 5$.
- *LOF*: $K = 9$.
- *ProxIF*: $T = 100, \quad S = 128, \quad D = \log_2(S)$ and $O - 2PH$ as training criterion. Please note that we used the default parametrization established in Chapter 3 and no additional analysis was made.

After having adequately set the parameters of each outlier detector, we make a non-parametric statistical analysis to compare all the seven approaches. The analysis, made of a Friedman test followed by a pairwise Nemenyi post-hoc test, is carried out on the following results: for each outlier detector we take the AUC values for each dataset and distance measure and average them across the 10 iterations. In Figure 5.7 the results of the statistical analysis are visualized as a CD diagram: we can observe that the first four outlier detectors in the ranking are comparable. In detail, they are: **KNNDist-Av**, the first in the ranking, followed by *KNNDist*, *LOF* and *ProxIF*. KNNd and K-Centers seem to be the worst performing overall.

Comparison of Distance Measures

This section is aimed at understanding how the different distance measures perform and whether we can establish a *best* one. We make three analyses: we compare the distances on the best outlier detector, then we try to infer whether we can group the datasets based on the preferred distance measure, and lastly we assess whether there is a best combination of distance measure and outlier detector.

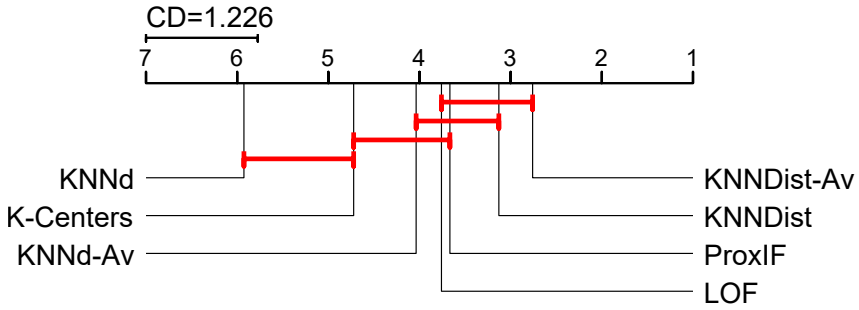


Figure 5.7: Comparing outlier detectors independently of the distance measure via a CD diagram.

Table 5.5: Evaluation of the 6 distance measures using KNNDist-Av.

Dataset	Shi	Zhu2	Zhu3	RatioRF	Ting	Aryal
Annthyroid	0.8560	0.8861	0.8647	0.8854	0.9044	**0.9147**
Arrhythmia	0.8131	0.8137	0.8129	**0.8210**	0.8179	0.7880
Cardiotocography	0.4564	0.4716	0.4654	**0.4935**	0.4116	0.4421
Hepatitis	0.6815	0.6989	0.6758	0.6906	0.7271	**0.7863**
Ionosphere	0.9492	0.9521	0.9466	0.9415	0.9701	**0.9786**
Pima	0.7189	0.7279	0.7184	0.7255	0.7389	**0.7538**
Spambase	0.8550	0.8589	0.8517	0.8580	0.8690	**0.8800**
Stamps	0.9236	0.9362	0.9310	**0.9395**	0.8525	0.7634
Wilt	0.7131	0.6735	0.6805	0.6827	0.7928	**0.8296**

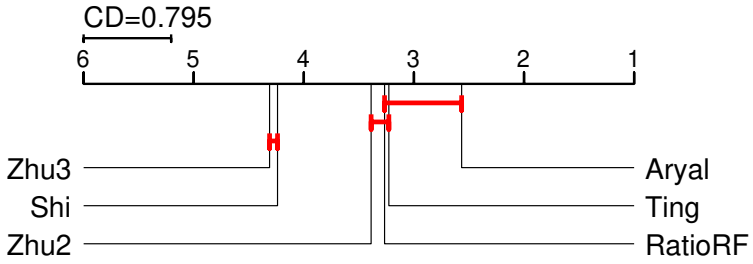


Figure 5.8: CD Diagram that compares the KNNDist-Av performances across the 6 distance measures.

The first analysis compares the performances of the 6 distance measures when using *KNNDist-Av*, the best outlier detector for this task, according to the analysis in the previous section. Concerning the distance-wise performances on the other 6 outlier detectors we used, please refer to Section C.2 of Appendix C. In Table 5.5, given a dataset and a distance, we report the average AUC across the iterations and indicate in **bold** the best result for each dataset. We can observe that all distance measures seem to have good and consistent performances across all datasets. The only exceptions are: *Wilt* for which using a non mass-based distance measure, i.e. neither Aryal nor Ting, seem to lead to much poorer performances and *Stamps* for which the converse holds, i.e. using either Aryal or Ting relevantly decreases the performances.

From Table 5.5 we can also observe that for each dataset, the best distance measure is either *Aryal* or *RatioRF*. We statistically assess these observations in Figure 5.8, which depicts the corresponding CD diagram. Indeed, from Figure 5.8 we can infer that *Aryal* is the best choice, even though it is comparable to both *RatioRF* and *Ting*. Further, we can observe that *Shi* and *Zhu3* are the worst distance measures –confirming the observations made on the first batch of experiments in Section 5.3.2.

Table 5.6: Best distance measures for each outlier detector.

Outlier Detector	**Best Distances**			
KNNDist-Av	Aryal	Ting	RatioRF	
KNNd	RatioRF	Aryal	Ting	Zhu2
KNNDist	Aryal	Zhu2	RatioRF	Ting
KNNd-Av	RatioRF	Zhu2	Aryal	Ting
K-Centers	RatioRF	Zhu2	Shi	Zhu3
LOF	RatioRF	Aryal	Zhu2	
ProxIF	Aryal	Ting		

We summarize the results of the analysis above and the ones in Section C.2 of Appendix C to understand which are the overall best distance measures and which are (if any) the least suitable ones. In detail, in Table 5.6 we report for each outlier detector the best distance measures, i.e. those that were either first in the ranking or comparable to the first-ranked measure according to the non-parametric statistical analyses we carried out. We report them left to right, from best to worst ranked. We can observe that *Aryal* and *RatioRF* are the only distance measures that are either the first-ranked or comparable to such measure for all outlier detectors but one. In detail, when using ProxIF, RatioRF is ranked third, and it is neither comparable to the best choice, Aryal, nor to the worst choice, Shi (see the CD diagram in Figure C.7 in Section C.2 of Appendix C for details). Instead, Aryal is the worst ranked when used as input to K-Centers (see the CD diagram in Figure C.5 in Section C.2 of Appendix C for details). From Table 5.6 we can also infer that using *Shi* and *Zhu3* is overall a bad choice as also observed in Section 5.3.2. In conclusion, it seems that the most robust distance measure is *RatioRF* since: it is the first ranked on 4 outlier detectors; it is comparable to the first-ranked distance measure on other 2 outlier detectors, and it is never comparable to the worst distance measure. We make an additional qualitative analysis aimed at understanding whether datasets as well have a preferred distance measure. To do that, given a dataset and a distance measure, we simply count the number of outlier detectors for which it is the best choice (please refer to Table 5.6 and those in Section C.2 of Appendix C for details). The results of this analysis are reported in Table 5.7. We can observe that *Aryal* is the best choice for 5 out of 9 datasets, whereas *RatioRF* for 3. On 1 dataset, the two distance measures are the best choice on the same number of outlier detectors. It seems that datasets that on average

Table 5.7: Best distance measures for each dataset. Between parenthesis, we report the number of outlier detectors for which the distance is the best choice.

Dataset	Best Distances	
Annthyroid	Aryal (5)	
Arrhythmia	RatioRF (4)	
Cardiotocography	RatioRF(6)	
Hepatitis	Aryal (6)	
Ionosphere	Aryal (6)	
Pima	Aryal (3)	RatioRF(3)
Spambase	Aryal (6)	
Stamps	RatioRF (6)	
Wilt	Aryal (4)	

prefers RatioRF as distance measure have a size ranging in $[300, 4000]$ –except for Ionosphere.

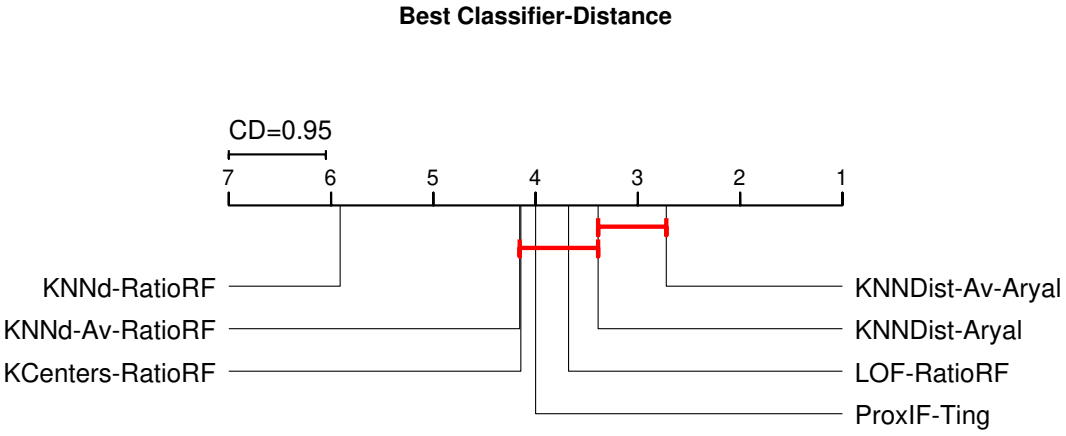


Figure 5.9: CD Diagram that compares each classifier combined with its best distance measure.

The last comparative analysis on RF-distances takes the first-ranked distance for each outlier detector and makes a global comparison, which results are depicted via the CD diagram in Figure 5.9. The analysis confirms that *KNNDist-Av* seems to be the best outlier detector, even though it is comparable to *KNNDist*, *LOF* and *ProxIF*. Three of the best ranked variants work with the mass-based dissimilarity measure *Aryal* confirming its suitability in detecting outliers.

Summarizing all the analyses presented in this section, there are two distance measures which seem to be the most suitable for outlier detection: **Aryal** and **RatioRF**. Nevertheless, it is difficult to establish which between the two is the best one, since their performance also depends on the dataset and on the outlier detector.

Table 5.8: Comparison with iForest

Dataset	iForest	Shi	Zhu2	Zhu3	RatioRF	Ting	Aryal
Annthyroid	0.8973	0.8560*	0.8861	0.8647	0.8854	0.9044	**0.9147**
Arrhythmia	**0.8561**	0.8131*	0.8137*	0.8129*	0.8210*	0.8179	0.7880*
Cardiotocography	**0.7559**	0.4564*	0.4716*	0.4654*	0.4935*	0.4116*	0.4421*
Hepatitis	0.6912	0.6815	0.6989	0.6758	0.6906	0.7271*	**0.7863***
Ionosphere	0.9016	0.9492*	0.9521*	0.9466*	0.9415*	0.9701*	**0.9786***
Pima	0.7316	0.7189	0.7279	0.7184	0.7255	0.7389	**0.7538***
Spambase	0.8787	0.8550*	0.8589	0.8517*	0.8580	0.8690	**0.8800**
Stamps	**0.9537**	0.9236	0.9362	0.9310	0.9395	0.8525*	0.7634*
Wilt	0.4618	0.7131*	0.6735*	0.6805*	0.6827*	0.7928*	**0.8296***

Comparison to iForest

In this section, we make an analysis analogous to that reported in Section 5.3.2: we compare the proposed approach to an iForest model which has been trained with default parameters. As to our approach, we report the results for all distance measure on the best outlier detector, *KNNDist-Av*. Results are depicted in Table 5.8 as averaged AUC across the iterations. We indicate in **bold** the best result for each dataset, and we put a * next to each result obtained using the proposed methodology if significantly different from iForest. The statistical analysis has been performed using a Wilcoxon signed-rank test where $\alpha = 0.05$ followed by a Bonferroni correction. The comparison confirms the power of the proposed methodology, analogously to what observed in Section 5.3.2: for 6 out of 9 datasets the best result is achieved using *Aryal* and for 4 out of 6 the results are significantly different from those of iForest. Instead, as to the 3 datasets for which iForest is the best choice, only for *Cardiotocography* the difference is statistically significant with respect to all six distance measures which are all bad performing. The reason behind this may be due to the fact that for this dataset it may not be informative to extract distances: indeed, recall that iForest employs the original dataset, which is vectorial.

Summarizing, the proposed approach, which can be interpreted as an enhanced *alternative testing phase* of iForest, shows very competitive results with respect to the standard methodology, which is a state-of-the-art outlier detector.

Comparison to Euclidean Distance

The last analysis is aimed at comparing the RF-distances with the Euclidean distance, one of the most used and known geometrical distance measures. In practice, we compute the Euclidean distance between all objects in a dataset and input the obtained representation to each of the outlier detectors that we have employed in the experimental evaluation. The parameters of the outlier detectors have been set as previously illustrated. For the proposed methodology, we employ the best distance for each outlier detector. Results

Table 5.9: Comparison on each outlier detector of the proposed methodology with the Eucldean-based distance matrix.

	KNNd		**KNNDist**		**KNNd-Av**		**KNNDist-Av**	
Dataset	**RatioRF**	**Euc**	**Aryal**	**Euc**	**RatioRF**	**Euc**	**Aryal**	**Euc**
Annthyroid	0.5722	**0.7897***	**0.9040***	0.8184	0.6041	**0.8885***	**0.9147***	0.8748
Arrhythmia	0.7268	**0.7418**	**0.7830**	0.7805	**0.8252**	0.7923	**0.7880**	0.7846
Cardiotocography	0.4984	**0.7542***	0.4496	**0.7057***	0.5000	**0.8499***	0.4421	**0.8595***
Hepatitis	**0.6402**	0.6348	0.7766	**0.7977**	0.7054	**0.7359**	**0.7863***	0.6234
Ionosphere	**0.9328**	0.9122	**0.9748**	0.9703	**0.9632***	0.9405	**0.9786***	0.9065
Pima	0.5855	**0.6205**	0.7468	**0.7528**	0.6408	**0.6674***	**0.7538***	0.6399
Spambase	0.5177	**0.7371***	**0.8777***	0.7300	0.5018	**0.8186***	**0.8800***	0.8152
Stamps	0.7246	**0.8442***	0.7466	**0.9315***	0.8093	**0.9207***	0.7634	**0.9190***
Wilt	**0.7611**	0.7485	**0.8277***	0.6566	**0.8704**	0.8668	0.8296	**0.8828***
Average	0.6621	0.7537	0.7874	0.7937	0.7134	0.8312	0.7930	0.8117

	K-Centers		**LOF**		**ProxIF**	
Dataset	**RatioRF**	**Euc**	**RatioRF**	**Euc**	**Aryal**	**Euc**
Annthyroid	**0.7594***	0.5230	0.6108	**0.8473***	**0.8993***	0.6620
Arrhythmia	**0.8023**	0.7746	**0.8242**	0.7846	**0.7996***	0.7076
Cardiotocography	0.6099	**0.7546***	0.4936	**0.8117***	0.5584	**0.6773***
Hepatitis	0.6436	**0.7350***	0.7368	**0.7798***	0.7353	**0.7842***
Ionosphere	0.9529	**0.9693**	**0.9726***	0.9335	**0.9165***	0.7696
Pima	**0.6328**	0.6323	0.6543	**0.6802***	**0.7721**	0.7493
Spambase	**0.8354***	0.5807	0.5210	**0.8020***	**0.8461***	0.4723
Stamps	**0.8560**	0.8155	0.8524	**0.9162***	0.8510	**0.9054***
Wilt	**0.4510***	0.3701	**0.8431***	0.8024	**0.5338***	0.3586
Average	0.7270	0.6839	0.7232	0.8175	0.7680	0.6763

are reported in Table 5.9 in terms of average AUC; we highlight in **bold** the best result for each outlier detector and for each dataset, and we put also a * if the difference is statistically significant. The statistical test we employed is the Wilcoxon signed-rank test, where $\alpha = 0.05$.

From Table 5.9 we can make several observations: i) on *KNNd* and the average counterpart, *KNNd-Av*, the Euclidean distance is the best choice and in many cases the difference is statistically significant; ii) on *KNNDist* and the average counterpart *KNNDist-Av*, *Aryal* is the best choice on a higher number of datasets confirming that using only the distance to the K^{th} neighbor is highly suitable; iii) even though on *K-Centers* the two approaches seem comparable, i.e. RatioRF and the Euclidean distance are respectively the best significant choice on 3 and 2 datasets, the very bad performance of the latter on Annthyroid and Spambase leads to *RatioRF* being overall a better choice; iv) on *LOF* it is more suitable to use the Euclidean distance; v) an analogous reasoning to point iii) can be made when using *ProxIF*, i.e. *Aryal* perform significantly better than the Euclidean counterpart on the majority of datasets, especially if we look at Annthryoid, Wilt and Spambase.

From this analysis we cannot draw a general conclusion on whether data-dependent distance measures are better than geometrical ones: indeed, the choice of the distance seems to be dependent on both the outlier detector and

the dataset. Nevertheless, as to the latter, there is no apparent meaningful criterion to partition the datasets such that the same distance can be used. In general, we can observe that the Euclidean distance works *surprisingly* well, in contrast to what assumed in literature [7, 151]. We are in the process of trying to understand the causes behind this behaviour: an explanation could be that to adequately exploit the nature of data-dependent measures we have to carefully choose the outlier detector, as also observed in Table 5.9. This result is quite interesting and opens new research directions related to the last step of the proposed pipeline.

Considerations on Complexity

In this section, we make some qualitative and quantitative considerations about the complexity of the proposed methodology. Our aim is not to make a classical complexity analysis, but to give only some insight on the computational properties of the proposed approach.

The methodology computes the entire distance matrix $\mathbf{D}$: this is the most burdensome step as to space complexity. In detail, the cost of storing the distance matrix $\mathbf{D}$ is the same independently of the distance measure: it depends only on the size of the dataset $\mathcal{O}$, i.e. the total space complexity amounts to $O(|\mathcal{O}|^2)$, which can become problematic if the dataset is too big. Please note that we need to store the whole distance matrix $\mathbf{D}$, since most outlier detectors need it as input. However, this issue is absent if we use *KNNDist* and variations thereof, since we only need the distances between the testing object and all training objects, i.e. a vector. Further, if we use *ProxIF* we can adopt a different approach: even though we need to know distances between training objects, we can compute them during the tree building procedure only when needed, thus saving in space complexity.

The time complexity of extracting $\mathbf{D}$ is instead dependent on the chosen distance measure. In the following, we make some considerations related to the computation of the distance between a pair of objects $\mathbf{x}$ and $\mathbf{y}$ within an iTree t:

- **All Distance Measures**: both $\mathbf{x}$ and $\mathbf{y}$ have to traverse the path from root to leaf. The time complexity depends on the length of the path, which depends on the depth of the tree; since trees are built with few examples (e.g. $S = 256$), this option is not so expensive. This is usually the first step and for each object we can record many different information such as: the reached leaf node, the identity of the traversed nodes, the answer to all tests, i.e. nodes, in the tree, and so on. This is crucial since it allows to save time during the distance extraction.

- **Shi**: given the previous step, the computation of Shi in a tree consists of a single comparison between the reached leaves.

- **Zhu2**: we have to make a few operations to compute this distance measure. To find the shared path we have to compare the two paths, i.e.

we compare each traversed node. The computation stops when the compared nodes do not coincide. This can take up to $\min(dp(\mathbf{x}), dp(\mathbf{y}))$, which is the length of the shortest path between the two.

- **Zhu3**: analogously to Zhu2, we have to compare the two paths until they do not coincide, i.e. to their LCA. Additionally, we have to make some other costly operations: we have to compute the number of objects of the dataset that go down to the LCA and those that go down the longest of the two paths. Please note that this can be done once for all objects during training time.

- **RatioRF**: like for Zhu2, we have to compare the two paths until the LCA is reached. Then we have to make other two operations: we have to find which are the nodes in the path of $\mathbf{x}$ to which $\mathbf{y}$ would answer in the same way and vice-versa. Please note that we can record the answers of an object $\mathbf{x}$ to all tests defined in a tree during the traversal, thus reducing this step to simple comparisons.

- **Ting, Aryal**: analogously to the other distance measures we have to compare the paths of $\mathbf{x}$ and $\mathbf{y}$ until they reach their LCA, and then we have to find the number of objects of the dataset that have reached such LCA.

In general, we can observe that *Shi* is the least burdening whereas the other distances, except maybe *Zhu2*, seem to be quite burdening since several operations are needed for their computation.

In Table 5.10 we assess our considerations from an experimental point of view: for each dataset we report the training time in seconds of building the iForest and then for each distance the seconds it takes to extract the whole matrix. The iForest was trained using the best parametrization $S = 64, T = 50$ according to Subsection 5.3.3 except for the depth for which we used only one value, $D = S - 1$, for all distances for comparison purposes. The small variations in training time across different datasets are due to the fact that their training sets have < 64 objects. As assumed, *Shi* does not introduce a relevant overhead independently of the size of the dataset. Instead, differently from what we hypothesized, the computation of *RatioRF* is quite fast, slower only than *Shi* and *Zhu2*, which are both very simple to compute –and also worse performing in terms of AUC. Lastly, as expected, *Zhu3, Ting* and *Aryal* do not scale well with the size of the dataset. Let us consider Annthyroid, a dataset of 7200 objects: to extract *Zhu3* it takes almost 9 minutes, whereas to extract *Ting* and *Aryal* almost 20.

5.4 Conclusions and Future Work

In this chapter, we presented a novel methodology for outlier detection that exploits iForests in an alternative and innovative fashion. Given a trained

Table 5.10: Training an iForest+extracting **D**: total time in seconds.

Dataset	**Training**	**Shi**	**Zhu2**	**Zhu3**	**RatioRF**	**Ting**	**Aryal**
Annthyroid	0.64s	16.57s	86.56s	510.19s	168.96s	1066.91s	1082.03s
Arrhythmia	0.83s	1.07s	1.6s	2.16s	1.81s	3.79s	4.05s
Cardiotocography	0.64s	4.87s	11.16s	21.51s	18.86s	86.28s	88.1s
Hepatitis	0.41s	0.33s	0.35s	0.39s	0.36s	0.43s	0.41s
Ionosphere	0.63s	0.8s	1.17s	1.59s	1.25s	2.47s	2.48s
Pima	0.6s	1.76s	3.08s	4.61s	3.77s	7.3s	7.8s
Spambase	0.74s	10.63s	31.75s	67.66s	65.87s	364.92s	372.52s
Stamps	0.63s	0.87s	1.13s	1.45s	1.2s	2.58s	2.57s
Wilt	0.62s	11.1s	11.78s	12.8s	11.73s	15.89s	15.93s
Average	0.57s	4.8s	14.86s	62.24s	27.38s	155.06s	157.59s

forest we extract a distance matrix which is then input to an outlier detector. The extracted RF-distances, being based on how data are distributed, i.e. how they are partitioned by each tree, are not geometric and should embed more information. In other words, this type of distances should capture more accurately the nature of the objects and their relations, thus improving the detection of outliers.

The suitability of this contribution has been tested on a total of 15 datasets. We made two separate analyses, with the second one being more thorough. It is shown how, aside from some sporadic cases, most distances are quite robust independently of how the forest is trained. In addition, we show that more refined distance measures are a better choice for all outlier detectors, leading to very good performances. We also assess that it is advantageous to use RF-distances with respect to the standard iForest. Nevertheless, no general observation can be made when we compare the proposed methodology to the same outlier detectors used in combination with the Euclidean distance: this proves that further research must be done to uncover whether the principles of some outlier detectors strongly rely on the geometric properties of the objects. The latter is especially true if we consider mass-based RF-distances which are not metrics, i.e. the identity of indescernibles does not hold.

Even though we cannot infer in which scenarios data-dependent measures are better than geometric distances, since the proposed methodology is easy to use, it is worth comparing their performances independently of the domain. Indeed, the computation of RF-distances is easy and fast and the only parameters to set are those of the iForest model, from which we extract the distance matrix. However, we provided clear guidelines on how to set them depending on the chosen RF-distance measure. Further, it is not necessary for the user to compute distances using all the analyzed RF-distance measures, since we have thoroughly shown that in all scenarios the best choice is either RatioRF or Aryal –we have additionally provided an intuition on when choosing RatioRF instead of Aryal based on the dataset size.

As to future research, in addition to trying to get a better understanding

concerning the use and applicability of RF-distances compared to geometric measures, another interesting aspect to investigate is the definition of novel RF-distances. Indeed, maybe the used distance measures may not be able to truly uncover the true nature of outliers. Since we are extracting distances from an iForest model, we could for example define a measure which explicitly models the difference between two objects in terms of their isolation capability. We provide additional insight into these ideas in Chapter 7.

Chapter 6

Characterization of Multiple Sclerosis Connectivity Networks

In this chapter, we propose **MS-ProxIF**, a variant of Proximity Isolation Forest proposed in Chapter 3 for characterizing Multiple Sclerosis (MS). In detail, we use MS-ProxIF as a feature extractor: the derived representation is called **RF-Isolation**.

