Utrecht University

Theoretical and Practical Aspects of Isolation Forest

Mark Sterkenburg

Supervised by: Prof. dr. ir. C. W. Oosterlee and L.A. Souto Arias MSc

Second reader: Dr. K. Dajani

Mathematical Sciences,
Utrecht University, The Netherlands,
Faculty of Science

August 25, 2022

Abstract

Outlier detection methods are becoming increasingly more popular, for example, in the financial world to detect fraudulent transactions. In this thesis, we explore the Isolation Forest (IF) algorithm which is a data-driven anomaly detection method. This method distinguishes itself from other outlier detecting methods, because it isolates the outliers directly instead of creating a profile of the normal instances. However, there is not much theory known behind this algorithm and therefore this is explored in this thesis. Theory based on the number of random splits needed to isolate a datapoint is developed and numerically validated with the IF algorithm. Moreover, new outlier detection methods are developed by combining the original IF algorithm with the theoretical formulas of this algorithm. With these methods, the outliers in multi-dimensional datasets can be detected due to the projection they are using to transform multi-dimensional datasets into one-dimensional ones. We test these methods rigorously for multiple different datasets and that shows us very good performances of some of these new methods. Furthermore, the original IF algorithm is used for further testing of multiple components of this algorithm. For example, the impact of pruning the isolation trees and the number of trees on the performance of the algorithm are tested. Also, different scoring functions are tested in combination with this algorithm. The IF algorithm is also compared with two other outlier detection methods and shows very good results when detecting the most outlying points of multiple random datasets. Finally, a real-world financial dataset is used to test the new methods and the original IF method on. On this dataset, the Transformed IF method gives more accurate results than the original IF algorithm.

Preface

This report contains the results of my master thesis project, which I have been fortunate enough to work on for the past 9 months. Before this all started, I had no idea of the immensity of such a project. Doing research on a scale like this was something I had never done and that was sometimes very challenging. Also, being in my room for the majority of this project was difficult at times. However, I have learned numerous things on the way and very interesting results were obtained. This gave me a lot of satisfaction.

I would like to express my sincerest gratitude to the people that have helped me with my research project. Prof. Kees Oosterlee, my academic supervisor and my daily supervisor, Luis Souto Arias, who have both helped me tremendously with their guidance, feedback and our weekly meetings. You were always there for me when I needed the help and that led to endless conversations via the mail. For this, I am extremely grateful.

Finally, I want to thank my parents for their support during this long process. They were always there for me when I needed it.

Mark Sterkenburg
August 25, 2022
Zeist

Contents

List of Figures **3**

List of Tables **5**

Nomenclature **9**

1 Introduction **11**
1.1 Research Questions . . . 12
1.2 Thesis Structure . . . 12

2 Outlier Detection **14**
2.1 Existing Methods . . . 14
2.1.1 Distance-Based Methods . . . 14
2.1.2 Non-Parametric Density-Based Methods . . . 15
2.1.3 Parametric Density-Based Methods . . . 16
2.1.4 Cluster-Based Methods . . . 18
2.1.5 Measures of Quality . . . 18
2.2 Isolation Forest . . . 19
2.2.1 Depth-Based Methods . . . 19
2.2.2 Basics of Isolation Forest . . . 19
2.3 Transformed Isolation Forest . . . 22
2.3.1 Center of Mass . . . 22
2.3.2 Distance Functions . . . 23
2.3.3 Local Transformation (kNN-distance) . . . 24
2.3.4 Counterexample for the Transformed Isolation Forest Based on Distance to the Center of Mass . . . 25
2.4 Summary . . . 26

3 Theoretical Results and Numerical Experiments **27**
3.1 Theory One-Dimensional Case . . . 27
3.1.1 Fringe Points . . . 27
3.1.2 Interior Points . . . 29
3.2 Theory Two-Dimensional Case . . . 34
3.2.1 Fringe Point in Both Directions . . . 34
3.3 Numerical Validation One-Dimensional Case . . . 37
3.3.1 Fringe Points . . . 37
3.3.2 Interior Points . . . 39
3.4 Numerical Validation Two-Dimensional Case . . . 41

3.4.1 Fringe Point in Both Directions . . . 42
3.5 Outlier Detection with Transformed Isolation Forest . . . 43
3.5.1 Counterexample for the Transformed Isolation Forest Based on Distance to the Center of Mass . . . 43
3.5.2 One-Dimensional Case . . . 44
3.5.3 Two-Dimensional Case . . . 46
3.5.4 Three-Dimensional Case . . . 49
3.5.5 Multi-Dimensional Case . . . 50
3.5.6 Outlier Detection in Low-Dimensional Subspaces . . . 56
3.6 Summary . . . 57

4 Further Rigorous Testing of Isolation Forest and Transformed Isolation Forest 59
4.1 Tests with Classical Isolation Forest Components . . . 59
4.1.1 Computing Times . . . 59
4.1.2 Number of Trees . . . 60
4.1.3 Pruning . . . 62
4.2 Performance of Advanced Isolation Forest Components . . . 63
4.2.1 Different Scoring Functions . . . 63
4.2.2 Different Outlier Detection Methods . . . 67
4.3 Summary . . . 70

5 Real-World Financial Dataset 72
5.1 Elliptic Dataset . . . 72
5.1.1 Tests with Original Isolation Forest and Transformed Isolation Forest . . . 72
5.1.2 Pruning . . . 75
5.1.3 Different Scoring Functions . . . 76
5.2 Summary . . . 78

6 Conclusion and Discussion 79
6.1 Conclusion . . . 79
6.2 Recommendations for Future Research . . . 83

A Additional Theoretical results 86
A.1 Fringe Points . . . 86
A.2 Interior Points . . . 88
A.3 Two-Dimensional Case . . . 89
A.4 Pruning . . . 91

List of Figures

2.1 2×2 confusion matrix of a binary classification problem. Figure is taken from [14]. 18

2.2 Given a Gaussian distribution (135 points), (a) a normal point x_i requires for example twelve random partitions to be isolated; (b) an anomaly requires only four partitions to be isolated. Figure taken from [18]. 20

2.3 An example of an isolation tree created from a small dataset. Figure is taken from [12]. 21

2.4 Center of mass for some simple geometric shapes (red dots). Figure is taken from [1]. 22

2.5 Center of mass of a small 2D dataset (red dot). 23

2.6 kNN for different numbers of neighbours. Figure is taken from [23]. 24

2.7 Two-dimensional dataset of 100 points with 4 clear outliers. 25

2.8 Transformed one-dimensional datasets of the two-dimensional dataset of Figure 2.7 26

3.1 Probability that the left fringe point and an interior point of a continuous uniform distributed dataset become isolated in s random splits for the Isolation Forest [8] method with 100000 trees. 40

3.2 Probability that the left fringe point and an interior point of a standard normally distributed dataset become isolated in s random splits for the Isolation Forest [8] method with 100000 trees. 41

3.3 Standard normally distributed dataset of 100 points. 44

3.4 l_1-distance between the CM and every datapoint in the standard normal distributed dataset of Figure 3.3. 45

3.5 Two-dimensional standard normally distributed dataset of 100 points. 46

3.6 Distance between the CM and every datapoint in the standard normally distributed dataset of Figure 3.5. 47

3.7 Two-dimensional uniformly distributed dataset of 100 points with the 10 most outlying points (green dots) detected by two different metrics. 48

3.8 Distance between the CM and every datapoint in the uniformly distributed dataset of Figure 3.7. 48

3.9 Three-dimensional standard normally distributed dataset of 100 points. 49

3.10 Distance between the CM and every datapoint in the standard normally distributed dataset of Figure 3.9. 50

3.11 Transformed one-dimensional datasets of the ten-dimensional standard normally distributed dataset. 51

3.12 Probability density function of one-dimensional t-distribution for different degrees of freedom (ν). 55

3.13 Transformed one-dimensional datasets of the ten-dimensional t-distribution with 1 degree of freedom. 55

4.1 Probability that the left fringe point gets isolated in 4 and 5 random splits for a different number of trees. 60
4.2 Probability that the interior point gets isolated in 4 and 5 random splits for a different number of trees. 61
4.3 Variance of the number of random splits that is needed to isolate the left fringe point for a different number of trees, for multiple samples of Isolation Forest. . . 61
4.4 Variance of the number of random splits that is needed to isolate the interior point for a different number of trees, for multiple samples of Isolation Forest. . . 61
4.5 Log-log plot of the absolute difference between the numerical and theoretical variance for an increasing number of trees, for multiple samples of Isolation Forest. 62
4.6 Probability that the left fringe point gets isolated in 4 and 5 random splits with pruned trees. 62
4.7 Probability that the interior point gets isolated in 4 and 5 random splits with pruned trees. 63
4.8 Distribution of a standard normal distribution Y with 1000 datapoints created with 2 different random seeds. 68
4.9 Methods of distinguishing the top outlier and top inlier in all the 100 trees from the Isolation Forest method. 70

5.1 t-SNE visualization of the Elliptic dataset. 72
5.2 Confusion matrix of the original IF method (10000 trees) used on the Elliptic dataset. 73
5.3 Transformed one-dimensional datasets of the Elliptic dataset. 73
5.4 Confusion matrices of the four transformed datasets of the Elliptic dataset. . . . 74
5.5 Confusion matrices of the original IF method (10000 trees) used on the four transformed datasets of the Elliptic dataset. 74
5.6 Confusion matrices of the original IF method (1000 trees) used on the Elliptic dataset for different height limits of the trees. 75
5.7 Confusion matrices of the original IF method (1000 trees) used with two different scoring functions for two different height limits. 76
5.8 Confusion matrices of the Transformed IF method with the new scoring function given by (4.3) used on the four transformed datasets of the Elliptic dataset. . . . 77
5.9 Confusion matrices of the Transformed IF method with the new scoring function given by (4.4) with $i = 3$ used on the four transformed datasets of the Elliptic dataset. 77

List of Tables

3.1 Theoretical and numerical probabilities that the left fringe point becomes isolated in s random splits. . . . 38

3.2 Expectation and variance of the number of random splits that is needed to isolate the left fringe point. . . . 38

3.3 Theoretical and numerical probabilities that the left fringe point becomes isolated in s random splits. . . . 39

3.4 Expectation and variance of the number of random splits that is needed to isolate the left fringe point. . . . 39

3.5 Theoretical and numerical probabilities that the interior point becomes isolated in s random splits. . . . 39

3.6 Expectation and variance of the number of random splits that is needed to isolate the interior point. . . . 40

3.7 Theoretical and numerical probabilities that the interior point becomes isolated in s random splits. . . . 41

3.8 Expectation and variance of the number of random splits that is needed to isolate the interior point. . . . 41

3.9 Theoretical and numerical probabilities that the left fringe point in both directions becomes isolated in s random splits. . . . 42

3.10 Expectation and variance of the number of random splits that is needed to isolate the left fringe point in both directions. . . . 42

3.11 Theoretical and numerical probabilities that the left fringe point in both directions becomes isolated in s random splits. . . . 43

3.12 Expectation and variance of the number of random splits that is needed to isolate the left fringe point in both directions. . . . 43

3.13 Number of mistakes made in detecting the most outlying points for a dataset of 100 points coming from a two-dimensional dataset with four clear outliers. . . . 44

3.14 Number of mistakes made in detecting the most outlying points for a dataset of 100 points coming from a standard normal distribution. Between brackets the number of mistakes if also the order is taken in consideration. . . . 45

3.15 Number of mistakes made in detecting the most outlying points for a dataset of 100 points coming from a two-dimensional standard normal distribution. Between brackets the number of mistakes if also the order is taken in consideration. . . . 47

3.16 Number of mistakes made in detecting the most outlying points for a dataset of 100 points coming from a two-dimensional uniform distribution. Between brackets the number of mistakes if also the order is taken in consideration. . . . 49

3.17 Number of mistakes made in detecting the most outlying points for a dataset of 100 points coming from a three-dimensional standard normal distribution. Between brackets the number of mistakes if also the order is taken in consideration. 50

3.18 Number of mistakes made in detecting the most outlying points for a dataset of 100 points coming from a ten-dimensional standard normal distribution. Between brackets the number of mistakes if also the order is taken in consideration. 52
3.19 Theoretical and numerical probabilities that the right fringe point in the transformed dataset becomes isolated in s random splits. 53
3.20 Theoretical and numerical probabilities that the right fringe point in the transformed dataset becomes isolated in s random splits. 53
3.21 Theoretical and numerical probabilities that the right fringe point in the transformed dataset becomes isolated in s random splits. 54
3.22 Expectation of the number of random splits that is needed to isolate the right fringe point in the transformed dataset. 54
3.23 Number of mistakes made in detecting the most outlying points for a dataset of 100 points coming from a ten-dimensional t-distribution with 1 degree of freedom. Between brackets the number of mistakes if also the order is taken in consideration. 56
3.24 Values of i for which the outlier given by $(4, 4, ..., 4)$ is detected in a ten-dimensional normally distributed dataset for different values of correlation. 57

4.1 Computation times of two different implementations of the theoretical formulas of the interior points used on a standard normally distributed dataset (average of 100 runs). 60
4.2 Number of mistakes made on average, over multiple random datasets, in detecting the most outlying points for a dataset of 1000 points while using the code of [8] and the scoring function from [13]. Between brackets the number of mistakes if also the order is taken in consideration. 64
4.3 Number of mistakes made in detecting the most outlying points for a dataset of 1000 points with 980 points coming from a standard normal distribution, 10 points from a normal distribution with mean -15 and variance 1 and 10 points from a normal distribution with mean 15 and variance 1. Between brackets the number of mistakes if also the order is taken in consideration. 64
4.4 Number of mistakes made in detecting the most outlying points for a dataset of 1000 points with 980 points coming from a standard normal distribution, 10 points from a normal distribution with mean -15 and variance 1 and 10 points from a normal distribution with mean 15 and variance 1. Between brackets the number of mistakes if also the order is taken in consideration. 65
4.5 Number of mistakes made in detecting the most outlying points for a dataset of 1000 points with 980 points coming from a standard normal distribution, 10 points from a normal distribution with mean -15 and variance 1 and 10 points from a normal distribution with mean 15 and variance 1. Between brackets the number of mistakes if also the order is taken in consideration. 66
4.6 Number of mistakes made in detecting the most outlying points for a dataset of 1000 points coming from a mixture distribution consisting of 3 different normal distributions. Between brackets the number of mistakes if also the order is taken in consideration. 67

4.7 Number of mistakes made on average, over multiple samples for Isolation Forest and over multiple random datasets, in detecting the most outlying points for a dataset of 1000 points. Between the brackets is the number of mistakes if also the order is taken in consideration. 68
4.8 Number of mistakes made on average, over multiple samples for Isolation Forest and over multiple random datasets, in detecting the most outlying points for a dataset of 1000 points with 980 points coming from a standard normal distribution, 10 points from a normal distribution with mean -15 and variance 1 and 10 points from a normal distribution with mean 15 and variance 1. Between brackets the number of mistakes if also the order is taken in consideration. 69
4.9 Variance of the scores of the 20 most outlying points for a dataset of 1000 points while using the code from [8] with 100 trees. 69

Nomenclature

Abbreviations

CM	Center of Mass
DB	Distance-Based
FN	False Negative
FP	False Positive
IF	Isolation Forest
IN	Interior
kNN	k-nearest Neighbors
LF	Left Fringe
LOF	Local Outlier Factor
PDF	Probability Density Function
RF	Right Fringe
RS	Random Splits
t-SNE	t-distributed Stochastic Neighbor Embedding
TIF	Transformed Isolation Forest
TN	True Negative
TP	True Positive

Notation

α	Confidence coefficient
α_n	Confidence coefficient after being corrected for multiple comparison tests
$\hat{\mu}_n$	Sample mean
$\hat{\sigma}_n$	Sample standard deviation
$\overline{\mathbf{x}}_n$	Sample mean vector
ψ	Sub-sampling size
$\mathbf{V}_n$	Sample covariance matrix
$\vec{r}_i$	Coordinates of datapoint i

$c(n)$	Average path length of an isolation tree that is built with n datapoints
D	Depth of a datapoint
d	Number of dimensions in a dataset
D_k	The contour of depth k
$E(h(x))$	Expected path length of instance x
$E(RS)$	Expectation of the number of random splits needed to isolate a datapoint
F	Target distribution
$h(x)$	Path length of instance x
k	Number of neighbours of a datapoint
l	Height limit of a tree
$lrd(O)$	Local reachability density of an object O
M_i	Mahalanobis distance for a multivariate datapoint i
m_i	Mass of a datapoint i
n	Number of observations in a dataset
O	Object
p	Split value of Isolation Forest
$P_{x_j}(RS = s)$	Probability that the datapoint x_j gets isolated in s random splits
Q_i	Feature value i
$s(x)$	Outlier score of instance x
T	Node of an isolation tree
V	Real vector space
V_D	Volume of a boll with center O and radius D
X	Dataset
x	Observation/instance in a dataset
z_q	q quintile of the $N(0, 1)$ distribution

1. Introduction

There may exist certain data patterns that have different characteristics than normal instances. We call these anomalies or outliers [2, 3, 10, 25]. Nowadays, we use huge amounts of data because of the fast technological development. That is why the detection of anomalies has become more relevant, for example, in the financial world to detect fraudulent transactions in order to stop money laundering. The majority of the well-known model-based methods use the same approach to detect these outliers. First, a profile of normal instances or inliers is constructed and then instances that do not conform to this profile are identified. This approach has two major disadvantages: namely, creating a profile of the normal instances gives us a method that is not meant for detecting outliers. It focuses mostly on mapping the normal data. Another weak point of these methods is the high computational complexity. That is why they can only process low dimensional data and small data sizes [4, 8, 18].

This thesis investigates a data-driven anomaly detection method that is also commonly used to detect fraudulent transactions. This method is called Isolation Forest (IF) [8, 18]. It is a very popular method and is applied in many different applications. Moreover, it does not suffer from the caveats that most well-known model-based anomaly detection methods have. It does not create a profile of the normal instances, but it isolates the anomalies directly. The special aspects of outliers ensure that they are more sensitive to isolation than the normal points. The advantages of this method are its robust performance and its low computational complexity. The algorithm is very easy to understand and implement, due to the small number of parameters that need to be optimized.

The theoretical properties of Isolation Forest are, however, not well-known, because of the lack of fundamental research in this field. It is known widely that it performs well, but it is not clearly understood why. Interesting theoretical results like in [5, 10, 17, 20, 24] will help us getting a better understanding of the method. It is based on random splits which are used to isolate points. This way of detecting anomalies performs well, because outlying points are isolated more easily from the rest of the points than inliers. The reason for this is that outliers are further away from other points than inliers, since they are different from normal instances. In other words, it takes fewer random splits to isolate an outlying point than an inlier, but it is not known what the probabilities are that a certain point gets isolated in a certain number of random splits. It is also unknown what the expected number of random splits is in which an outlier or inlier gets isolated. Hence, there is much behind this method that is still unknown and that is why this is the main direction the thesis will go. The main goal is to broaden such studies exploring the theoretical properties of Isolation Forest.