6.1 Introduction

Multiple Sclerosis is a chronic autoimmune disease of the central nervous system that causes demyelination and eventually neurodegeneration [12, 40]. The exact mechanisms causing MS are still unknown, even though several factors are known to be involved: low exposure to the sun, low vitamin D, positivity to the Epstein Barr virus, smoking, obesity during childhood, environmental and genetic factors [40]. Further, MS is more common in females and its usual onset is between the ages of 20 and 40 years. In the past, MS was generally considered as a two-stage disease. The first phase, called Relapsing-Remitting (RR) is characterized by periods of severe inflammation followed by periods of remission in which symptoms are almost absent. The second phase is known as Progressive (P) and its main feature is neurodegeneration leading to an increased disability; we can further differentiate this phase into primary (PP) and secondary progressive (SP). However, in recent years, this interpretation of MS is losing validity: several drugs have been developed to slow down, if not stop, the progression of the disease. This has led to an innovative perspective where MS patients lay on a spectrum that goes from inflammation to neurodegeneration: the mode of intervention depends on where the patient is on the spectrum [40]. In other words, a person suffering of MS does not necessarily advance to a progressive stage.

From a pathological point of view, the main hallmark of MS is the presence of plaques rich in inflammatory materials in various locations, such as perivenular plaques. The inflammation of these plaques causes demyelination, i.e. lesions affecting white matter tracts and causing disconnection of gray matter regions. In advanced MS, axonal damage can be observed as well [40]. To correctly diagnose MS, the physician follows the McDonald's criteria [126]. The first and most important diagnostic test consists of analyzing the cerebrospinal fluid collected via a lumbar puncture. Nevertheless, other paraclinical tests must be performed to draw a final diagnosis. Following the latest

revision of the McDonald's criteria, one of the most important is the Magnetic Resonance Imaging (MRI) of the brain. The aim of an MRI is to identify the macrolesions caused by MS, i.e. the demyelination process [40]. Demyelination, analogously to what causes the onset of MS, is still rather obscure: there is no complete knowledge about the mechanisms of how this disease acts on the brain. However, thanks to MR images, it is possible to start uncovering such mechanisms by identifying, and subsequently study, which brain regions are the most involved in the MS demyelination process.

Recently it has been discovered that all subjects, independently of their disease status, present structural disconnection which is pathophysiologically relevant [30, 123]. This has led to an increasing amount of researches focusing on the analysis of structural disconnection: the core principle of many of these studies consists of analyzing the brain connectivity network of a subject, which must be estimated starting from the MRI. A brain connectivity network encodes the connectivity strength between each pair of brain regions of interest (ROIs) –please note that in the rest of the chapter, we will use the terms ROI and brain region interchangeably. The extraction of a meaningful connectivity network is challenging: there exist several alternatives of tractography algorithms [104], i.e. approaches which map neuronal pathways in the white matter, but unfortunately most of them present several limitations, e.g. computationally burdening, locally optimized or overall inaccurate. Only recently an approach has been proposed [34] which combines such tractograms with local microstructure properties leading to higher quality brain connectivity networks.

The quality of the brain connectivity networks is important because via their analysis we aim at discovering the ROIs involved in an MS lesion: we expect their connectivity strength in an MS subject to be lower than in a healthy control. Often, instead of analyzing directly the network, network-derived measures are extracted and analyzed [23, 29, 50, 103, 137, 169]. These measures are defined to capture the importance in terms of connectivity of: i) each ROI; ii) the whole network. An example of the former, called *local measures*, is the *node strength*, which measures the total connectivity strength of a ROI to all other brain regions. As to the latter category, an example is the *density*, which measures how many connections are present in a brain connectivity network with respect to the total possible number of connections. These are known as *global measures*. After having extracted these measures for all subjects, the usual approach consists of carrying out a comparative analysis with the healthy cohort, similarly to what is done when analyzing the connectivity network itself. Even though several network-derived measures have been shown to be relevant for MS [119, 138], there is still room for methodological improvement. Actually, the main limit of all these measures is that they are *subject-wise*: they are computed starting from a subject's connectivity network and the computation is independent of all other subjects. In other words, these representations do not capture the nature of MS in a global fashion but rather the nature of how MS manifests in a specific subject, which can be an obstacle

in the study of the disease, in particular in the identification of the ROIs the most involved in the disease.

In this chapter we propose a novel representation of structural connectivity networks, called *RF-Isolation*, aimed at characterizing MS. In detail, the main features of our approach, *RF-Isolation*, are:

- It characterizes a brain connectivity network via the disconnection degree of each ROI, measured with respect to all the other ROIs.
- The representation is extracted by reformulating the problem as an outlier detection task; indeed, outliers (disconnected ROIs) are few and different from the rest of the data (other ROIs). The representation exploits a variant of ProxIF, the methodology proposed in Chapter 3, by making the following analogies: i) each ROI is an object, which we recall is described in terms of proximity to other objects; ii) the connectivity network is interpreted as a similarity matrix. In other words, the higher the connectivity strength between two ROIs x and y the higher their similarity. The anomaly score assigned to each object is the degree of isolation (disconnection) of the ROI.
- A model is trained on the entire MS population, and subsequently it is used as a feature extractor to derive the *RF-Isolation* vector for any subject, independently of whether they suffer from MS or not. This characteristic makes the extracted representation *disease-wise*: the model captures the characteristics of the disease and not only of one subject. In other words, the obtained representation may be more informative for the discovery of MS-related mechanisms. This is a crucial difference with respect to the previously mentioned standard network-derived metrics.

To evaluate the suitability of the proposed representation, we made several analyses based on the classification of a cohort of 79 subjects, 55 of which suffering from MS. The obtained results are promising, even when compared to standard network-derived measures. We also report some preliminary analyses on the identification of the most relevant ROIs for the proposed representation.

This work is the result of a collaboration with the University of Genova, La Sapienza University of Rome and the Icahn School of Medicine at Mount Sinai, New York, US. In addition, part of this chapter has been accepted for the 17^{th} edition of the Computational Intelligence Methods for Bioinformatics and Biostatistics conference (CIBB2021) [112], and, following the invitation of the conference chairs, submitted to Lecture Notes in Bioinformatics.

This chapter is organized into three main sections. Section 6.2 presents the proposed methodology, which is divided in two steps, each thoroughly described. In Section 6.3 we make a thorough experimental evaluation, preceded by an exhaustive description of the dataset. Lastly, in Section 6.4 we make some final statements and illustrate some future ideas.

6.2 RF-Isolation

In this section, we thoroughly describe the novel representation that we propose, *RF-Isolation*. RF-Isolation is a local representation of structural connectivity networks, in other words it is a vector of length equal to the number of ROIs. The proposed approach is based on the following ideas:

- Each feature of the RF-Isolation vector encodes how much one ROI is disconnected with respect to the remainder of the ROIs, i.e. its disconnection degree. Our choice seems reasonable, since structural disconnection is one of the hallmarks of MS. Please note that this methodology does not detect a specific lesion, which is defined by a pair of ROIs, whereas it aims at capturing how much *one* ROI is involved in the demyelination process.

- To capture the disconnection degree of a ROI, we cast the problem as an outlier detection task. As recalled many times in this thesis, outliers are rare patterns which have been generated by a different mechanism than the one behind the generation of inliers. In other words, outliers do not conform to the rest of the data and can be very different from them [68]. In the context of brain connectivity networks, ROIs with a high degree of disconnection can be interpreted as outliers: indeed these ROIs are usually few with respect to the rest, and they have a different relation in terms of connectivity with other brain regions. There may exist two types of such ROIs: i) ROIs which are involved in MS-related lesions. In detail, we expect these ROIs to be highly disconnected in the MS population or at least a part of it; ii) ROIs which have naturally a low degree of connection to other regions. These are outliers in the whole population independently of whether they have MS or not, unless they are further disconnected by a demyelination process. Summarizing, we can measure the disconnection degree of a brain region of interest via the quantification of its "outlierness" level.

- *RF-Isolation* is a disease-wise representation, i.e. it is extracted for all subjects from the same model in order to better capture MS-related mechanisms. This can be obtained by building the model on the entire MS population. The latter choice must be preferred over the alternatives, which are training using: i) the healthy cohort; ii) both populations; iii) one subject only. In detail, in the first scenario MS-related ROIs have a normal connection degree in the training population and therefore when extracting their disconnection degree for an MS subject they are likely to get a lower disconnection value. Analogously it holds for scenario ii) except that the effect is reduced due to the presence in the training population also of MS subjects. Nevertheless, an additional issue may present itself in scenario ii): using both populations may hide the peculiar characteristic specific to only one of the two populations. As to scenario

iii), one subject would be unable to correctly characterize the true nature of MS: we would capture just the static image of the connectivity network under analysis. In conclusion, to adequately characterize MS, we have to build the model using the MS population.

In detail, our approach is based on two steps:

1. Train a model using the MS subjects.
2. Given a subject, we can obtain its RF-Isolation vector by employing the model from Step 1.

As to the first step, our approach is based on ProxIForest, the RF-based outlier detection technique that we proposed in Chapter 3. The choice was made starting from the following reasoning: iForest is one of the most successful outlier detection techniques. Nevertheless, we cannot use iForest since a connectivity network is a non-vectorial representation. However, as explained in the previous section, we can make a straightforward analogy between a connectivity network and a similarity matrix: a stronger connection between two ROIs can be interpreted as a higher similarity value. This interpretation allows us to build our model starting from ProxIForest since it works with proximities between objects. In detail, we extend and adapt ProxIForest for the context at hand: we denote the obtained model as *MS-ProxIF*.

In Figure 6.1 we represent the pipeline of the methodology. In Figure 6.1 (a) we depict the building procedure of an MS-ProxIF model: given the input set of subjects, each represented via their connectivity network, we train a ProxIForest on the MS population. In Figure 6.1 (b) given a generic subject x, we use the MS-ProxIF model from Figure 6.1 (a) as a feature extractor to obtain their RF-Isolation representation. More details are given in the following two subsections.

6.2.1 Step 1: Building the MS-ProxIF Model

The first step consists of building the MS-ProxIF $\mathcal{F}$. In Chapter 3 we exhaustively explained how to build a ProxIF. In this section, we thoroughly describe its adaptation to the context and the extensions that we made with respect to standard ProxIF.

An MS-ProxIF is made of several MS-ProxITs where each tree t represents a recursive partitioning of the ROIs. In detail, the input of each MS-ProxIT is a set of pairwise similarities, which corresponds to the set of connectivity values between each pair of ROIs belonging to one connectivity network. Recall that each connectivity network represents a subject.

Please note that as input we have similarity values, whereas in Chapter 3 we used dissimilarities. We can manage this difference in two ways:

- The first consists of transforming the connectivity strength $c(x, y)$ between two ROIs x and y in the same way a similarity is transformed into

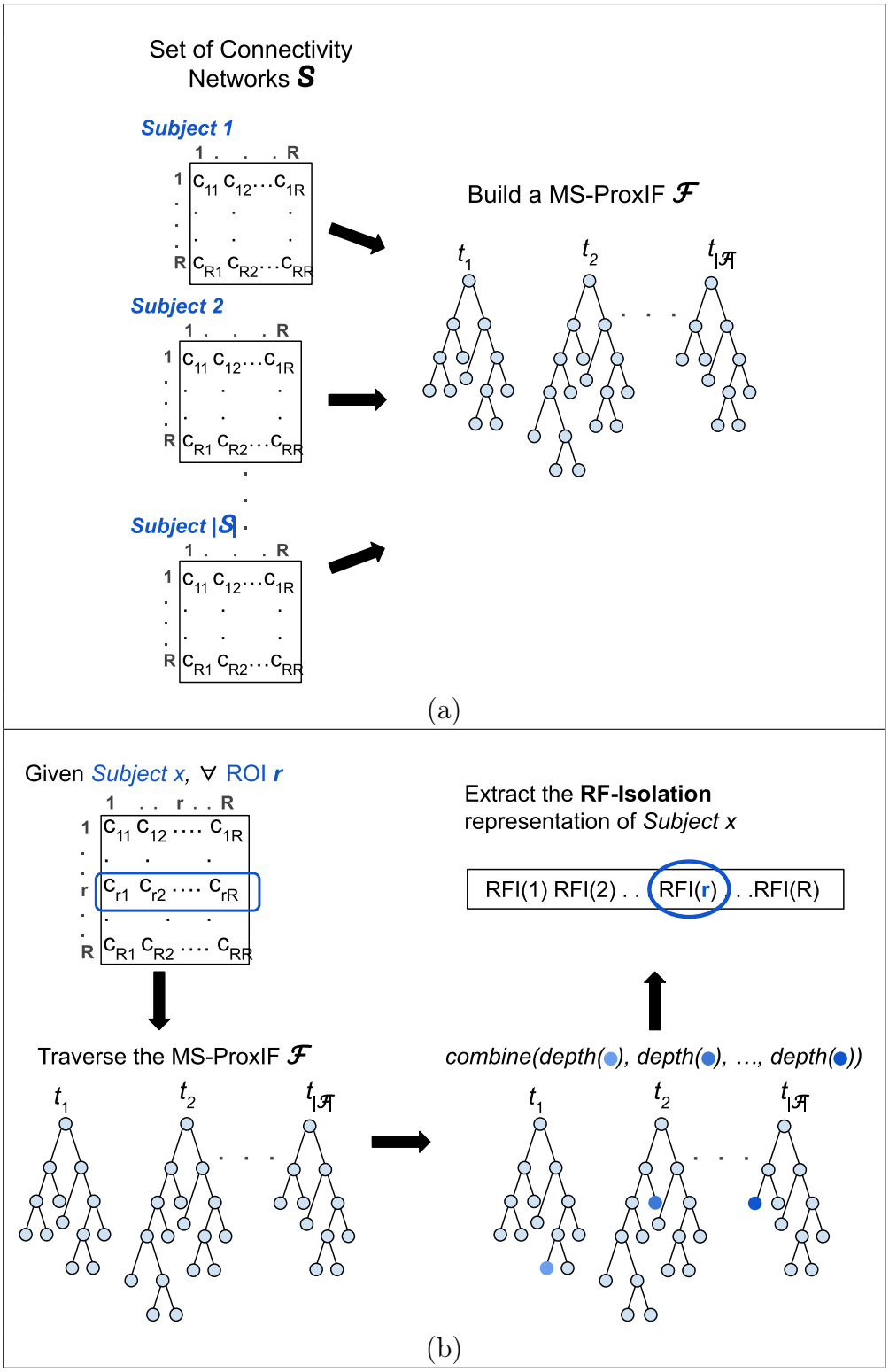


Figure 6.1: Pipeline of the proposed approach: (a) building an MS-ProxIF (b) Extracting the RF-Isolation vector for a subject x. *Combine* stands for the function computing the anomaly score at forest level via aggregation of the tree scores.

a distance. In other words, two ROIs which are strongly connected are at a small distance. We denote this distance-like representation as $d(x, y)$.

- The second way consists of modifying the learning algorithm by changing the direction of the inequality evaluating whether an object in an internal node n should traverse the left or the right edge. In detail, a ROI ends up in the same node as a prototype ROI if their connectivity strength is *greater*, instead of lower, than a fixed threshold (or than another connectivity value).

The main extension with respect to ProxIF is that each MS-ProxIT can be built using several sets of pairwise connectivity strengths, i.e. multiple connectivity networks, each representing a different subject. This extension is crucial since it allows having a tree structure that captures disease-wise characteristics. Indeed, employing more subjects to build one tree can be, not only very informative, but it can also help, for example, when in a connectivity network of a subject some MS-characterizing lesion l is absent. In detail, if we used only the latter network to build the MS-ProxIT t, the brain regions involved in the lesion l would end up deeper in t than anticipated, i.e. their disconnection degrees would be lower than expected. We can limit this phenomenon by building t using an additional connectivity network, i.e. a different subject. Summarizing, in addition to training each MS-ProxIT with a subset of S ROIs randomly sampled without replacement, we also randomly subsample without replacement C connectivity networks.

As to how a ROI y should traverse a node n in the tree, we adopt the standard ProxIT traversal procedure. We briefly recall the two modalities:

1. In a node n, a ROI P is chosen as prototype along with a threshold on the connectivity strength θ. θ is either

$$\theta = [\min_{x \in n} d(x, P), \max_{x \in n} d(x, P)]$$

or

$$\theta \in \{th | th = d(x, P) \forall x \in P\}$$

depending on the training strategy. Therefore, the ROI y will go to the left child n_L, along with P, if it is connected to P strongly enough with respect to θ, otherwise it will end up in n_R, the right child.

2. In a node n two different ROIs, P_L and P_R are chosen as prototypes, each representing the putative left and right child respectively. The ROI y will end up in n_L if its connection to P_L is stronger than its connection to P_R.

In the context of connectivity networks, it seems more natural to employ the first traversal modality. In detail, when using only one prototype, by analyzing how strongly connected is the ROI y to the ROI P, we obtain a partial contribution to the total disconnection degree of y, which computation is related to

the final aim of the proposed methodology. Instead, from the second traversal modality we obtain a different piece of information: we would be able to infer only to which prototype the ROI y has a stronger connection to, which is not what we are seeking.

Whereas a ProxIF is aimed at detecting outliers, the aim of an MS-ProxIF is completely different: we use the forest in an innovative way to extract features for brain connectivity networks. We can rewrite the aim in such terms: in an MS-ProxIT t, splitting a node n should generate two children such that one of them contains many ROIs strongly connected to each other and the other child contains few ROIs. The former will be the root of a big subtree and thus the isolation process of the contained regions will take longer; instead the ROIs in the smaller sibling will be more likely to be isolated soon, i.e. of being disconnected. Ideally, we would like that the split of node n results in the generation of two pure nodes: one containing only highly connected ROIs, and the other containing only disconnected ROIs.

The training scheme underneath the tree construction, i.e. deciding the test for each node, consists of the choice of either a pair of prototypes or a prototype and a threshold, as thoroughly illustrated in Chapter 3. However, in addition to that, when building an MS-ProxIT with $C > 1$ connectivity networks, i.e. subjects, we have to adapt the methodology such that it can manage multiple similarity matrices in input. In detail, when $C > 1$ the test on a node n is characterized by three elements: a prototype ROI P, a threshold on the connectivity strength θ (or two ROIs as prototypes P_L and P_R, depending on the training scheme) and a connectivity network cn, chosen among the C subjects sampled to build the tree. The connectivity network cn is necessary to partition the ROIs in n: in detail we have to evaluate

$$d^{cn}(x, P) \leq \theta,$$

that is the connectivity strength in terms of distance between a ROI x and the prototype ROI P evaluated on connectivity network cn (analogously it holds for the ProxIF variants with two prototypes). It is important to note that:

- Only ROIs, which are our objects, traverse the nodes.
- The connectivity network cn is needed *only* during the tree building procedure. In detail, cn is needed to find the best pair of prototype and threshold $(\hat{P}, \hat{\theta})$ (respectively $(\hat{P_L}, \hat{P_R})$ for $2P$) and to decide during the training procedure how to partition the ROIs.
- In other words, we do not have to retain the information about which cn was used to partition the ROIs in n after the tree building procedure has ended.
- Therefore, traversing the tree t is independent of which and how many subjects were used to build it.

The latter point is also the reason why we have to select a specific *cn* to partition n: if we partitioned independently the ROIs of each of the C subjects, we would not be able to carry out the isolation procedure correctly since in some nodes, e.g. a pair of sibling nodes, we would have duplicates of the same ROI belonging to different subjects. In other words, t would not be anymore a partitioning of the ROIs, but rather a partitioning of the ROIs of the subjects used to build t. In the latter model, not only it does not make sense, but in addition it would not carry out its function as feature extractor, i.e. it would not be able to correctly isolate a generic ROI.

The rest of the building procedure of an MS-ProxIT is identical to that of a ProxIT independently of the value of C: either the choice of the best pair $(\hat{P}, \hat{\theta})$ (or $(\hat{P_L}, \hat{P_R})$) is randomly performed, or it is optimized according to an impurity function. In addition to the training criteria proposed in Chapter 3 we design other three strategies, which core consists in a slightly different notion of scatter as the one defined in Chapter 3. In detail, when we build t with multiple connectivity networks, and we want to split a node n using a scatter-based training strategy, we have two options:

1. Compute the scatter across all ROIs of all C subjects. This leads to the previously defined criteria *O-1PS*$_D$, *O-2PS*$_D$ and *O-1PS*$_D$. We just have to slightly adapt the formulation of ScatterD and ScatterP as defined in Eqs. (3.4) and (3.5) respectively. Consider the whole matrix of connectivity strengths, in terms of distances, $\mathbf{D}$ of size $M \times |n|$ where $M = C \times |n|$ and $|n|$ is the number of ROIs in node n. This matrix is the row-concatenation of the C networks used to build the MS-ProxIT. Having defined $\mathbf{D}$, we define *ScatterD* as:

$$S_D(\mathbf{D}) = \frac{1}{M|n|} \sum_{i=1}^{M} \sum_{j=1}^{|n|} d(i,j) \tag{6.1}$$

   and *ScatterP* as:

$$S_P(\mathbf{D}, P) = \frac{1}{M} \sum_{i=1}^{M} d(i,P). \tag{6.2}$$

2. Compute the scatter across all ROIs of the connectivity network *cn* that has generated the split under evaluation. This alternative perspective on the scatter makes sense if we consider that the split of n into n_L and n_R is determined not only by the pair (P, θ) (or (P_L, P_R)) but also by the connectivity network *cn*.

The latter option leads to the design of three new training schemes. First, we redefine the *ScatterD* and *ScatterP* functions by simply replacing $\mathbf{D}$ by the matrix $\mathbf{D^{cn}}$ of size $|n| \times |n|$, which is the connectivity network, in terms of

distances, of subject *cn*. Formally we redefine *ScatterD* as:

$$S_{SD}(\mathbf{D}^{cn}) = \frac{1}{|n|^2}\sum_{i=1}^{|n|}\sum_{j=1}^{|n|} d^{cn}(i,j) \tag{6.3}$$

and *ScatterP* as:

$$S_{SP}(\mathbf{D}^{cn}, P) = \frac{1}{|n|}\sum_{i=1}^{|n|} d^{cn}(i,P). \tag{6.4}$$

Having defined S_{SD} and S_{SP}, we can employ these measures as impurity functions which are analogous to Eqs. (3.4) and (3.5) defined in Chapter 3. Please, for details on the meaning of the functions and notation, refer to Chapter 3. We define the impurity function employing S_{SD} as:

$$I_{S_{SD}}(n_L, n_R, cn) = p_L S_{SD}(\mathbf{D}^{cn}_{n_L}) + p_R S_{SD}(\mathbf{D}^{cn}_{n_R}). \tag{6.5}$$

The second function employs S_{SP} as an impurity function, and it is defined as:

$$\begin{aligned}\Delta I_{S_{SP}}(n, n_L, n_R, P_L, P_R, cn) = \frac{1}{2}(S_{SP}(\mathbf{D}^{cn}_{n}, P_L) + S_{SP}(\mathbf{D}^{cn}_{n}, P_R)) \\ -p_L S_{SP}(\mathbf{D}^{cn}_{n_L}, P_L) - p_R S_{SP}(\mathbf{D}^{cn}_{n_R}, P_R).\end{aligned} \tag{6.6}$$

Please recall that since the computation of S_{SP} is dependent on the prototypes, the evaluation of the scatter within node n is not constant and thus must be taken into account in the computation of $\Delta I_{S_{SP}}$.

Before defining the novel training criteria based on the above impurity functions, we briefly recall some concepts related to the learning stage defined in Chapter 3 and define some new ones specific to MS-ProxIF. Given a node n that we are about to split, we observe the following:

- For $1P$ variants, we create the child nodes n_L and n_R as follows:

  $$n_L = \{x | x \in n \wedge d^{cn}(x,P) \leq \theta\}$$

  and

  $$n_R = \{x | x \in n \wedge d^{cn}(x,P) > \theta\}.$$

  Please note that if $C = 1$, $d^{cn}(x,P) = d(x,P)$. In other words, n_L contains the ROIs which are strongly connected to the prototype, where the strength is defined with respect to a threshold and a connectivity network.

- For $2P$ variants, we generate n_L and n_R as follows:

  $$n_L = \{x | x \in n \wedge d^{cn}(x,P_L) \leq d^{cn}(x,P_R)\}$$

and

$$n_R = \{x | x \in n \wedge d^{cn}(x, P_L) > d^{cn}(x, P_R)\}.$$

By splitting n, each ROI is assigned to the child whose representative prototype is closer. The evaluation is done with respect to a connectivity network cn.

- In an MS-ProxIT the split of a node n is determined by a triplet of the form (P, θ, c) if the variant is $1P$; if the training criterion is $2P$ the triplet becomes (P_L, P_R, c). This holds for all training schemes. Note that if $C = 1$ then the triplet becomes the classical pair (P, θ) (or (P_L, P_R) for criteria with 2 prototypes) defined for the standard ProxIT.

- Analogously to ProxIT, in an MS-ProxIT each triplet determines a different partitioning of n.

- In an MS-ProxIT the number of triplets to evaluate in a node n is $\frac{N^2-N}{2}C$. Whereas for random criteria there is no evaluation, when employing an optimized training strategy we do not consider the entire pool of possible triplets since it would be too computationally burdening. In detail, in each node we evaluate (up to) $r = 20$ triplets following the observations made in Chapter 3.

- Please recall that the above reasoning holds only for the training procedure: a ROI that has simply to traverse n of a given MS-ProxIT will be evaluated only with respect to (P, θ) for $1P$ schemes and with respect to (P_L, P_R) for $2P$ ones.

Given these concepts, we define in the following three new *scatter-based* training criteria.

1. **O-1PS**$_{SD}$ Generate *randomly* r triplets (P, θ, cn) where $P \in n$ and $\theta \in \{d^{cn}(x, P) | x \in n\}$ and $cn \in \mathcal{C}$ where $\mathcal{C}$ is the set of C connectivity networks used to build t. Each triplet (P, θ, cn) determines a different partitioning of the node n. Each triplet is evaluated using Eq. (6.5) in order to find the best $(\hat{P}, \hat{\theta}, \hat{cn})$ selected via the following optimization function:

$$(\hat{P}, \hat{\theta}, \hat{cn}) = \min_{(P,\theta,cn)} I_{S_{SD}}(n_L, n_R, cn). \tag{6.7}$$

In other words, *O-1PS*$_{SD}$ aims at finding the triplet that minimizes the scatter of $\mathbf{D}^{cn}_{n_L}$ and $\mathbf{D}^{cn}_{n_R}$, which are respectively the submatrices of $\mathbf{D}_{n_L}$ and $\mathbf{D}_{n_R}$ relative to the connectivity network cn.

2. **O-2PS**$_{SD}$: This criterion chooses *randomly* r triplets (P_L, P_R, cn), with the two prototypes being different ROIs and cn being one of the C connectivity networks. Each triplet generates a different partition of n and is evaluated using Eq. (6.5). The best triplet $(\hat{P_L}, \hat{P_R}, \hat{cn})$ is the solution

to the following optimization problem:

$$(\hat{P_L}, \hat{P_R}, \hat{cn}) = \min_{(P_L, P_R, cn)} I_{S_{SD}}(n_L, n_R, cn). \tag{6.8}$$

The discussion is identical to that done for *O-1PS*$_{SD}$, the main difference being in the use of two prototypes instead of only one.

3. **O-2PS**$_{SP}$: This criterion chooses randomly up to r triplets of the form (P_L, P_R, cn) where $P_L \neq P_R$ are ROIs in n and $c \in \mathcal{C}$. Each triplet splits the internal node n into different child nodes n_L and n_R. The evaluation is done using Eq. (6.6) and the best triplet $(\hat{P_L}, \hat{P_R}, \hat{cn})$ is found by:

$$(\hat{P_L}, \hat{P_R}, \hat{cn}) = \max_{(P_L, P_R, cn)} \Delta I_{S_{SP}}(n, n_L, n_R, P_L, P_R, cn). \tag{6.9}$$

This criterion restricts the evaluation of the scatter to: the two prototypes and the connectivity network *cn*. Please note that it does not make sense to define a one prototype version since only one of the two child nodes will contain the prototype ROI P, as thoroughly explained in Chapter 3.

6.2.2 Step 2: RF-Isolation Extraction

As illustrated in Figure 6.1 (b), we use a trained MS-ProxIF $\mathcal{F}$ as a feature extractor for any brain connectivity network. The only constraint being that the brain regions used to build the model must be present also in the latter network.

Formally given an MS-ProxIT t, a subject represented via their connectivity network $\mathbf{D}$ and the ROI x described as the set of connectivity strengths to all other ROIs, we can obtain the outlierness degree of x in t, i.e. anomaly score, by making x traverse t. The outlierness degree of a ROI should reflect its disconnection degree.

Concerning how a t can be traversed, we can follow one of the two modalities presented in the previous section, depending on the training variant used to build t (one or two prototypes). In detail, we recall that:

- The traversal of a node n depends on the connectivity strength between the ROI x that is traversing the tree and a prototype P with respect to a threshold θ (analogously it holds if the training criterion uses two prototypes).

- Traversing t is independent of which and how many connectivity networks were used to build the MS-ProxIT. In other words, the evaluation of whether a ROI should follow the left or right edge depends exclusively on the connectivity network of the subject that is traversing t.

More formally, a ROI x traverses a node n in an MS-ProxIT in the following way: if $d(x, P) \leq \theta$ then x will traverse the left edge and go to n_L, otherwise to n_R. Analogously, it holds if two prototypes were used.

After traversing t, i.e. after the ROI x reaches a leaf in t, we can recover its disconnection degree RFI_x^t as a function of the depth of the reached leaf. Indeed, as previously explained, ROIs which are more disconnected are more likely to be isolated sooner in the tree due to their outlier-like nature, and therefore we would like them to have a higher anomaly score. To do that, we employ the original formulation of the iForest anomaly score defined in Eq. (2.4) in Chapter 2: we assign a higher anomaly score to brain regions which end up in leaves at smaller depth, since it means that their isolation process takes fewer splits. In other words, the smaller the depth, the more likely the ROI x is highly disconnected. The final anomaly score of x across the forest, i.e. its disconnection degree, RFI_x, is obtained by aggregating the tree anomaly scores –see Eq. (2.4) in Chapter 2 for details[1].

The entire *RF-Isolation* vector of a subject s, $\mathbf{RFI}_s$ is extracted by repeating this procedure for all R ROIs. Formally, we define it as:

$$\mathbf{RFI}_s = [RFI_{s1} \quad RFI_{s2} \quad \dots RFI_{sR}] \tag{6.10}$$

where RFI_{si} is the disconnection degree of the i^{th} ROI of subject s extracted from the MS-ProxIF $\mathcal{F}$ and R is the total number of ROIs.

As already explained, it is likely that the outlierness degree extracted from an MS-ProxIF model $\mathcal{F}$ is a good feature for MS subjects. Indeed, let us consider a ROI x that is usually highly disconnected in the MS population: there is a high chance that in several MS-ProxITs of the forest the weak connection of this ROI is detected during the training stage. Therefore, when extracting RFI_{sx} for an MS subject s, the ROI x is likely to get isolated after few steps and therefore to get an overall high anomaly score, i.e. disconnection degree. An analogous reasoning holds if the subject for which we extract the RF-Isolation vector is healthy. Indeed, consider the same ROI x of the previous example, which is highly disconnected in the MS population alone, and the same MS-ProxIF $\mathcal{F}$, which we recall to have been trained on an MS population. If we make a healthy subject traverse the same tree structure, the same ROI is likely to traverse a longer path, since the connectivity strength to the other ROIs which are involved in the lesion is stronger, thus leading to a lower anomaly score. Therefore, we may conclude that the model can be a good feature extractor for both MS and healthy subjects and at the same time it seems to capture some differences between the two. In other words, by using

[1]Please note that we could have used all the enhanced anomaly scores proposed in Chapter 4. However, the preliminary aim of this proposal is to test the suitability and soundness of the representation, and therefore we decided to focus only on one scoring function, the standard one. Nevertheless, the use of other anomaly scores will be the object of future research.

the *RF-Isolation* representation, there seems to be the chance of discriminating the two populations.

6.3 Experimental Evaluation

In this section, we evaluate the suitability and robustness of the proposed approach via a classification analysis. First, in Subsection 6.3.1 we describe the dataset we used for the experiments. Then in Subsection 6.3.2 we present all experimental details. The remaining subsections cover different aspects of the proposed methodology: in Subsection 6.3.3 we make several analyses to choose the best MS-ProxIF parametrization; subsequently in Subsection 6.3.4 we compare the proposed representation to standard network-derived measures, whereas in Subsection 6.3.5 we compare the proposed methodology to a variant of RF-Isolation obtained by extracting the representation from a different model for each subject. Lastly, in Subsection 6.3.6 we make a preliminary analysis, following the good classification results, on the importance of the ROIs, made to identify the most important regions involved in the disease.

6.3.1 Dataset

In this subsection we first describe the population of the study and then we briefly describe how to obtain the brain connectivity network, input of the proposed methodology, from an MRI.

Study Population

The dataset we employ consists of 79 subjects[2]: 55 suffering from MS and 24 healthy controls (HC). 13 of MS subjects are RR, while the remaining 42 are progressive. As described in detail in [138], the MS subjects are 36 females and 19 males, and at the time of the collection of data they were 50.45 ± 11.28 years old on average, and they were suffering from MS for 15.5 ± 11.6 years on average. In addition, MS subjects were not restricted to wheelchair, i.e. their Expanded Disability Status Scale (EDSS) score [91] was ≤ 7 and they satisfied the diagnostic McDonald criteria [126]. Other criteria had to be satisfied: the absence of any major systemic disease, women were not to be pregnant, no addiction to drugs or alcohol and many others, as exhaustively listed in [138]. In addition, all patients underwent a clinical examination one week within the MRI to assess the EDSS, evaluate motor and cognitive performances, depression and fatigue.

All subjects have signed a written informed consent prior to the beginning of the whole study, as the Declaration of Helsinki states. The Institutional

[2]The data have been collected at the Mount Sinai Hospital of New York (US) by the group of Prof. Matilde Inglese, affiliated with the University of Genova, Italy and the Icahn School of Medicine at Mount Sinai, New York. The dataset is not publicly available.

Table 6.1: List of the 85 parcellated gray matter ROIs, in **bold** we indicate those belonging to the motor cortex.

Nr.	Acronym	Name
1	L.BSTS	ctx-lh-bankssts
2	L.CACG	ctx-lh-caudalanteriorcingulate
3	L.CMFG	ctx-lh-caudalmiddlefrontal
4	L.CU	ctx-lh-cuneus
5	L.EC	ctx-lh-entorhinal
6	L.FG	ctx-lh-fusiform
7	L.IPG	ctx-lh-inferiorparietal
8	L.ITG	ctx-lh-inferiortemporal
9	L.ICG	ctx-lh-isthmuscingulate
10	L.LOG	ctx-lh-lateraloccipital
11	**L.LOFG**	**ctx-lh-lateralorbitofrontal**
12	L.LG	ctx-lh-lingual
13	**L.MOFG**	**ctx-lh-medialorbitofrontal**
14	L.MTG	ctx-lh-middletemporal
15	L.PHIG	ctx-lh-parahippocampal
16	**L.PaCG**	**ctx-lh-paracentral**
17	L.POP	ctx-lh-parsopercularis
18	L.POR	ctx-lh-parsorbitalis
19	L.PTR	ctx-lh-parstriangularis
20	L.PCAL	ctx-lh-pericalcarine
21	**L.PoCG**	**ctx-lh-postcentral**
22	L.PCG	ctx-lh-posteriorcingulate
23	**L.PrCG**	**ctx-lh-precentral**
24	**L.PCU**	**ctx-lh-precuneus**
25	L.RACG	ctx-lh-rostralanteriorcingulate
26	**L.RMFG**	**ctx-lh-rostralmiddlefrontal**
27	**L.SFG**	**ctx-lh-superiorfrontal**
28	**L.SPG**	**ctx-lh-superiorparietal**
29	L.STG	ctx-lh-superiortemporal
30	**L.SMG**	**ctx-lh-supramarginal**
31	L.FP	ctx-lh-frontalpole
32	L.TP	ctx-lh-temporalpole
33	L.TTG	ctx-lh-transversetemporal
34	L.IN	ctx-lh-insula
35	**L.CER**	**Left-Cerebellum-Cortex**
36	**L.TH**	**Left-Thalamus-Proper**
37	**L.CA**	**Left-Caudate**
38	**L.PU**	**Left-Putamen**
39	**L.PA**	**Left-Pallidum**
40	L.HI	Left-Hippocampus
41	L.AM	Left-Amygdala
42	L.AC	Left-Accumbens-area
43	**R.TH**	**Right-Thalamus-Proper**
44	**R.CA**	**Right-Caudate**
45	**R.PU**	**Right-Putamen**
46	**R.PA**	**Right-Pallidum**
47	R.HI	Right-Hippocampus
48	R.AM	Right-Amygdala
49	R.AC	Right-Accumbens-area
50	R.BSTS	ctx-rh-bankssts
51	R.CACG	ctx-rh-caudalanteriorcingulate
52	R.CMFG	ctx-rh-caudalmiddlefrontal
53	R.CU	ctx-rh-cuneus
54	R.EC	ctx-rh-entorhinal
55	R.FG	ctx-rh-fusiform
56	R.IPG	ctx-rh-inferiorparietal
57	R.ITG	ctx-rh-inferiortemporal
58	R.ICG	ctx-rh-isthmuscingulate
59	R.LOG	ctx-rh-lateraloccipital
60	**R.LOFG**	**ctx-rh-lateralorbitofrontal**
61	**R.LG**	**ctx-rh-lingual**
62	**R.MOFG**	**ctx-rh-medialorbitofrontal**
63	R.MTG	ctx-rh-middletemporal
64	R.PHIG	ctx-rh-parahippocampal
65	**R.PaCG**	**ctx-rh-paracentral**
66	R.POP	ctx-rh-parsopercularis
67	R.POR	ctx-rh-parsorbitalis
68	R.PTR	ctx-rh-parstriangularis
69	R.PCAL	ctx-rh-pericalcarine
70	**R.PoCG**	**ctx-rh-postcentral**
71	R.PCG	ctx-rh-posteriorcingulate
72	**R.PrCG**	**ctx-rh-precentral**
73	**R.PCU**	**ctx-rh-precuneus**
74	R.RACG	ctx-rh-rostralanteriorcingulate
75	**R.RMFG**	**ctx-rh-rostralmiddlefrontal**
76	**R.SFG**	**ctx-rh-superiorfrontal**
77	**R.SPG**	**ctx-rh-superiorparietal**
78	R.STG	ctx-rh-superiortemporal
79	**R.SMG**	**ctx-rh-supramarginal**
80	R.FP	ctx-rh-frontalpole
81	R.TP	ctx-rh-temporalpole
82	R.TTG	ctx-rh-transversetemporal
83	R.IN	ctx-rh-insula
84	**R.CER**	**Right-Cerebellum-Cortex**
85	B.Stem	Brainstem

Review Board of the Icahn School of Medicine at Mount Sinai, New York (US) approved the protocol.

MRI Acquisition and Processing

The MRI acquisition procedure was performed identically for all subjects using a Siemens Skyra 3T scanner (Siemens, Erlangen, Germany) with a 32-channels head coil –details of the acquisition procedure can be found in [138]. As to the processing of the brain images, the pipeline described in [138] was followed: briefly, the images were segmented obtaining a cortical parcellation in 85 gray matter ROIs using the Desikan–Killiany atlas [36]. At the same time of the segmentation, from the MR images, the tractography was computed using a state-of-the-art probabilistic algorithm. Lastly, the COMMIT framework [34] was employed to derive a structural connectivity network that better encodes the quantitative aspect of the MRI along with the local microstructure properties of the tissue.

The final result, which consists of the combination of the output of COMMIT with the segmentation outcome, is a connectivity network for each subject. The obtained connectivity network encodes the connectivity strength between 85 gray matter ROIs, parcellated from the segmentation step. In Table 6.1 we report the acronym and the name of each ROI: if the acronym starts with *L.* it means that the ROI is in the left brain hemisphere whereas if the prefix is *R.* it belongs to the right hemisphere. There is also a ROI, the last one, which is denoted as *B.Stem*: this region lies between the two hemispheres, hence the name. Please note that we have highlighted in **bold** numbers corresponding to motor brain regions, since they are known to be highly involved in the MS demyelination process [138, 152].

6.3.2 Experimental Details

To find the most adequate representation and evaluate the general suitability and robustness of the methodology, we generated many different MS-ProxIF models from which to extract the *RF-Isolation* vectors, by varying:

- Number of trees T in a forest: $50, 100, 200$.
- Number of regions S used to build each tree t sampled without replacement: $43, 64, 85$ which correspond respectively to the $50\%, 75\%$, and 100% of ROIs.
- Number of connectivity networks C used to build each tree t sampled without replacement: $1, 5, 10, 20, All$.
- Number of training criteria: the 9 presented in Chapter 3 and the 3 presented in Section 6.2 which we recall are *O-1PS*$_{SD}$, *O-2PS*$_{SD}$ and *O-2PS*$_{SP}$.
- Maximum depth D: $\log_2(S), S - 1$.
- Training population: MS or HC –we recall that HC stands for Healthy Controls. Please note that our initial assumption was that the former is

the most appropriate choice, nevertheless to confirm it from an experimental point of view we also extracted the *RF-Isolation* representation starting from an *MS-ProxIF* model trained using only healthy subjects[3].

We extract from each trained forest the *RF-Isolation* vector for each of the 79 subjects. Then we classify the subjects using the obtained representation. Classification is performed using a Leave One Out Protocol (LOO) [147] and 3 very well known binary classifiers: Support Vector Machines (SVM), Random Forests (RF) and K-Nearest Neighbor (KNN). As to SVM, we employ the linear kernel and the other parameters are optimized using the built-in MATLAB function. Instead, RFs are composed by 100 trees each trained using all subjects. Finally, as to KNN we use the Euclidean distance and as to other parameters they are optimized using the built-in MATLAB function. As to the LOO protocol, please note that it is applied only to the classification stage: each classifier is trained 79 times on a dataset of 78 subjects and tested on the left out subject. For robustness purposes, we repeat the whole procedure, i.e. from training the model to classification, 5 times.

Classification performances are measured using the normalized Matthews Correlation Coefficient (MCC): this measure has been shown to be more adequate than accuracy, especially when dealing with unbalanced datasets [31]. In detail, while accuracy simply counts the number of correctly classified subjects, MCC takes into account all four entries of the confusion matrix, thus reaching a high value only if both classes are correctly identified. Formally, we compute the MCC as follows:

$$MCC = \frac{(TP \cdot TN) - (FP \cdot FN)}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}} \tag{6.11}$$

where we recall TP and FP are respectively true and false positives whereas TN and FN represent true and false negatives –please refer to Chapter 2 for the definition. The MCC index ranges in $[-1, 1]$, with 1 being completely correct classification. The normalized version of MCC ranges in $[0, 1]$ and is obtained as follows:

$$nMCC = \frac{(MCC + 1)}{2} \tag{6.12}$$

please note that in the remainder of the chapter, we will use the normalized version and simply note it as MCC.

6.3.3 Analysis of the MS-ProxIF model

The first analysis aims at understanding the most suitable parametrization of the MS-ProxIF model. We decided not to follow the guidelines of Chapter 3 to set the parameters for two reasons: i) the final task is very different from the standard outlier detection experiments performed in Chapter 3; ii) the

[3]We decided not to train the model using *both* populations, following our initial assumption that this choice would probably lead to a non-discriminative representation.

MS-ProxIF model is an extension of ProxIF, and we have to set additional parameters and training strategies.

The analysis of each parameter is done as follows: we fix the parameter under analysis to one of the possible values, we fix a training criterion, and we fix a classifier. Then we compute the average MCC across all other parameters. For each analysis, we carry out a Wilcoxon signed-rank test or a Friedman test followed by a post-hoc Nemenyi test, depending on whether the comparison is pairwise or not. The significance level is identical for all tests, set to $\alpha = 0.05$. Please note that unless expressly stated otherwise, each analysis is independent of the others. Further, note that MCC values may be smaller than expected because averaged across several parametrizations –including non-beneficial ones.

Table 6.2: Comparing models trained with MS subjects with those trained with HCs.

	SVM		**KNN**		**RF**	
Criterion	**MS**	**HC**	**MS**	**HC**	**MS**	**HC**
R-1P	**0.7882**	0.7605	**0.7010**	0.6765	**0.7802**	0.7687
R-2P	**0.6280**	0.5932	**0.5437**	0.5352	**0.6234**	0.6066
O-1PS$_D$	**0.8077**	0.7998	**0.7429**	0.7299	**0.7940**	0.7869
O-2PS$_D$	**0.6323**	0.5549	**0.6098**	0.5860	**0.7039**	0.6424
O-1PS$_{SD}$	**0.8094**	0.8027	**0.7407**	0.7289	**0.7938**	0.7911
O-2PS$_{SD}$	**0.6297**	0.5604	**0.5981**	0.5701	**0.6891**	0.6318
O-2PS$_P$	**0.6294**	0.6258	**0.5459**	0.5356	**0.6213**	0.5898
O-2PS$_{SP}$	**0.6175**	0.6118	**0.5589**	0.5325	**0.6249**	0.5950
O-1PH	**0.6975**	0.5884	**0.6237**	0.5354	**0.7195**	0.6125
O-2PH	**0.6068**	0.5824	**0.5594**	0.5220	**0.6103**	0.5674
O-1PRD	**0.8332**	0.7826	**0.7413**	0.6709	**0.8244**	0.7530
O-2PRD	0.6113	**0.6262**	**0.5281**	0.5218	0.5989	**0.6066**

Table 6.3: Results of the Wilcoxon signed-rank test comparing the two training populations: MS vs HC.

MS Mean Rank	**HC Mean Rank**	**P-value**
1.056	1.944	**0.0049**

The first parameter we analyze is the training population: in Table 6.2 we report the results in terms of averaged MCC, where the average has been computed following the just described procedure. We highlight in **bold** the best result for each algorithm and training criterion. We can clearly observe that independently of the classifier and training criterion the best choice is to train using the MS population: in detail, except for one case, i.e. *O-2PRD* when used in combination with SVM and RF, it is always better and in some cases the difference in terms of MCC is quite big. To confirm our observations we carried out a non-parametric Wilcoxon signed-rank test comparing the two training populations; the test was carried out using 36 observations, i.e. MCC values: 12 training variants analyzed across 3 classifiers. We depict the results of the statistical test in Table 6.3: we report the mean rank of both training

populations and the p-value output by the test, in **bold** if the null hypothesis is rejected. We can observe that our hypothesis is confirmed: using the MS population for training the model is significantly better than using the healthy cohort, i.e. the p-value is much lower than α and in addition the difference between the mean ranks is almost maximum.

Since choosing **MS** as training population is one of the core assumptions of our approach, the subsequent analyses are restricted to the experiments in which the MS-ProxIF model has been built on the MS population only.

Table 6.4: Analyzing the behaviour of different values for the forest size T.

	SVM			KNN			RF		
Criterion	**T=50**	**T=100**	**T=200**	**T=50**	**T=100**	**T=200**	**T=50**	**T=100**	**T=200**
R-1P	0.7847	**0.7955**	0.7843	0.6890	**0.7122**	0.7020	0.7699	0.7836	**0.7873**
R-2P	0.6291	**0.6310**	0.6239	**0.5583**	0.5544	0.5183	0.6128	**0.6296**	0.6277
O-1PS$_D$	0.8008	**0.8136**	0.8086	0.7302	**0.7570**	0.7413	0.7863	0.7946	**0.8011**
O-2PS$_D$	**0.6369**	0.6310	0.6292	0.6136	**0.6183**	0.5975	0.6840	0.7102	**0.7174**
O-1PS$_{SD}$	0.7995	**0.8179**	0.8108	0.7257	**0.7520**	0.7443	0.7813	0.7958	**0.8043**
O-2PS$_{SD}$	0.6262	0.6266	**0.6363**	0.6012	**0.6056**	0.5876	0.6640	0.6956	**0.7078**
O-2PS$_P$	0.6257	**0.6399**	0.6227	0.5430	**0.5593**	0.5355	0.6099	**0.6278**	0.6263
O-2PS$_{SP}$	0.6025	0.6198	**0.6300**	0.5546	**0.5721**	0.5501	0.6032	0.6281	**0.6433**
O-1PH	0.6808	0.7057	**0.7059**	0.6223	**0.6339**	0.6148	0.7008	0.7221	**0.7355**
O-2PH	0.6037	**0.6180**	0.5985	0.5654	**0.5654**	0.5473	0.6071	**0.6125**	0.6112
O-1PRD	0.8088	0.8440	**0.8468**	0.7242	**0.7566**	0.7431	0.7950	0.8336	**0.8446**
O-2PRD	**0.6244**	0.6105	0.5991	**0.5403**	0.5378	0.5064	0.5951	0.5999	**0.6016**

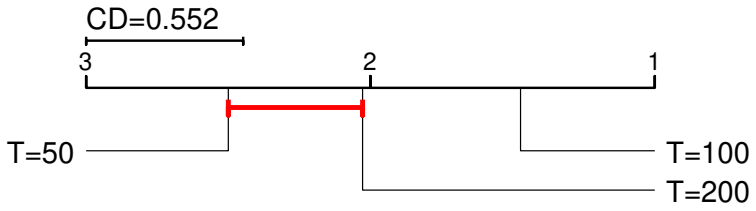


Figure 6.2: CD diagram comparing the different options for the forest size T.

The second parameter we want to find the best setting of is the forest size T: in Table 6.4 we report the averaged MCC across all other parametrization given a classifier, a value of T and a training criterion, highlighting in **bold** the best option for each classifier and criterion. From the table, we can observe that for the majority of the training strategies the best option is $T = 100$ aside from when classifying using RF. We can in general observe that $T = 50$ seems to be too small to be considered as a valuable option. To understand whether there is an option which is significantly better than the others, we perform, as mentioned in the preamble of the section, a Friedman test followed by a post-hoc Nemenyi test, which results are depicted via a CD diagram in Figure 6.2: it is clear that $\mathbf{T = 100}$ is the best option in a statistically significant way, whereas $T = 200$ and $T = 50$ are actually comparable. Please note that setting $T = 100$ is not uncommon for random forest-based methodologies [96, 97, 128] even though there is no golden standard.