Furthermore, in order to exploit the theoretical results obtained in this thesis, new outlier detection methods are developed based on a projection that first transforms multi-dimensional datasets into one-dimensional ones. With these new one-dimensional datasets, the outliers can be detected in the same way as the original IF method. Therefore, we call these methods Transformed Isolation Forest. The first new transformation method calculates the distance of every datapoint in a set to the center of mass of the whole dataset. All these distances conform a new one-dimensional dataset. This method uses three different metrics to calculate the distance of every point to the center of mass. Finally, a second transformation is developed based on the k-nearest neighbours distance. The k-nearest neighbours distance of a datapoint is then the

mean of the distances to its k nearest neigbours. With all the distances of every datapoint in the set, we again obtain a one-dimensional dataset.

Once the theory is developed, the findings can be checked experimentally with the algorithm itself. This will be done in this thesis by making use of the Scikit-learn implementation and also the implementation developed in [8]. For different datasets, theory will be developed and then checked with the numerical values by running the Isolation Forest algorithm on the same datasets. Moreover, the new transformation methods will be tested rigorously for multiple random datasets. The number of outliers detected by the new methods will be compared with the number of outliers detected by the original IF method.

Finally, Isolation Forest will be compared with other common anomaly detection algorithms. The two methods that are used for comparison are K-means clustering [25] and the Local Outlier Factor algorithm (LOF) [15, 25]. The concepts of these methods are different from the IF method: K-means clustering detects anomalies based on the cluster membership, distance from other clusters and the size of the closest clusters [8] and LOF is based on local density estimates. Hence, we present a broad comparison between different concepts of anomaly detection. The goal is to better understand the advantages and disadvantages of these methods with respect to each other.

1.1 Research Questions

After motivating the topic of this thesis, the next step is to state the research questions. These are stated below:

1. **Can we relate the concept of Isolation Forest with density- and distance-based outlier detection methods?**
2. **How can we mathematically define the concept of isolation? And can we use this new framework to develop better outlier detection methods?**
3. **What is the theory behind the Isolation Forest method?**
4. **Confirm the theoretical findings with numerical experiments.**

Therefore, this thesis's aim is to introduce new theoretical results of the Isolation Forest method that have not yet been explored in the literature before.

1.2 Thesis Structure

In Chapter 2, first a summary of the necessary background literature and related work is given. First, the concept of an outlier is introduced and explained. However, there is no single and clear definition of an outlier. Multiple definitions are stated in the literature and, for reasons of space, only the most relevant ones will be mentioned. After this has been stated, multiple anomaly detection algorithms, which are based on these definitions of an outlier, are introduced and elaborated on. Special emphasis is given to the description of the Isolation Forest method, because it is the main topic of this thesis. After that, a new method is described to transform multi-dimensional datasets to one-dimensional datasets. This transformation ensures that the

theoretical formulas for the original Isolation Forest method to be developed in Chapter 3 for the one-dimensional case can be used. The new transformed datasets are namely one-dimensional and therefore can be used in the same way as the original IF method to detect the outliers with the theoretical formulas developed in Chapter 3. As possible transformations, we consider first the distance to the center of mass, and finally the kNN-distance.

In Chapter 3, new theory behind the existing Isolation Forest is developed. First, the dataset is split into fringe points and interior points and the definition of both is stated. This is because the theory is slightly different for these types of observations. The theory for the fringe points is the starting point before moving on to the interior points of the dataset. Moreover, theory is developed for the two-dimensional case. The theory behind the Isolation Forest method is now developed and stated and thus the next step is checking it experimentally with the numerical results. This is done by validating the theoretical formulas with the numerical Isolation Forest. After this validation, the methods that transform the multi-dimensional datasets into one-dimensional ones are tested numerically. This is done by looking at the number of mistakes that these methods make in detecting the most outlying points of multiple datasets.

Moreover, classical components of the Isolation Forest method are tested numerically in Chapter 4, like the impact of pruning the isolation trees or the number of trees. The computing times of the theoretical formulas for the interior points in the one-dimensional case are also tested. Furthermore, different scoring functions and the performances of three outlier detection methods, including the Isolation Forest method, are compared. The results of these performance tests are summarized in the last section of Chapter 4.

In Chapter 5, a real-world financial dataset is used to test how the methods perform on such a dataset. It is a labelled dataset and thus we can compare the outliers detected by the original IF method and the Transformed IF method with the transactions that are labelled as illicit beforehand. This is done at the beginning of this chapter. Moreover, the impact of pruning on these results is also tested and summarized. Finally, different scoring functions are tested on this real-world financial dataset.

The final chapter, Chapter 6, provides a conclusion of the results derived from this thesis, and issues recommendations into promising further research directions.

2. Outlier Detection

In this chapter, an overview of the relevant background information and related work is provided. Therefore, multiple existing outlier detecting methods are mentioned and explained. Moreover, a new method is introduced. In the first section, the notion of an outlier is introduced and multiple definitions of an outlier with the outlier detection methods that are based on them are given. In the final subsection of the first section, a method to measure the quality and effectiveness of our outlier detection methods is introduced, namely the confusion matrix. In the second section, the main topic of this thesis, the Isolation Forest method, is introduced. Finally, the Transformed Isolation Forest (TIF) is developed and explained in the last section of this chapter. This method uses different distance metrics to transform multi-dimensional datasets into one-dimensional ones. With these one-dimensional datasets, the outliers can be detected in the same way as the original Isolation Forest method.

2.1 Existing Methods

There is no single, generally accepted, formal definition of an outlier. Therefore, a collection of multiple different definitions of an outlier is created. The first one is the definition of an outlier by Hawkins [2, 10, 15] and it definitely captures the spirit.

Definition 2.1 (Hawkins' definition of an outlier)**.** *An outlier is an observation that deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism.*

Notice that this is not a mathematical definition. Now multiple definitions that are more mathematically grounded are provided together with the methods based on these definitions. The definitions and methods are separated based on distance, non-parametric density, parametric density, cluster, depth and isolation.

2.1.1 Distance-Based Methods

First, the distance-based definitions and methods are described. There is the following distance based definition of an outlier stated by [10].

Definition 2.2 (Distance-Based (DB) outlier)**.** *An object O in a dataset X is a $DB(p, D)$-outlier if at least a fraction p of the objects in X lies at a distance greater than D from O.*

This definition is similar to Hawkins' definition, but the foundation behind it is more mathematical. The advantage of this definition is that it can be used in the case that the observed data distribution does not fit any standard distribution or when there is no discordancy test developed to check if a datapoint is an outlier. A discordancy test verifies whether an object is significantly larger (or smaller) in relation to the distribution F. Another advantage of Definition 2.2 is that it is well-defined for d-dimensional datasets for any value of d. If the distribution of the data is known, then there are criteria for being an outlier. This is different for every distribution. Therefore, if an object is an outlier based on a specific discordancy test, then it is also a distance-based outlier for specific values p and D. This is given by the following definition [10].

Definition 2.3. *An object O is an outlier according to a specific discordancy test if and only if there exist values p_0 and D_0 such that O is a $DB(p_0, D_0)$-outlier.*

In order to use this definition, the underlying data distribution is required.

In the univariate case, the detection of outliers is done by looking at each variable independently, but in many cases this will not lead to the correct result if there is a relationship between the different variables. In that case, multivariate outlier definitions are more useful. One of them is the Mahalanobis distance [3], which is given by:

Definition 2.4. *(Mahalanobis distance) The Mahalanobis distance for each multivariate datapoint i,$i = 1, ..., n$ in a d-dimensional dataset, is denoted by M_i and given by*

$$M_i = \left(\sum_{i=1}^{n} (\boldsymbol{x}_i - \overline{\boldsymbol{x}}_n)^T \, \boldsymbol{V}_n^{-1} (\boldsymbol{x}_i - \overline{\boldsymbol{x}}_n) \right)^{1/2}, \tag{2.1}$$

with the sample mean vector denoted by $\overline{\mathbf{x}}_n$ and the sample covariance matrix denoted by $\mathbf{V}_n$. The instances with a large Mahalanobis distance are marked as outliers. If the assumption is made that the data is generated from a multivariate normal distribution, then the squared Mahalanobis distances are approximately chi-squared distributed with d degrees of freedom. Hence, then it is stated that an instance is a multivariate outlier if its squared robust Mahalanobis distance exceeds the value of the quantile 0.975 of the distribution χ^2_d.

Distance-based definitions of an outlier can also be interpreted as density-based definitions. For example, Definition 2.2 can also be understood as local density estimate centered around points. This is explained in Subsection 2.1.2. Distance-based methods are based on evaluating distances between datapoints. Anomaly scores are often based on neighbour distances. It is clear that anomalies have larger separation distances compared to their neighbours. One of the advantages of these methods is that this separation can be easily visualized.

One of the distance-based methods that is stated in the literature makes use of Definition 2.2 without the threshold p. This definition can be used to give every object $O \in X$ with distance$(x, O) > D$ for data objects $x \in X$ an outlier score in the range $[0, 1]$ [25]:

$$score(O) = \frac{|\{x \in X : \text{distance}(x, O) > D\}|}{|X|}. \tag{2.2}$$

Therefore, parameter p of Definition 2.2 can be seen as fixing a decision threshold for the score. Scores that are larger than the threshold label the object O as an outlier.

2.1.2 Non-Parametric Density-Based Methods

Non-parametric density-based methods investigate specific regions of a dataset, determining the local density within these regions with the number of points. Hence, these methods partition the dataspace. Outliers are in this situation the points in the regions with lower local densities. These methods do not look at the specific density distribution.

Definition 2.2 can also be used for a local density estimate centered at a point $O \in X$. This is given by [25]:

$$score(O) = \frac{|\{x \in X : \text{distance}(x, O) \leq D\}|}{V_D}, \tag{2.3}$$

with V_D the volume of the D-range (i.e. a boll with center O and radius D). The lower this estimated density is, the more O is considered to be an outlier. Hence, distance-based definitions of an outlier can be used to create density-based methods for detecting outliers.

We can also derive another density-based score via the k-nearest neighbors algorithm (kNN) [25] by using the sum of distances to all points within the set of k-nearest neighbours as an outlier degree:

$$score(O) = \frac{k}{k\text{-dist}(O)}, \tag{2.4}$$

with the distance of O to its kth nearest neighbour denoted as k-dist(O). Hence the lower the score, the lower the estimated density is and the more the object O is considered as an outlier.

The advantages of the methods above are the following: all of them make use of local density estimates, but do not look at the density distribution itself. Thus, even if the specific distribution of the dataset is not known, these methods would still work. These methods are global methods, because the outlierness of an object is evaluated in comparison to all other objects. Disadvantages of these methods are that all of them have difficulties with handling data that contain clusters of different local density. Such datasets are handled better by the Local Outlier Factor algorithm (LOF) [15, 25]. The LOF method finds local outliers instead of global outliers that are found by the methods above.

The LOF method compares the density of each object O of a dataset X with the density of the k-nearest neighbours of O. The density of an object O is estimated by a value called the local reachability density and it is defined as:

$$lrd(O) := 1/\frac{\sum_{o \in kNN(O)} reach\text{-}dist_k(O, o)}{|kNN(O)|}, \tag{2.5}$$

with $|kNN(O)|$ the number of k nearest neighbours of the object O. In this formula, there is the term $reach\text{-}dist_k(O, o)$, which is the reachability distance and it is defined as:

$$reach\text{-}dist_k(O, o) = \max\{k\text{-}dist(o), d(O, o)\}, \tag{2.6}$$

with $k\text{-}dist(o)$ the distance of o to its kth nearest neighbour and $d(o, O)$ the distance between the objects o and O. The final score of the LOF algorithm is defined as:

$$LOF_k(O) = \frac{1}{|kNN(O)|} \sum_{o \in kNN(O)} \frac{lrd_k(o)}{lrd_k(O)}. \tag{2.7}$$

This can be interpreted as follows: a LOF value around 1 means that the object is located within a cluster, so in a region of homogeneous density and is thus not an outlier. The LOF value is the highest if the density estimate of an object is small relative to the estimates of its nearest neighbours. Therefore, high LOF values are given to the most outlying points.

2.1.3 Parametric Density-Based Methods

There are also definitions and methods that look at the specific density distribution. These are parametric density-based methods.

In the chapter written by Ben-Gal [3], there is a definition for being an outlier in the univariate parametric case where the data is normally distributed, but with a small number of observations that are randomly sampled from distributions different from the target distribution. It is the following definition [6]:

Definition 2.5. *For any confidence coefficient α, $0 < \alpha < 1$, the α-outlier region of the $N(\mu, \sigma^2)$ is defined by:*

$$out(\alpha, \mu, \sigma^2) = \{x : |x - \mu| > z_{1-\alpha/2}\sigma\}, \tag{2.8}$$

with z_q the q quintile of the $N(0, 1)$ distribution. By this definition, a datapoint x is classified as an α-outlier with respect to the target distribution F (which in this case is $N(\mu, \sigma^2)$) if $x \in \text{out}(\alpha, \mu, \sigma^2)$. This definition can be extended to different target distributions than the normal distribution, namely any unimodal symmetric distribution with positive density function, including the multivariate case. A disadvantage of this definition is that it does not identify which of the observations is sampled from the distributions that differs from the target distribution, but it only indicates the points that lie in the outlier region.

In Section 3.1 of [3], it is stated that there are single-step and sequential procedures for outlier detection. In single-step outlier detection procedures, all outliers are detected at once while in sequential outlier detection procedures at each step one instance is tested for being an outlier.

Definition 2.6. *If we look at the single-step procedures, there is another definition for detecting outliers, which is given by*

$$out(\alpha_n, \hat{\mu}_n, \hat{\sigma}_n^2) = \{x : |x - \hat{\mu}_n| > g(n, \alpha_n)\hat{\sigma}_n\}, \tag{2.9}$$

with n the size of the sample; $\hat{\mu}_n$ and $\hat{\sigma}_n$ the estimated mean and standard deviation of the target distribution based on the sample respectively. In this definition we also have α_n, which is the confidence coefficient after being corrected for multiple comparison tests. The limits of the outlier regions are defined by $g(n, \alpha_n)$. Thus, it defines the number of standard deviations that a datapoint should differ from the estimated mean to be an outlier.

Traditionally, $\hat{\mu}_n$ and $\hat{\sigma}_n$ are estimated respectively by the sample mean and sample standard deviation. Since these estimates are highly affected by outliers, procedures often replace them by other, more robust, estimates. John Tukey [3] introduced better estimators for $\hat{\mu}_n$ and $\hat{\sigma}_n$, namely $\hat{\mu}_n = (Q_1 + Q_3)/2$ and $\hat{\sigma}_n = Q_3 - Q_1$ with Q_1 and Q_3 the first and third quartiles. The first quartile Q_1 is the value that is greater or equal than $\frac{1}{4}$ of the dataset and the third quartile Q_3 is the value that is greater or equal than $\frac{3}{4}$ of the dataset. One of the definitions, based on this [22], is given by:

Definition 2.7. *If Q_1 and Q_3 are the lower and upper quartiles, then an outlier is an instance that falls outside the following range:*

$$[Q_1 - k(Q_3 - Q_1), Q_3 + k(Q_3 - Q_1)],$$

for some non-negative constant k.

John Tukey proposed this test where $k = 1.5$ indicated a weak outlier and $k = 3$ a strong outlier.

In the case of Definition 2.6, instead of a single observation, a set of n instances is used for the test. While a given α-value may be appropriate to decide whether a single observation lies in

the outlier region, this is not the case for a set of several comparisons. In order to avoid spurious positives, the α-value needs to be lowered to account for the number of performed comparisons. This can be done by the Bonferroni's correction [3] which sets the α-value for each comparison equal to α/n. Another correction method sets $\alpha_n = 1 - (1 - \alpha)^{1/n}$. The value $g(n, \alpha_n)$ is then obtained by numerical procedures.

A major disadvantage of all of these definitions is the assumption of a specific data distribution which is required in order to apply a specific test. This can lead to a considerable model error when such assumptions do not reflect the true distribution.

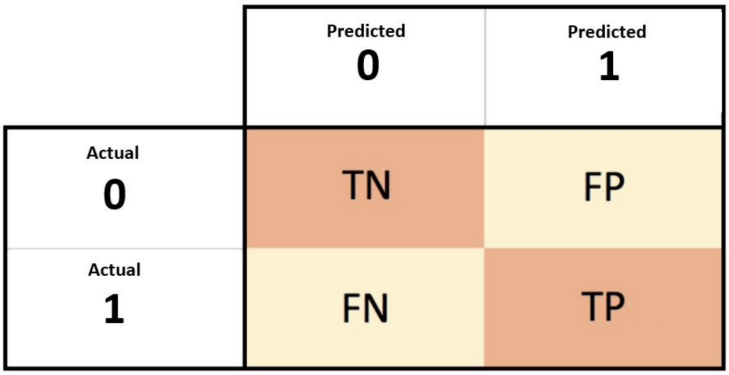
2.1.4 Cluster-Based Methods

Cluster-based methods partition the datapoints instead of the dataspace. Anomaly scores are based on cluster membership, distance from other clusters and the size of the closest clusters [8]. Thus, these methods classify a datapoint as an outlier if it is not a member of a cluster of points. The most popular clustering algorithm is K-means clustering [25]. This method calculates for each datapoint in a dataset the distance to the center of the nearest cluster. The distance metric that can be used is the Euclidean distance, for example. The larger this distance, the more outlying a point will be.

2.1.5 Measures of Quality

In this subsection, we will introduce a measure to assess the quality and effectiveness of our outlier detection methods. This measure is called the confusion matrix. It is a performance measurement for algorithms, typically supervised ones. Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class. It is typically used in the case where we only have 2 classes and thus a binary classification problem. In our case that would be the following 2 classes: outlier and inlier. The four outcomes can be formulated in a 2×2 confusion matrix, which can be seen in Figure 2.1.

Figure 2.1: 2×2 confusion matrix of a binary classification problem. Figure is taken from [14].



In Figure 2.1, we see four different outcomes, namely TN, FP, FN and TP. We will explain what these four outcomes mean: True Negative (TN) is the outcome where the algorithm predicted negative and the true outcome is also negative, False Positive (FP) is the outcome where the algorithm predicted positive and the true outcome is negative, False Negative (FN) is the outcome where the algorithm predicted negative and the true outcome is positive and True

Positive (TP) is the outcome where the algorithm predicted positive and the true outcome is also positive.

2.2 Isolation Forest

The next definition of an outlier that will be discussed is based on the notion of isolation. This is related to the concept of depth, which we will introduce in the next subsection.