Table 6.5: Analyzing the behaviour of different number of ROIs S used to build each tree.

	SVM			KNN			RF		
Criterion	**S=50%**	**S=75%**	**S=100%**	**S=50%**	**S=75%**	**S=100%**	**S=50%**	**S=75%**	**S=100%**
R-1P	**0.7971**	0.7955	0.7718	**0.7114**	0.7011	0.6905	0.7833	0.7703	**0.7871**
R-2P	0.5600	0.6040	**0.7199**	0.5309	0.5353	**0.5649**	0.5953	0.6244	**0.6504**
O-1PS$_D$	**0.8145**	0.8051	0.8034	**0.7507**	0.7294	0.7485	0.7928	0.7886	**0.8007**
O-2PS$_D$	**0.6382**	0.6331	0.6257	0.6054	0.5927	**0.6315**	**0.7108**	0.6986	0.7021
O-1PS$_{SD}$	**0.8225**	0.8056	0.8002	**0.7557**	0.7273	0.7390	**0.7987**	0.7863	0.7964
O-2PS$_{SD}$	0.6148	0.6352	**0.6391**	0.5815	0.5907	**0.6222**	0.6701	0.6889	**0.7083**
O-2PS$_P$	0.5786	0.6032	**0.7064**	0.5248	0.5355	**0.5775**	0.6125	0.6067	**0.6448**
O-2PS$_{SP}$	0.5690	0.5932	**0.6902**	0.5327	0.5597	**0.5843**	0.6075	0.6091	**0.6580**
O-1PH	0.6854	0.6864	**0.7207**	**0.6383**	0.6127	0.6200	0.7104	0.7143	**0.7337**
O-2PH	0.5677	0.5614	**0.6911**	0.5388	0.5375	**0.6018**	0.5780	0.5852	**0.6677**
O-1PRD	0.8155	0.8322	**0.8518**	0.7324	0.7450	**0.7464**	0.8071	0.8280	**0.8381**
O-2PRD	0.5726	0.5620	**0.6994**	0.5134	0.5087	**0.5622**	0.5645	0.5821	**0.6499**

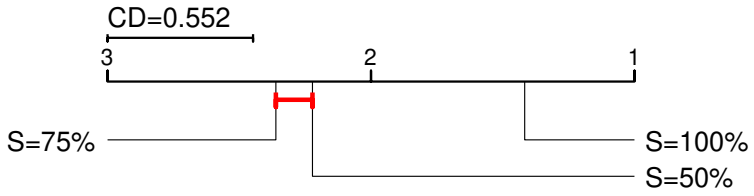


Figure 6.3: CD diagram comparing the different options for the number of ROIs S used to build each tree.

Table 6.6: Analyzing the behaviour of different values for the maximum depth D.

	SVM		KNN		RF	
Criterion	**D** $= \log_2(\mathbf{S})$	**D** $= \mathbf{S} - 1$	**D** $= \log_2(\mathbf{S})$	**D** $= \mathbf{S} - 1$	**D** $= \log_2(\mathbf{S})$	**D** $= \mathbf{S} - 1$
R-1P	**0.7932**	0.7831	**0.7100**	0.6920	**0.7819**	0.7785
R-2P	**0.6369**	0.6191	**0.5501**	0.5373	0.6228	**0.6240**
O-1PS$_D$	**0.8148**	0.8005	**0.7545**	0.7313	**0.7957**	0.7924
O-2PS$_D$	**0.6415**	0.6232	**0.6130**	0.6066	0.7021	**0.7056**
O-1PS$_{SD}$	**0.8196**	0.7992	**0.7506**	0.7307	**0.7960**	0.7916
O-2PS$_{SD}$	**0.6409**	0.6185	**0.6051**	0.5912	0.6867	**0.6915**
O-2PS$_P$	**0.6480**	0.6108	**0.5595**	0.5323	0.6188	**0.6239**
O-2PS$_{SP}$	**0.6341**	0.6008	**0.5750**	0.5428	**0.6253**	0.6244
O-1PH	**0.7045**	0.6905	**0.6343**	0.6130	0.7190	**0.7200**
O-2PH	**0.6190**	0.5946	**0.5713**	0.5474	**0.6108**	0.6098
O-1PRD	**0.8332**	0.8332	0.7383	**0.7443**	0.8186	**0.8302**
O-2PRD	**0.6280**	0.5947	**0.5452**	0.5111	**0.6016**	0.5961

Table 6.7: Results of the Wilcoxon signed-rank test comparing the two options for D.

Mean Rank		
D $= \log_2(\mathbf{S})$	**D** $= \mathbf{S} - 1$	**P-value**
1.1944	1.8056	$4.88 \cdot 10^{-4}$

Further, we analyze the number of ROIs used to build each tree. We performed experiments by setting S to 3 possible values, as already described in Section 6.3.2: $S = 50\% = 43$, $S = 75\% = 64$ and $S = 100\% = 85$, i.e. all regions. In Table 6.5 we report the results of the classification experiments obtained and visualized analogously to T. We can make several observations: i) $S = 75\%$ is never the best option; ii) if $S = 50\%$ is the best choice we tend to have decreasing performance as S increases, i.e. in many of such cases $S = 75\%$ is better than $S = 100\%$; iii) similarly it holds if $S = 100\%$: performances decrease as S does. The statistical analysis, depicted in Figure 6.3 via a CD diagram, assesses that using all ROIs is the best choice, whereas the other two options lead to comparable representations. Please note that one could assume that using all ROIs would remove completely the randomization from the training procedure, but that is not true: first in each node only a small subset of tests is evaluated, and secondly, and most importantly, each tree is built using a random subsample of connectivity networks as well.

In Tables 6.6 and 6.7 we make the same analysis but for the depth parameter D. By looking at the averaged MCC in the Table, except when using RF as classifier, it is clear that the best option is to use $D = \log_2(S)$, a common choice in the context of isolation-based approaches [24, 57, 96, 98] which also represents the average tree height [89]. This is not true for RF, for which instead it seems that almost all $1P$ learning strategies prefer bigger trees. To assess whether there is a best significant option for D, we performed a pairwise comparison via a Wilcoxon signed-rank test. In Table 6.7 we report the $p - value$ returned by the test along with the mean ranks assigned to the two values of D: we can observe that the differences in terms of ranks is quite astonishing. Indeed, the statistical analysis confirms that in general, using a smaller depth, i.e. $\mathbf{D = \log_2(S)}$, is the best significant option.

One of the most interesting parameters is C, the number of connectivity networks, i.e. subjects, used to build each tree. Differently from ProxIF, which takes as input only one distance matrix, MS-ProxIF can manage multiple ones in input. In detail, we analyzed five different options for C: from 1 up to all subjects ($C = 55$). In Table 6.8 we report the results of the analysis of C: we can observe that for the majority of the training strategies, independently of the classifier, the best results are reached with $\mathbf{C = 10}$. The statistical analysis makes some clarity: from the CD diagram presented in Figure 6.4 we confirm that $C = 10$ is the best option even though in a non-significant way, since it is comparable to all other values of C except for $C = 1$, which is ranked last. This observation confirms the usefulness of employing multiple connectivity networks to build each tree, in other words, setting $C > 1$ allows to obtain a more informative *RF-Isolation* representation. We decided to set $C = 10$.

The last analysis compares the different training strategies, since even though in the previous analyses we have reported the results for each of them they have not been the main focus. We report the results in terms of averaged MCC in Table 6.9: *O-1PRD* is the best choice for both RF and SVM, and it is the second best performing on KNN as well, after *O-1PS*$_D$. In addition to

Table 6.8: Analyzing the behaviour of different values for the number of connectivity networks C used to build each MS-ProxIT.

	SVM					KNN				
Criterion	**C=1**	**C=5**	**C=10**	**C=20**	**C=55**	**C=1**	**C=5**	**C=10**	**C=20**	**C=55**
R-1P	0.7795	**0.7966**	0.7909	0.7871	0.7868	0.6955	**0.7124**	0.7070	0.6920	0.6983
R-2P	0.6189	0.6321	**0.6521**	0.6244	0.6124	0.5385	**0.5607**	0.5493	0.5326	0.5373
O-1PS$_D$	0.7984	0.8078	**0.8131**	0.8084	0.8106	0.7375	0.7437	**0.7452**	0.7465	0.7413
O-2PS$_D$	0.6090	0.6251	**0.6482**	0.6464	0.6330	0.5973	0.6161	**0.6218**	0.5927	0.6213
O-1PS$_{SD}$	0.7984	0.8047	**0.8192**	0.8090	0.8158	0.7375	0.7326	**0.7408**	0.7456	0.7469
O-2PS$_{SD}$	0.6090	0.6273	**0.6394**	0.6375	0.6352	0.5973	**0.6013**	0.5892	0.5962	0.6067
O-2PS$_P$	0.6159	0.6195	**0.6414**	0.6317	0.6385	0.5524	0.5515	**0.5547**	0.5308	0.5402
O-2PS$_{SP}$	0.6159	**0.6486**	0.6114	0.6163	0.5951	0.5524	**0.5635**	0.5571	0.5668	0.5547
O-1PH	0.6779	0.7038	**0.7060**	0.6867	0.7130	0.6092	**0.6243**	0.6215	0.6396	0.6237
O-2PH	0.5960	0.6019	0.6087	0.6063	**0.6209**	0.5461	0.5615	**0.5654**	0.5521	0.5717
O-1PRD	0.8377	0.8320	**0.8394**	0.8263	0.8306	0.7428	0.7315	**0.7569**	0.7327	0.7425
O-2PRD	0.6036	**0.6387**	0.6042	0.5819	0.6282	0.5275	**0.5293**	0.5205	0.5334	0.5300

	RF				
Criterion	**C=1**	**C=5**	**C=10**	**C=20**	**C=55**
R-1P	0.7697	0.7803	**0.7841**	0.7875	0.7795
R-2P	0.6224	0.6178	**0.6328**	0.6282	0.6157
O-1PS$_D$	0.7859	0.7938	**0.7954**	0.7992	0.7958
O-2PS$_D$	0.6928	0.7002	**0.7135**	0.7071	0.7057
O-1PS$_{SD}$	0.7859	0.7888	**0.8013**	0.7978	0.7952
O-2PS$_{SD}$	**0.6928**	0.6791	0.6923	0.6898	0.6915
O-2PS$_P$	**0.6282**	0.6088	0.6224	0.6243	0.6231
O-2PS$_{SP}$	0.6282	**0.6322**	0.6124	0.6361	0.6155
O-1PH	0.7074	0.7221	**0.7246**	0.7308	0.7124
O-2PH	0.6098	0.6080	**0.6172**	0.6040	0.6124
O-1PRD	0.8220	0.8177	**0.8351**	0.8279	0.8192
O-2PRD	0.5835	**0.6073**	0.5973	0.5880	0.6181

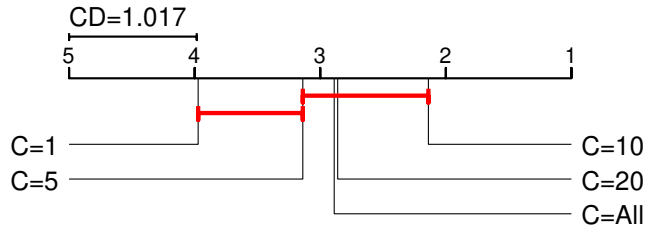


Figure 6.4: CD diagram comparing the 5 options for the number of connectivity networks used to build each tree C.

reporting the results for each single variant, we also report the average MCC across all criteria using one prototype and across all criteria that employ two, to assess our assumption that in this context using $1P$ training schemes is the most suitable choice. From the table, we can conclude that results largely confirm our hypothesis. The CD diagram in Figure 6.5 confirms what we observed from the table: **O-1PRD** is the best choice, comparable only to *O-1PS$_D$* and *O-1PS$_{SD}$*. In addition, training strategies that use two prototypes are all ranked worse than all criteria using one prototype, and none of the former is comparable to any of the 1P training schemes. Please note that the input of

Table 6.9: Analyzing the behaviour of different training strategies.

Criterion	SVM	KNN	RF
R-1P	0.7882	0.7010	0.7802
R-2P	0.6280	0.5437	0.6234
O-1PS$_D$	0.8077	**0.7429**	0.7940
O-2PS$_D$	0.6323	0.6098	0.7039
O-1PS$_{SD}$	0.8094	0.7407	0.7938
O-2PS$_{SD}$	0.6297	0.5981	0.6891
O-2PS$_P$	0.6294	0.5459	0.6213
O-2PS$_{SP}$	0.6175	0.5589	0.6249
O-1PH	0.6975	0.6237	0.7195
O-2PH	0.6068	0.5594	0.6103
O-1PRD	**0.8332**	0.7413	**0.8244**
O-2PRD	0.6113	0.5281	0.5989
1P Avg.	0.7872	0.7099	0.7824
2P Avg.	0.6221	0.5634	0.6388

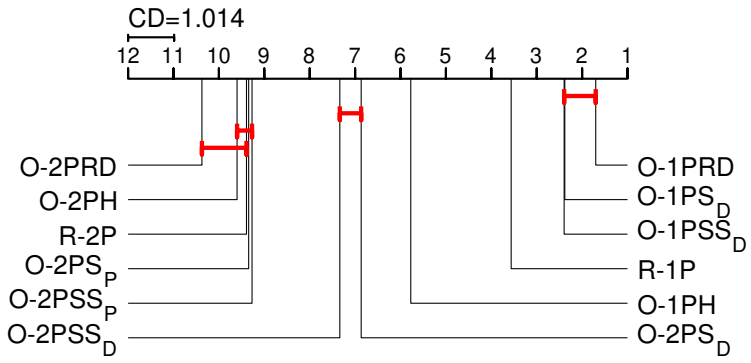


Figure 6.5: CD diagram comparing the 12 training strategies.

this test is different from the previous ones, since for a given training strategy we consider the MCC results across all parametrizations and classifiers.

From these analyses, we have derived the most suitable parametrization for the proposed approach, which we will adopt in all the remaining analyses. In detail, after confirming our assumption that a *MS-ProxIF* must be trained only with MS subjects, we have set: $\mathbf{S = 100\%}$, $\mathbf{C = 10}$, $\mathbf{T = 100}$, $\mathbf{D = \log_2(S)}$ and the training scheme to **O-1PRD**.

6.3.4 Comparison to Standard Network Measures

In this section, we compare the proposed methodology to standard network-derived measures. We have selected the metrics which have shown to be the most relevant in MS, as illustrated in [138] and the references therein. For most of them we have used the related function implemented in the Brain Connectivity Toolbox [133]; only if the function was absent we implemented it ourselves. We extracted both local and global measures. Local measures, analogously to *RF-Isolation*, have one feature for each ROI; global measures instead describe a connectivity network and its components via one value alone.

In detail, from each subject's connectivity network, where each ROI is a node and a connection between two ROIs an edge, we extracted the following metrics –for a more thorough description please refer to `https://sites.google.com/site/bctnet/measures/list`:

- *Node Strength*: it is the sum of the connections of a node to all other nodes, i.e. how connected it is to all other nodes.
- *Local Efficiency*: it is the average inverse shortest path length in the neighborhood of the node; the higher the connection between two neighbors the higher their contribution.
- *Assortativity*: it is measured in terms of node strength. If positive, it means that nodes tend to link to other nodes which node strength is similar.
- *Clustering Coefficient*: it is the sum of each node clustering coefficient, i.e. the number of neighbors that are neighbors of each other.
- *Density*: it is calculated strictly on the presence or absence of an edge between two nodes, i.e. the unweighted version of the network. It is the ratio between the number of existing connections to the number of possible connections.
- *Global Efficiency*: it is the average inverse shortest path length in the network. It is related to the characteristic path length.
- *Mean Strength*: it is the average node strength computed across all nodes.
- *Modularity*: a network can be partitioned into subnetworks such that the within strength is maximized and the between strength is minimized. The modularity is a statistic describing the goodness of such partitioning.

Since our representation has the novelty of being extracted from a previously trained *MS-ProxIF* model, we have to set the model-related parameters. In detail, we set them according to the analysis in the previous section: $S = 100\%$, $C = 10$, $T = 100$, $D = \log_2(S)$ and the training strategy to *O-1PRD*, other than building the forest using only the MS subjects.

In Table 6.10 we report for each classifier the MCC for each representation except for *RF-Isolation* for which we report the average MCC across the 5 iterations of the model –please recall that we iterate the model for robustness purposes. We indicate in **bold** the best representation for each classifier, i.e. the one reaching the highest performance. Further, for this analysis, we report only the standard errors of the mean to validate the obtained results. We made this choice since: the standard error of the mean already gives an idea of the significance of the results; and the recently introduced tests for LOO [158], which are based on the comparison of the predicted labels, rely on the strong assumption that labels must be normally distributed.

Table 6.10: Comparison in terms of MCC of the proposed methodology, RF-Isolation, with standard network-derived metrics.

Representation	**SVM**	**KNN**	**RF**
RF-Isolation	**0.8686(0.0128)**	0.7822(0.0192)	**0.8554(0.0139)**
Local Efficiency	0.7997(0.0180)	**0.7843(0.0190)**	0.7799(0.0193)
Node Strength	0.8167(0.0168)	0.6559(0.0254)	0.8127(0.0171)
Assortativity	0.5000(0.0281)	0.5000(0.0281)	0.4790(0.0281)
Clustering Coefficient	0.5000(0.0281)	0.5950(0.0271)	0.6110(0.0267)
Density	0.5000(0.0281)	0.4369(0.0277)	0.6325(0.0262)
Global Efficiency	0.6226(0.0264)	0.7373(0.0218)	0.5818(0.0274)
Mean Strength	0.7261(0.0224)	0.6760(0.0246)	0.6102(0.0268)
Modularity	0.5812(0.0274)	0.7487(0.0212)	0.6392(0.0259)

Obtained results suggest that *RF-Isolation*, a disease-wise representation extracted from a trained model, can represent a valid alternative to standard metrics for connectivity networks, which are instead computed independently for each subject. In detail, better results are achieved with respect to all representations independently of the employed classifier. This observation is always true, aside for one case: *RF-Isolation* is slightly less performing than Local Efficiency when using KNN –even though it is the second-best representation.

6.3.5 Comparison with Subject-Wise RF-Isolation

In this subsection we compare the proposed methodology to a technique which is a *compromise* between our representation, RF-Isolation, and subject-wise network-derived metrics.

We call this *intermediate* representation *Subject-Wise RF-Isolation*, in short SWRF-Isolation. To obtain the SWRF-Isolation representation for a subject s, which is represented by their connectivity network $\mathbf{D}$, the procedure is the following:

1. Build an *MS-ProxIF* model $\mathcal{F}$ using as training population only the subject s. Each *MS-ProxIT* will thus be built with $C = 1$ connectivity networks, that of subject s.

2. Extract from $\mathcal{F}$ the *SWRF-Isolation* vector for the subject s.

In other words, the model $\mathcal{F}$ is used as feature extractor only for s: if a new subject s' comes we have to train another *MS-ProxIF* model $\mathcal{F}'$ following the above procedure, which is summarized in Figure 6.6. In detail, instead of having 1 model like for *RF-Isolation*, to extract the *SWRF-Isolation* vector for $|\mathcal{S}|$ subjects we have to build $|\mathcal{S}|$ different models.

We perform a comparative analysis between *SWRF-Isolation* and *RF-Isolation* to assess whether and how much the increase in performances observed using *RF-Isolation* with respect to standard network-derived metrics is

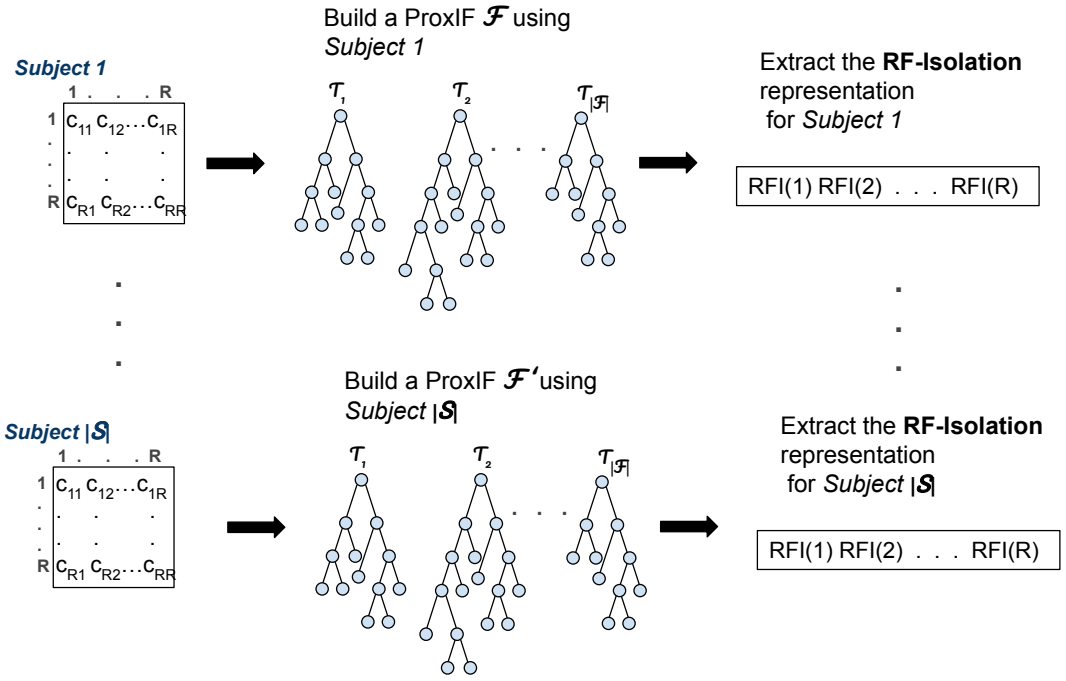


Figure 6.6: Pipeline of the Subject-wise RF-Isolation representation.

Table 6.11: Comparison between RF-Isolation ad SWRF-Isolation in terms of MCC.

Criterion	RF-Isolation	SWRF-Isolation
O-1PS$_D$ (SVM)	**0.8207**	0.5075
O-1PS$_D$(KNN)	**0.7635**	0.4885
O-1PS$_D$ (RF)	**0.7936**	0.5520
O-1PS$_{SD}$ (SVM)	**0.8166**	0.5075
O-1PS$_{SD}$ (KNN)	**0.7569**	0.4885
O-1PS$_{SD}$ (RF)	**0.7999**	0.5520
O-1PRD (SVM)	**0.8686**	0.5167
O-1PRD (KNN)	**0.7822**	0.5967
O-1PRD (RF)	**0.8554**	0.5788
Mean Rank	1	2

due to the fact it is a disease-wise representation. We set the parameters according to the analysis in Section 6.3.2 aside for the learning strategies. As to the latter we report the results for three learning strategies: *O-1PS$_D$*, *O-1PS$_{SD}$* and *O-1PRD*, that is all those criteria which are comparable to the first ranked training strategy according to Figure 6.5. In Table 6.11 we report the average MCC across the 5 iterations for each classifier and training variant for both representations. We highlight in **bold** the best model between the two, and in the last row we also report the mean rank. We can observe that *RF-Isolation* is the best option independently of the training criterion and classifier; in addition, the differences between the MCC values are in all cases quite relevant.

On the MCC values reported in Table 6.11 we performed a Wilcoxon signed-rank test, which returned a p-value of $0.0039 < 0.05$, confirming again the superiority of the proposed technique.

In conclusion, this analysis further assesses the informativeness and suitability of deriving and employing a disease-wise representation. In addition, it discards as a valid alternative this *compromise* representation, *SWRF-Isolation*.

6.3.6 Preliminary analysis of ROI importance

In this section, we make a preliminary analysis to understand whether the importance of the ROIs can be extracted from the model, *MS-ProxIF*. In general, it is of crucial importance to understand the relevance of a ROI within both the model and the representation. Indeed, there is a good possibility that the most relevant ROIs are those more associated to MS and thus possibly involved in the MS demyelination process. If the latter is true –which can be assessed via other types of analyses– we can use this knowledge as a starting point to uncover unknown mechanisms on how MS works on the brain.

Since MS-ProxIF is an RF-based approach, we can extract the importance of each ROI by adopting similar approaches to those used for standard RFs. There exist several approaches to compute how relevant is a feature (which in our context is a ROI) within an RF. In the following we describe two of the most used, studied and extended approaches:

- **Permutation Feature Importance**: this technique was introduced by [19] in the context of RF, but it is commonly used also in other scenarios. The main idea is that we can retrieve the relevance of a feature i by measuring the misclassification rate in a validation set where the values along feature i have been exchanged among the objects. In detail, given a trained RF $\mathcal{F}$ we recover the importance of a feature i as follows:

  1. Fix a tree t and consider its set of out-of-bag samples, i.e. those objects which were not used to build the tree.
  2. Shuffle the values of the out-of-bag samples along feature i.
  3. Make each out-of-bag sample traverse t and record the predicted label.
  4. Repeat the procedure for all trees.
  5. Assign to each object x the most frequent label assigned by those trees in which x is an out-of-bag sample.
  6. By comparing the predicted labels to the ground truth, we obtain the misclassification rate of the out-of-bag samples with respect to the permuted feature i.
  7. The importance of feature i is measured by the ratio of the value obtained in the previous step with the misclassification rate of the out-bag-samples computed without performing Step 2., i.e. the shuffling procedure.

- **Mean Decrease Impurity (MDI) Feature Importance**: this approach was originally designed in the context of CARTs [17] but it has been widely employed also in RF scenarios. The starting assumption is that the importance of a feature is correlated to its contribution in decreasing the impurity within a tree. In detail, we can measure the importance of a feature i in a built RF $\mathcal{F}$ as follows:

  1. Fix a tree $t \in \mathcal{F}$.
  2. Retrieve the nodes in which the split has been performed along feature i.
  3. For each node n of the above, retrieve:
     - The value $\Delta I(n, n_L, n_R)$ of the impurity decrease caused by splitting n. The impurity is measured using one of the functions illustrated in Chapter 2.
     - The importance of the node in the tree. This is commonly encoded as the percentage of samples that have reached n, i.e. $\frac{|n|}{S}$.
  4. Combine the two elements via multiplication, $\Delta I(n, n_L, n_R)\frac{|n|}{S}$[4].
  5. Sum the total impurity decrease in tree t determined by the i^{th} feature.
  6. The importance of the feature i in t is the ratio between the sum obtained at Step 5. and the total impurity decrease in the tree t.
  7. Repeat steps 1.-6. for all trees in $\mathcal{F}$.
  8. The importance of the feature i in $\mathcal{F}$ is its averaged importance across all trees.

Please note that these techniques have been developed and used in the field of classification and regression, whereas their application in the context of outlier detection is completely unexplored.

In our context, we adopt as starting point the latter technique, since more consistent with our approach. We start by making a preliminary analysis which takes the approach to an extreme: we measure the importance of a ROI by counting the number of nodes in which it has been chosen as prototype weighted by the importance of such nodes, and leaving out the factor related to the impurity function. In detail, given a trained MS-ProxIF $\mathcal{F}$ we compute the importance of a ROI i as follows:

1. Fix an MS-ProxIT $t \in \mathcal{F}$.
2. Retrieve the nodes in which the split has been performed using the ROI i as prototype.

[4]Please note that this may depend on the used impurity function. For example, Gini already includes the latter factor in the impurity computation.

3. For each node n of the above, retrieve its importance in the tree via computing $\frac{|n|}{S}$.

4. The importance of the ROI i in t is the sum of the importance of the nodes in which the i^{th} ROI is used as prototype.

5. Repeat the previous Steps 1.-4. for all MS-ProxITs which include ROI i in their training set. Indeed, not all ROIs are necessarily used to build each tree.

6. The importance of the ROI i in $\mathcal{F}$ is its averaged importance across all trees in which the ROI i is present.

7. Repeat the previous steps for all the 5 iterations of the model and average the importance of the ROI i across them.

Table 6.12: ROIs most used as prototypes: (a) standard MS-ProxIF parametrization (b) standard MS-ProxIF parametrization aside for $D = S - 1$.

Rank	**Name**
#**1**	L.BSTS
#**2**	R.FG
#**3**	R.BSTS
#**4**	L.FG
#**5**	**L.LOFG**
#**6**	R.CU
#**7**	L.EC
#**8**	**L.CER**
#**9**	L.POP
#**10**	**R.PU**

Rank	**Name**
#**1**	L.BSTS
#**2**	**L.CER**
#**3**	R.BSTS
#**4**	**R.PU**
#**5**	**L.LOFG**
#**6**	R.FG
#**7**	L.EC
#**8**	R.CU
#**9**	L.FG
#**10**	R.HI

We use the above approach on the MS-ProxIF model built with the best parametrization according to Section 6.3.2. According to the last step, we repeat the computation of the importance of each ROI for each iteration of the model, and then we average the importance of each ROI across said iterations. The result of the analysis is depicted in Table 6.12 (a): we report the 10 regions that are most used as prototypes. We highlight in **bold** those ROIs that belong to the motor cortex. In general, to validate whether a ROI is good we either need the confirmation from an expert or to correlate the findings with, for example, the clinical profiles of the population. Nevertheless, from Table 6.12 (a) we can observe that 3 of the found ROIs belong to the motor area, which is known to be one of the most involved in MS [138], but they all have a rank ≥ 5. This initial result is not bad, however there is large room for improvements. Indeed, we wondered whether an improvement could be observed if the same trees were built to their maximum depth, since longer trees are likely to be more informative. We present the results of such analysis in Table 6.12 (b): the identified motor brain regions are the same, but they are

better ranked, i.e. all have a rank ≤ 5. Further, the other ROIs are the same ones identified in Table 6.12 (a) –except for one. In conclusion, resuming the building procedure of an *MS-ProxIT* t which has not been built to the end, seems to suggest that a more reliable estimate of which are the most relevant ROIs can be obtained.

We decided to take a further step and adopt an approach more similar to the *MDI feature importance* technique that we presented at the beginning of this section. Indeed, with respect to the latter the differences are just two: i) we adopt as impurity functions the optimization functions we use to build an MS-ProxIT; ii) when aggregating the importance of a ROI i across the forest, we average across only those trees in which the ROI is present.

In Table 6.13 we present the results of such analysis, performed on the same MS-ProxIF model of Table 6.12. The impurity decrease is measured in terms of Rényi divergence, since the adopted learning scheme is *O-1PRD*. We report the results for MS-ProxITs built with $D = \log_2(S)$ and $D = S - 1$ in Table 6.13 (a) and (b) respectively. We can observe that whereas in Table 6.13 (a) only two motor ROIs are detected, in Table 6.13 (b) this number increases to 4. In addition, in the latter table the first two ROIs in the ranking belong to the motor area of the brain, confirming the informativeness of using deeper trees. Further, most ROIs are present in both tables and many of them are also present in Tables 6.12 (a) and (b).

Table 6.13: ROI importance measured via the Rényi impurity function: (a) standard MS-ProxIF parametrization (b) standard MS-ProxIF parametrization aside for $D = S - 1$.

Rank	**Name**
#**1**	L.BSTS
#**2**	R.FG
#**3**	**L.LOFG**
#**4**	**R.PU**
#**5**	L.LOG
#**6**	R.BSTS
#**7**	L.POP
#**8**	L.EC
#**9**	L.FG
#**10**	R.IPG

Rank	**Name**
#**1**	**R.PU**
#**2**	**L.LOFG**
#**3**	L.BSTS
#**4**	**L.CER**
#**5**	R.BSTS
#**6**	L.POP
#**7**	L.CMFG
#**8**	**L.PrCG**
#**9**	L.EC
#**10**	R.HI

In conclusion, results seem to suggest that an *MS-ProxIF* model hinders relevant information related to the importance of a ROI. In both our analyses, even the first one which is rather approximate, motor brain regions appeared among the best ranked in terms of importance. Nevertheless, we need to perform further analyses, also in relation with clinical data, and design a more ad hoc approach to estimate the relevance of a ROI in order to make some more elaborate and interesting conclusions.

6.4 Conclusions and Future Work

In this chapter, we proposed a novel isolation-based RF approach to characterize multiple sclerosis connectivity networks. In detail, the model, called *MS-ProxIF*, works as a feature extractor to derive a novel network-based measure of quantitative structural connectomes for MS patients. In detail, MS is an autoimmune disease that can have a huge impact on life and which mechanisms are still under study. Indeed, a big part of MS research focuses on uncovering such mechanisms by extracting and comparing network-derived metrics. Nevertheless, the proposed representation, *RF-Isolation*, is rather unique since: i) it exploits information coming from all subjects for one subject's representation; ii) the model, *MS-ProxIF*, is trained on the MS population; and iii) it sees the problem as an outlier detection task. The first two points make the representation very descriptive from a disease-point of view: indeed, the representation depends on a multitude of subjects and not on a single one, like other network-derived measures present in the literature. The other novelty consists in using an outlier detection methodology in a unique way: the anomaly scores extracted from the forest represent the features of the novel representation of the connectivity network. In detail, each feature relates to a brain region and the anomaly score is interpreted as the disconnection degree of the region with respect to all the other ROIs.

We make a thorough experimental analysis to assess the suitability and robustness of *RF-Isolation* and to confirm the several hypotheses we made throughout the chapter about its informativeness. The analysis consists mostly in analyzing the classification results, which confirm its suitability. Since comparative results to other representations were promising, we also present some preliminary results on extracting MS-relevant ROIs.

The proposed methodology is highly user-friendly: in detail, even though, when building the model, several parameters need to be set, we provide clear guidelines, thus making the methodology easy to use also for those who are not Computer Science experts. Further, unless the dataset drastically changes, when a new subject arrives, the user has only to extract the RF-Isolation vector, without retraining the model. Lastly, both the RF-Isolation representation, as previously said, and the MS-ProxIF model from which the former is extracted, are highly interpretable. In detail, the MS-ProxIF model can provide the user with a list of possibly relevant ROIs, that can be directly analyzed by healthcare experts.

Indeed, finding the ROIs the most involved in the demyelination process of MS, should be at the core of future research, in order to uncover MS-related mechanisms. In detail, we should focus on refining the techniques to identify –but also validate– these regions within either the model or the *RF-Isolation* representation itself. Another idea to be investigated is to use *RF-Isolation* to classify different subtypes of MS, since it is still clinically challenging. A rather different idea would be to focus on other demyelinating diseases to see if the methodology can be applied in those contexts as well.

Lastly, as to the representation itself, it would be interesting to investigate whether the informativeness can be improved by employing and adapting the scoring functions proposed in Chapter 4 –we give some intuition of this idea in Chapter 7.

Chapter 7

Conclusions and Future Work

This thesis focused on Random Forest-based techniques for outlier detection. Outlier detection is the task of identifying those objects that are likely to have been generated by a different mechanism than the one underlying the remainder of the data. Finding outliers is fundamental for several reasons, which may vary depending on the task: in many cases a fast identification is required, for example to block network intruders from stealing users' data; or an accurate identification may be needed, for example to evaluate whether a mass in an image is a tumour. There exist many different types of outlier detection techniques, designed to face the numerous amount of application scenarios and the different aspects of an outlier detection problem. Among them, there is a category of approaches which works very well in many different situations: Random Forest-based techniques.

A Random Forest is an ensemble of Decision Trees originally designed for classification and regression tasks. These learners are characterized by a high degree of flexibility, robustness, high generalization capabilities, and they are computationally fast. Over the last few years, research on RF for learning tasks different from classification and regression has substantially increased: this is due to both the good theoretical characteristics of RFs and their success in many application fields. In particular, in the latest years, an increasing amount of RF-based methodologies for outlier detection has been proposed and, as mentioned before, they have been demonstrated to be a very suitable choice. In particular, there exists a category of techniques solving the task by isolating each object from the rest via the use of unsupervised Random Forests. Outliers are encoded as objects having a high isolation capability, i.e. they are likely to be separated after few splits of the tree. This characteristic can be easily retrieved during the testing phase and used to compute the anomaly score, i.e. the outlierness degree, of the object traversing the tree. The precursor of these techniques is iForest, introduced in 2008 and since then greatly exploited and extended.

This thesis inserted itself into the above context and worked on methodological aspects that either had yet to be faced or were only partially researched; in addition, it exploited the flexibility of Random Forests to design novel applications. In detail, we moved along the following four research directions:

- The first research direction investigated in this thesis led to a novel variant of RFs for outlier detection, *Proximity Isolation Forest*, that works with pairwise distances via an adaptation of the principle of isolation. Further, the measure used to compute the pairwise distances does not

have to satisfy any particular mathematical requirements. Therefore, Proximity Isolation Forests can easily manage non-vectorial objects: indeed, in literature many powerful distance measures have been proposed for many different types of non-vectorial data. Describing non-vectorial objects via distances is a suitable choice since there is no loss of information related to the structure and to the relations between the objects, differently from what happens when high-level features are extracted. In detail, we proposed several training schemes to build the trees in the most optimal way. We can divide them into four categories depending on the core idea:

- Random criteria: the test on each node is randomly defined, in a similar fashion to iForests.
- Scatter-based criteria: the intuition is that when we remove outliers from a set of data, the standard deviation tends to decrease. Since we cannot measure standard deviation within a distance matrix, we compute its *scatter*, i.e. the sparseness of the distance values. The aim is to partition the node such that in each child node the scatter is minimized.
- Separation-based criteria: the aim is to find the test in the node which generates the two well separated children. This can be done by using the Hausdorff distance, which measures how much two sets, i.e. nodes, resemble each other. This measure relies on the pairwise distances computed between the objects in one child node with the objects in its sibling.
- Information theoretic criteria: the aim is to obtain two child nodes that contain the maximum amount of different information. To measure the difference in terms of information between two sets of objects, we estimate the Rényi divergence. In detail, the estimation is based on the concept of K-Nearest Neighbor, i.e. only pairwise distances need to be known.

To assess the goodness and suitability of our proposal, we made a thorough evaluation on several benchmark datasets for non-vectorial data, also comparing to state-of-the-art techniques quite successfully. This contribution was made in collaboration with TU Delft, the Netherlands.

- The second research direction focused on defining novel scoring functions to compute the anomaly score of isolation-based methodologies. In detail, we studied this aspect from two different perspectives. The core assumption of the first perspective is that each node of a tree contains a lot of information that can be used to further specify the outlierness degree of an object. In detail, we can define a novel way to score the objects by giving to each traversed node a weight. We employed different information that have led to the definition of five different weighting

functions. The second perspective we adopted relies on ensemble theory and probability: we interpret the anomaly score of a tree from a probabilistic point of view and extend this notion at forest level. The latter leads to the definition of a novel anomaly score, less restrictive with respect to the original formulation. The used aggregation function is a common and successful combiner rule in the ensemble learning field. Both proposals were thoroughly evaluated on a pool of 16 benchmark datasets for outlier detection. We made several analyses: not only we compared both our contributions to the original scoring function, but we also studied and analyzed their combination. Further, we compared the best variant to more refined isolation-based techniques, obtaining highly competitive results.

- The third contribution uses RF in an alternative way. Random Forests are data partitioners, and they can be efficiently used to extract non-geometrical distance measures. Whereas this concept has been widely exploited for clustering, it has hardly been investigated for outlier detection. Our contribution started from the assumption that via using an unsupervised tree structure that retrieves the isolation capability of the objects, we can effectively measure the pairwise distances such that inliers and outliers can be easily discerned. In detail, our approach aimed at assessing the suitability for outlier detection of well known and/or recent RF-distance measures. We extracted these informative distances for each pair of objects and input the obtained distance matrix to a variety of distance-based outlier detectors. The suitability of this contribution was assessed via a thorough experimental evaluation on a large pool of outlier detection problems: we analyzed and compared the different distance measures and outlier detectors; but we also compared to the standard iForest and to the Euclidean distance. Some results have explicitly shown that we need to investigate further along this research direction to improve the proposed approach. This contribution was made in collaboration with TU Delft, the Netherlands.

- Whereas the above are all methodological contributions to the researched field, the last one is applicative. Briefly, we designed a variant of RF, *MS-ProxIF* for characterizing multiple sclerosis brain connectivity networks. A brain connectivity network encodes the connectivity strength between each pair of brain regions and in MS some of these regions are highly disconnected. The starting point of the proposed approach was to cast the problem as an outlier detection task since we can make a natural analogy between brain regions that are highly disconnected, which are the hallmark of MS, and outliers. Subsequently, we proposed *MS-ProxIF*, an extension of Proximity Isolation Forests adapted for the context: the approach manages connectivity networks by interpreting them as similarity matrices. In detail, the higher the connectivity strength between two regions, the higher their similarity. The built model is then used as a

feature extractor to derive the novel representation *RF-Isolation* of structural connectivity networks, which encodes the outlierness degree of each brain region. We evaluated the methodology on a dataset provided by the Mount Sinai Ichan School of Medicine of New York (US), in collaboration with another group within our department and with the University of Genova and La Sapienza University of Rome. The proposed approach showed promising results when compared to other network-derived measures. Further, we also made an initial and preliminary analysis on the identification of relevant ROIs.

Each methodological contribution of this thesis, whereas it focuses on a relevant and peculiar aspect of this research field, at the same it is inserted into a wider scenario to further prove its relevance. Further, this thesis supports and further shows that RF-based approaches for outlier detection are becoming very popular due to their success and suitability for the task, other than being characterized by an easiness of use and high interpretability. However, this thesis also shows that there is still large room for improvements: we can exploit the innate flexibility of RFs to design novel methods aimed at facing unsolved aspects of outlier detection, or at identifying outliers within a specific application field which has its own peculiar characteristics.

In the following, we illustrate some future research ideas related to the limitations and uninvestigated aspects of each research direction studied in this thesis.

- As to Proximity Isolation Forests, it would be interesting to test the methodology on a real-life dataset which original aim is outlier detection. An example is the detection of heart abnormalities via the analysis of ECG data: while time-series outlier detection is a very investigated field of research, it could benefit of the use of ProxIF. In detail, often using only one distance measure to evaluate the similarity between two time sequences may be limiting, e.g. the DTW distance is not able to detect whether two time sequences grow at the same rate as well as a temporal correlation coefficient does. In this scenario, we could use a ProxIF in the following way: we could train a part of the ProxITs with DTW pairwise distances computed between the time sequences, and the remaining trees using a temporal correlation coefficient. In this way we could use the same model, without any further adjustment, to extract different information and combine them together at forest level. Another idea to investigate concern the design of novel training schemes able to capture different aspects characterizing an outlier. An example could be the combination of a separation-based criterion with a training strategy based on the scatter. In detail, the former is aimed at finding the two child nodes that are the most separated, and thus it is based on evaluating the distance between objects belonging to different child nodes. Instead, the focus of scatter-based strategies is only on one of the two putative children: whereas the aim is to find the two children that lead

to the minimum dispersion of distance values, the dispersion is measured within each child independently of its sibling.

- Related to Chapter 4, in which we define novel functions to detect outliers, the most natural idea would be to further study whether novel weights or other aggregation rules could lead to even more informative scores. An example of the latter could be to aggregate the tree scores by computing the median, instead of using the sum rule. Indeed, the median is even more robust to a skewed distribution of the scores, i.e. trees in which the isolation capability of an object is either over or underestimated. Further, the median should be able to detect which is the tree structure encoding the most probable isolation degree of the testing object. Another compelling idea related to this research direction would be to employ the proposed scoring functions with more refined isolation-based tree structures, such as those we compared to in the state-of-the-art evaluation. Please note that this would probably also lead to novel path-weights to take into account tree-specific characteristics. For example, if we wanted to weigh the path using $V3$, the variant based on the proxy, and we were working with $K - ary$ trees with $K > 2$, we would have to redefine the proxy function to manage the existence of more than two child nodes. Another topic that requires investigation is to understand the reasons behind the poor performances of neighborhood-based variants. A starting point could be to verify whether, in the context of isolation-based outlier detection, assigning a lower weight to nodes with a big neighborhood is the most informative choice.

- The third research direction investigated in this thesis is maybe the one which needs the most research due to the inconclusive comparison with the Euclidean distance. Future ideas concern designing novel proximity measures that better capture the isolation of the objects and subsequently the existing differences between inliers and outliers. A starting point would be to explicitly include in the distance computation the difference between the depths of the leaves reached by the two objects: indeed in an iForest the outlierness degree is measured via the depth of the reached leaf and since we extract the distances from such model, it would maybe lead to more informative measures. Another compelling topic to investigate consists of the outlier detectors used at the end of the pipeline: maybe some of them rely too strongly on the geometrical nature of distances that they cannot completely benefit of the information embedded in data-dependent distances. Thus, we may need to develop novel detectors or adequately adapt the existing ones. This intuition holds especially if we work with mass-based data-dependent dissimilarities, which we recall define also a data-dependent self-similarity. In detail whereas for Nearest Neighbor-based techniques this feature is not a problem since according to mass-based measures, the closest neighbor to an object is still the object itself, this may become problematic

for clustering-based outlier detectors, such as K-Centers, which rely on different principles.

- As to the application proposed in Chapter 6, the extracted representation seems already very robust, however there is still a lot of work and research to do. In detail, the further step is to analyze both the model and the representation to understand whether multiple sclerosis-specific brain regions can be identified. After their identification, we can correlate them to some clinical data and effectively uncover unknown MS mechanisms. A different but related idea would be to assess the suitability of a similar approach for other demyelinating diseases, such as Amyotrophic Lateral Sclerosis or Parkinson's disease. Further, it would be interesting to assess whether using an enhanced anomaly score, such as those proposed in Chapter 4, can lead to an even more informative representation. For example, weighting the path could better highlight the difference in terms of disconnection degree between those brain regions which are slightly disconnected, maybe because the demyelination process has just started, with respect to those that are normally connected. The described situation is analogous to when a marginal outlier needs to be distinguished from some marginal inliers, an issue we highlighted and faced in Chapter 4.

In conclusion, this doctoral thesis further demonstrates that using RFs within an outlier detection pipeline is a suitable choice, leading to very successful techniques. In detail, not only we face unsolved methodological aspects concerning the detection of outliers using a Random Forest model, but we also investigate and study alternative uses of Random Forests. Indeed, we use RF for outlier detection in two innovative ways. The first consists of extracting pairwise distances from an iForest model, i.e. we exploit the innate nature of an RF model, which partitions data based on their distribution in the space. The obtained distance matrix is then input to an outlier detector. The second way consists of defining an RF-based approach for outlier detection to characterize multiple sclerosis brain connectivity networks. In detail, we can derive a representation encoding for each region its disconnection degree, which is a hallmark of multiple sclerosis, as its degree of outlierness. Finally, this thesis showed the importance and versatility of RF approaches for outlier detection by thoroughly investigating along very different open research directions and by highlighting the limitations and the uninvestigated aspects.

Bibliography

[1] M. C. Abba, H. Sun, K. A. Hawkins, J. A. Drake, Y. Hu, M. I. Nunez, S. Gaddis, T. Shi, S. Horvath, A. Sahin, and C. M. Aldaz. "Breast Cancer Molecular Signatures as Determined by SAGE: Correlation with Lymph Node Status". In: *Mol. Cancer Res.* 5.9 (2007), pp. 881–890.

[2] C. C. Aggarwal and S. Sathe. "Theoretical foundations and algorithms for outlier ensembles". In: *ACM SIGKDD Explor. Newsl.* 17.1 (2015), pp. 24–47.

[3] S. Ahmed, Y. Lee, S.-H. Hyun, and I. Koo. "Unsupervised machine learning-based detection of covert data integrity assault in smart grid networks utilizing isolation forest". In: *IEEE Trans. Inf. Forensics Security* 14.10 (2019), pp. 2765–2777.

[4] A. Aksay, A. Temizel, and A. E. Cetin. "Camera tamper detection using wavelet analysis for video surveillance". In: *2007 IEEE Int. Conf. Adv. Video Signal Based Surveill.* IEEE. 2007, pp. 558–562.

[5] Y. Amit and D. Geman. "Shape quantization and recognition with randomized trees". In: *Neural Comput.* 9.7 (1997), pp. 1545–1588.

[6] B. Araújo, F. A. Rodrigues, T. C. Silva, and L. Zhao. "Identifying Abnormal Nodes in Protein-Protein Interaction Networks". In: *2010 11th Braz. Symp. Neural Netw.* 2010, pp. 97–102.

[7] S. Aryal, K. M. Ting, T. Washio, and G. Haffari. "A comparative study of data-dependent approaches without learning in measuring similarities of data objects". In: *Data Min. Knowl. Disc.* 34.1 (2020), pp. 124–162.

[8] S. Aryal, K. M. Ting, T. Washio, and G. Haffari. "Data-dependent dissimilarity measure: an effective alternative to geometric distance measures". In: *Knowl. Inf. Syst.* 53.2 (2017), pp. 479–506.

[9] S. Balakrishnan and D. Madigan. "Decision trees for functional variables". In: *6th Int. Conf. Data Min.* 2006, pp. 798–802.

[10] R. J. Beckman and R. D. Cook. "Outlier. . . . . s". In: *Technometrics* 25.2 (1983), pp. 119–149.

[11] C. L. Beirão and M. A. Figueiredo. "Defect detection in textile images using Gabor filters". In: *Proc. Int. Conf. Image Anal. Recognit.* Springer. 2004, pp. 841–848.

[12] M. Bester, M. Petracca, and M. Inglese. "Neuroimaging of multiple sclerosis, acute disseminated encephalomyelitis, and other demyelinating diseases." In: *Semin. Roentgenol.* Vol. 49. 1. 2013, pp. 76–85.

[13] M. Bicego. "Dissimilarity Random Forest Clustering". In: *2020 IEEE Int. Conf. Data Mining (ICDM)*. IEEE. 2020, pp. 936–941.

[14] M. Bicego. "K-Random Forests: a K-means style algorithm for Random Forest clustering". In: *Proc. Int. Jt. Conf. Neural Netw.* 2019.

[15] M. Bicego, F. Cicalese, and A. Mensi. "RatioRF: a novel measure for Random Forest clustering based on the Tversky's Ratio model". In: *IEEE Trans. Knowl. Data Eng.* (2021).