2.2.1 Depth-Based Methods

In the univariate case, there is the following definition for the depth of a point [17]:

Definition 2.8. *The depth of a point z relative to a one-dimensional dataset, $X = \{x_1, x_2, ..., x_n\}$, is defined as the minimum of the number of datapoints to the left of z, and the number of datapoints to the right of z:*

$$depth(z; X) = min(\#\{i; x_i \leq z\}, \#\{i; x_i \geq z\}). \tag{2.10}$$

For a multivariate dataset, John Tukey introduced the notion of "half-space depth of a point" [17]. This is defined as:

Definition 2.9. *The half-space depth of a point $z \in \mathbb{R}^d$ relative to a d-dimensional data cloud $X = \{x_1, x_2, ..., x_n\}$ is defined as the smallest depth of z in any one-dimensional projection of the dataset.*

In [17], it is also stated that if the dataset contains m outliers, only the m outermost depth contours can be strongly affected by them. These depth contours are given by $D_k = \{x \in \mathbb{R}^d; \text{depth}(x; X) \geq k\}$. Hence, the convex hull of the dataset will be affected by outliers, whereas sufficiently deep contours remain robust.

In the one-dimensional, univariate case, the points of a dataset can be ranked. The extreme points are given depth one. The values with second lowest and second highest rank are given depth two and so on. The median will then be the point with the highest depth. In higher dimensions, the depth of a point gives an indication of how deep the point is inside the cloud. In [17], they use the following definition in higher dimensions: for a dataset $X \subset \mathbb{R}^d$, we have the set of depth contours. Hence, the interior points of D_k have depth of at least k and the points at the boundary have depth of exactly k. Moreover, D_k is introduced as the contour of depth k. The outermost contour D_1 is the convex hull of the dataset X.

2.2.2 Basics of Isolation Forest

The definitions of Section 2.1 are based on the normal instances, while the concept of isolation looks directly at the outliers. In this case, an instance will be separated or isolated from the rest of the instances. Hence, the definition of an isolation-based outlier [18] is given by

Definition 2.10. *An isolation-based outlier is an observation that is separated from the rest of the instances.*

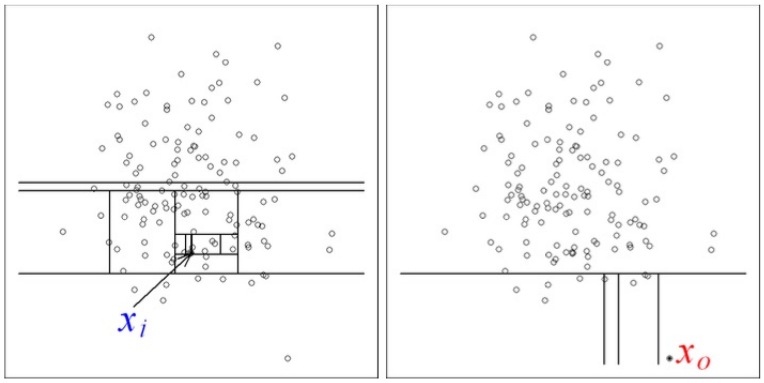
Anomalies are points that deviate from other points, and also there are fewer outliers than inliers, therefore they are more susceptible to isolation. The Isolation Forest algorithm, which

is the main topic of this thesis, is based on the notion of isolation. The Isolation Forest method does not create a profile of the normal instances, but instead it isolates the anomalies directly. A tree structure can be used to isolate the outliers and the method that uses this tree structure is called Isolation Forest (IF) [4, 8, 18]. Such a tree is called an Isolation Tree. An isolation tree is a proper binary tree. That means that each node in the tree has exactly zero or two daughter nodes. Here the assumption is made that T is the node of an isolation tree. External nodes have no daughter nodes and internal nodes have exactly two daughter nodes (T_L, T_R) with one test.

Suppose there is a dataset X of n instances from a d-variate distribution and IF is used on it, then X is recursively divided by randomly selecting the feature $Q_i \in Q_1, ..., Q_d$ with equal probability from the set of features and a split value p which gives an isolation tree. After the feature Q_i is chosen, the value belonging to this specific feature $X(Q_i)$ is compared with the split value p for every datapoint. If $X(Q_i) < p$, the datapoint will go to T_L and otherwise it will go to T_R. This is done until the tree reaches a height limit[1] or there is only one datapoint from the set X left or all data in X have the same values. This is the first stage of the model and it is called the training stage. In this stage, the isolation trees are constructed from a sub-sample of the data. In Figure 2.3, an example of an isolation tree created from a small dataset can be seen.

In isolation trees, instances are partitioned recursively until all of them are isolated. Anomalies are isolated earlier in the trees than the normal instances, because of their distinguishable attribute-values. To illustrate that outliers are more susceptible to isolation than inliers, an example is given in Figure 2.2.

Figure 2.2: Given a Gaussian distribution (135 points), (a) a normal point x_i requires for example twelve random partitions to be isolated; (b) an anomaly requires only four partitions to be isolated. Figure taken from [18].

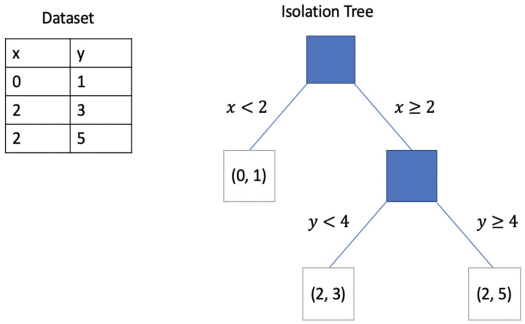


(a) Isolating x_i (b) Isolating x_o

The second stage is the evaluation stage, where an anomaly score s is derived from the expected path length $E(h(x))$ for each instance. The path length $h(x)$ of a datapoint x is represented by the number of edges from the root node to a terminating node as this point x passes through the isolation tree. The expected path length is derived by passing all the datapoints through each isolation tree in the isolation forest. It is then the average value of $h(x)$ from all the isolation trees that were built. However, the trees have a height limit and thus it can happen that the tree is not fully grown. These are early terminated nodes, which means that these nodes will contain more than one datapoint. If this is the case, an extra constant $c(n)$ is added to the path

[1]The trees are cut off at the pre-set height limit to reduce the computation time of the IF algorithm.

Figure 2.3: An example of an isolation tree created from a small dataset. Figure is taken from [12].



length of the instance in the early terminated node. This $c(n)$ is the average path length of an isolation tree that is built with n datapoints. Finally, the anomaly score of a datapoint x for a dataset of size n is given by:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}, \tag{2.11}$$

with $E(h(x))$ the average value of $h(x)$ from all the isolation trees that were built. We see the following:

$$\begin{aligned}
&\text{When } E(h(x)) \to c(n) : s(x, n) = 2^{-\frac{c(n)}{c(n)}} = 2^{-1} = 0.5,\\
&\text{When } E(h(x)) \to 0 : s(x, n) = 2^{-\frac{0}{c(n)}} = 2^{0} = 1,\\
&\text{When } E(h(x)) \to n-1 : s(x, n) = 2^{-\frac{n-1}{c(n)}} = 2^{-\frac{1}{2} \cdot \frac{n-1}{\ln n - 1 + 0.5772 - 1}} \to 0 \text{ as } n \to \infty.
\end{aligned}$$

This score s can be used to identify outliers. If $s(x)$ is close to 1, then x is an outlier. If $s(x)$ is much smaller than 0.5, then x is normal observation. If $s(x)$ is around 0.5 for all instances, then the set of instances does not contain clear outliers.

It is clear that anomalies will be isolated closer to the root of the tree than inliers. IF uses multiple isolation trees for a given dataset to isolate the anomalies. The input of this method only consists of two variables: the number of trees to build and the sub-sampling size. The sub-sampling size controls the training size of the algorithm. A sample of the overall data is taken randomly and used to construct an isolation tree. Two major advantages of this method are that the detection performance converges quickly with a very small number of trees and it only requires a small sub-sampling size to achieve high detection performance with high efficiency. Better isolation trees are built from small sample sizes, because the swamping and masking effects are reduced. Swamping happens when the method wrongly labels normal instances as outliers. Masking is the case if there are too many outliers. Another characteristic from Isolation Forest is that it does not need additional measures to detect the outliers. This reduces the computational cost. It can also easily handle large data sizes. Moreover, it can be explained why a datapoint is an outlier in a certain dataset by looking at the specific paths in the isolation trees and the different features used at every split.

2.3 Transformed Isolation Forest

In this section, a new method is presented to detect the most outlying points of a dataset. The method we will discuss is based on a projection that first transforms a multi-dimensional dataset into a one-dimensional dataset. This one-dimensional dataset can then be used to detect the outliers in the same way as the original IF method. The first transformation that is introduced is based on the distance of every point to the center of mass of the dataset. This transformation will be discussed in the next subsection.

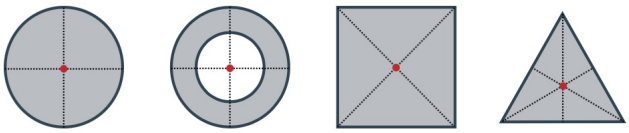
2.3.1 Center of Mass

Before we explain how this transformation works in detail, we first define the center of mass. This is defined as follows [1, 21]:

Definition 2.11. *(Center of Mass) An object's* ***Center of Mass (CM)*** *is the (hypothetical) point where the total mass of an object can be treated as a point mass.*

Hence, this is the point where all the masses of a set of points are concentrated. It is a hypothetical point, so it does not have to be contained in the dataset itself. For objects or datasets with uniform density, the center of mass will always be the geometric center. This is called the centroid of the object. Non-uniform datasets contain points which are denser than other points. Therefore, the CM will not be the centroid of the dataset in this case. In Figure 2.4, the center of mass of multiple two-dimensional geometric shapes can be seen.

Figure 2.4: Center of mass for some simple geometric shapes (red dots). Figure is taken from [1].



The center of mass can be calculated for all datasets. The formula of the CM is given by (2.12).

$$CM = \frac{1}{M}\sum_{i=1}^{M} m_i\vec{r}_i, \tag{2.12}$$

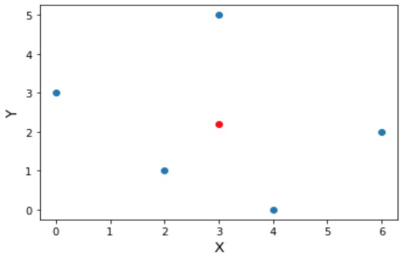
with M the number of points of the dataset, m_i the mass of each individual point and $\vec{r}_i$ the coordinates of each individual point.

The coordinates of the 5 points in this dataset are equal to $\{(3,0),(2,1),(3,5),(4,0),(6,2)\}$ with the mass of every point equal to 1. The calculation of the CM for this dataset is given by (2.13).

$$CM = \frac{1}{M}\sum_{i=1}^{M} m_i\vec{r}_i = \frac{1}{5}\sum_{i=1}^{5}\vec{r}_i = \frac{1}{5}(\sum_{i=1}^{5} x_i, \sum_{i=1}^{5} y_i) = \frac{1}{5}(15,11) = (3,2.2) \tag{2.13}$$

Hence, the CM of this dataset is equal to the point $(3, 2.2)$, which is not a point that is contained in the dataset itself. This can be seen in Figure 2.5.

Figure 2.5: Center of mass of a small 2D dataset (red dot).



2.3.2 Distance Functions

The idea is to look at the distance of each point to the CM of the whole dataset. This will transform a multi-dimensional dataset into a one-dimensional dataset. The aim is thus to focus on a one-dimensional set-up for a multi-dimensional dataset. For this transformation a proper distance function is needed. In this subsection, multiple distance functions are described.

For the distance function, we use a norm. This is a function from a real (in our case) or complex vector space to the non-negative real numbers that gives information like the distance from the origin. In our case, we have real vector spaces, which are the one- or multidimensional datasets. In this case, the norm on a real vector space V is a function $|| \cdot || : V \to \mathbb{R}$ with the following well-known properties [11]:

$$||x|| > 0 \text{ for any nonzero } x \in V, \qquad \textbf{(positive)}$$

$$||ax|| = |a|||x|| \text{ for any } a \in R \text{ and } x \in V, \qquad \textbf{(homogeneous)}$$

$$||x + y|| \leq ||x|| + ||y|| \text{ for any } x, y \in V. \qquad \textbf{(triangle inequality)}$$

Hence, a norm is a function $|| \cdot || : \mathbb{R}^n \to \mathbb{R}$ and thus suitable for the envisioned projection. Common examples of norms on $\mathbb{R}^n$ are the l_p norms, with $1 \leq p \leq \infty$, defined by:

$$l_p(x) = \Big(\sum_{i=1}^{n} |x_i|^p \Big)^{\frac{1}{p}} \qquad \text{if} \qquad 1 \leq p < \infty, \tag{2.14}$$

$$l_p(x) = \max_{1 \leq i \leq n} |x_i| \qquad \text{if} \qquad p = \infty, \tag{2.15}$$

for any $x = (x_1, ..., x_n)^t \in \mathbb{R}^n$. If $0 < p < 1$, the triangle inequality is not satisfied and thus not a norm.

The most well-known norm is the Euclidean distance [11], which is the 2-norm. This is often used to calculate the distance between two points in space. It is defined by (2.16).

$$l_2(x) = ||x||_2 = \Big(\sum_{i=1}^{n} |x_i|^2 \Big)^{\frac{1}{2}} = \sqrt{\Big(\sum_{i=1}^{n} x_i^2 \Big)} = \sqrt{x_1^2 + ...x_n^2}. \tag{2.16}$$

Another common norm is the 1-norm, which is the taxicab distance [11]. It is defined by (2.17).

$$l_1(x) = ||x||_1 = \sum_{i=1}^{n} |x_i| = |x_1| + ...|x_n|. \tag{2.17}$$

It is called the taxicab distance, because it measures the distance for a taxicab to drive from $(0, 0)$ to (x, y). The l_2-norm measures the straight line distance between the two points.

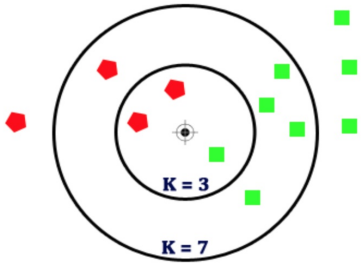
Our goal is to find all the outliers preferably with the correct ordering in a dataset, by making use of these norms. We expect the l_2-norm to be a better choice for this experiment, because it squares the values and thus increases the cost of outliers exponentially. Therefore, we expect this norm to perform better than the l_1-norm for the specific experiments.

2.3.3 Local Transformation (kNN-distance)

In this subsection, we also discuss a transformation. This will transform a multi-dimensional dataset into a one-dimensional one. The transformation is different from the transformation discussed in Subsection 2.3.1, because it uses a local measure of distance instead of a global one.

The k-nearest neighbours distance (kNN-distance) of every point is used for the transformation, already introduced in Subsection 2.1.2. The kNN-distance of a point is derived via the k-nearest neighbors algorithm [25] by looking at the distance of every datapoint to its k nearest neighbours. The kNN-distance of a datapoint is then the mean of the distances to its k nearest neighbours. If we calculate the kNN-distance of every datapoint in a dataset, we again get a one-dimensional dataset. In Figure 2.6, we see how kNN works for different numbers of neighbours.

Figure 2.6: kNN for different numbers of neighbours. Figure is taken from [23].



In Figure 2.6, we see kNN used for a classification problem. We see some pentagons in red and squares in green. The datapoint in the middle needs to be classified as a square or a pentagon. If $k = 3$, we look at the three nearest neighbours of this new point and that are 2 pentagons and 1 square. In this case, the new datapoint will be classified as a pentagon. However, if $k = 7$, we look at the seven nearest neighbours of this new point and then the majority of the nearest neighbours are squares and thus it will be classified as a square. Classification is one of the most important applications of this algorithm, but we won't use it for classification. We will only use the distances to the k nearest neighbours. What is interesting to see in this figure is how the algorithm finds the nearest neighbours of a point. This is done with circles around the point considered.

Unlike the transformation based on the distance to the CM, this new transformation is a local transformation. It considers the neighbourhood of each datapoint separately, because every datapoint has other points that are the k nearest neighbours. The transformations based on the distance to the CM are global transformations, because the CM is applicable to the whole dataset. Therefore, the distance of every point to the same (hypothetical) point is considered.

However, the TIF in combination with this local transformation does find the global outliers. The outlierness of a point is evaluated in direct comparison to all other points, because all local kNN-distances are compared with each other on a global scale. Every datapoint has different nearest neighbours, but the kNN-distances of every point in a dataset together are a new one-

dimensional dataset. By comparing these local kNN-distances with each other in a new dataset, we get a new global comparison.

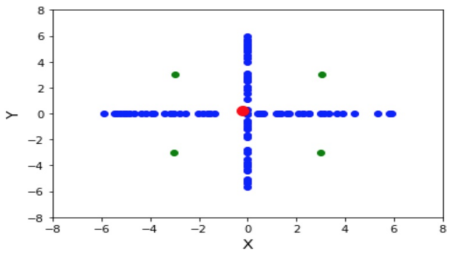
In this case, we use the kNN-distances in combination with the theoretical IF. The kNN algorithm makes use of these kNN-distances directly to detect the anomalies. The combination is better for high dimensional datasets, because it transforms these sets to one-dimensional datasets. The kNN algorithm itself does not work well with high dimensional sets, because it becomes computationally expensive.

The distance metric here can be any metric measure just like with the distance to the CM, but the Euclidean distance is the most common choice.

2.3.4 Counterexample for the Transformed Isolation Forest Based on Distance to the Center of Mass

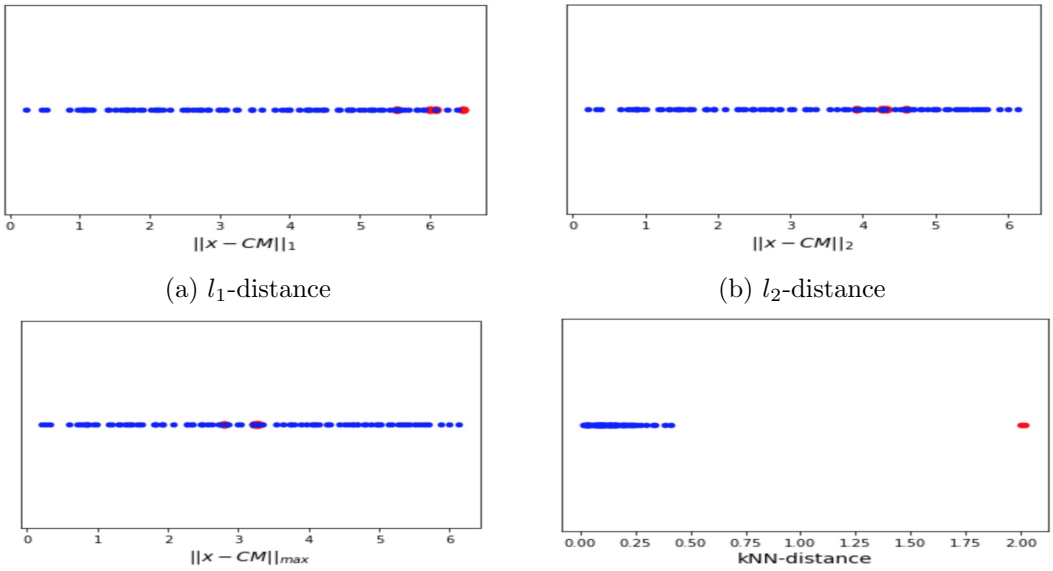
The method we described in Subsection 2.3.1 and Subsection 2.3.2 uses a global transformation and therefore there are also datasets for which this method cannot detect any of the most outlying points. An example of a dataset that could be difficult to handle for these type of methods can be seen in Figure 2.7.