[16] M. Bicego and F. Escolano. "On learning Random Forests for Random Forest-clustering". In: *2020 25th Int. Conf. Pattern Recognit. (ICPR)*. 2021, pp. 3451–3458.

[17] L. Breiman, J. Friedman, C. Stone, and R. Olshen. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984.

[18] L. Breiman. "Bagging predictors". In: *Mach. Learn.* 24.2 (1996), pp. 123–140.

[19] L. Breiman. "Random Forests". In: *Mach. Learn.* 45.1 (2001), pp. 5–32.

[20] L. Breiman. *Using adaptive bagging to debias regressions*. Tech. rep. Technical Report 547, Statistics Dept. UCB, 1999.

[21] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. "LOF: identifying density-based local outliers". In: *Proc. 2000 ACM SIGMOD Int. Conf. Manag. Data*. 2000, pp. 93–104.

[22] J. Van den Broeck, S. Argeseanu Cunningham, R. Eeckels, and K. Herbst. "Data cleaning: detecting, diagnosing, and editing data abnormalities". In: *PLoS medicine* 2.10 (2005), e267.

[23] C. R. Buchanan, M. E. Bastin, S. J. Ritchie, D. C. Liewald, J. W. Madole, E. M. Tucker-Drob, I. J. Deary, and S. R. Cox. "The effect of network thresholding and weighting on structural brain networks in the UK Biobank". In: *NeuroImage* 211 (2020), p. 116443.

[24] S. Buschjäger, P.-J. Honysz, and K. Morik. "Randomized outlier detection with trees". In: *Int. J. Data Sci. Anal.* (2020), pp. 1–14.

[25] G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle. "On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study". In: *Data Min. Knowl. Disc.* 30.4 (2016), pp. 891–927.

[26] L. Cazzanti and M. R. Gupta. "Information-theoretic and set-theoretic similarity". In: *2006 IEEE Int. Symp. Inf. Theory - Proc.* IEEE. 2006, pp. 1836–1840.

[27] S.-H. Cha. "Comprehensive survey on distance/similarity measures between probability density functions". In: *City* 1.2 (2007), p. 1.

[28] V. Chandola, A. Banerjee, and V. Kumar. "Anomaly detection: A survey". In: *ACM Comput. Surv.* 41.3 (2009), pp. 1–58.

[29] T. Charalambous, C. Tur, F. Prados, B. Kanber, D. T. Chard, S. Ourselin, J. D. Clayden, C. A. G. Wheeler-Kingshott, A. J. Thompson, and A. T. Toosy. "Structural network disruption markers explain disability in multiple sclerosis". In: *J. Neurol. Neurosurg. Psychiatry* 90.2 (2019), pp. 219–226.

[30] D. T. Chard, A. A. Alahmadi, B. Audoin, T. Charalambous, C. Enzinger, H. E. Hulst, M. A. Rocca, À. Rovira, J. Sastre-Garriga, M. Schoonheim, and the MAGNIMS Study Group. "Mind the gap: From neurons to networks to outcomes in multiple sclerosis". In: *Nat. Rev. Neurol.* 17.3 (2021), pp. 173–184.

[31] D. Chicco and G. Jurman. "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation". In: *BMC Genom.* 21.1 (2020), pp. 1–13.

[32] T. Cover and P. Hart. "Nearest neighbor pattern classification". In: *IEEE Trans. Inf. Theory* 13.1 (1967), pp. 21–27.

[33] T. M. Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[34] A. Daducci, A. Dal Palù, A. Lemkaddem, and J.-P. Thiran. "COMMIT: Convex optimization modeling for microstructure informed tractography". In: *IEEE Trans. Med. Imag.* 34.1 (2014), pp. 246–257.

[35] J. Demšar. "Statistical comparisons of classifiers over multiple data sets". In: *J. Mach. Learn. Res.* 7 (2006), pp. 1–30.

[36] R. S. Desikan, F. Ségonne, B. Fischl, B. T. Quinn, B. C. Dickerson, D. Blacker, R. L. Buckner, A. M. Dale, R. P. Maguire, B. T. Hyman, M. S. Alberti, and R. J. Killiany. "An automated labeling system for subdividing the human cerebral cortex on MRI scans into gyral based regions of interest". In: *Neuroimage* 31.3 (2006), pp. 968–980.

[37] C. Désir, S. Bernard, C. Petitjean, and L. Heutte. "One class random forests". In: *Pattern Recognit.* 46 (2013), pp. 3490–3506.

[38] T. G. Dietterich. "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization". In: *Mach. Learn.* 32 (1998), pp. 1–22.

[39] Y. Djenouri and A. Zimek. "Outlier detection in urban traffic data". In: *Proc. 8th Int. Conf. Web Intell. Min. Semant.* 2018, pp. 1–12.

[40] R. Dobson and G. Giovannoni. "Multiple sclerosis–a review". In: *Eur. J. Neurol.* 26.1 (2019), pp. 27–40.

[41] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui. "A comparative evaluation of outlier detection algorithms: Experiments and analyses". In: *Pattern Recognit.* 74 (2018), pp. 406 –421.

[42] A. Douzal-Chouakria and C. Amblard. "Classification trees for time series". In: *Pattern Recognit.* 45.3 (2012), pp. 1076–1091.

[43] D. Dua and C. Graff. *UCI Machine Learning Repository*. 2017. URL: `http://archive.ics.uci.edu/ml`.

[44] A. Duraj and L. Chomatek. "Supporting Breast Cancer Diagnosis with Multi-objective Genetic Algorithm for Outlier Detection". In: *Advanced Solutions in Diagnostics and Fault Tolerant Control*. Ed. by J. M. Kościelny, M. Syfert, and A. Sztyber. Cham: Springer Int. Publishing, 2018, pp. 304–315.

[45] J. K. Dutta, B. Banerjee, and C. K. Reddy. "RODS: Rarity based outlier detection in a sparse coding framework". In: *IEEE Trans. Knowl. Data Eng.* 28.2 (2015), pp. 483–495.

[46] M. El Azami, A. Hammers, J. Jung, N. Costes, R. Bouet, and C. Lartizien. "Detection of lesions underlying intractable epilepsy on T1-weighted MRI as an outlier detection problem". In: *PloS one* 11.9 (2016), e0161498.

[47] A. F. Emmott, S. Das, T. Dietterich, A. Fern, and W.-K. Wong. "Systematic Construction of Anomaly Detection Benchmarks from Real Data". In: *Proc. ACM SIGKDD Workshop Outl. Detect. Desc.* 2013, pp. 16–21.

[48] C. Englund and A. Verikas. "A novel approach to estimate proximity in a random forest: An exploratory study". In: *Expert Syst. Appl.* 39.17 (2012), pp. 13046–13050.

[49] F. Escolano Ruiz, P. S. Pérez, and B. I. Bonev. *Information theory in computer vision and pattern recognition*. Springer Science & Business Media, 2009.

[50] A. Fornito, A. Zalesky, and E. Bullmore. *Fundamentals of brain network analysis*. Academic Press, 2016.

[51] B. J. Frey and D. Dueck. "Clustering by passing messages between data points". In: *Science* 315.5814 (2007), pp. 972–976.

[52] M. Friedman. "The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance". In: *J. Am. Stat. Assoc.* 32.200 (1937), pp. 675–701.

[53] P. Geurts, D. Ernst, and L. Wehenkel. "Extremely randomized trees". In: *Mach. Learn.* 63.1 (2006), pp. 3–42.

[54] G. Giacinto, R. Perdisci, M. Del Rio, and F. Roli. "Intrusion detection in computer networks by a modular ensemble of one-class classifiers". In: *Inf. Fusion* 9.1 (2008). Special Issue on Applications of Ensemble Methods, pp. 69–82.

[55] G. Giacinto, F. Roli, and L. Didaci. "Fusion of multiple classifiers for intrusion detection in computer networks". In: *Pattern Recognit. Lett.* 24.12 (2003), pp. 1795–1803.

[56] C. Gini. "Variabilità e mutabilità". In: *Vamu* (1912).

[57] N. Goix, N. Drougard, R. Brault, and M. Chiapino. "One Class Splitting Criteria for Random Forests". In: *Proc. 9th Asian Conf. Mach. Lear.* Ed. by M.-L. Zhang and Y.-K. Noh. Vol. 77. Proc. Mach. Lear. Res. 2017, pp. 343–358.

[58] L. Goldfarb. "A unified approach to pattern recognition". In: *Pattern Recognit.* 17.5 (1984), pp. 575–582.

[59] M. Goldstein and S. Uchida. "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data". In: *PloS one* 11.4 (2016).

[60] N. Görnitz, M. Kloft, K. Rieck, and U. Brefeld. "Toward supervised anomaly detection". In: *J. Artif. Intell. Res.* 46 (2013), pp. 235–262.

[61] K. R. Gray, P. Aljabar, R. A. Heckemann, A. Hammers, and D. Rueckert. "Random forest-based similarity measures for multi-modal classification of Alzheimer's disease". In: *NeuroImage* 65 (2013), pp. 167 – 175.

[62] C. M. Grinstead and J. L. Snell. *Introduction to probability.* American Mathematical Soc., 2012.

[63] S. Guha, N. Mishra, G. Roy, and O. Schrijvers. "Robust Random Cut Forest Based Anomaly Detection on Streams". In: *Proc. 33rd Int. Conf. Mach. Lear.* Vol. 48. 2016, 2712–2721.

[64] S. Haghiri, D. Garreau, and U. von Luxburg. "Comparison-based random forests". In: *Proc. 35th Int. Conf. Mach. Learn.* (2018).

[65] S. Hariri, M. C. Kind, and R. J. Brunner. "Extended Isolation Forest". In: *IEEE Trans. Knowl. Data Eng.* 33.4 (2021), pp. 1479–1489.

[66] A. Harol, E. Pękalska, S. Verzakov, and R. P. Duin. "Augmented embedding of dissimilarity data into (pseudo-) Euclidean spaces". In: *Joint IAPR Int. Workshops Stat. Tech. Pattern Recognit. (SPR) Struct. Syntactic Pattern Recognit. (SPR) (S+SSPR 2006)*. Springer. 2006, pp. 613–621.

[67] P. E. Hart, D. G. Stork, and R. O. Duda. *Pattern classification.* Wiley Hoboken, 2000.

[68] D. M. Hawkins. *Identification of outliers.* Vol. 11. Springer, 1980.

[69] K. M. van Hespen, J. J. Zwanenburg, J. W. Dankbaar, M. I. Geerlings, J. Hendrikse, and H. J. Kuijf. "An anomaly detection approach to identify chronic brain infarcts on MRI". In: *Sci. Rep.* 11.1 (2021), pp. 1–10.

[70] T. K. Ho. "The random subspace method for constructing decision forests". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 20.8 (1998), pp. 832–844.

[71] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. "Comparing images using the Hausdorff distance". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 15.9 (1993), pp. 850–863.

[72] P. Indyk and R. Motwani. "Approximate nearest neighbors: towards removing the curse of dimensionality". In: *Proc. 30th Ann. ACM Symp. Theory Comput.* 1998, pp. 604–613.

[73] H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer. "Random survival forests". In: *Ann. Appl. Stat.* 2.3 (2008), pp. 841–860.

[74] A. K. Jain, R. P. W. Duin, and J. Mao. "Statistical pattern recognition: A review". In: *IEEE Conf. Pattern Anal. Mach. Intell.* 22.1 (2000), pp. 4–37.

[75] N. Javed and T. Wolf. "Automated sensor verification using outlier detection in the internet of things". In: *2012 32nd Int. Conf. Distrib. Comput. Syst. Workshops*. IEEE. 2012, pp. 291–296.

[76] P. Jehl, F. Sievers, and D. G. Higgins. "OD-seq: outlier detection in multiple sequence alignments". In: *BMC Bioinform.* 16.1 (2015), pp. 1–11.

[77] A. Joly, P. Geurts, and L. Wehenkel. "Random forests with random projections of the output space for high dimensional multi-label classification". In: *Jt. Eur. Conf. Mach. Learn. Knowl. Disc. Databases*. Springer. 2014, pp. 607–622.

[78] P. Kanhere and H. Khanuja. "A survey on outlier detection in financial transactions". In: *Int. J. Comput. Appl.* 108.17 (2014), pp. 23–25.

[79] R. Kannan, H. Woo, C. C. Aggarwal, and H. Park. "Outlier detection for text data". In: *Proc. 2017 SIAM Int. Conf. Data Min.* SIAM. 2017, pp. 489–497.

[80] P. Karczmarek, A. Kiersztyn, and W. Pedrycz. "Fuzzy set-based isolation forest". In: *2020 IEEE Int. Conf. Fuzzy Syst.* IEEE. 2020, pp. 1–6.

[81] P. Karczmarek, A. Kiersztyn, W. Pedrycz, and E. Al. "K-Means-based isolation forest". In: *Knowl. Based Syst.* 195 (2020), p. 105659.

[82] 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 Int. Conf. Fuzzy Syst.* IEEE. 2021, pp. 1–8.

[83] P. Karczmarek, A. Kiersztyn, W. Pedrycz, and D. Czerwiński. "Fuzzy C-Means-based Isolation Forest". In: *Appl. Soft Comput.* 106 (2021), p. 107354.

[84] I. Karlsson, P. Papapetrou, and H. Boström. "Generalized random shapelet forests". In: *Data Min. Knowl. Disc.* 30.5 (2016), pp. 1053–1085.

[85] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. Vol. 344. John Wiley & Sons, 2009.

[86] F. Keller, E. Muller, and K. Bohm. "HiCS: High contrast subspaces for density-based outlier ranking". In: *IEEE Int. Conf. Data Eng.* IEEE. 2012, pp. 1037–1048.

[87] J. Kim, H. Naganathan, S.-Y. Moon, W. K. Chong, and S. T. Ariaratnam. "Applications of clustering and isolation forest techniques in real-time building energy-consumption data: Application to LEED certified buildings". In: *J. Energ. Eng.* 143.5 (2017), p. 04017052.

[88] J. Kittler. "Combining Classifiers: A Theoretical Framework". In: *Pattern Anal. Appl.* 1.1 (1998), pp. 18–27.

[89] D. E. KNUTH. "The Art of Computer Programming". In: *Sorting and Searching* 3 (1973), Ch. 6.

[90] Kurnianingsih, L. E. Nugroho, Widyawan, L. Lazuardi, and A. S. Prabuwono. "Detection of Anomalous Vital Sign of Elderly Using Hybrid K-Means Clustering and Isolation Forest". In: *TENCON 2018-2018 IEEE Region 10 Conf.* IEEE. 2018, pp. 0913–0918.

[91] J. F. Kurtzke. "Rating neurologic impairment in multiple sclerosis: an expanded disability status scale (EDSS)". In: *Neurol.* 33.11 (1983), pp. 1444–1444.

[92] A. Lazarevic and V. Kumar. "Feature bagging for outlier detection". In: *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.* 2005, pp. 157–166.

[93] C. Leistner, A. Saffari, and H. Bischof. "MIForests: Multiple-instance learning with randomized trees". In: *Eur. Conf. Comput. Vis.* Springer. 2010, pp. 29–42.

[94] D. Lin. "An Information-theoretic Definition of Similarity". In: *Proc. Int. Conf. Mach. Learn.* Vol. 98. 1998, pp. 296–304.

[95] Z. Lin, X. Liu, and M. Collu. "Wind power prediction based on high-frequency SCADA data along with isolation forest and deep learning neural networks". In: *Int. J. Electr. Power Energ. Syst.* 118 (2020), p. 105835.

[96] F. T. Liu, K. M. Ting, and Z. H. Zhou. "Isolation Forest". In: *2008 IEEE Int. Conf. Data Min.* 2008, pp. 413–422.

[97] F. T. Liu, K. M. Ting, and Z.-H. Zhou. "Isolation-Based Anomaly Detection". In: *ACM Trans. Knowl. Discov. Data* 6.1 (2012), 3:1–3:39.

[98] F. T. Liu, K. M. Ting, and Z.-H. Zhou. "On detecting clustered anomalies using SCiForest". In: *Mach. Learn. Knowl. Disc. Databases (ECML PKDD)*. 2010, pp. 274–290.

[99] X. Liu, C.-T. Lu, and F. Chen. "Spatial outlier detection: Random walk based approaches". In: *Proc. 18th SIGSPATIAL Int. Conf. Adv. Geogr. Inf. Syst.* 2010, pp. 370–379.

[100] A. Lourenço, H. Silva, C. Carreiras, and A. Fred. "Outlier detection in non-intrusive ECG biometric system". In: *Int. Conf. Image Anal. Recognit.* 2013, pp. 43–52.

[101] B. Lucas, A. Shifaz, C. Pelletier, L. O'Neill, N. Zaidi, B. Goethals, F. Petitjean, and G. I. Webb. "Proximity forest: an effective and scalable distance-based classifier for time series". In: *Data Min. Knowl. Disc.* 33.3 (2019), pp. 607–635.

[102] Y. Ma, Q. Zhang, J. Ding, Q. Wang, and J. Ma. "Short term load forecasting based on iForest-LSTM". In: *2019 14th IEEE Conf. Ind. Electron. Appl. (ICIEA)*. IEEE. 2019, pp. 2278–2282.

[103] G. Mangeat, A. Badji, R. Ouellette, C. A. Treaba, E. Herranz, T. Granberg, C. Louapre, N. Stikov, J. A. Sloane, P. Bellec, C. Mainero, and J. Cohen-Adad. "Changes in structural network are associated with cortical demyelination in early multiple sclerosis". In: *Hum. Brain Mapp.* 39.5 (2018), pp. 2133–2146.

[104] J.-F. Mangin, P. Fillard, Y. Cointepas, D. Le Bihan, V. Frouin, and C. Poupon. "Toward global tractography". In: *Neuroimage* 80 (2013), pp. 290–296.

[105] T. Maszczyk and D. Wlodzislaw. "Comparison of Shannon, Renyi and Tsallis Entropy Used in Decision Trees". In: vol. 5097. June 2008, pp. 643–651.

[106] G. J. McLachlan and K. E. Basford. *Mixture models: Inference and applications to clustering*. Vol. 38. M. Dekker New York, 1988.

[107] A. Mensi and M. Bicego. "A novel anomaly score for isolation forests". In: *Int. Conf. Image Analysis and Processing (ICIAP 2019)*. Springer. 2019, pp. 152–163.

[108] A. Mensi and M. Bicego. "Enhanced Anomaly Scores for Isolation Forests". In: *Pattern Recognit.* (2021), p. 108115.

[109] A. Mensi, M. Bicego, and D. M. Tax. "Proximity Isolation Forests". In: *2020 25th Int. Conf. Pattern Recognit.* IEEE. 2021, pp. 8021–8028.

[110] A. Mensi, F. Cicalese, and M. Bicego. "Using Random Forest Distances for Outlier Detection". In: *Int. Conf. Image Analysis and Processing (ICIAP 2022)*. Springer International Publishing, 2022, pp. 75–86.

[111] A. Mensi, A. Franzoni, D. M. Tax, and M. Bicego. "An alternative exploitation of isolation forests for outlier detection". In: *Joint IAPR Int. Workshops Stat. Tech. Pattern Recognit. (SPR) Struct. Syntactic Pattern Recognit. (SPR) (S+SSPR 2021)*. Springer. 2021, pp. 34–44.

[112] A. Mensi, S. Schiavi, M. Petracca, N. Graziano, A. Daducci, M. Inglese, and M. Bicego. "RF-Isolation: a Novel Representation of Structural Connectivity Networks for Multiple Sclerosis Classification". In: *CIBB 2021* (2021).

[113] B. Micenková, B. McWilliams, and I. Assent. "Learning outlier ensembles: The best of both worlds–supervised and unsupervised". In: *Proc. SIGKDD Workshop Outl. Detect. and Desc.* 2014, pp. 51–54.

[114] J. Mingers. "An empirical comparison of pruning methods for decision tree induction". In: *Mach. Learn.* 4.2 (1989), pp. 227–243.

[115] F. Moosmann, B. Triggs, and F. Jurie. "Fast discriminative visual codebooks using randomized clustering forests". In: *Advances in neural information processing systems*. 2007, pp. 985–992.

[116] M. Müller. "Dynamic time warping". In: *Inf. Ret. Music Motion* (2007), pp. 69–84.

[117] J. Ning, L. Chen, and J. Chen. "Relative density-based outlier detection algorithm". In: *Proc. 2018 2nd Int. Conf. Comput. Sci. Artif. Intell.* 2018, pp. 227–231.

[118] M. Noshad, K. R. Moon, S. Y. Sekeh, and A. O. Hero. "Direct estimation of information divergence using nearest neighbor ratios". In: *2017 IEEE Int. Symp. Inf. Theory (ISIT)*. IEEE. 2017, pp. 903–907.

[119] E. Pagani, M. A. Rocca, E. De Meo, M. A. Horsfield, B. Colombo, M. Rodegher, G. Comi, and M. Filippi. "Structural connectivity in multiple sclerosis and modeling of disconnection". In: *Multiple Sclerosis J.* 26.2 (2020), pp. 220–232.

[120] D. Pál, B. Póczos, and C. Szepesvári. "Estimation of Rényi entropy and mutual information based on generalized nearest-neighbor graphs". In: *Adv. Neural Inf. Process. Syst.* 23 (2010).

[121] E. Pękalska. "The Dissimilarity representations in pattern recognition. Concepts, theory and applications." PhD thesis. The University of Manchester, 2005.

[122] V. Pérez-Rosas, B. Kleinberg, A. Lefevre, and R. Mihalcea. "Automatic detection of fake news". In: *arXiv preprint arXiv:1708.07104* (2017).

[123] M Petracca, S Schiavi, M Battocchio, M. El Mendili, L Fleysher, A Daducci, and M Inglese. "Streamline density and lesion volume reveal a postero–anterior gradient of corpus callosum damage in multiple sclerosis". In: *Eur. J. Neurol.* 27.6 (2020), pp. 1076–1082.

[124] J. Pizarro, E. Guerrero, and P. L. Galindo. "Multiple comparison procedures applied to model selection". In: *Neurocomputing* 48.1-4 (2002), pp. 155–173.

[125] P. Pławiak. *ECG signals (1000 fragments)*. 2017.

[126] C. H. Polman, S. C. Reingold, B. Banwell, M. Clanet, J. A. Cohen, M. Filippi, K. Fujihara, E. Havrdova, M. Hutchinson, L. Kappos, F. D. Lublin, X. Montalban, P. O'Connor, M. Sandberg-Wollheim, A. J. Thompson, E. Waubant, B. Weinshenker, and J. S. Wolinsky. "Diagnostic criteria for multiple sclerosis: 2010 revisions to the McDonald criteria". In: *Ann. Neurol.* 69.2 (2011), pp. 292–302.

[127] B. R. Preiss. *Data structures and algorithms.* John Wiley & Sons, Inc., 1999.

[128] P. Probst and A.-L. Boulesteix. "To tune or not to tune the number of trees in random forest." In: *J. Mach. Learn. Res.* 18.1 (2017), pp. 6673–6690.

[129] S. B. E. Raj and A. A. Portia. "Analysis on credit card fraud detection methods". In: *2011 Proc. Int. Conf. Comput. Commun. Electr. Technol. (ICCCET)*. IEEE. 2011, pp. 152–156.

[130] S. I. Rennard, N. Locantore, B. Delafont, R. Tal-Singer, E. K. Silverman, J. Vestbo, B. E. Miller, P. Bakke, B. Celli, P. M. Calverley, A Coxson, C Crim, L. A. Edwards, D. A. Lomas, W MacNee, E. F. M. Wouters, J. C. Yates, I Coca, and A Agusti. "Identification of five chronic obstructive pulmonary disease subgroups with different prognoses in the ECLIPSE cohort using cluster analysis". In: *Ann. Am. Thorac. Soc.* 12.3 (2015), pp. 303–312.

[131] A. Rényi. "On measures of entropy and information". In: *Proc. 4th Berkeley Symp. on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. University of California Press. 1961, pp. 547–561.

[132] E. Rodrigues. "Combining Minkowski and Chebyshev: New distance proposal and survey of distance metrics using k-nearest neighbours classifier". In: *Pattern Recognit. Lett.* 110 (2018), pp. 66–71.

[133] M. Rubinov and O. Sporns. "Complex network measures of brain connectivity: uses and interpretations". In: *Neuroimage* 52.3 (2010), pp. 1059–1069.

[134] O. Sagi and L. Rokach. "Ensemble learning: A survey". In: *Wiley Interdiscip. Rev. Data Min. Knowl. Disc.* 8.4 (2018).

[135] M. Sakurada and T. Yairi. "Anomaly detection using autoencoders with nonlinear dimensionality reduction". In: *Proc. MLSDA 2014 2nd Workshop Mach. Learn. Sens. Data Anal.* 2014, pp. 4–11.

[136] S. Sathe and C. C. Aggarwal. "Similarity forests". In: *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.* 2017, pp. 395–403.

[137] S. Schiavi, A. Azzari, A. Mensi, N. Graziano, A. Daducci, M. Bicego, M. Inglese, and M. Petracca. "Classification of multiple sclerosis patients based on structural disconnection: A robust feature selection approach". In: *J. Neuroimaging* (2022).