Figure 2.7: Two-dimensional dataset of 100 points with 4 clear outliers.



In Figure 2.7, we see four clear outliers. These are the green points and the CM is the red dot. If we now calculate the l_1-, l_2- and l_{max}-distances between the CM and every datapoint in this set, we will see that these four clear outliers will not be seen as clear outliers anymore in the three transformed datasets given by these three distance metrics. The kNN-distance of every datapoint in this set is also calculated and the number of neighbours that is considered is three for each datapoint. The four new transformed datasets can be seen in Figure 2.8.

In Figure 2.8, the red dots are the four outliers of the dataset of Figure 2.7. Hence, it can be concluded that these points are mixed with the inliers and aren't clear outliers anymore for the three global transformations. For the l_1-distance, we see that the four outliers have almost the largest distances of the dataset. For the l_2- and l_{max}-distance, we see that many of the inliers have much larger distances to the CM than the distances of the four outliers to the CM. However, in the dataset of the kNN-distance the four outliers are still clear outliers. In Subsection 2.3.2, we expected the l_2-norm to be the best choice for the experiments. However, for this specific dataset, our expectation is that the l_1-norm perhaps will detect some of the outliers, the l_2- and l_{max}-distance won't detect any of the outliers and the kNN-distance will detect all four clear outliers.



(a) l_1-distance (b) l_2-distance

(c) l_{max}-distance (d) kNN-distance

Figure 2.8: Transformed one-dimensional datasets of the two-dimensional dataset of Figure 2.7

2.4 Summary

Summarizing, we have provided multiple definitions of an outlier together with some of the methods that are based on these definitions. First, the notion of an outlier was introduced in Section 2.1. After that, the distance-based definitions and methods were described in Subsection 2.1.1. In Subsection 2.1.2, non-parametric density-based methods were explained. These are methods that do not look at the specific data distribution. There are also methods that do look at this and these are the parametric density-based methods, which were described in Subsection 2.1.3. The cluster-based methods were explained in Subsection 2.1.4. Moreover, methods that are based on the concept of depth were introduced in Subsection 2.2.1. In Subsection 2.1.5, we introduced the confusion matrix which is a measure of quality for our outlier detection methods. The main topic of this thesis, namely the notion of isolation, was introduced together with the IF method. This was done in Subsection 2.2.2. In the last section of this chapter, Section 2.3, a new method for outlier detection was developed. This method transforms multi-dimensional datasets into one-dimensional ones and then it detects outliers just like the IF method.

3. Theoretical Results and Numerical Experiments

The first section of this chapter introduces some theory behind the original IF algorithm, discussed in Section 2.2. This is especially applicable to one-dimensional datasets. Moreover, Section 3.2 expands the theory to the two-dimensional case. For multi-dimensional datasets, the Transformed Isolation Forest introduced in Section 2.3 is used to transform these datasets to one-dimensional datasets and then use the one-dimensional theoretical formulas. These formulas are validated in Section 3.3 and Section 3.4. In Section 3.5, the results of outlier detection with the Transformed Isolation Forest described in Section 2.3 are presented and explained. The new global and local methods are compared in detecting the most outlying points of multiple datasets with the original Isolation Forest method.

3.1 Theory One-Dimensional Case

In this section, the theory behind the original IF algorithm, discussed in Section 2.2 is developed. There is not much theory known for this method and therefore we want to explore this more thoroughly. In Appendix A of [19], there is some theory developed for simple one-dimensional datasets. We will expand this theory to make it applicable to more complex and bigger datasets. For simplicity, we start with the fringe points of a one-dimensional dataset.

3.1.1 Fringe Points

The theory that is developed in this section is based on the number of random splits that the IF algorithm needs in order to isolate a point. A random split (RS) is the partition of the dataset into two subsets. The setting is as follows: the dataset is equal to $(x_1, ..., x_n)$ and thus consist of n points. All the observations are assumed different and the dataset is sorted from minor to major. The question is now: what are the probabilities that a certain point in this set can be isolated in s random splits with $1 \leq s \leq n-1$? The first points that are considered are the fringe points of the dataset. In the one-dimensional case, these are the minimum and maximum points of the dataset. The theory starts with the point x_1, the left fringe point of the dataset, which is the minimum point of the dataset. We find the recursion given by Equation (3.1).

$$P_{x_1}^{LF}(RS = s+1) = \sum_{i=s+1}^{n-1} \frac{x_{i+1} - x_i}{x_n - x_1} P_{x_1}^{LF}(RS = s|(x_1, ..., x_i)). \tag{3.1}$$

In Equation (3.1), $P_{x_1}^{LF}(RS = s+1)$ is the probability that the left fringe point x_1 gets isolated in $s+1$ random splits. After splitting between the point x_i and x_{i+1}, we only look at the remaining subset $(x_1, ..., x_i)$. This gives us $P_{x_1}^{LF}(RS = s|(x_1, ..., x_i))$, which is the probability that the left fringe point x_1 in the remaining dataset $(x_1, ..., x_i)$ gets isolated in s random splits. This leads to the recursion formula of Equation (3.1). This formula is computationally very demanding and therefore we look for an expression that is faster to implement. Hence, the particular cases $RS = 1, 2, 3, 4$ are written out to determine a formula for all cases that is also fast to implement.

$$P_{x_1}^{LF}(RS=1) = \frac{x_2 - x_1}{x_n - x_1}, \tag{3.2}$$

$$\begin{aligned} P_{x_1}^{LF}(RS=2) &= \sum_{i=2}^{n-1} \frac{x_{i+1} - x_i}{x_n - x_1} \cdot \frac{x_2 - x_1}{x_i - x_1} \\ &= \frac{x_2 - x_1}{x_n - x_1} \sum_{i=2}^{n-1} \frac{x_{i+1} - x_i}{x_i - x_1} = P_{x_1}^{LF}(RS=1) \sum_{i=2}^{n-1} \frac{x_{i+1} - x_i}{x_i - x_1}, \end{aligned} \tag{3.3}$$

$$\begin{aligned} P_{x_1}^{LF}(RS=3) &= \sum_{j=3}^{n-1} \sum_{i=2}^{j-1} \frac{x_{j+1} - x_j}{x_n - x_1} \cdot \frac{x_{i+1} - x_i}{x_j - x_1} \cdot \frac{x_2 - x_1}{x_i - x_1} \\ &= \frac{x_2 - x_1}{x_n - x_1} \sum_{j=3}^{n-1} \sum_{i=2}^{j-1} \frac{x_{j+1} - x_j}{x_j - x_1} \cdot \frac{x_{i+1} - x_i}{x_i - x_1} \\ &= \frac{x_2 - x_1}{x_n - x_1} \sum_{j=3}^{n-1} \frac{x_{j+1} - x_j}{x_j - x_1} \sum_{i=2}^{j-1} \frac{x_{i+1} - x_i}{x_i - x_1} \\ &= \sum_{j=3}^{n-1} \frac{x_{j+1} - x_j}{x_j - x_1} \Big(P_{x_1}^{LF}(RS=2) - \frac{x_2 - x_1}{x_n - x_1} \sum_{i=j}^{n-1} \frac{x_{i+1} - x_i}{x_i - x_1} \Big). \end{aligned} \tag{3.4}$$

In Equation (3.4), we see that the first split is between x_j and x_{j+1} for $j = 3, ..., n-1$. The value of j couldn't be smaller for $RS = 3$, because otherwise the first split will be between x_j and x_{j+1} for $j = 1, 2$ and then the remaining dataset $(x_1, .., x_j)$ consists of less than 3 datapoints. Therefore, there cannot be more than 1 RS in this remaining dataset and thus x_1 cannot be isolated in 3 RS. Hence, we have the remaining dataset $(x_1, .., x_j)$ after the first split and in this set x_1 should be isolated in 2 RS. Thus, for this dataset we use the formula of Equation (3.3).

$$\begin{aligned} P_{x_1}^{LF}(RS=4) &= \sum_{k=4}^{n-1} \sum_{j=3}^{k-1} \sum_{i=2}^{j-1} \frac{x_{k+1} - x_k}{x_n - x_1} \cdot \frac{x_{j+1} - x_j}{x_k - x_1} \cdot \frac{x_{i+1} - x_i}{x_j - x_1} \cdot \frac{x_2 - x_1}{x_i - x_1} \\ &= \sum_{k=4}^{n-1} \sum_{j=3}^{k-1} \sum_{i=2}^{j-1} \frac{x_{k+1} - x_k}{x_k - x_1} \cdot \frac{x_{j+1} - x_j}{x_j - x_1} \cdot \frac{x_{i+1} - x_i}{x_i - x_1} \cdot \frac{x_2 - x_1}{x_n - x_1} \\ &= \frac{x_2 - x_1}{x_n - x_1} \sum_{k=4}^{n-1} \frac{x_{k+1} - x_k}{x_k - x_1} \sum_{j=3}^{k-1} \frac{x_{j+1} - x_j}{x_j - x_1} \sum_{i=2}^{j-1} \frac{x_{i+1} - x_i}{x_i - x_1} \\ &= \sum_{k=4}^{n-1} \frac{x_{k+1} - x_k}{x_k - x_1} \Big(P_{x_1}^{LF}(RS=3) - \frac{x_2 - x_1}{x_n - x_1} \sum_{j=k}^{n-1} \frac{x_{j+1} - x_j}{x_j - x_1} \sum_{i=2}^{j-1} \frac{x_{i+1} - x_i}{x_i - x_1} \Big). \end{aligned} \tag{3.5}$$

Equation (3.5) can be expanded to a general formula for s random splits with $1 \leq s \leq n-1$:

$$P_{x_1}^{LF}(RS=s) = \sum_{k=s}^{n-1} \frac{x_{k+1}-x_k}{x_k-x_1}\Big(P_{x_1}^{LF}(RS=s-1) - \frac{x_2-x_1}{x_n-x_1} \sum_{j=k}^{n-1} \frac{x_{j+1}-x_j}{x_j-x_1} \sum_{i=s-2}^{j-1} \frac{x_{i+1}-x_i}{x_i-x_1} \cdots \sum_{a=2}^{b-1} \frac{x_{a+1}-x_a}{x_a-x_1}\Big). \tag{3.6}$$

Equation (3.6) can be efficiently implemented in Python by making use of cumulative summations of the previous terms.

With the formulas above, the expectation and variance of the number of RS that are needed to isolate the left fringe point x_1 can also be derived. These formulas are given as follows:

$$E_{x_1}^{LF}(RS) = \sum_{i=1}^{n-1} i P_{x_1}^{LF}(RS=i), \tag{3.7}$$

$$Var_{x_1}^{LF}(RS) = \sum_{i=1}^{n-1} i^2 P_{x_1}^{LF}(RS=i) - \Big(E_{x_1}^{LF}(RS)\Big)^2. \tag{3.8}$$

After the probabilities of the left fringe point of the dataset have been calculated, the same can be done for the right fringe point. These probabilities are very similar to those of the left fringe point. Notice that the dataset can be sorted in the opposite direction, i.e. from major to minor, and then the probabilities of the left fringe point can be used again. The explicit probabilities of the right fringe point x_n are given in Appendix A.1.

3.1.2 Interior Points

The next step is computing the probabilities that an interior point gets isolated in s random splits with $2 \leq s \leq n-1$. In the one-dimensional case, an interior point is a point which is not the minimum nor the maximum of the dataset. If the same dataset $(x_1, ..., x_n)$ is used, then these probabilities are as follows for the point x_j in the dataset:

$$\begin{aligned} P_{x_j}^{IN}(RS=s) = &\sum_{i=1}^{j-2} \frac{x_{i+1}-x_i}{x_n-x_1} P_{x_j}^{IN}(RS=s-1|x_{i+1}, ..., x_n) \\ &+ \sum_{i=j+1}^{n-1} \frac{x_{i+1}-x_i}{x_n-x_1} P_{x_j}^{IN}(RS=s-1|x_1, ..., x_i) \\ &+ \frac{x_j-x_{j-1}}{x_n-x_1} P_{x_j}^{LF}(RS=s-1|x_j, ..., x_n) \\ &+ \frac{x_{j+1}-x_j}{x_n-x_1} P_{x_j}^{RF}(RS=s-1|x_1, ..., x_j). \end{aligned} \tag{3.9}$$

In this formula, we can see that if the first split is to the left of the point x_j, but not between x_j and x_{j-1}, then it is between x_i and x_{i+1} with $1 \leq i \leq j-2$ and x_j is still an interior point. In

that case, the remaining subset is $(x_{i+1}, ..., x_n)$ with now only $s-1$ splits. The same happens if the first split is to the right of the point x_j, but not between x_j and x_{j+1} and thus between x_i and x_{i+1} with $j+1 \leq i \leq n-1$. In this case, we will look at the remaining subset $(x_1, ..., x_i)$ and use the formula for s splits. If the first split is between x_j and x_{j-1}, then we will look at the remaining subset $(x_j, ..., x_n)$. In this subset, the point x_j is a left fringe point and therefore we use the fringe formula. Finally, if the first split is between x_j and x_{j+1}, then we will look at the remaining subset $(x_1, ..., x_j)$. In this subset, the point x_j is a right fringe point and therefore we use the fringe formula. Equation (3.9) is the combination of these four possibilities for the first random split. It is a recursion formula and thus computationally very expensive. Again, we are looking for an expression that is faster to implement. Therefore, the particular cases $RS = 2, 3$ are written out.

$$P_{x_j}^{IN}(RS=2) = \frac{x_j - x_{j-1}}{x_n - x_1} \cdot \frac{x_{j+1} - x_j}{x_n - x_j} + \frac{x_{j+1} - x_j}{x_n - x_1} \cdot \frac{x_j - x_{j-1}}{x_j - x_1}, \tag{3.10}$$

$$\begin{aligned} P_{x_j}^{IN}(RS=3) = &\sum_{i=1}^{j-2} \frac{x_{i+1} - x_i}{x_n - x_1} P_{x_j}^{IN}(RS=2|(x_{i+1}, ..., x_n)) \\ &+ \sum_{i=j+1}^{n-1} \frac{x_{i+1} - x_i}{x_n - x_1} P_{x_j}^{IN}(RS=2|(x_1, ..., x_i)) \\ &+ \frac{x_j - x_{j-1}}{x_n - x_1} P_{x_j}^{LF}(RS=2|(x_j, ..., x_n)) \\ &+ \frac{x_{j+1} - x_j}{x_n - x_1} P_{x_j}^{RF}(RS=2|(x_1, ..., x_j)). \end{aligned} \tag{3.11}$$

This formula is also written out for 4 random splits in Appendix A.2. Notice that these formulas can be written in terms of the fringe formulas. Starting with Equation (3.10), we find:

$$\begin{aligned} P_{x_j}^{IN}(RS=2) &= \frac{(x_{j+1} - x_j)(x_j - x_{j-1})}{x_n - x_1}\left(\frac{1}{x_j - x_1} + \frac{1}{x_n - x_j}\right) = \frac{(x_{j+1} - x_j)(x_j - x_{j-1})}{(x_n - x_j)(x_j - x_1)} \\ &= P_{x_j}^{LF}(RS=1|(x_j, ..., x_n)) P_{x_j}^{RF}(RS=1|(x_1, ..., x_j)). \end{aligned} \tag{3.12}$$

And starting with Equation (3.11), we get:

$$\begin{aligned} P_{x_j}^{IN}(RS=3) = &\sum_{i=1}^{j-2} \frac{x_{i+1} - x_i}{x_n - x_1} \cdot \frac{(x_{j+1} - x_j)(x_j - x_{j-1})}{(x_n - x_j)(x_j - x_{i+1})} + \sum_{i=j+1}^{n-1} \frac{x_{i+1} - x_i}{x_n - x_1} \cdot \frac{(x_{j+1} - x_j)(x_j - x_{j-1})}{(x_i - x_j)(x_j - x_1)} \\ &+ \frac{x_j - x_{j-1}}{x_n - x_1} \sum_{i=j+1}^{n-1} \frac{x_{i+1} - x_i}{x_n - x_j} \cdot \frac{x_{j+1} - x_j}{x_i - x_j} + \frac{x_{j+1} - x_j}{x_n - x_1} \sum_{i=1}^{j-2} \frac{x_{i+1} - x_i}{x_j - x_1} \cdot \frac{x_j - x_{j-1}}{x_j - x_{i+1}} \end{aligned}$$

$$= \frac{(x_{j+1}-x_j)(x_j-x_{j-1})}{(x_n-x_j)(x_j-x_1)}\Big(\sum_{i=1}^{j-2}\frac{x_{i+1}-x_i}{x_j-x_{i+1}}+\sum_{i=j+1}^{n-1}\frac{x_{i+1}-x_i}{x_i-x_j}\Big)$$

$$= \frac{x_{j+1}-x_j}{x_n-x_j}P_{x_j}^{RF}(RS=2|(x_1,...,x_j))+\frac{x_j-x_{j-1}}{x_j-x_1}P_{x_j}^{LF}(RS=2|(x_j,...,x_n))$$

$$= P_{x_j}^{LF}(RS=1|(x_j,...,x_n))P_{x_j}^{RF}(RS=2|(x_1,...,x_j))$$

$$+ P_{x_j}^{RF}(RS=1|(x_1,...,x_j))P_{x_j}^{LF}(RS=2|(x_j,...,x_n)). \tag{3.13}$$

The formulas for isolating in 2 and 3 random splits hint at the recursion formula given by Lemma 3.1. This formula is proven by an induction proof given below.