[138] S. Schiavi, M. Petracca, M. Battocchio, M. M. El Mendili, S. Paduri, L. Fleysher, M. Inglese, and A. Daducci. "Sensory-motor network topology in multiple sclerosis: Structural connectivity analysis accounting for intrinsic density discrepancy". In: *Hum. Brain Mapp.* 41.11 (2020), pp. 2951–2963.

[139] C. E. Shannon. "A mathematical theory of communication". In: *Bell Syst. Tech. J.* 27.3 (1948), pp. 379–423.

[140] S. Shekhar, C.-T. Lu, and P. Zhang. "Detecting graph-based spatial outliers: algorithms and applications (a summary of results)". In: *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.* 2001, pp. 371–376.

[141] T. Shi, D. Seligson, A. Belldegrun, A. Palotie, and S. Horvath. "Tumor Classification by Tissue Microarray Profiling: Random Forest Clustering Applied to Renal Cell Carcinoma". In: *Modern Pathol.* 18 (2005), pp. 547–557.

[142] T. Shi and S. Horvath. "Unsupervised Learning With Random Forest Predictors". In: *J. Comput. Graph. Stat.* 15 (2005).

[143] J. Shotton, M. Johnson, and R. Cipolla. "Semantic texton forests for image categorization and segmentation". In: *2008 IEEE Conf. Comput. Vis. Pattern Recognit.* 2008, pp. 1–8.

[144] J. Sternby, E. Thormarker, and M. Liljenstam. "Anomaly Detection Forest". In: *24th Eur. Conf. Artif. Intell.* 2020.

[145] G. A. Susto, A. Beghi, and S. McLoone. "Anomaly detection through on-line isolation Forest: An application to plasma etching". In: *2017 28th Annu. SEMI Adv. Semicond. Manuf. Conf. (ASMC)*. 2017, pp. 89–94.

[146] D. Tax. "One-class classification; concept-learning in the absence of counter-examples". PhD thesis. Delft University of Technology, 2001.

[147] S. Theodoridis and K. Koutroumbas. *Pattern Recognition (Fourth Edition)*. Ed. by S. Theodoridis and K. Koutroumbas. Fourth Edition. Academic Press, 2009.

[148] R. Tibshirani, G. Walther, and T. Hastie. "Estimating the number of clusters in a data set via the gap statistic". In: *J. R. Stat. Soc.: Series B (Statistical Methodology)* 63.2 (2001), pp. 411–423.

[149] K. Ting, Y. Zhu, M. Carman, Y. Zhu, and Z.-H. Zhou. "Overcoming Key Weaknesses of Distance-based Neighbourhood Methods Using a Data Dependent Dissimilarity Measure". In: *Proc. Int. Conf. Knowl. Disc. Data Min.* 2016, pp. 1205–1214.

[150] G. D. Tourassi, B. Harrawood, S. Singh, J. Y. Lo, and C. E. Floyd. "Evaluation of information-theoretic similarity measures for content-based retrieval and detection of masses in mammograms". In: *Med. Phys.* 34.1 (2007), pp. 140–150.

[151] A. Tversky. "Features of similarity." In: *Psychol. Rev.* 84.4 (1977), p. 327.

[152] P. Valsasina, M. A. Rocca, M. Absinta, M. P. Sormani, L. Mancini, N. De Stefano, A. Rovira, A. Gass, C. Enzinger, and F. Barkhof. "A multicentre study of motor functional connectivity changes in patients with multiple sclerosis". In: *Eur. J. Neurosci.* 33.7 (2011), pp. 1256–1263.

[153] U. Von Luxburg. "A tutorial on spectral clustering". In: *Stat. Comput.* 17.4 (2007), pp. 395–416.

[154] H. Wang, M. J. Bah, and M. Hammad. "Progress in outlier detection techniques: A survey". In: *IEEE Access* 7 (2019), pp. 107964–108000.

[155] R. Wang, F. Nie, Z. Wang, F. He, and X. Li. "Multiple features and isolation forest-based fast anomaly detector for hyperspectral imagery". In: *IEEE Trans. Geosci. Remote Sens.* 58.9 (2020), pp. 6664–6676.

[156] S. Watanabe. *Pattern recognition: human and mechanical.* John Wiley & Sons, Inc., 1985.

[157] F. Wilcoxon. "Individual comparisons by ranking methods". In: *Biometrics Bull.* 1.6 (1945), pp. 80–83.

[158] T.-T. Wong. "Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation". In: *Pattern Recognit.* 48.9 (2015), pp. 2839–2846.

[159] J Wu, J Wang, and W Shen. "Identification of MAGEA12 as a prognostic outlier gene in gastric cancers." In: *Neoplasma* 64.2 (2017), pp. 238–243.

[160] W. Wu and Y. Chen. "Application of isolation forest to extract multivariate anomalies from geochemical exploration data". In: *Glob. Geol.* 21.1 (2018), pp. 36–47.

[161] Y. Yamada, E. Suzuki, H. Yokoi, and K. Takabayashi. "Decision-tree induction from time-series data based on a standard-example split test". In: *Proc. 20th Int. Conf. Mach. Learn.* 2003, pp. 840–847.

[162] L. Ye and E. Keogh. "Time series shapelets: a new primitive for data mining". In: *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.* 2009, pp. 947–956.

[163] K.-A. Yoon, O.-S. Kwon, and D.-H. Bae. "An Approach to Outlier Detection of Software Measurement Data using the K-means Clustering Method". In: *First Int. Symposium on Empirical Software Eng. and Measurement (ESEM 2007).* 2007, pp. 443–445.

[164] J. Zhang and M. Zulkernine. "Anomaly Based Network Intrusion Detection with Unsupervised Outlier Detection". In: *2006 IEEE Int. Conf. Commun.* Vol. 5. 2006, pp. 2388–2393.

[165] X. Zhang, W. Dou, Q. He, R. Zhou, C. Leckie, R. Kotagiri, and Z. Salcic. "LSHiForest: A generic framework for fast tree isolation based ensemble anomaly analysis". In: *2017 IEEE 33rd Int. Conf. Data Eng. (ICDE)*. IEEE. 2017, pp. 983–994.

[166] L. Zhu, A. Sun, and B. Choi. "Online spam-blog detection through blog search". In: *Proc. 17th ACM Conf. Inf. Knowl. Manag.* 2008, pp. 1347–1348.

[167] X. Zhu, C. Loy, and S. Gong. "Constructing Robust Affinity Graphs for Spectral Clustering". In: *Proc. Int. Conf. Comput. Vis. Pattern Recognit.* 2014, pp. 1450–1457.

[168] A. Zimek and P. Filzmoser. "There and back again: Outlier detection between statistical reasoning and data mining algorithms". In: *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* 8.6 (2018), e1280.

[169] M. Zurita, C. Montalba, T. Labbé, J. P. Cruz, J. D. da Rocha, C. Tejos, E. Ciampi, C. Cárcamo, R. Sitaram, and S. Uribe. "Characterization of relapsing-remitting multiple sclerosis patients using support vector machine classifications of functional and diffusion MRI data". In: *NeuroImage: Clin.* 20 (2018), pp. 724–730.

Appendix A

Complete Results Chapter 3

This additional section includes Tables A.1, A.2 and A.3, which respectively refer to Figures 3.2, 3.4 and Table 3.2 in Section 3.4.2 of Chapter 3. In particular, for each dataset, fixed a specific parameter, such as the sample size S in Table A.1, the parametrization with the maximum averaged AUC across the different iterations is chosen.

The first observation to make on all tables is that for most parametrizations the best training criterion is **O-2PS**$_D$, and if it is not the best first choice on average, is the second one. This training variant is closely followed by *O-2PH* which is the best average choice in the remaining cases and in general very performing. Also *R-2P* and *O-2PS*$_P$ perform rather well. In addition the worst performing variants on average employ always one prototype due to the bad performance on some datasets; this is true except for *O-1PH* which works rather well and comparably to *O-2PR*, a *2P* training scheme.

In Table A.1 the best results, fixed S and the training strategy, are depicted. For three datasets, BrainMRI, Chickenpieces and Protein, the training sample size is bigger than the whole training set; in these cases the reported AUC values are those of the experiments with the biggest possible training sample size–there is a $*$ next to the dataset name for such cases.
For most datasets, the best results are achieved using a small sample size, such as **S=64, 128** aside from three datasets. In addition, for the majority of the problems, when changing S, the best training criterion tends to be the same for at least 3 out of 4 values, such as Chickenpieces. Nevertheless there are some datasets for which there does not seem to be one best choice, such as DelftGestures. Another relevant observation is that differences between variants in terms of AUC range from very small to quite striking, such as for WoodyPlants and Zongker when using 2 prototypes instead of one.

Table A.2 depicts the best results as to the depth and training sample size, while the training criterion and the forest size are fixed, averaged across the iterations. Performances increase as T does: indeed the best AUC in terms of average is reached for **T=500**. Nevertheless for many datasets and variants other than the best one, the best choice is achieved with smaller forests. Therefore using always such a big forest would be quite myopic. Analogously to Table A.1 we can infer that the best variant for a dataset is the same across different T in most cases.

Lastly, Table A.3 shows the best averaged AUC results in terms of forest and training sample size, while the training criterion and the depth are fixed.

We can observe that for most datasets, using a smaller depth is more advantageous than employing $D = S - 1$, which is the better choice only in 2 cases out of 10. This must be kept in mind when looking at the averages across the different variants, which might be biased by the difference on said few datasets. Nevertheless the best result is achieved when employing $O - 2PS_D$ and $D = \log_2(S)$, the latter being advantageous also from a computational point of view.

Table A.1: Behaviour of the nine training variants in terms of average AUC when varying S.

S=64	**R-1P**	**R-2P**	**O-1PS$_D$**	**O-2PS$_D$**	**O-2PS$_P$**	**O-1PH**	**O-2PH**	**O-1PRD**	**O-2PRD**
BrainMRI*	0.7770	0.7287	0.8033	0.6877	0.7148	0.7623	0.7098	**0.8164**	0.6861
Chickenpieces	0.8425	0.8520	0.8338	0.8492	0.8423	0.8999	**0.9072**	0.8199	0.8207
CoversSongs	0.9908	0.9919	0.9853	**0.9932**	0.9926	0.9901	0.9931	0.9875	0.9927
DelftGestures	0.8397	**0.9610**	0.8032	0.9271	0.9534	0.9034	0.9114	0.8685	0.9088
DelftPedestrians	0.7359	0.7763	0.7377	0.7962	0.7306	0.7874	0.7979	**0.7989**	0.7989
Flowcyto	0.5905	**0.737**	0.5805	0.7233	0.6967	0.6722	0.6841	0.5869	0.6167
Pendigits	0.6923	0.6983	0.6801	0.7618	0.7381	0.7575	0.7967	0.697	**0.7978**
Protein	0.9944	0.9818	0.9927	0.9627	0.9459	**0.9996**	0.9871	0.9992	0.9692
Woodyplants	0.5453	0.9176	0.4966	0.9318	**0.9335**	0.7712	0.7983	0.6587	0.7443
Zongker	0.5978	0.7418	0.5927	0.7792	0.7264	0.7247	**0.8517**	0.7046	0.8457
Average	0.7606	0.8386	0.7506	0.8412	0.8274	0.8268	**0.8437**	0.7937	0.8181
S=128	**R-1P**	**R-2P**	**O-1PS$_D$**	**O-2PS$_D$**	**O-2PS$_P$**	**O-1PH**	**O-2PH**	**O-1PRD**	**O-2PRD**
BrainMRI*	0.7770	0.7287	0.8033	0.6877	0.7148	0.7623	0.7098	**0.8164**	0.6861
Chickenpieces*	0.8293	0.8517	0.8125	0.8457	0.8364	0.8931	**0.9071**	0.8104	0.8091
CoversSongs	0.9913	0.9912	0.9899	0.9916	0.9917	0.9906	0.9914	0.9894	**0.9924**
DelftGestures	0.8572	**0.9674**	0.8067	0.9428	0.9648	0.9153	0.9217	0.8763	0.9067
DelftPedestrians	0.7391	0.7658	0.7372	0.7944	0.7287	0.7987	0.8035	0.797	**0.8094**
Flowcyto	0.5987	**0.7397**	0.5812	0.7174	0.6900	0.6813	0.6904	0.5483	0.6148
Pendigits	0.7033	0.7037	0.7014	0.7615	0.7465	0.7602	0.7959	0.6577	**0.7968**
Protein*	**0.9923**	0.9874	0.9852	0.9851	0.9858	0.9872	0.9834	0.987	0.9861
Woodyplants	0.4887	0.9259	0.4613	0.9307	**0.9309**	0.7965	0.8166	0.5444	0.7330
Zongker	0.5666	0.7538	0.5682	0.7896	0.7425	0.7299	**0.8589**	0.6742	0.8485
Average	0.7544	0.8415	0.7447	0.8446	0.8332	0.8315	**0.8479**	0.7701	0.8183
S=256	**R-1P**	**R-2P**	**O-1PS$_D$**	**O-2PS$_D$**	**O-2PS$_P$**	**O-1PH**	**O-2PH**	**O-1PRD**	**O-2PRD**
BrainMRI*	0.7770	0.7287	0.8033	0.6877	0.7148	0.7623	0.7098	**0.8164**	0.6861
Chickenpieces	0.8147	0.8289	0.7894	0.8323	0.8307	0.8002	**0.8337**	0.7933	0.8274
CoversSongs	0.9913	0.9912	0.9899	0.9916	0.9917	0.9906	0.9914	0.9894	**0.9924**
DelftGestures	0.8517	0.9772	0.8163	0.9598	**0.9780**	0.9261	0.9335	0.8832	0.9043
DelftPedestrians	0.7435	0.7543	0.7484	0.8057	0.7316	0.8018	0.8105	0.7976	**0.8229**
Flowcyto	0.5887	**0.7371**	0.5857	0.7135	0.6863	0.6977	0.6909	0.5296	0.6132
Pendigits	0.6873	0.7005	0.6857	0.7530	0.7402	0.7555	0.7848	0.5899	**0.7986**
Protein*	**0.9923**	0.9874	0.9852	0.9851	0.9858	0.9872	0.9834	0.987	0.9861
Woodyplants	0.4597	0.9159	0.4608	0.9169	**0.9222**	0.8176	0.8283	0.4441	0.7322
Zongker	0.5839	0.7720	0.5950	0.8116	0.7501	0.7291	0.8483	0.6133	**0.8544**
Average	0.7490	0.8393	0.7460	**0.8457**	0.8331	0.8268	0.8415	0.7444	0.8218
S=512	**R-1P**	**R-2P**	**O-1PS$_D$**	**O-2PS$_D$**	**O-2PS$_P$**	**O-1PH**	**O-2PH**	**O-1PRD**	**O-2PRD**
BrainMRI*	0.7770	0.7287	0.8033	0.6877	0.7148	0.7623	0.7098	**0.8164**	0.6861
Chickenpieces*	0.8147	0.8289	0.7894	0.8323	0.8307	0.8002	**0.8337**	0.7933	0.8274
CoversSongs	0.9913	0.9912	0.9899	0.9916	0.9917	0.9906	0.9914	0.9894	**0.9924**
DelftGestures	0.8504	0.9753	0.782	0.9617	**0.9779**	0.9222	0.9293	0.8523	0.8640
DelftPedestrians	0.7349	0.7580	0.7482	0.7571	0.7574	0.7416	0.7913	0.7208	**0.7923**
Flowcyto	0.5977	**0.7418**	0.5888	0.7376	0.7415	0.5105	0.6065	0.4965	0.6105
Pendigits	0.7102	0.6994	0.6958	0.7476	0.7368	0.7481	0.7745	0.5405	**0.7881**
Protein*	**0.9923**	0.9874	0.9852	0.9851	0.9858	0.9872	0.9834	0.987	0.9861
Woodyplants	0.4401	0.9090	0.4542	**0.9139**	0.9126	0.5507	0.758	0.5269	0.7616
Zongker	0.5547	0.7851	0.5583	0.8198	0.7659	0.7294	**0.8476**	0.5546	0.8455
Average	0.7464	0.8405	0.7395	**0.8434**	0.8415	0.7743	0.8226	0.7278	0.8154

Table A.2: Behaviour of the nine training variants in terms of average AUC when varying T.

T=50	**R-1P**	**R-2P**	**O-1PS$_D$**	**O-2PS$_D$**	**O-2PS$_P$**	**O-1PH**	**O-2PH**	**O-1PRD**	**O-2PRD**
BrainMRI	0.7557	0.6246	**0.7721**	0.5992	0.641	0.6508	0.6320	0.7459	0.5721
Chickenpieces	0.8406	0.8544	0.8266	0.8511	0.8432	0.8981	**0.9078**	0.8169	0.8252
CoversSongs	0.9916	0.9900	0.9887	0.9896	**0.9919**	0.9902	0.9914	0.9881	0.9914
DelftGestures	0.8585	0.9743	0.7873	0.9640	**0.9770**	0.9257	0.9311	0.8795	0.9135
DelftPedestrians	0.7518	0.7833	0.7623	0.8028	0.7678	0.8084	0.8162	0.7984	**0.8172**
Flowcyto	0.6109	**0.7433**	0.5998	0.7301	0.7317	0.6953	0.6937	0.5831	0.6227
Pendigits	0.7264	0.7030	0.7041	0.7611	0.7457	0.7622	0.7897	0.6889	**0.8024**
Protein	0.9932	0.9844	0.9846	0.9776	0.9812	**0.9991**	0.9858	0.9988	0.9818
Woodyplants	0.5435	0.9144	0.5257	0.9264	**0.9290**	0.8125	0.8261	0.6198	0.7572
Zongker	0.5994	0.7649	0.5973	0.7919	0.7581	0.7335	0.8559	0.6970	**0.8587**
Average	0.7672	0.8337	0.7549	0.8394	0.8367	0.8276	**0.8430**	0.7816	0.8142
T=100	**R-1P**	**R-2P**	**O-1PS$_D$**	**O-2PS$_D$**	**O-2PS$_P$**	**O-1PH**	**O-2PH**	**O-1PRD**	**O-2PRD**
BrainMRI*	0.7328	0.6369	**0.777**	0.6262	0.6820	0.7049	0.6533	0.7721	0.5541
Chickenpieces	0.8379	0.8500	0.8267	0.8491	0.8458	0.8973	**0.9073**	0.8179	0.8263
CoversSongs	0.9907	0.9907	0.9886	**0.9926**	0.9917	0.9910	0.9923	0.9892	0.9919
DelftGestures	0.8609	0.9713	0.8138	0.9613	**0.9794**	0.9253	0.9295	0.8839	0.9109
DelftPedestrians	0.7402	0.7763	0.7505	0.8015	0.7661	0.8070	0.8090	0.7984	**0.8199**
Flowcyto	0.6025	**0.7416**	0.5885	0.7296	0.7336	0.6919	0.6904	0.5753	0.6209
Pendigits	0.7167	0.7061	0.6972	0.7629	0.7413	0.7586	0.7930	0.6865	**0.7997**
Protein	0.9937	0.9847	0.9889	0.9826	0.9829	**0.9993**	0.9865	0.999	0.9823
Woodyplants	0.5367	0.9236	0.5145	0.9299	**0.9312**	0.8126	0.8211	0.6287	0.7558
Zongker	0.5869	0.7713	0.6058	0.8041	0.7623	0.7309	0.8534	0.6979	**0.8561**
Average	0.7599	0.8352	0.7552	**0.8440**	0.8416	0.8319	0.8436	0.7849	0.8118
T=200	**R-1P**	**R-2P**	**O-1PS$_D$**	**O-2PS$_D$**	**O-2PS$_P$**	**O-1PH**	**O-2PH**	**O-1PRD**	**O-2PRD**
BrainMRI	0.6697	0.6770	**0.7754**	0.6295	0.6738	0.7098	0.6344	0.7607	0.6049
Chickenpieces	0.8341	0.8462	0.8260	0.8483	0.8446	0.8981	**0.9063**	0.8153	0.8262
CoversSongs	0.9905	0.9912	0.9883	**0.9926**	0.9924	0.9907	0.9923	0.9894	0.9916
DelftGestures	0.8481	0.9769	0.8081	0.9600	**0.9790**	0.9246	0.9295	0.8735	0.9121
DelftPedestrians	0.7353	0.7705	0.7462	0.8033	0.7679	0.8089	0.8079	0.7990	**0.8119**
Flowcyto	0.5967	**0.7418**	0.5883	0.7330	0.7357	0.6893	0.6920	0.5721	0.6202
Pendigits	0.7059	0.7046	0.7022	0.7595	0.7417	0.7588	0.7959	0.6837	**0.7987**
Protein	0.9931	0.9843	0.9890	0.9840	0.9841	**0.9995**	0.9868	0.9989	0.9842
Woodyplants	0.5120	0.9262	0.4890	**0.9310**	0.9296	0.8090	0.8208	0.6352	0.7520
Zongker	0.5781	0.7812	0.5866	0.8199	0.7604	0.7282	**0.8558**	0.6962	0.8497
Average	0.7463	0.8400	0.7499	**0.8461**	0.8409	0.8317	0.8422	0.7824	0.8151
T=500	**R-1P**	**R-2P**	**O-1PS$_D$**	**O-2PS$_D$**	**O-2PS$_P$**	**O-1PH**	**O-2PH**	**O-1PRD**	**O-2PRD**
BrainMRI	0.6574	0.6615	**0.7721**	0.6426	0.6459	0.7410	0.6525	0.7508	0.6377
Chickenpieces	0.8318	0.8440	0.8256	0.8482	0.8425	0.8969	**0.9066**	0.8155	0.8274
CoversSongs	0.9904	0.9910	0.9886	**0.9924**	**0.9924**	0.9899	0.9919	0.9889	0.9917
DelftGestures	0.8460	0.9740	0.8172	0.9580	**0.9782**	0.9226	0.9313	0.8719	0.9126
DelftPedestrians	0.7347	0.7689	0.7404	0.8004	0.7615	0.8050	0.8105	0.7983	**0.8133**
Flowcyto	0.5903	**0.7407**	0.5860	0.7348	0.7355	0.6892	0.6914	0.5729	0.6226
Pendigits	0.6961	0.7021	0.7067	0.7583	0.7398	0.7577	0.7962	0.6819	**0.7978**
Protein	0.9936	0.9859	0.9922	0.9838	0.9842	**0.9995**	0.9864	0.9990	0.9845
Woodyplants	0.4941	0.9251	0.4743	**0.9342**	0.9318	0.8122	0.8173	0.6429	0.7511
Zongker	0.5658	0.7700	0.5809	0.8140	0.7599	0.7273	**0.8550**	0.6971	0.8464
Average	0.7400	0.8363	0.7484	**0.8467**	0.8372	0.8341	0.8439	0.7819	0.8185

Table A.3: Behaviour of the nine training variants in terms of average AUC when varying D.

D=S-1	**R-1P**	**R-2P**	**O-1PS$_D$**	**O-2PS$_D$**	**O-2PS$_P$**	**O-1PH**	**O-2PH**	**O-1PRD**	**O-2PRD**
BrainMRI	0.7311	0.6967	0.8016	0.6574	0.6959	0.7574	0.6828	**0.8131**	0.6680
Chickenpieces	0.8402	0.8523	0.8334	0.8513	0.8449	0.8981	**0.9047**	0.8198	0.8261
CoversSongs	0.9916	0.9919	0.9898	**0.9931**	0.9929	0.9913	0.9930	0.9902	0.9929
DelftGestures	0.8607	0.9791	0.8266	0.9651	**0.9814**	0.9217	0.9310	0.8806	0.9151
DelftPedestrians	0.7521	0.7817	0.7682	0.8039	0.7746	**0.8143**	0.7999	0.8000	0.7920
Flowcyto	0.6150	**0.7459**	0.6068	0.7365	0.7401	0.6973	0.6926	0.5872	0.6262
Pendigits	0.7305	0.7098	0.7215	0.7655	0.7486	0.7648	0.7979	0.6975	**0.8012**
Protein	0.9946	0.9870	0.9926	0.9840	0.9844	**0.9996**	0.9872	0.9991	0.9843
Woodyplants	0.5508	0.9267	0.5349	**0.9357**	0.9353	0.8181	0.8297	0.6564	0.7662
Zongker	0.6046	0.7878	0.6297	0.8214	0.7671	0.7354	**0.8617**	0.7046	0.8575
Average	0.7671	0.8459	0.7705	**0.8514**	0.8465	0.8398	0.8480	0.7948	0.8229
D=log$_2$(S)	**R-1P**	**R-2P**	**O-1PS$_D$**	**O-2PS$_D$**	**O-2PS$_P$**	**O-1PH**	**O-2PH**	**O-1PRD**	**O-2PRD**
BrainMRI	0.7770	0.7098	0.7934	0.6770	0.7066	0.7426	0.6885	**0.8033**	0.6754
Chickenpieces	0.8417	0.8567	0.8334	0.8545	0.8490	0.8989	**0.9095**	0.8224	0.8305
CoversSongs	0.9917	0.9919	0.9901	**0.9933**	0.9930	0.9918	0.9932	0.9900	0.9931
DelftGestures	0.8614	**0.9792**	0.8265	0.9651	0.9791	0.9282	0.9341	0.8904	0.9175
DelftPedestrians	0.7423	0.7843	0.7675	0.8099	0.7752	0.7896	0.8189	0.7975	**0.8246**
Flowcyto	0.5949	**0.7450**	0.6038	0.7375	0.7410	0.6895	0.6960	0.5764	0.6207
Pendigits	0.7207	0.7090	0.7206	0.7662	0.7484	0.7606	0.7988	0.6847	**0.8050**
Protein	0.9936	0.9870	0.9923	0.9845	0.9854	**0.9996**	0.9865	0.9992	0.9859
Woodyplants	0.5397	0.9282	0.5343	**0.9370**	0.9364	0.8044	0.8211	0.6442	0.7655
Zongker	0.6015	0.7862	0.6295	0.8243	0.7654	0.7176	**0.8630**	0.6848	0.8620
Average	0.7665	0.8477	0.7692	**0.8549**	0.8480	0.8323	0.8509	0.7893	0.8280

Appendix B

Complete Results Chapter 4

In this additional section we present in Tables B.1 and B.2 results related respectively to Sections 4.4.2 and 4.4.4 of Chapter 4. As briefly pointed out in the related sections, both tables present the dataset-wise results when training the Isolation Forest with default parameter values for S, T and F, set respectively to 256, 100, f, and a different value for the depth, which has been set to $D = S - 1$. Table B.1 differs from Table B.2 in terms of the used combiner function: in the former we employ $s(x)$ while in the latter $p(x)$. Results are presented in terms of average AUC across the 10 iteration. We also performed a Wilcoxon signed-rank test, which results are visualized in the same way as done for Tables 4.2 and 4.4 in Chapter 4.