Lemma 3.1. *For a particular $2 \leq s \leq n-2$, it holds that*

$$P_{x_j}^{IN}(RS=s)=\sum_{i=1}^{s-1}P_{x_j}^{LF}(RS=i|(x_j,...,x_n))P_{x_j}^{RF}(RS=s-i|(x_1,...,x_j)). \tag{3.14}$$

Proof. The base case for isolating in 2 random splits is already proven above, so the only aspect that is left to prove is the inductive step. Thus, the goal is to prove that it also holds that:

$$P_{x_j}^{IN}(RS=s+1)=\sum_{i=1}^{s}P_{x_j}^{LF}(RS=i|(x_j,...,x_n))P_{x_j}^{RF}(RS=(s+1)-i|(x_1,...,x_j)).$$

It is clear from Equation (3.9) that the following formula holds:

$$P_{x_j}^{IN}(RS=s+1)=\sum_{i=1}^{j-2}\frac{x_{i+1}-x_i}{x_n-x_1}P_{x_j}^{IN}(RS=s|(x_{i+1},...,x_n))$$

$$+\sum_{i=j+1}^{n-1}\frac{x_{i+1}-x_i}{x_n-x_1}P_{x_j}^{IN}(RS=s|(x_1,...,x_i))$$

$$+\frac{x_j-x_{j-1}}{x_n-x_1}P_{x_j}^{LF}(RS=s|(x_j,...,x_n))$$

$$+\frac{x_{j+1}-x_j}{x_n-x_1}P_{x_j}^{RF}(RS=s|(x_1,...,x_j)).$$

Now, Equation (3.14) is substituted in this formula and that leads to the following:

$$P_{x_j}^{IN}(RS=s+1)=\sum_{i=1}^{j-2}\frac{x_{i+1}-x_i}{x_n-x_1}\Big(\sum_{k=1}^{s-1}P_{x_j}^{LF}(RS=k|(x_j,...,x_n))P_{x_j}^{RF}(RS=s-k|(x_{i+1},...,x_j))\Big)$$

$$+ \sum_{i=j+1}^{n-1} \frac{x_{i+1}-x_i}{x_n-x_1}\Big(\sum_{k=1}^{s-1} P_{x_j}^{LF}(RS=k|(x_j,...,x_i))P_{x_j}^{RF}(RS=s-k|(x_1,...,x_j))\Big)$$

$$+ \frac{x_j-x_{j-1}}{x_n-x_1}P_{x_j}^{LF}(RS=s|(x_j,...,x_n))$$

$$+ \frac{x_{j+1}-x_j}{x_n-x_1}P_{x_j}^{RF}(RS=s|(x_1,...,x_j))$$

$$= \sum_{k=1}^{s-1} P_{x_j}^{LF}(RS=k|(x_j,...,x_n)) \sum_{i=1}^{j-2} \frac{x_{i+1}-x_i}{x_n-x_1}P_{x_j}^{RF}(RS=s-k|(x_{i+1},...,x_j))$$

$$+ \sum_{k=1}^{s-1} P_{x_j}^{RF}(RS=s-k|(x_1,...,x_j)) \sum_{i=j+1}^{n-1} \frac{x_{i+1}-x_i}{x_n-x_1}P_{x_j}^{LF}(RS=k|(x_j,...,x_i))$$

$$+ \frac{x_j-x_{j-1}}{x_n-x_1}P_{x_j}^{LF}(RS=s|(x_j,...,x_n))$$

$$+ \frac{x_{j+1}-x_j}{x_n-x_1}P_{x_j}^{RF}(RS=s|(x_1,...,x_j)).$$

The first and the third terms are multiplied with $\frac{x_j-x_1}{x_j-x_1}$ and the second and the fourth terms with $\frac{x_n-x_j}{x_n-x_j}$, respectively. This gives us the following result:

$$P_{x_j}^{IN}(RS=s+1) = \sum_{k=1}^{s-1} P_{x_j}^{LF}(RS=k|(x_j,...,x_n)) \sum_{i=1}^{j-2} \frac{x_{i+1}-x_i}{x_n-x_1}\frac{x_j-x_1}{x_j-x_1}P_{x_j}^{RF}(RS=s-k|(x_{i+1},...,x_j))$$

$$+ \sum_{k=1}^{s-1} P_{x_j}^{RF}(RS=s-k|(x_1,...,x_j)) \sum_{i=j+1}^{n-1} \frac{x_{i+1}-x_i}{x_n-x_1}\frac{x_n-x_j}{x_n-x_j}P_{x_j}^{LF}(RS=k|(x_j,...,x_i))$$

$$+ \frac{x_j-x_{j-1}}{x_n-x_1}\frac{x_j-x_1}{x_j-x_1}P_{x_j}^{LF}(RS=s|(x_j,...,x_n))$$

$$+ \frac{x_{j+1}-x_j}{x_n-x_1}\frac{x_n-x_j}{x_n-x_j}P_{x_j}^{RF}(RS=s|(x_1,...,x_j))$$

$$= \frac{x_j-x_1}{x_n-x_1}\sum_{k=1}^{s-1} P_{x_j}^{LF}(RS=k|(x_j,...,x_n)) \sum_{i=1}^{j-2} \frac{x_{i+1}-x_i}{x_j-x_1}P_{x_j}^{RF}(RS=s-k|(x_{i+1},...,x_j))$$

$$+ \frac{x_n - x_j}{x_n - x_1} \sum_{k=1}^{s-1} P_{x_j}^{RF}(RS = s-k|(x_1, ..., x_j)) \sum_{i=j+1}^{n-1} \frac{x_{i+1} - x_i}{x_n - x_j} P_{x_j}^{LF}(RS = k|(x_j, ..., x_i))$$

$$+ \frac{x_j - x_1}{x_n - x_1} P_{x_j}^{RF}(RS = 1|(x_1, ..., x_j)) P_{x_j}^{LF}(RS = s|(x_j, ..., x_n))$$

$$+ \frac{x_n - x_j}{x_n - x_1} P_{x_j}^{LF}(RS = 1|(x_j, ..., x_n)) P_{x_j}^{RF}(RS = s|(x_1, ..., x_j)).$$

The term $\frac{x_{i+1}-x_i}{x_j-x_1}$ is the probability to isolate the points to the left of x_{i+1} from the dataset $(x_1, ..., x_j)$. The term $P_{x_j}^{RF}(RS = s-k|(x_{i+1}, ..., x_j))$ is the probability that the right fringe point x_j gets isolated in $s-k$ random splits in the dataset $(x_{i+1}, ..., x_j)$. When these two terms are multiplied, this leads to one extra split due to the term $\frac{x_{i+1}-x_i}{x_j-x_1}$, so $s-k$ changes in $(s+1)-k$. Moreover, the dataset $(x_{i+1}, ..., x_j)$ changes in the whole dataset $(x_1, .., x_j)$, because the term $\frac{x_{i+1}-x_i}{x_j-x_1}$ is the split in the whole dataset $(x_1, ..., x_j)$. Together with the summation over i and the term, this leads to the following term: $P_{x_j}^{RF}(RS = (s+1)-k|(x_1, ..., x_j))$. A similar result holds for the second term. This leads to the following:

$$P_{x_j}^{IN}(RS = s+1) = \frac{x_j - x_1}{x_n - x_1} \sum_{k=1}^{s-1} P_{x_j}^{LF}(RS = k|(x_j, ..., x_n)) P_{x_j}^{RF}(RS = (s+1)-k|(x_1, ..., x_j))$$

$$+ \frac{x_n - x_j}{x_n - x_1} \sum_{k=1}^{s-1} P_{x_j}^{RF}(RS = s-k|(x_1, ..., x_j)) P_{x_j}^{LF}(RS = k+1|(x_j, ..., x_n))$$

$$+ \frac{x_j - x_1}{x_n - x_1} P_{x_j}^{RF}(RS = 1|(x_1, ..., x_j)) P_{x_j}^{LF}(RS = s|(x_j, ..., x_n))$$

$$+ \frac{x_n - x_j}{x_n - x_1} P_{x_j}^{LF}(RS = 1|(x_j, ..., x_n)) P_{x_j}^{RF}(RS = s|(x_1, ..., x_j))$$

$$= \frac{x_j - x_1}{x_n - x_1} \Big(\sum_{k=1}^{s-1} P_{x_j}^{LF}(RS = k|(x_j, ..., x_n)) P_{x_j}^{RF}(RS = (s+1)-k|(x_1, ..., x_j))$$

$$+ P_{x_j}^{RF}(RS = 1|(x_1, ..., x_j)) P_{x_j}^{LF}(RS = s|(x_j, ..., x_n)) \Big)$$

$$+ \frac{x_n - x_j}{x_n - x_1} \Big(\sum_{k=1}^{s-1} P_{x_j}^{RF}(RS = s-k|(x_1, ..., x_j)) P_{x_j}^{LF}(RS = k+1|(x_j, ..., x_n))$$

$$+ P_{x_j}^{LF}(RS = 1|(x_j, ..., x_n)) P_{x_j}^{RF}(RS = s|(x_1, ..., x_j)) \Big)$$

$$= \frac{x_j - x_1}{x_n - x_1} \sum_{k=1}^{s} P_{x_j}^{LF}(RS = k|(x_j, ..., x_n)) P_{x_j}^{RF}(RS = (s+1) - k|(x_1, ..., x_j))$$

$$+ \frac{x_n - x_j}{x_n - x_1} \sum_{k=1}^{s} P_{x_j}^{RF}(RS = (s+1) - k|(x_1, ..., x_j)) P_{x_j}^{LF}(RS = k|(x_j, ..., x_n))$$

$$= \frac{x_j - x_1 + x_n - x_j}{x_n - x_1} \sum_{k=1}^{s} P_{x_j}^{LF}(RS = k|(x_j, ..., x_n)) P_{x_j}^{RF}(RS = (s+1) - k|(x_1, ..., x_j))$$

$$= \frac{x_n - x_1}{x_n - x_1} \sum_{k=1}^{s} P_{x_j}^{LF}(RS = k|(x_j, ..., x_n)) P_{x_j}^{RF}(RS = (s+1) - k|(x_1, ..., x_j))$$

$$= \sum_{k=1}^{s} P_{x_j}^{LF}(RS = k|(x_j, ..., x_n)) P_{x_j}^{RF}(RS = (s+1) - k|(x_1, ..., x_j)).$$

□

Hence, by induction the recursion formula given by Equation (3.14) holds for all $2 \leq s \leq n-1$. With the formulas above, the expectation and variance of the number of RS that are needed to isolate the interior point x_j can also be derived. These formulas are given as follows:

$$E_{x_j}^{IN}(RS) = \sum_{i=2}^{n-1} i P_{x_j}^{IN}(RS = i), \tag{3.15}$$

$$Var_{x_j}^{IN}(RS) = \sum_{i=2}^{n-1} i^2 P_{x_j}^{IN}(RS = i) - \left(E_{x_j}^{IN}(RS)\right)^2. \tag{3.16}$$

3.2 Theory Two-Dimensional Case

The random splitting theory that is derived in the previous section holds for one-dimensional datasets. The next goal is to expand this to higher dimensions, particularly to two-dimensional datasets. Instead of only x_j, the points are now denoted by (x_j, y_j). In this case, there are four possible types of points. It could be a fringe point in both directions, an interior point in both directions, an interior point in the x-direction and a fringe point in the y-direction or a fringe point in the x-direction and an interior point in the y-direction. These formulas will turn out to be long, involved and cumbersome. Therefore, only theoretical formulas are developed for points that are fringe points in both directions.

3.2.1 Fringe Point in Both Directions

For simplicity, a point that is a left fringe point in both directions is looked at first. This point is thus the point (x_1, y_1). Such a point may not always exist, but in this dataset it is assumed that it does. It is known that IF chooses randomly which direction to split. In particular, it

is uniformly random. This means that for every split it holds that the probability that the split will be in one direction or the other is equal to $\frac{1}{2}$. Here, the assumption is made that the points are ordered in the x-direction only, so the following dataset $\{(x_1, y_1), (x_2, \tilde{y}_2), ..., (x_n, \tilde{y}_n)\}$ is considered. The $\tilde{y}$ are the points in the y-direction according to the sorting in the x-direction. The mapping $f : i_{\tilde{y}} \to i_y$ ensures that the points are ordered in the y-direction. This mapping has as input the index of the point in the unsorted y-direction and gives as output the same index in the sorted y-direction. For example, $\tilde{y}_3 = y_{f(3)}$. There is also the mapping for the other direction $g : i_{\tilde{x}} \to i_x$. If y is the sorted sample, then it gives rise to the following dataset $\{(x_1, y_1), (\tilde{x}_2, y_2), ..., (\tilde{x}_n, y_n)\}$. Before every random split, a sorting function is used in order to get the dataset sorted in the right direction, meaning, in the direction of the upcoming split. As already mentioned, for every split it holds that the probability that the split will be in one direction or the other is equal to $\frac{1}{2}$. With this in mind, the formula that the point (x_1, y_1) gets isolated in s random splits with $1 \leq s \leq n-1$ can be derived. It is given by:

$$\begin{aligned} P^{LF,LF}_{(x_1,y_1)}(RS = s) = \frac{1}{2} \cdot \sum_{j=s}^{n-1} \frac{x_{j+1} - x_j}{x_n - x_1} P^{LF,LF}_{(x_1,y_1)}(RS = s-1|((x_1, y_1), ..., (x_j, \tilde{y}_j))) \\ + \frac{1}{2} \cdot \sum_{j=s}^{n-1} \frac{y_{j+1} - y_j}{y_n - y_1} P^{LF,LF}_{(x_1,y_1)}(RS = s-1|((x_1, y_1), ..., (\tilde{x}_j, y_j))). \end{aligned} \tag{3.17}$$

In Equation (3.17), $P^{LF,LF}_{(x_1,y_1)}(RS = s)$ is the probability that the left fringe point in both directions (x_1, y_1) gets isolated in s random splits. If the first split is in the x-direction, we will look at the remaining subset $((x_1, y_1), ..., (x_j, \tilde{y}_j))$ with x the sorted sample. This gives us $P^{LF,LF}_{(x_1,y_1)}(RS = s-1|((x_1, y_1), ..., (x_j, \tilde{y}_j)))$, which is the probability that the left fringe point in both directions (x_1, y_1) in the remaining dataset $((x_1, y_1), ..., (x_j, \tilde{y}_j))$ gets isolated in $s-1$ random splits. If the first split is in the y-direction, we will look at the remaining subset $((x_1, y_1), ..., (\tilde{x}_j, y_j))$ with y the sorted sample.

This recursive formula is written out for the particular cases $RS = 1, 2, 3$, as follows:

$$\begin{aligned} P^{LF,LF}_{(x_1,y_1)}(RS = 1) &= \frac{1}{2} \cdot P^{LF}_{x_1}(RS = 1) + \frac{1}{2} \cdot P^{LF}_{y_1}(RS = 1) \\ &= \frac{1}{2} \cdot \frac{x_2 - x_1}{x_n - x_1} + \frac{1}{2} \cdot \frac{y_2 - y_1}{y_n - y_1}. \end{aligned} \tag{3.18}$$

In Equation (3.18), we see that the probability that the point (x_1, y_1) gets isolated in 1 RS is equal to $\frac{1}{2}$ multiplied by the probability that x_1 gets isolated in 1 RS together with $\frac{1}{2}$ multiplied by the probability that y_1 gets isolated in 1 RS. This has to do with the fact that there is only one RS and this RS can only be in one direction. The probability that it is in either direction

is equal to $\frac{1}{2}$. For this specific point to be isolated in 2 random splits, we find:

$$
\begin{aligned}
P^{LF,LF}_{(x_1,y_1)}(RS=2) &= \frac{1}{4}\cdot P^{LF}_{x_1}(RS=2) + \frac{1}{4}\cdot P^{LF}_{y_1}(RS=2) \\
&+ \frac{1}{4}\cdot\sum_{i=2}^{n-1}\frac{x_{i+1}-x_i}{x_n-x_1}P^{LF}_{y_1}(RS=1|((x_1,y_1),...,(x_i,\tilde{y}_i))) \\
&+ \frac{1}{4}\cdot\sum_{i=2}^{n-1}\frac{y_{i+1}-y_i}{y_n-y_1}P^{LF}_{x_1}(RS=1|((x_1,y_1),...,(\tilde{x}_i,y_i))) \\
&= \frac{1}{4}\cdot\frac{x_2-x_1}{x_n-x_1}\sum_{i=2}^{n-1}\frac{x_{i+1}-x_i}{x_i-x_1} + \frac{1}{4}\cdot\frac{y_2-y_1}{y_n-y_1}\sum_{i=2}^{n-1}\frac{y_{i+1}-y_i}{y_i-y_1} \\
&+ \frac{1}{4}\cdot\sum_{i=2}^{n-1}\frac{x_{i+1}-x_i}{x_n-x_1}\frac{y_{f(2)}-y_1}{y_{f(i)}-y_1} + \frac{1}{4}\cdot\sum_{i=2}^{n-1}\frac{y_{i+1}-y_i}{y_n-y_1}\frac{x_{g(2)}-x_1}{x_{g(i)}-x_1},
\end{aligned}
\tag{3.19}
$$

While the formula for three splits grows to:

$$
\begin{aligned}
P^{LF,LF}_{(x_1,y_1)}(RS=3) &= \frac{1}{2}\cdot\sum_{j=3}^{n-1}\frac{x_{j+1}-x_j}{x_n-x_1}P^{LF,LF}_{(x_1,y_1)}(RS=2|((x_1,y_1),...,(x_j,\tilde{y}_j))) \\
&+ \frac{1}{2}\cdot\sum_{j=3}^{n-1}\frac{y_{j+1}-y_j}{y_n-y_1}P^{LF,LF}_{(x_1,y_1)}(RS=2|((x_1,y_1),...,(\tilde{x}_j,y_j))) \\
&= \frac{1}{8}\cdot\sum_{j=3}^{n-1}\sum_{i=2}^{j-1}\frac{x_{j+1}-x_j}{x_n-x_1}\cdot\frac{x_{i+1}-x_i}{x_j-x_1}\cdot\frac{x_2-x_1}{x_i-x_1} \\
&+ \frac{1}{8}\cdot\sum_{j=3}^{n-1}\sum_{i=2}^{j-1}\frac{y_{j+1}-y_j}{y_n-y_1}\cdot\frac{y_{i+1}-y_i}{y_j-y_1}\cdot\frac{y_2-y_1}{y_i-y_1} \\
&+ \frac{1}{8}\cdot\sum_{j=3}^{n-1}\sum_{i=2}^{j-1}\frac{x_{j+1}-x_j}{x_n-x_1}\cdot\frac{x_{i+1}-x_i}{x_j-x_1}\cdot\frac{y_{f(2)}-y_1}{y_{f(i)}-y_1} \\
&+ \frac{1}{8}\cdot\sum_{j=3}^{n-1}\sum_{i=2}^{j-1}\frac{x_{j+1}-x_j}{x_n-x_1}\cdot\frac{y_{f(i+1)}-y_{f(i)}}{y_{f(j)}-y_1}\cdot\frac{x_{g(2)}-x_1}{x_{g(i)}-x_1} \\
&+ \frac{1}{8}\cdot\sum_{j=3}^{n-1}\sum_{i=2}^{j-1}\frac{x_{j+1}-x_j}{x_n-x_1}\cdot\frac{y_{f(i+1)}-y_{f(i)}}{y_{f(j)}-y_1}\cdot\frac{y_2-y_1}{y_i-y_1} \\
&+ \frac{1}{8}\cdot\sum_{j=3}^{n-1}\sum_{i=2}^{j-1}\frac{y_{j+1}-y_j}{y_n-y_1}\cdot\frac{x_{g(i+1)}-x_{g(i)}}{x_{g(j)}-x_1}\cdot\frac{y_{f(2)}-y_1}{y_{f(i)}-y_1} \\
&+ \frac{1}{8}\cdot\sum_{j=3}^{n-1}\sum_{i=2}^{j-1}\frac{y_{j+1}-y_j}{y_n-y_1}\cdot\frac{y_{i+1}-y_i}{y_j-y_1}\cdot\frac{x_{g(2)}-x_1}{x_{g(i)}-x_1} \\
&+ \frac{1}{8}\cdot\sum_{j=3}^{n-1}\sum_{i=2}^{j-1}\frac{y_{j+1}-y_j}{y_n-y_1}\cdot\frac{x_{g(i+1)}-x_{g(i)}}{x_{g(j)}-x_1}\cdot\frac{x_2-x_1}{x_i-x_1}.
\end{aligned}
\tag{3.20}
$$

In Equation (3.20), we see all the 8 possibilities for the point (x_1, y_1) to be isolated in 3 RS. There are 2 directions, namely x and y and 3 random splits. For every RS, the probability is

$\frac{1}{2}$ that the split is in either direction. Hence, there are $2^3 = 8$ possibilities and each possibility has a probability of $\frac{1}{2^3} = \frac{1}{8}$.

With the formulas above, the expectation and variance of the number of RS that is needed to isolate the left fringe point in both directions (x_1, y_1) can also be derived:

$$E^{LF,LF}_{(x_1,y_1)}(RS) = \sum_{i=1}^{n-1} iP^{LF,LF}_{(x_1,y_1)}(RS = i), \tag{3.21}$$

$$Var^{LF,LF}_{(x_1,y_1)}(RS) = \sum_{i=1}^{n-1} i^2 P^{LF,LF}_{(x_1,y_1)}(RS = i) - \left(E^{LF,LF}_{(x_1,y_1)}(RS)\right)^2. \tag{3.22}$$

The next step is to look at one of the other four possibilities for the type of point we deal with. The point that is explored next is a point that is an interior point in the x-direction and a fringe point in the y-direction. For these types of points, there are also formulas developed which turned out to be very cumbersome. These formulas were extremely difficult to implement in Python and also to prove. Due to the lack of efficient implementations, these formulas cannot be used for large datasets. Therefore, these formulas are only placed in Appendix A.3.

Hence, we will not go further with developing formulas for the two-dimensional case. Instead, we will use new methods to transform multi-dimensional datasets into one-dimensional set-ups in order to make use of the developed theoretical formulas for the one-dimensional case. Before that, we validate numerically the theory developed for one-dimensional datasets.

3.3 Numerical Validation One-Dimensional Case

In this section, the theoretical results from the previous sections will be compared with numerical results that are obtained from the Isolation Forest algorithm [8]. The theoretical formulas have been implemented in Python 3.9 for the fringe and interior points in a one-dimensional dataset. With these implementations, it is now possible to calculate the probabilities that a fringe or interior point in a one-dimensional dataset gets isolated in s random splits. The expectation and variance of the number of random splits needed to isolate a point can also be calculated theoretically. Numerically, this is also possible with the Isolation Forest method. Hence, we will compare the two to see if the theoretical and numerical probabilities match. We start with a numerical validation of the theoretical formulas developed in Section 3.1.

3.3.1 Fringe Points

The first dataset that is used for the comparison is a continuous uniformly distributed dataset consisting of 10 points. The Isolation Forest algorithm is tested with and without pruning. Pruned trees have a height limit l as already mentioned in Section 2.2. The height limit that is used here is the standard height limit of the algorithm, namely $l = \lceil \log_2(\psi) \rceil$ with ψ the sub-sampling size. The sub-sampling size controls the training size of the algorithm. A sample of the overall data is taken randomly and used to construct an isolation tree. The sub-sampling size is the minimum of 256 and the size of the dataset and thus in our case equal to the size of the dataset.

Before looking at the interior points, the fringe points are investigated. For the left fringe point of the dataset, the theoretical probabilities (3.6) that it becomes isolated in s random splits are compared with the numerical probabilities. This comparison is presented in Table 3.1.

Table 3.1: Theoretical and numerical probabilities that the left fringe point becomes isolated in s random splits.

Number of random splits	Theoretical probability	Isolation Forest (with fully grown trees)	Isolation Forest (with pruning)
1	0.00587	0.00579	0.00583
2	0.26545	0.26607	0.26378
3	0.41991	0.42129	0.42055
4	0.23728	0.23580	0.23871
5	0.06239	0.06196	0.07064
6	0.00847	0.00838	0.00049
7	0.00061	0.00066	0
8	0.00002	0.00005	0
9	0.0000003	0	0

In Table 3.2, the theoretical expectation (3.7) and variance (3.8) of the number of random splits that is needed to isolate the left fringe point are compared with the numerical expectation and variance.

Table 3.2: Expectation and variance of the number of random splits that is needed to isolate the left fringe point.

	Expectation	**Variance**
Theoretical	3.113	0.850
IF (with fully grown trees)	3.110	0.848
IF (with pruning)	3.106	0.802

By looking at Table 3.1 and Table 3.2, it can be seen that the theoretical probabilities, expectation and variance of this left fringe point match very well with the numerical probabilities for both the pruned and fully grown trees. For the fully grown trees, the difference is smaller between the numerical and theoretical results than for the forest with pruning. The numerical probabilities, when making use of pruned trees, are equal to zero after fewer random splits than with fully grown trees. This is because in this case the trees have been cut off at a certain point. The right fringe point of this dataset has also been looked at and that led to similar results as the left fringe point.

All these results are based on a uniform distribution, so with no clear outliers. Therefore, we also look at a dataset that comes from a standard normal distribution $\mathcal{N}(0, 1)$. The dataset consists of 10 points again. The left fringe point of this dataset has been explored first before experimenting with the interior points. The theoretical probabilities (3.6) that it becomes isolated in s random splits are compared with the numerical probabilities. The results are summarized in Table 3.3. The expectation (3.7) and variance (3.8) are shown in Table 3.4.

Just like with the uniform distribution, it can be seen that the theoretical probabilities, expectation and variance of this left fringe point match very well with the numerical probabilities for both the pruned and fully grown trees. The right fringe point of this dataset has also been looked at and that led to similar results as the left fringe point.

Further validation of these formulas is done in Section 3.5 using the Transformed Isolation Forest.

Table 3.3: Theoretical and numerical probabilities that the left fringe point becomes isolated in s random splits.

Number of random splits	Theoretical probability	Isolation Forest (with fully grown trees)	Isolation Forest (with pruning)
1	0.07234	0.07087	0.07270
2	0.31931	0.32266	0.31982
3	0.35913	0.35949	0.35715
4	0.18704	0.18443	0.18892
5	0.05291	0.05294	0.05976
6	0.00848	0.00875	0.00165
7	0.00075	0.00084	0
8	0.00003	0.00002	0
9	0.0000005	0	0

Table 3.4: Expectation and variance of the number of random splits that is needed to isolate the left fringe point.

	Expectation	**Variance**
Theoretical	2.857	1.076
IF (with fully grown trees)	2.856	1.074
IF (with pruning)	2.848	1.030

3.3.2 Interior Points

Now that the numerical probabilities of the fringe points have been compared with the theory for 2 different datasets, the same will be done for the interior points. First, an interior point of the same uniform distribution used in Subsection 3.3.1 is explored. In Table 3.5, the theoretical (3.14) and numerical probabilities that the point gets isolated in s random splits are compared. The expectation (3.15) and variance (3.16) are shown in Table 3.6.

Table 3.5: Theoretical and numerical probabilities that the interior point becomes isolated in s random splits.

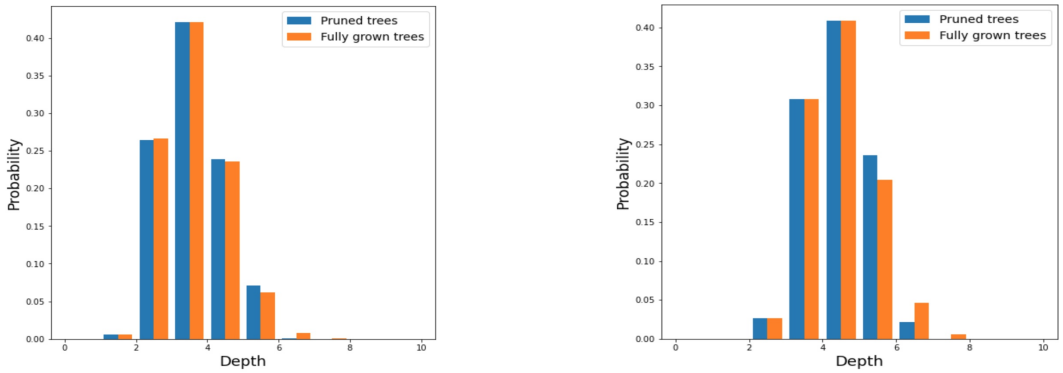
Number of random splits	Theoretical probability	Isolation Forest (with fully grown trees)	Isolation Forest (with pruning)
1	0	0	0
2	0.02588	0.02657	0.02639
3	0.30876	0.30813	0.30784
4	0.40954	0.40894	0.40851
5	0.20386	0.20429	0.23579
6	0.04658	0.04655	0.02147
7	0.00513	0.00527	0
8	0.00025	0.00024	0
9	0.000003	0.00001	0

Table 3.6: Expectation and variance of the number of random splits that is needed to isolate the interior point.

	Expectation	**Variance**
Theoretical	3.953	0.850
IF (with fully grown trees)	3.953	0.854
IF (with pruning)	3.918	0.728

Thus, we can conclude from Table 3.5 and Table 3.6 that for this interior point the theoretical probabilities also match with the numerical probabilities just like the expectation and variance at least for these experiments. This is also the case for another interior point in this dataset.

In Figure 3.1, the comparison between the distribution of random splits of the left fringe point and an interior point of the uniformly distributed dataset, for pruned and fully grown trees, is shown. We see two different distributions of the depth. For the left fringe point, the depth is much smaller than for the interior point. For the pruned trees, there are some differences for more than five random splits. This has to do with the cutting point of the trees that in a dataset with 10 points has been set at $l = \lceil \log_2(\psi) \rceil = \lceil \log_2(10) \rceil = 4$. This implies that up to four random splits the probabilities of the pruned trees and fully grown trees will be similar and after that there will be differences.



(a) Left fringe point

(b) Interior point

Figure 3.1: Probability that the left fringe point and an interior point of a continuous uniform distributed dataset become isolated in s random splits for the Isolation Forest [8] method with 100000 trees.

The next step is to experiment with the interior points of the same standard normally distributed dataset used in Subsection 3.3.1. The theoretical (3.14) and numerical probabilities for an interior point of this dataset are shown in Table 3.7 and the expectation (3.15) and variance (3.16) are shown in Table 3.8.

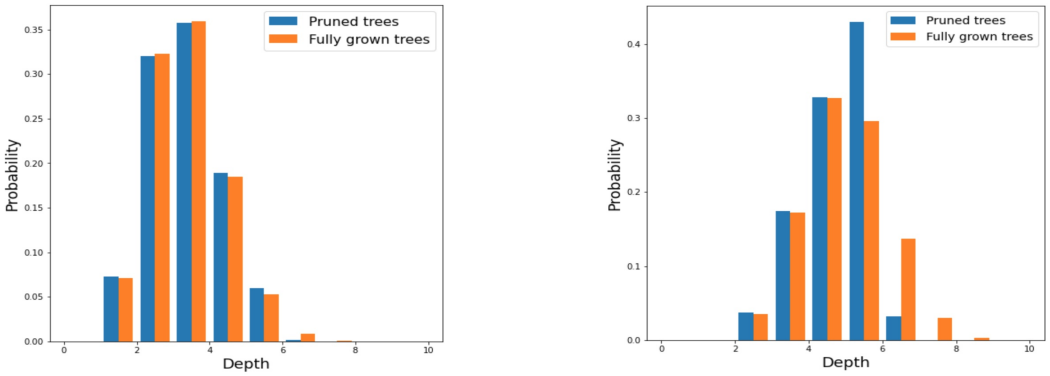
Hence, we can conclude from Table 3.7 and Table 3.8 that for this interior point the theoretical probabilities also match with the numerical probabilities just like the expectation and variance. This is also the case for another interior point in this dataset. In Figure 3.2, the comparison between the numerical probabilities for pruned and fully grown trees of the left fringe point and an interior point of the standard normally distributed dataset is shown. There are similar results as for the comparison between the left fringe point and an interior point of the uniform distribution, with differences in probabilities between the pruned trees and fully grown trees after the trees have been cut off.

Table 3.7: Theoretical and numerical probabilities that the interior point becomes isolated in s random splits.

Number of random splits	Theoretical probability	Isolation Forest (with fully grown trees)	Isolation Forest (with pruning)
1	0	0	0
2	0.03621	0.03510	0.03668
3	0.17466	0.17219	0.17398
4	0.32584	0.32701	0.32759
5	0.29613	0.29596	0.43009
6	0.13507	0.13696	0.03166
7	0.02910	0.02976	0
8	0.00288	0.00296	0
9	0.00011	0.00006	0

Table 3.8: Expectation and variance of the number of random splits that is needed to isolate the interior point.

	Expectation	**Variance**
Theoretical	4.419	1.291
IF (with fully grown trees)	4.429	1.289
IF (with pruning)	4.246	0.817



(a) Left fringe point

(b) Interior point

Figure 3.2: Probability that the left fringe point and an interior point of a standard normally distributed dataset become isolated in s random splits for the Isolation Forest [8] method with 100000 trees.

3.4 Numerical Validation Two-Dimensional Case

Just like for the 1D case, the theoretical formulas for the 2D case have been implemented in Python. There are more possible points now, because a point can be a fringe point or an interior point in both directions. Hence, four possible points should be distinguished. We only derived formulas for the fringe points in both directions, because deriving formulas for the other types of points turned out to be very computationally demanding.

3.4.1 Fringe Point in Both Directions

First, points that are a left fringe point in both the x-direction as the y-direction are investigated. These formulas have been implemented in Python and are tested numerically. The first dataset that is used for comparison is a uniform distributed set, consisting of 10 points and that has a left fringe point in both directions. The comparison between the theoretical (3.17) and numerical probabilities that this point gets isolated in s random splits are summarized in Table 3.9.

Table 3.9: Theoretical and numerical probabilities that the left fringe point in both directions becomes isolated in s random splits.

Number of random splits	Theoretical probability	Isolation Forest (with fully grown trees)	Isolation Forest (with pruning)
1	0.04862	0.04946	0.04739
2	0.21752	0.21919	0.21737
3	0.37930	0.37582	0.3819
4	0.25763	0.25818	0.2557
5	0.08209	0.08238	0.09648
6	0.01371	0.01381	0.00116
7	0.00109	0.00110	0
8	0.00004	0.00006	0
9	0.0000005	0	0

In Table 3.10, we see the theoretical expectation (3.21) and variance (3.22) compared with the numerical expectation and variance that this point gets isolated in s random splits.

Table 3.10: Expectation and variance of the number of random splits that is needed to isolate the left fringe point in both directions.

	Expectation	**Variance**
Theoretical	3.153	1.116
IF (with fully grown trees)	3.151	1.126
IF (with pruning)	3.140	1.039

From Table 3.9 and Table 3.10, it can be concluded that the theoretical probabilities match very well with the numerical probabilities for this specific left fringe point in both directions in this specific dataset. However, there are some differences after the height limit of the pruned trees is reached between the theoretical probabilities and numerical probabilities calculated with pruned trees.

To expand on this, a second dataset is used for comparison. This dataset is standard normally distributed, consists of 10 points and also has a left fringe point in both directions. The theoretical formulas (3.17) have been validated numerically again on this point, which can be seen in Table 3.11. The comparison between the theoretical expectation (3.21) and variance (3.22) and numerical expectation and variance that this point gets isolated in s random splits is summarized in Table 3.12.

The overall conclusion that can be drawn from the validations of the theoretical formulas is that the theoretical probabilities, expectations and variances are almost equal to the numerical ones with fully grown trees. For pruned trees, there are some differences due to the trees cut off at a certain point.

Table 3.11: Theoretical and numerical probabilities that the left fringe point in both directions becomes isolated in s random splits.

Number of random splits to isolate the point	Theoretical probability	Isolation Forest (with fully grown trees)	Isolation Forest (with pruning)
1	0.02992	0.02984	0.02995
2	0.17624	0.17823	0.17914
3	0.32081	0.31967	0.31931
4	0.27692	0.27529	0.276
5	0.14725	0.14802	0.1889
6	0.04293	0.04347	0.0067
7	0.00565	0.00526	0
8	0.00027	0.00022	0
9	0.000004	0	0

Table 3.12: Expectation and variance of the number of random splits that is needed to isolate the left fringe point in both directions.

	Expectation	**Variance**
Theoretical	3.488	1.407
IF (with fully grown trees)	3.486	1.410
IF (with pruning)	3.435	1.202

3.5 Outlier Detection with Transformed Isolation Forest

In Section 3.1, formulas were developed for the points in a one-dimensional dataset. The effort to expand this to a two-dimensional dataset turned out to be impractical. In this case, only formulas were developed for fringe points in both directions. For the other kinds of points, this was extremely difficult. However, Transformed Isolation Forest was developed which, with a transformation, turned a multi-dimensional dataset into a one-dimensional set-up. In this case, the one-dimensional theoretical formulas can again be applied. This method was described in Section 2.3. After this method is used, we have a new one-dimensional dataset. On this set-up, we can then use the formulas we derived in Section 3.1 for the one-dimensional case. The idea is then to calculate the theoretical expectation of the number of random splits that is needed to isolate every datapoint in this new dataset. The lower this expectation, the more outlying a point. Outliers are more susceptible to isolation than inliers and therefore they are isolated in less random splits than inliers. Hence, their expectation will be lower than the expectation of the inliers.

3.5.1 Counterexample for the Transformed Isolation Forest Based on Distance to the Center of Mass

The dataset described in Subsection 2.3.4 is used for the experiments with the new transformation methods. The four new transformed datasets can be seen in Figure 2.8 and by looking at this figure we expect the TIF in combination with the local transformation based on the kNN distance to detect all the four clear outliers and the TIF in combination with the global transformations to detect none of the clear outliers.