We can observe that in Table B.1 there is at least one path-weighted variant significantly outperforming $s(x)$ on 11 datasets. In particular **V5** appears to be the best variant and all proposed scoring functions, except $V3$, perform significantly better than $s(x)$ on at least 7 datasets. Lastly $s(x)$ is the best significant choice only for one dataset.

Analogously to Table B.1, in Table B.2 we can infer that all scoring functions except $V3$ work well. We can also make two novel observations: i) **V5** and **V4** perform comparably well, i.e. there is no single best variant; and ii) the goodness of the novel aggregation scheme is confirmed by the good performances of $p(x)$.

Table B.1: Comparison between $s(x)$ and path-weighted scores when employing the standard iForest parametrization except for the depth set to $D = S - 1$.

Dataset	s(x)	V1	V2	V3	V4	V5
Adult	0.659	**0.660**	0.659	0.649	0.658	0.656
Annthyroid	0.919	0.937*	0.933*	0.909	0.929*	**0.949***
Arrhythmia	0.757	0.763	0.763	0.751	0.758	**0.774***
Cardiotocography	**0.747**	0.697*	0.655*	0.629*	0.735*	0.732
ForestCover	0.834	0.865*	0.856	0.820	0.845*	**0.934***
Hepatitis	0.701	0.717	0.724	0.680	0.717*	**0.738***
Http	0.992	**0.996***	**0.996***	0.995	0.994*	0.994
Ionosphere	0.910	0.941*	**0.955***	0.941*	0.928*	0.949*
PageBlocks	0.811	**0.887***	0.833	0.825	0.845*	0.872*
Pendigits	0.834	**0.927***	0.926*	**0.927***	0.868*	0.874*
Pima	**0.731**	0.697*	0.712*	0.681*	0.724*	0.701*
Shuttle	0.996	0.996	0.996	0.987*	0.997*	**0.998***
Smtp	0.911	0.922	**0.929***	0.917	0.923*	0.919
Spambase	0.833	0.796*	0.824	0.748*	0.833	**0.840**
Stamps	**0.957**	0.928*	0.939	0.832*	0.956	0.951
Wilt	0.531	0.698*	0.684*	**0.722***	0.57*	0.664*

Table B.2: Comparison between $p(x)$ and path-weighted scores, when employing the standard iForest parametrization except for the depth set to $D = S - 1$.

Dataset	p(x)	V1	V2	V3	V4	V5
Adult	0.657	**0.660**	0.658	0.649	0.658	0.656
Annthyroid	0.928	0.941	0.937	0.913	0.943*	**0.949***
Arrhythmia	0.768	0.765	0.767	0.752	0.772	**0.774**
Cardiotocography	**0.747**	0.698*	0.676*	0.639*	0.745	0.732*
ForestCover	0.925	0.877*	0.890	0.831*	0.931	**0.934**
Hepatitis	**0.745**	0.718	0.727	0.674*	0.742	0.738
Http	0.993	**0.995***	**0.995***	**0.995**	0.994	0.994
Ionosphere	0.934	0.943	**0.956***	0.940	0.945*	0.949*
PageBlocks	0.843	**0.888***	0.855	0.838	0.859*	0.872*
Pendigits	0.787	**0.926***	0.917*	0.924*	0.837*	0.874*
Pima	0.703	0.697	**0.709**	0.680*	**0.709**	0.701
Shuttle	0.997	0.997	0.996	0.988*	**0.998***	**0.998**
Smtp	0.911	0.921	**0.924**	0.915	0.919	0.919
Spambase	0.835	0.800*	0.836	0.756*	**0.847**	0.840
Stamps	0.949	0.929	0.941	0.835*	**0.951**	**0.951**
Wilt	0.538	0.706*	0.687*	**0.727***	0.600*	0.664*

Appendix C

Complete Results Chapter 5

This Appendix refers to Chapter 5 and it is divided into two sections: in the first we report the preliminary analyses made on each outlier detector for setting the various parameters; in the second section we report distance-wise results for each outlier detector except for *KNNDist-Av* which results are reported in Chapter 5.

C.1 Setting the Parametrizations of the Outlier Detectors

In this first section, as mentioned in the preamble, we make an analysis aimed at inferring the best parametrization for each outlier detector independently of the distance measure given in input. For each approach the analysis is performed as follows: given the range of values for the parameter to be set, e.g. the number of clusters K, we select one value, one dataset and one distance measure and average the AUC across the iterations. Starting from these results, we perform a non-parametric statistical analysis, analogously to what done throughout the thesis, to find the best value for the parameter to be set. We visualize the results via a CD diagram.

In Figure C.1 (a) we analyze the behaviour of changing the size of neighborhood K where $K \in [1, 20]$ when employing the *KNNd* outlier detector: we can observe that higher Ks tend to be better ranked, with $K = 20$ being the first one in the ranking. However, the performance reaches a plateau since $K = 20$ is comparable to all $K \geq 8$; we therefore choose to set $K = 8$ to save some computational time and because it may leads to poor performances as the size of the dataset decreases. For example for Hepatitis $K = 20$ may be too big and lead to very similar anomaly scores for all objects.
A similar behaviour can be observed for *KNNDist*: K ranges again in $[1, 20]$ and in Figure C.1 (b) the CD diagram comparing the different options is depicted: analogously to KNNd we can observe that greater Ks are ranked better. However for *KNNDist* the best $K = 18$ is comparable to both bigger and smaller Ks, with $K = 4$ being the smallest comparable option. For the same reasons as KNNd we therefore set $K = 4$.
The observations are the same also for the other two variants of KNN-based outlier detectors, *KNNd-Av* and *KNNDist-Av*, which CD diagrams are depicted respectively in Figure C.1 (c) and (d), leading to respectively setting $K = 16$ and $K = 8$.

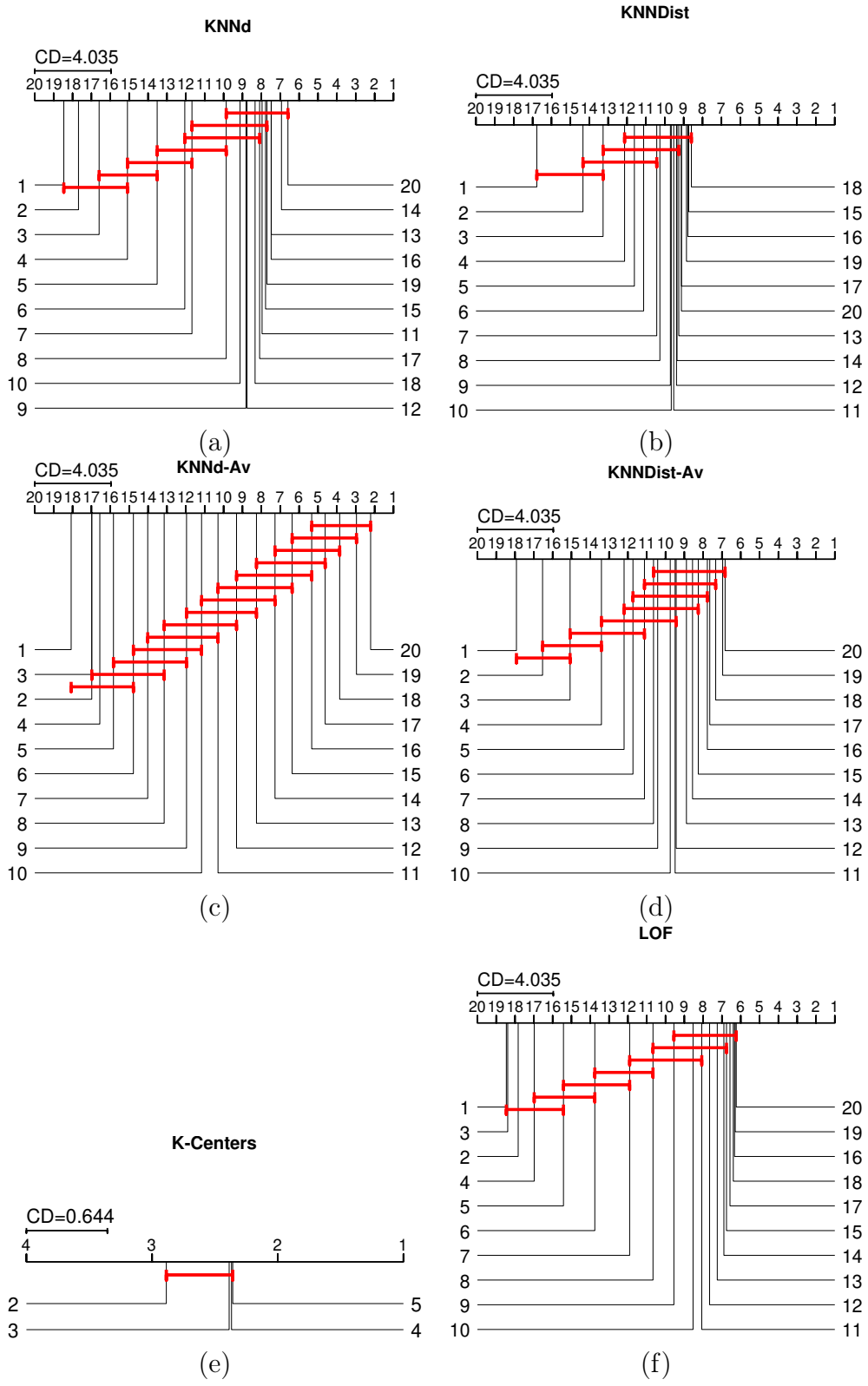


Figure C.1: Comparison of the different values the K parameter can assume for each outlier detector via a CD diagram.

As to *K-Centers* we varied the number of clusters from 2 up to 5 and analyzed the different results: from the CD diagram in Figure C.1 (e) we can infer that the number of clusters is not crucial, i.e. performances are comparable. Since the values of K are all small, i.e. the computational overhead in using an higher K is not big, we set $K = 5$ which is the first ranked variant.
Lastly in Figure C.1 (f) we analyze the behaviour of LOF as the size of the analyzed neighborhood K changes, where $K \in [1, 20]$. We can infer that a bigger neighborhood is more informative overall, even though the performances of $K \in [9, 20]$ are all comparable. We therefore set $K = 9$ for computational purposes and to take into account that some datasets, e.g. Hepatitis, are very small and could be impacted negatively by a greater K.

C.2 Comparison of Distance Measures

In this section we illustrate the performances of all outlier detectors on all 6 distance measures, aside for KNNDist-Av which analysis is discussed in Chapter 5. For each outlier detector we perform the analysis as follows:

- Given a dataset and a distance measure we compute the average AUC across the iterations and report the results in the related table.
- In each table we highlight in **bold** the best result.
- On the results of each table, we perform a Friedman test followed by a post-hoc Nemenyi test to assess whether there is a statistically significant best distance measure where $\alpha = 0.05$.
- The results of the statistical analysis are depicted via a CD diagram.

In Table C.1 we report the average AUC for **KNNd**. We can observe that all distances have similar performances on the majority of the datasets but there are some exceptions. In detail on *Hepatitis* we can observe a relevant difference between *Shi*, the worst performing and *Aryal*, the best choice; whereas on *Spambase* mass-based distance measures lead to relevant improvements and finally *Stamps* seem to reach a good performance only when employing *RatioRF*. In general, RatioRF and Aryal seem to be the best performing distance measures: to confirm this hypothesis we compare the different distances from a statistical point of view. From the CD diagram in Figure C.2 we can infer that **RatioRF** is the best choice, comparable to some other distance measures, whereas *Shi* and *Zhu3* are the worst ones.

In Table C.2 the same analysis is reported for **KNNDist**: all distances seem to be overall well performing and aside from *Stamps*, analogously to Table C.1, and *Wilt*, there does not seem to be a dataset for which some distance measure has a relevant performance decrease. The best distances seem to be *Aryal* and *RatioRF*. In Figure C.3 the CD diagram is reported: the

Table C.1: Evaluation of the 6 distance measures-KNNd

Dataset	Shi	Zhu2	Zhu3	RatioRF	Ting	Aryal
Annthyroid	0.5171	0.5602	0.5260	0.5722	0.5622	**0.5748**
Arrhythmia	0.7063	0.7245	**0.7467**	0.7268	0.7255	0.6788
Cardiotocography	0.4839	0.4944	0.4828	**0.4984**	0.4839	0.4704
Hepatitis	0.5631	0.6141	0.6279	0.6402	0.6350	**0.7020**
Ionosphere	0.8960	0.9318	0.9162	0.9328	0.9292	**0.9379**
Pima	0.5498	0.5828	0.5482	**0.5855**	0.5675	0.5748
Spambase	0.4906	0.5275	0.5027	0.5177	0.6258	**0.6608**
Stamps	0.6205	0.6988	0.6436	**0.7246**	0.6191	0.5722
Wilt	0.7588	0.7582	0.7626	0.7611	**0.7776**	0.7670

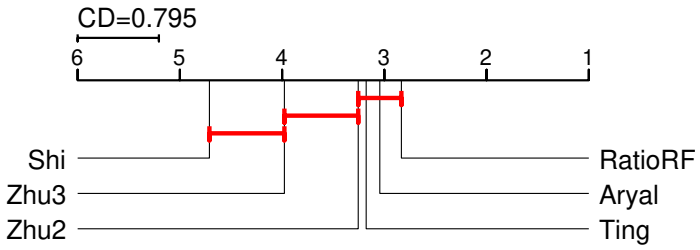


Figure C.2: CD Diagram that compares the KNNd performances across the 6 distance measures.

Table C.2: Evaluation of the 6 distance measures-KNNDist.

Dataset	Shi	Zhu2	Zhu3	RatioRF	Ting	Aryal
Annthyroid	0.8533	0.8822	0.8602	0.8821	0.8917	**0.9040**
Arrhythmia	0.8090	0.8152	0.8135	**0.8212**	0.8085	0.7830
Cardiotocography	0.4630	0.4752	0.4689	**0.4988**	0.4154	0.4496
Hepatitis	0.6839	0.7036	0.6726	0.6801	0.7302	**0.7766**
Ionosphere	0.9485	0.9510	0.9464	0.9404	0.9681	**0.9748**
Pima	0.7205	0.7274	0.7203	0.7289	0.7353	**0.7468**
Spambase	0.8538	0.8583	0.8510	0.8572	0.8672	**0.8777**
Stamps	0.9203	0.9343	0.9273	**0.9398**	0.8348	0.7466
Wilt	0.7083	0.6729	0.6803	0.6805	0.7909	**0.8277**

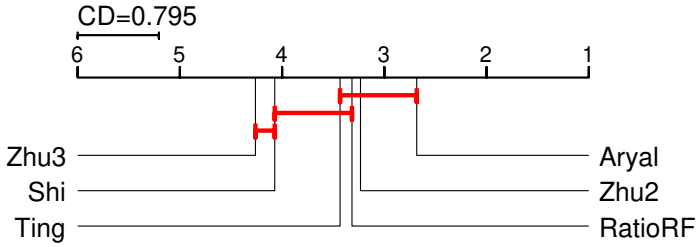


Figure C.3: CD Diagram that compares the KNNDist performances across the 6 distance measures.

Table C.3: Evaluation of the 6 distance measures-KNNd-Av.

Dataset	Shi	Zhu2	Zhu3	RatioRF	Ting	Aryal
Annthyroid	0.5189	0.5962	0.5426	0.6041	0.5993	**0.6277**
Arrhythmia	0.8018	0.8085	0.8043	**0.8252**	0.7970	0.7568
Cardiotocography	0.4787	0.4981	0.4892	**0.5000**	0.4748	0.4526
Hepatitis	0.6772	0.7239	0.6949	0.7054	0.7276	**0.7709**
Ionosphere	0.9461	0.9655	0.9534	0.9632	0.9609	**0.9723**
Pima	0.6036	0.6285	0.6064	**0.6408**	0.6013	0.6281
Spambase	0.4504	0.5472	0.5119	0.5018	0.7021	**0.7611**
Stamps	0.6858	0.7755	0.7185	**0.8093**	0.6701	0.6401
Wilt	0.8713	0.8600	0.8688	0.8704	**0.8804**	0.8769

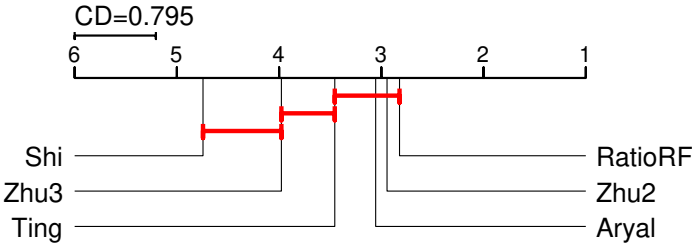


Figure C.4: CD Diagram that compares the KNNd-Av performances across the 6 distance measures.

best distance measure is **Aryal** comparable to all other measures but *Zhu3* and *Shi* which are the worst choice, analogously to *KNNd*.

In Table C.3 we report the results for **KNNd-Av**: also for this outlier detector only for *Spambase* and *Stamps* there are some relevant differences between different distance measures. Overall **Aryal** and **RatioRF** seem again to be the best choice: this is confirmed by the statistical analysis which results are depicted via the the CD diagram in Figure C.4.

In Table C.4 and Figure C.5 we analyze the performances of the different distance measures when input to **K-Centers** with $K = 5$. From Table C.4 we can make a novel observation with respect to the analyses on the KNN-based detectors: mass-based measures, i.e. *Ting* and *Aryal*, are rarely the best choice–even though only for *Stamps* the difference in terms of AUC is impacting. *RatioRF* seems instead to be a very good choice also for this clustering-based technique. Indeed these observations are confirmed by the CD diagram in Figure C.4: **RatioRF** is the best choice, along with all non mass-based measures and *Aryal* and *Ting* are the worst ranked.

In Table C.5 we report the averaged AUC results obtained using LOF with $K = 9$. The conclusions are analogous to those made on KNN-based methods: *RatioRF* and *Aryal* are the best choice and only for few datasets there are relevant differences when employing different distance measures. In Figure C.6 we present the related CD diagram which confirms our observations, i.e. **RatioRF** is the best choice comparable to *Aryal* and *Zhu2*.

Lastly we report in Table C.6 the performances of the different distance

Table C.4: Evaluation of the 6 distance measures-K-Centers.

Dataset	Shi	Zhu2	Zhu3	RatioRF	Ting	Aryal
Annthyroid	**0.7926**	0.7370	0.7829	0.7594	0.6824	0.6691
Arrhythmia	0.7905	**0.8052**	0.8020	0.8023	0.7526	0.7392
Cardiotocography	**0.6741**	0.6469	0.6363	0.6099	0.5553	0.6185
Hepatitis	0.5265	0.5308	0.4920	**0.6436**	0.5724	0.6385
Ionosphere	0.8990	0.9446	0.9297	**0.9529**	0.9174	0.9144
Pima	0.6329	**0.6546**	0.6479	0.6328	0.6193	0.6269
Spambase	0.8262	0.8278	0.8141	**0.8354**	0.8101	0.7122
Stamps	0.7960	0.8283	**0.8646**	0.8560	0.6474	0.5744
Wilt	0.4584	0.4540	0.4385	0.4510	0.5475	**0.5584**

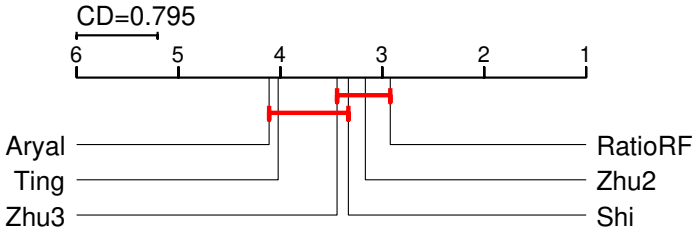


Figure C.5: CD Diagram that compares the K-Centers performances across the 6 distance measures.

Table C.5: Evaluation of the 6 distance measures-LOF.

Dataset	Shi	Zhu2	Zhu3	RatioRF	Ting	Aryal
Annthyroid	0.3717	0.6025	0.5656	0.6520	0.6434	**0.6747**
Arrhythmia	0.8154	0.8106	0.8097	**0.8242**	0.8069	0.7787
Cardiotocography	0.4653	0.4861	0.4775	**0.4936**	0.4768	0.4530
Hepatitis	0.6792	0.7416	0.7231	0.7368	0.7467	**0.7652**
Ionosphere	0.9572	0.9696	0.9585	0.9726	0.9657	**0.9753**
Pima	0.6260	0.6360	0.6153	**0.6543**	0.6024	0.6396
Spambase	0.4641	0.5294	0.4996	0.5210	0.6900	**0.7405**
Stamps	0.7474	0.8161	0.7774	**0.8524**	0.6544	0.6388
Wilt	0.8463	0.8268	0.8379	0.8431	**0.8634**	0.8607

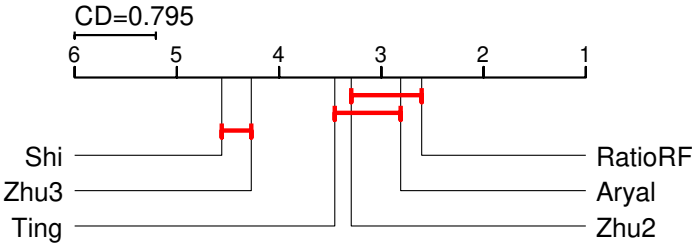


Figure C.6: CD Diagram that compares the LOF performances across the 6 distance measures.

measures employed as input to a ProxIForest which has been trained using the

Table C.6: Evaluation of the 6 distance measures-ProxIF.

Dataset	Shi	Zhu2	Zhu3	RatioRF	Ting	Aryal
Annthyroid	0.8416	0.8515	0.8278	0.8624	**0.9190**	0.8993
Arrhythmia	0.7640	0.7609	0.7690	0.7610	**0.8027**	0.7996
Cardiotocography	0.5982	0.6346	0.6080	**0.6579**	0.5597	0.5584
Hepatitis	0.6591	0.6781	0.6436	0.6664	0.6960	**0.7353**
Ionosphere	0.8224	0.8135	0.7997	0.7555	0.8859	**0.9165**
Pima	0.7338	0.7421	0.7349	0.7481	0.7599	**0.7721**
Spambase	0.8320	0.8411	0.8242	0.8423	0.8410	**0.8461**
Stamps	0.9512	0.9475	0.9489	**0.9541**	0.8998	0.8510
Wilt	0.4197	0.4004	0.3992	0.4028	0.4881	**0.5338**

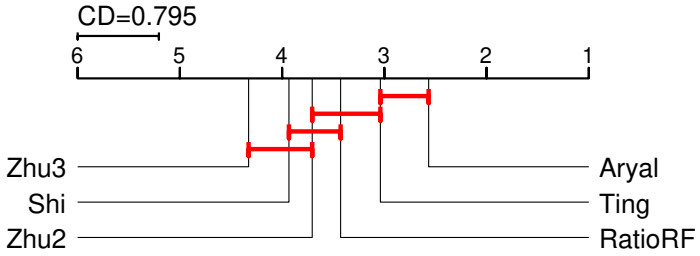


Figure C.7: CD Diagram that compares the ProxIF performances across the 6 distance measures.

default parametrization. Even though there are relevant differences between different distance measures on a given dataset, there is not a bad performing distance measure. Also for this outlier detector, which is rather different from the others, mass-based measures and *RatioRF* seem to be the best choice. However the CD diagram in Figure C.7 shows that **Aryal** is the best ranked variant and it is comparable only to the other mass-based similarity measure *Ting*.