Thus, the expectation of the number of random splits needed to isolate each datapoint in the datasets of Figure 2.8 is calculated. The results of this can be seen in Table 3.13.

Table 3.13: Number of mistakes made in detecting the most outlying points for a dataset of 100 points coming from a two-dimensional dataset with four clear outliers.

Outlier detection method	**Mistakes made in detecting the 4 most outlying points**
Isolation Forest (1000 trees)	0
$E[RS]$ **of** $l_1(x_i, CM)$	3
$E[RS]$ **of** $l_2(x_i, CM)$	4
$E[RS]$ **of** $l_{max}(x_i, CM)$	4
$E[RS]$ **of kNN-distance with 3 neighbours**	0

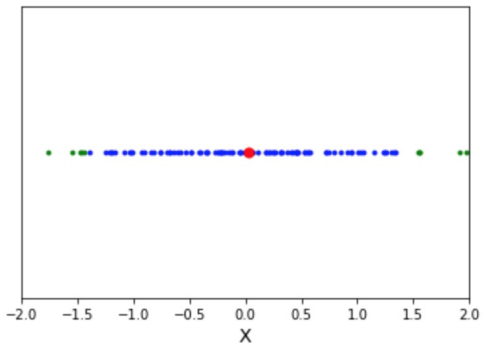
From Table 3.13, it can be concluded that our expectation was correct. Hence, the TIF in combination with the global transformations does not work on this dataset. The TIF in combination with the local transformation detects all four clear outliers and that also matches our expectation. Hence, for this dataset the local transformation gives more accurate results than the global transformations.

3.5.2 One-Dimensional Case

The datasets considered in this subsection are already one-dimensional. For these datasets, we will now first calculate the CM and then the distance of every datapoint to the CM. All these distances together will form a new one-dimensional dataset and on this dataset the formulas derived in Section 3.1 are used. With these formulas, the theoretical expectation of the number of random splits needed to isolate every point in this new dataset is calculated. The lower this expectation, the more outlying a point.

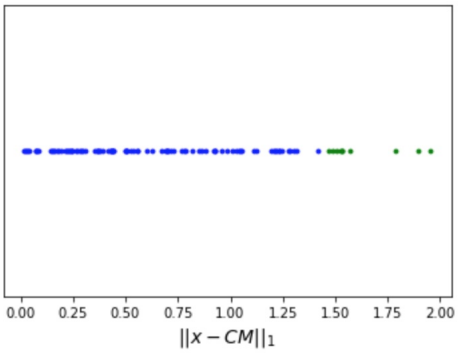
The first example that is considered is a standard normal distributed dataset of 100 points. This dataset can be seen in Figure 3.3.

Figure 3.3: Standard normally distributed dataset of 100 points.



The red dot in Figure 3.3 is the center of mass of this dataset. The green points are the 10 most outlying points in this dataset. We calculated the l_1-distance between the CM and every datapoint of this dataset. For one-dimensional datasets, the l_1-,l_2- and l_{max}-distances are the same. Therefore, we only checked the results for one of these metrics. This new dataset can be seen in Figure 3.4. In this figure, the green points are also the 10 most outlying points.

Figure 3.4: l_1-distance between the CM and every datapoint in the standard normal distributed dataset of Figure 3.3.



For the dataset of Figure 3.3, the most outlying points are calculated beforehand. This is done with 3 different metrics. The first one is the l_2-distance (2.16) to the mean of the dataset. The most outlying points have the largest distance to the mean. The second metric is the kNN-distance, already discussed in Subsection 2.3.3, which is the mean distance of every datapoint to his k-nearest neighbours. The bigger this kNN-distance is, the more outlying a point is. The third metric is the probability density function (PDF) of the distribution. The probability density function is a function that provides the likelihood that the value of a random variable will fall between a certain range of values. Therefore, the lower the PDF of a point, the more outlying it is.

The most outlying points calculated by these three metrics are compared with the most outlying points calculated by looking at the expectation of the number of random splits needed to isolate each datapoint in the dataset of Figure 3.4. The ordering of the outliers according to these three measures, is also taken into account. The results of this comparison can be seen in Table 3.14.

Table 3.14: Number of mistakes made in detecting the most outlying points for a dataset of 100 points coming from a standard normal distribution. Between brackets the number of mistakes if also the order is taken in consideration.

Outlier detection method and metric for the outliers	**Mistakes made in detecting the 5 most outlying points**	**Mistakes made in detecting the 10 most outlying points**
IF (1000 trees) and l_2-distance to the mean.	2 (5)	1 (9)
IF (1000 trees) and kNN-distance with $k = 50$.	1 (4)	1 (8)
IF (1000 trees) and PDF of the distribution.	2 (5)	1 (9)
IF (Theory) and l_2-distance to the mean.	2 (5)	2 (10)
IF (Theory) and kNN-distance with $k = 50$.	1 (4)	2 (9)
IF (Theory) and PDF of the distribution.	2 (5)	2 (10)
$E[RS]$ **of** $l_1(x_i, CM)$ **and l_2-distance to the mean.**	2 (2)	5 (6)
$E[RS]$ **of** $l_1(x_i, CM)$ **and kNN-distance with $k = 50$.**	1 (2)	5 (6)
$E[RS]$ **of** $l_1(x_i, CM)$ **and PDF of the distribution.**	2 (2)	5 (6)

In Table 3.14, we see in the mistakes columns two numbers. The first number is the number of mistakes made in detecting the most outlying points and the second number, between the brackets, is the number of ordering mistakes in detecting the most outlying points. The results of the theoretical Isolation Forest are calculated by using the theoretical formulas of Section 3.1

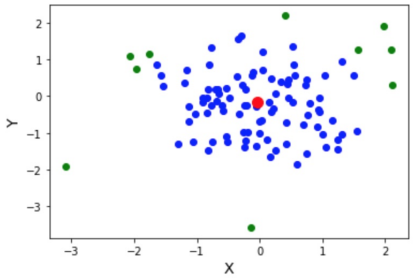
to get the theoretical expectation of the number of random splits needed to isolate every point in the original dataset. With these expectations, the scores (2.11) of the theoretical IF can be calculated and with these scores we can determine the number of outlying points detected by the theoretical IF method. We see that the expectation of the number of random splits needed to isolate every point in the dataset of Figure 3.4 gives us less ordering mistakes than both the theoretical and numerical IF on the original dataset. It also gives very accurate results when the detecting the 5 most outlying points.

3.5.3 Two-Dimensional Case

In this section, two-dimensional datasets are explored. In Chapter 3, we only derived formulas for fringe points in both directions for the two-dimensional case. Therefore, the distance of every datapoint to the CM of the dataset is calculated. This gives us a new one-dimensional dataset and on this dataset the theoretical formulas for the one-dimensional case are used. Hence, we can use the same techniques as we did for the one-dimensional case in Subsection 3.5.2.

The first two-dimensional dataset that is used for our experiment is a standard normally distributed dataset of 100 points. This dataset can be seen in Figure 3.5.

Figure 3.5: Two-dimensional standard normally distributed dataset of 100 points.

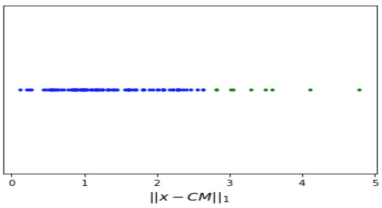


In Figure 3.5, the green dots are the 10 most outlying points in the dataset and the red dot is the CM of the dataset. For this dataset, the l_1-, l_2- and l_{max}-distances between the CM and every datapoint are calculated. This can be seen in Figure 3.6.

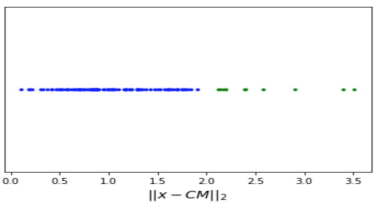
In Figure 3.6, it can be seen that these 3 distances are not the same for this two-dimensional dataset.

The most outlying points of the dataset of Figure 3.5 are calculated with the same three metrics used in Subsection 3.5.2. These are compared with the most outlying points calculated by looking at the expectation of the number of random splits needed to isolate each datapoint in the datasets of Figure 3.6. The ordering of the points is also taken into account. The results can be seen in Table 3.15. From there, it can be concluded that the numerical IF on the original two-dimensional dataset does a perfect job in detecting the 10 most outlying points. It also performs the best in ordering these points. If we look at the 5 most outlying points, the new transformation method in combination with the l_2-norm gives us very accurate results. It only makes 1 mistake when detecting the 5 most outlying points. Hence, the l_2-norm performs the best of the three norms. In Section 2.3, we also expected the l_2-norm to be the best choice for the new method based on the distance to the CM.

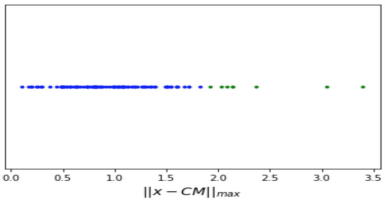
We also want to do this same experiment with a different dataset. Hence, the dataset we will use for the next experiment is a uniformly distributed set. Thus, it has no clear outliers and therefore it will be interesting to see which points will be detected as the most outlying points by each of the different norms. The PDF of the uniform distribution is constant for every



(a) l_1-distance



(b) l_2-distance



(c) l_{max}-distance

Figure 3.6: Distance between the CM and every datapoint in the standard normally distributed dataset of Figure 3.5.

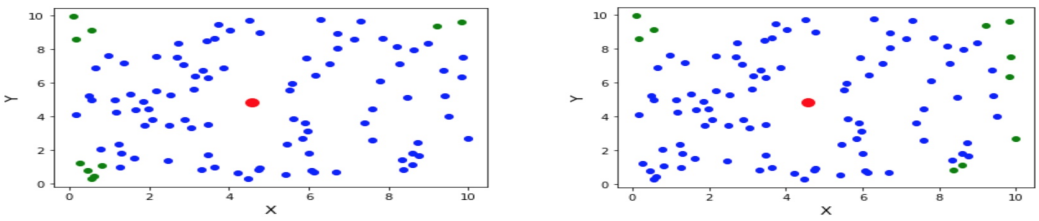
Table 3.15: Number of mistakes made in detecting the most outlying points for a dataset of 100 points coming from a two-dimensional standard normal distribution. Between brackets the number of mistakes if also the order is taken in consideration.

Outlier detection method and metric for the outliers	**Mistakes made in detecting the 5 most outlying points**	**Mistakes made in detecting the 10 most outlying points**
IF (1000 trees) and l_2-distance to the mean.	2 (2)	0 (4)
IF (1000 trees) and kNN-distance with $k = 50$.	1 (2)	0 (7)
IF (1000 trees) and PDF of the distribution.	2 (2)	0 (4)
$E[RS]$ **of** $l_1(x_i, CM)$ **and l_2-distance to the mean.**	1 (2)	4 (7)
$E[RS]$ **of** $l_1(x_i, CM)$ **and kNN-distance with $k = 50$.**	2 (3)	4 (8)
$E[RS]$ **of** $l_1(x_i, CM)$ **and PDF of the distribution.**	1 (2)	4 (7)
$E[RS]$ **of** $l_2(x_i, CM)$ **and l_2-distance to the mean.**	1 (2)	4 (6)
$E[RS]$ **of** $l_2(x_i, CM)$ **and kNN-distance with $k = 50$.**	1 (2)	4 (7)
$E[RS]$ **of** $l_2(x_i, CM)$ **and PDF of the distribution.**	1 (2)	4 (6)
$E[RS]$ **of** $l_{max}(x_i, CM)$ **and l_2-distance to the mean.**	3 (5)	4 (10)
$E[RS]$ **of** $l_{max}(x_i, CM)$ **and kNN-distance with $k = 50$.**	2 (5)	4 (10)
$E[RS]$ **of** $l_{max}(x_i, CM)$ **and PDF of the distribution.**	3 (5)	4 (10)

datapoint in the set. Hence, we cannot use this metric to calculate the most outlying points for this dataset. Therefore, we are only using the other two metrics in this case. The dataset is shown in Figure 3.7.

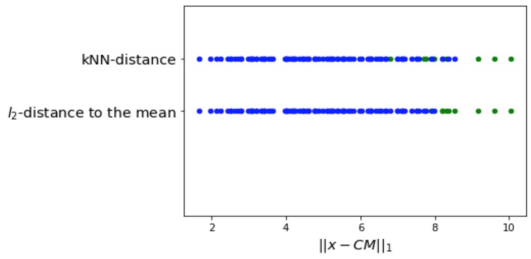
For the standard normally distributed dataset of Figure 3.5, the most outlying points detected by the three different metrics were the same points. In Figure 3.7, we see that this is not the case for this uniform distribution. Each metric considers different points as the most outlying points. This can also be seen if we calculate the l_1-, l_2- and l_{max}-distances between the CM and every datapoint. In Figure 3.8, we can see these distances with the most outlying points detected by the different metrics in green.

In Figure 3.8, we see differences between the three norms. The distances are from a different order and also the most outlying points are at different places in the three datasets. Moreover,

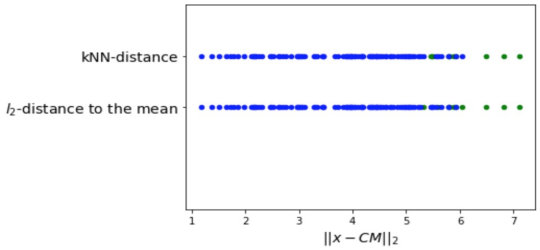


(a) l_2-distance to the mean (b) kNN-distance with 50 neighbours

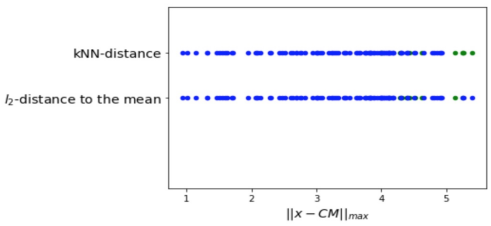
Figure 3.7: Two-dimensional uniformly distributed dataset of 100 points with the 10 most outlying points (green dots) detected by two different metrics.



(a) l_1-distance



(b) l_2-distance



(c) l_{max}-distance

Figure 3.8: Distance between the CM and every datapoint in the uniformly distributed dataset of Figure 3.7.

the most outlying points are different for the metrics we used to calculate them. We already saw this in Figure 3.7.

The most outlying points of the dataset of Figure 3.7 are thus calculated with the same three metrics used for the standard normally distributed dataset. These are compared with the most outlying points calculated by looking at the expectation of the number of random splits needed to isolate each datapoint in the datasets of Figure 3.8. The ordering of the points is also taken into account. The results of this can be seen in Table 3.16.

From Table 3.16, we can conclude that for this dataset all methods make more mistakes than for the standard normally distributed dataset. The reason for this may be the distribution itself, because an uniform distribution has no clear outliers. Therefore, it is more difficult to detect the outliers and that leads to more mistakes. Moreover, we notice that the new transformation method in combination with the l_1- and l_2-norms gives the most accurate results in terms of the detecting the 5 most outlying points. The numerical IF on the original two-dimensional dataset performs the best of all the methods considered, because it only makes 1 mistake when detecting the most outlying points. If the order is taken into consideration, then all methods make many mistakes.

Table 3.16: Number of mistakes made in detecting the most outlying points for a dataset of 100 points coming from a two-dimensional uniform distribution. Between brackets the number of mistakes if also the order is taken in consideration.

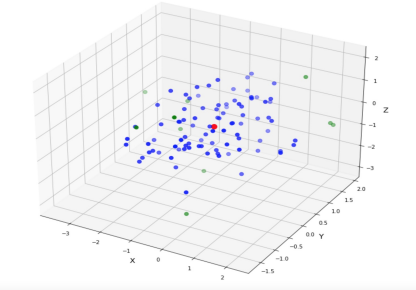
Outlier detection method and metric for the outliers	**Mistakes made in detecting the 5 most outlying points**	**Mistakes made in detecting the 10 most outlying points**
IF (1000 trees) and l_2-distance to the mean.	3 (4)	4 (9)
IF (1000 trees) and kNN-distance with $k = 50$.	1 (5)	3 (10)
$E[RS]$ **of** $l_1(x_i, CM)$ **and l_2-distance to the mean.**	3 (5)	5 (10)
$E[RS]$ **of** $l_1(x_i, CM)$ **and kNN-distance with $k = 50$.**	2 (4)	7 (9)
$E[RS]$ **of** $l_2(x_i, CM)$ **and l_2-distance to the mean.**	3 (5)	6 (10)
$E[RS]$ **of** $l_2(x_i, CM)$ **and kNN-distance with $k = 50$.**	2 (4)	7 (9)
$E[RS]$ **of** $l_{max}(x_i, CM)$ **and l_2-distance to the mean.**	4 (5)	9 (10)
$E[RS]$ **of** $l_{max}(x_i, CM)$ **and kNN-distance with $k = 50$.**	3 (5)	6 (9)

3.5.4 Three-Dimensional Case

The goal is to also perform the same experiments as we did with the one- and two-dimensional sets with datasets that are three-dimensional. These experiments are described in this subsection. These experiments are exactly the same as the ones for the two-dimensional datasets, but the only difference is now that the dataset has an extra dimension.

The first three-dimensional dataset that is used for this experiment is a standard normally distributed dataset of 100 points. This dataset can be seen in Figure 3.9.

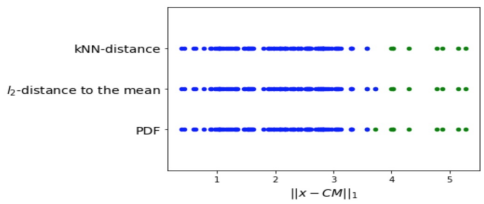
Figure 3.9: Three-dimensional standard normally distributed dataset of 100 points.



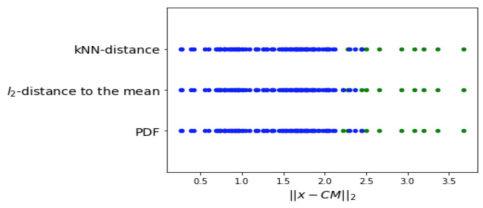
In Figure 3.9, the green points are the most outlying points. The red dot is the CM of this dataset. The three metrics we used for the two-dimensional set to calculate the most outlying points consider the same points as the outliers. However, there is one exception: the kNN-distance considers one point as an outlier, which the other two metrics does not consider as an outlier. The other 9 points are exactly the same. Therefore, there is only one figure of this dataset shown.

Again, the distances between the CM and every point in the dataset of Figure 3.9 are calculated for the three different metrics used in Subsection 3.5.3. This can be seen in Figure 3.10.

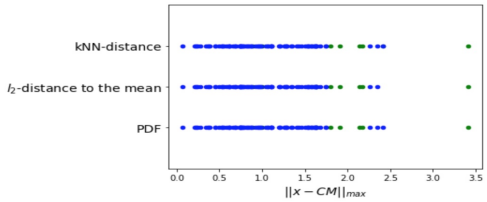
From Figure 3.10, we can conclude that there are differences between the three norms. In the dataset of the l_{max}-distance, the outliers are more mixed with the other points than for the datasets of the other two norms. Between the three metrics to calculate the most outlying points of the dataset beforehand, there are no major differences.



(a) l_1-distance



(b) l_2-distance



(c) l_{max}-distance

Figure 3.10: Distance between the CM and every datapoint in the standard normally distributed dataset of Figure 3.9.

The same experiment is done with these three new datasets as we have done with all the previous datasets we have considered in this section. The results of this can be seen in Table 3.17.

Table 3.17: Number of mistakes made in detecting the most outlying points for a dataset of 100 points coming from a three-dimensional standard normal distribution. Between brackets the number of mistakes if also the order is taken in consideration.

Outlier detection method and metric for the outliers	**Mistakes made in detecting the 5 most outlying points**	**Mistakes made in detecting the 10 most outlying points**
IF (1000 trees) and l_2-distance to the mean.	1 (3)	2 (8)
IF (1000 trees) and kNN-distance with $k = 50$.	1 (3)	2 (8)
IF (1000 trees) and PDF of the distribution.	1 (3)	2 (8)
$E[RS]$ **of** $l_1(x_i, CM)$ **and l_2-distance to the mean.**	2 (3)	4 (8)
$E[RS]$ **of** $l_1(x_i, CM)$ **and kNN-distance with $k = 50$.**	2 (3)	5 (8)
$E[RS]$ **of** $l_1(x_i, CM)$ **and PDF of the distribution.**	2 (3)	4 (8)
$E[RS]$ **of** $l_2(x_i, CM)$ **and l_2-distance to the mean.**	2 (3)	4 (8)
$E[RS]$ **of** $l_2(x_i, CM)$ **and kNN-distance with $k = 50$.**	2 (3)	4 (8)
$E[RS]$ **of** $l_2(x_i, CM)$ **and PDF of the distribution.**	2 (3)	4 (8)
$E[RS]$ **of** $l_{max}(x_i, CM)$ **and l_2-distance to the mean.**	4 (4)	5 (9)
$E[RS]$ **of** $l_{max}(x_i, CM)$ **and kNN-distance with $k = 50$.**	4 (4)	4 (9)
$E[RS]$ **of** $l_{max}(x_i, CM)$ **and PDF of the distribution.**	4 (4)	5 (9)

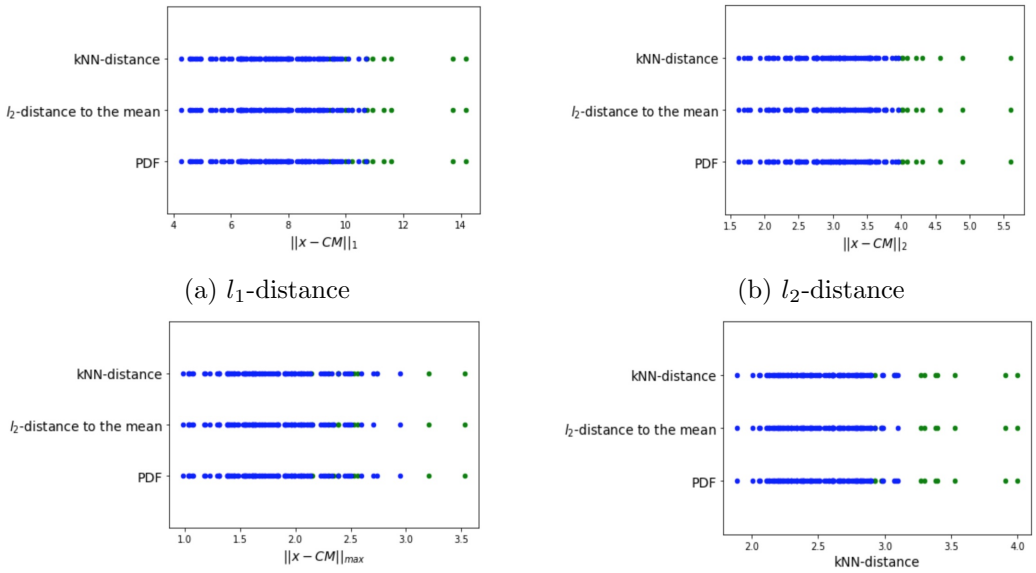
In Table 3.17, we see that the most accurate results come from the numerical IF on the original three-dimensional dataset. The new transformation method together with the l_{max}-norm has the least accurate results of the three norms considered.

3.5.5 Multi-Dimensional Case

The datasets that are considered in this section are multi-dimensional datasets, namely ten-dimensional datasets. With these datasets, the same experiments are done as with the lower

dimensional datasets of the previous subsections. This is done in order to see how the results of the lower dimensional datasets expand to higher dimensions.

The first multi-dimensional dataset that is used for these experiments is a ten-dimensional standard normally distributed dataset of 100 points. This dataset cannot be plotted, but we can visualize the l_1-, l_2- and l_{max}-distances between the CM and every datapoint in this set. For this set, we have also calculated the kNN-distance of every point with the number of neighbours equal to 5. Hence, this is the first set on which we use the TIF in combination with the local transformation based on the kNN-distance of every point. The new transformed datasets can be seen in Figure 3.11.



(a) l_1-distance (b) l_2-distance

(c) l_{max}-distance (d) kNN-distance

Figure 3.11: Transformed one-dimensional datasets of the ten-dimensional standard normally distributed dataset.

In Figure 3.11, we see differences between the four new datasets. For the kNN-distance, the most outlying points are separated the most from the rest of the points. The l_2-distance gives us a new dataset with more than half of the most outlying point separated from the inliers. If we look at the l_{max}-distance, we see that only 2 of the most outlying points are separated from the rest of the points. The other 8 most outlying points are mixed with the inliers. The l_1-distance gives us almost half of the outlying points clearly, but the other half is mixed with the inliers. Hence, we expect the kNN-distance to have the most accurate results followed by the l_2-distance.

For these four new datasets, the theoretical expectation of the number of random splits needed to isolate every datapoint is calculated and compared with the most outlying points calculated with the three metrics we used to calculate the outliers beforehand. The results of this experiment can be seen in Table 3.18. From this table, it can be concluded that the kNN-distance gives the most accurate results from the four methods if we are considering the 10 most outlying points. If we are considering the 5 most outlying points, then the new transformation method in combination with the l_1-norm gives us the most accurate results. However, the results of the new transformation method in combination with the l_2-norm and the kNN-distance do not deviate much from the results of the l_1-norm. The numerical IF still performs better than the new methods we consider

Table 3.18: Number of mistakes made in detecting the most outlying points for a dataset of 100 points coming from a ten-dimensional standard normal distribution. Between brackets the number of mistakes if also the order is taken in consideration.

Outlier detection method and metric for the outliers	**Mistakes made in detecting the 5 most outlying points**	**Mistakes made in detecting the 10 most outlying points**
IF (1000 trees) and l_2-distance to the mean.	1 (1)	2 (6)
IF (1000 trees) and kNN-distance with $k = 50$.	1 (2)	1 (6)
IF (1000 trees) and PDF of the distribution.	1 (1)	2 (6)
$E[RS]$ **of** $l_1(x_i, CM)$ **and l_2-distance to the mean.**	1 (4)	5 (9)
$E[RS]$ **of** $l_1(x_i, CM)$ **and kNN-distance with $k = 50$.**	2 (5)	5 (10)
$E[RS]$ **of** $l_1(x_i, CM)$ **and PDF of the distribution.**	1 (4)	5 (9)
$E[RS]$ **of** $l_2(x_i, CM)$ **and l_2-distance to the mean.**	2 (3)	5 (8)
$E[RS]$ **of** $l_2(x_i, CM)$ **and kNN-distance with $k = 50$.**	2 (3)	5 (8)
$E[RS]$ **of** $l_2(x_i, CM)$ **and PDF of the distribution.**	2 (3)	5 (8)
$E[RS]$ **of** $l_{max}(x_i, CM)$ **and l_2-distance to the mean.**	3 (4)	8 (9)
$E[RS]$ **of** $l_{max}(x_i, CM)$ **and kNN-distance with $k = 50$.**	3 (4)	7 (9)
$E[RS]$ **of** $l_{max}(x_i, CM)$ **and PDF of the distribution.**	3 (4)	8 (9)
$E[RS]$ **of kNN-distance with $k = 5$ and l_2-distance to the mean.**	2 (4)	5 (9)
$E[RS]$ **of kNN-distance with $k = 5$ and kNN-distance with $k = 50$.**	2 (4)	5 (8)
$E[RS]$ **of kNN-distance with $k = 5$ and PDF of the distribution.**	2 (4)	5 (9)
$E[RS]$ **of kNN-distance with $k = 10$ and l_2-distance to the mean.**	2 (3)	3 (8)
$E[RS]$ **of kNN-distance with $k = 10$ and kNN-distance with $k = 50$.**	2 (3)	3 (7)
$E[RS]$ **of kNN-distance with $k = 10$ and PDF of the distribution.**	2 (3)	3 (8)

If we look back at the two- and three-dimensional standard normally distributed datasets, we see the same as for the ten-dimensional set. The new transformation method in combination with the l_1- and l_2-norms gives the most accurate results from the global norms. Our expectation, in Section 2.3, was that the l_2-distance is the best global norm for this experiment, because it squares the values and thus increases the cost of outliers exponentially. The numerical IF performs the best out of all the methods we consider.

However, all these methods to detect the most outlying points still make some mistakes. Therefore, we look at the numerical probabilities that the datapoint in the original ten-dimensional dataset gets isolated in s random splits and compare this with the theoretical and numerical probabilities of the three transformed datasets. This is done to explore what the impact is of those transformations. When the number of dimensions is decreased, there will always be some information lost. However, the advantage of decreasing the number of dimensions to one is that we can use the theoretical formulas for the one-dimensional case and thus it helps with interpretable results. This helps us to detect the outliers of multi-dimensional datasets with the new transformation methods. Hence, there is always a field of tension between interpretability on the one side and loss of information on the other side with these transformations. With these new transformation methods, we tried to get a balance between those two. To explore what the impact is of the transformations, the probabilities for the right fringe point of the dataset from Figure 3.11a are shown in Table 3.19. This is the point with the largest distance to the CM of the dataset and thus should also be an outlier in the original dataset. Only the probabilities for the first 15 RS are considered and the numerical probabilities are calculated with fully grown trees

Table 3.19: Theoretical and numerical probabilities that the right fringe point in the transformed dataset becomes isolated in s random splits.

Number of random splits	Isolation Forest (original dataset)	Isolation Forest (Transformed dataset)	Theoretical probability (Transformed dataset)
1	0.0084	0.04670	0.04400
2	0.0296	0.2763	0.27665
3	0.0672	0.3382	0.33729
4	0.1083	0.2103	0.21508
5	0.1419	0.09260	0.09059
6	0.1397	0.02720	0.02803
7	0.1405	0.00730	0.00678
8	0.1167	0.00120	0.00133
9	0.0905	0.00020	0.00022
10	0.0613	0	0.00003
11	0.0398	0	0.000004
12	0.0269	0	0.0000004
13	0.0146	0	0.00000004
14	0.008	0	0.000000003
15	0.0041	0	0.0000000002

The probabilities for the right fringe point of the dataset from Figure 3.11b can be seen in Table 3.20.

Table 3.20: Theoretical and numerical probabilities that the right fringe point in the transformed dataset becomes isolated in s random splits.

Number of random splits	Isolation Forest (original dataset)	Isolation Forest (Transformed dataset)	Theoretical probability (Transformed dataset)
1	0.03680	0.1799	0.17508
2	0.05960	0.3311	0.32820
3	0.09250	0.2757	0.27890
4	0.1130	0.1426	0.14578
5	0.1179	0.05240	0.05340
6	0.1211	0.01420	0.01475
7	0.1156	0.00370	0.00322
8	0.1009	0.00030	0.00058
9	0.08220	0.00010	0.00009
10	0.05410	0	0.00001
11	0.04350	0	0.000001
12	0.02610	0	0.0000001
13	0.01530	0	0.00000001
14	0.01150	0	0.0000000008
15	0.00530	0	0.00000000005

The probabilities for the right fringe point of the dataset from Figure 3.11c can be seen in Table 3.21.

Table 3.21: Theoretical and numerical probabilities that the right fringe point in the transformed dataset becomes isolated in s random splits.

Number of random splits	Isolation Forest (original dataset)	Isolation Forest (Transformed dataset)	Theoretical probability (Transformed dataset)
1	0.03680	0.1293	0.12562
2	0.05960	0.293	0.29747
3	0.09250	0.3021	0.29909
4	0.1130	0.1725	0.17769
5	0.1179	0.0699	0.07214
6	0.1211	0.0245	0.02173
7	0.1156	0.007	0.00512
8	0.1009	0.0011	0.00098
9	0.08220	0.0006	0.00016
10	0.05410	0	0.00002
11	0.04350	0	0.000002
12	0.02610	0	0.0000003
13	0.01530	0	0.00000002
14	0.01150	0	0.000000002
15	0.00530	0	0.0000000001

The expected values following from Table 3.19, Table 3.20 and Table 3.21 are also calculated. The results of that are shown in Table 3.22.

Table 3.22: Expectation of the number of random splits that is needed to isolate the right fringe point in the transformed dataset.

	IF (Original dataset)	IF (Transformed dataset)	Theoretical (Transformed dataset)
$E(RS\|RS \leq 15)$ **of Table 3.19**	6.71	3.144	3.151
$E(RS\|RS \leq 15)$ **of Table 3.20**	6.330	2.616	2.635
$E(RS\|RS \leq 15)$ **of Table 3.21**	6.330	2.871	2.865

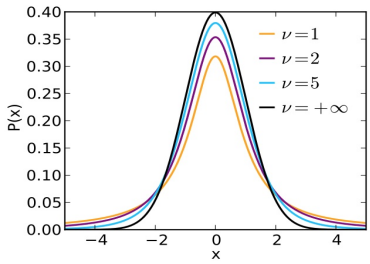
From Table 3.19, Table 3.20, Table 3.21 and Table 3.22, it can be concluded that the theoretical and numerical probabilities and expectations of the transformed datasets are matching for the right fringe point of these 3 sets. This is what we also saw when we numerically validated the theoretical formulas developed in Section 3.1. Hence, no surprise there.

What we also see in these four tables is that for the first 4 splits the numerical probabilities for the original dataset are much smaller than for the transformed datasets. However, for the last splits ($RS \geq 5$) the probabilities for the original dataset are much larger than the probabilities for the transformed datasets. Hence, we lose information in both cases. For the first 4 RS, the probabilities are too big in comparison with the original dataset and for the last 11 RS ($5 \leq RS \leq 15$) the probabilities are too small. Moreover, the (conditional) expectation of the number of RS needed to isolate these points is larger for the original dataset than for the transformed datasets. The reason for this could be the decrease of dimensions, because in multi-dimensional datasets you may need more random splits to isolate an outlier than in a one-dimensional dataset. This only has to do with the number of dimensions.

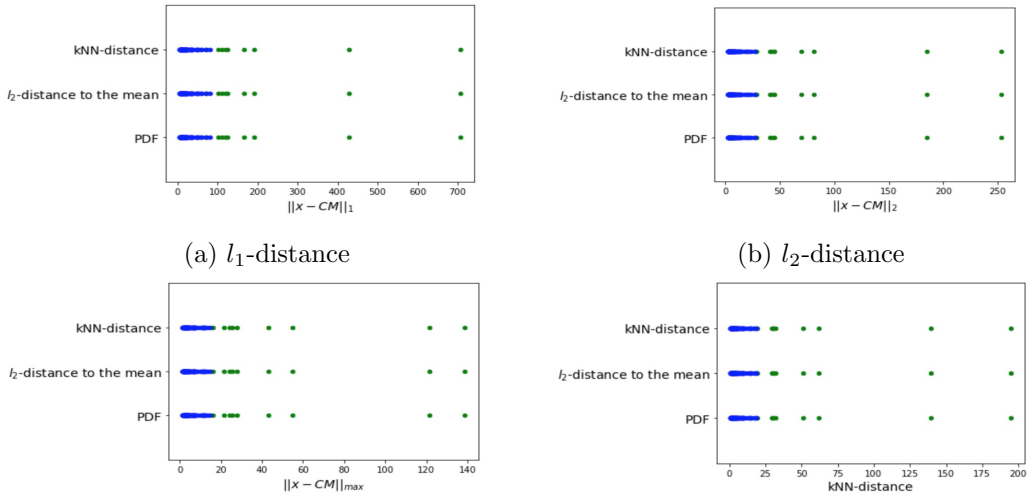
Finally, we are going to perform the same experiments as we did for the ten-dimensional standard normal distribution with a ten-dimensional t-distribution. The t-distribution is a family of continuous distributions that look almost identical to the normal distribution curve. The only

differences are that the curves are a bit shorter and it has fatter tails. The t-distribution has a parameter, the degrees of freedom, that can be used to change the curve of the distribution. The curve of the t-distribution comes closer to the curve of the normal distribution as the degrees of freedom increases. The probability density function of this distribution is shown in Figure 3.12.

Figure 3.12: Probability density function of one-dimensional t-distribution for different degrees of freedom (ν).



Hence, we see in Figure 3.12 that the tails are the fattest when the distribution has only 1 degree of freedom. In that case, the distribution has the most clear outliers. Therefore, we will consider a ten-dimensional t-distribution with 1 degree of freedom of 100 points. This is a dataset with clear outliers and thus the outliers should be easier detected than when we considered the standard normal distribution. Hence, we again have calculated the l_1-, l_2- and l_{max}-distances between the CM and every datapoint in this set. For this set, we have also calculated the kNN-distance of every point with the number of neighbours equal to 5. The new transformed datasets can be seen in Figure 3.13.



(a) l_1-distance

(b) l_2-distance

(c) l_{max}-distance

(d) kNN-distance

Figure 3.13: Transformed one-dimensional datasets of the ten-dimensional t-distribution with 1 degree of freedom.

In Figure 3.13, we see that for all the 4 new transformed one-dimensional datasets the outliers are clearly separated from the rest of the points. Thus, we expect that the transformation methods detect almost all the outliers for these specific datasets. The theoretical expectation of the number of random splits needed to isolate every datapoint is calculated and compared with the most outlying points calculated with the three metrics we used to calculate the outliers beforehand. The results of this experiment can be seen in Table 3.23.

Table 3.23: Number of mistakes made in detecting the most outlying points for a dataset of 100 points coming from a ten-dimensional t-distribution with 1 degree of freedom. Between brackets the number of mistakes if also the order is taken in consideration.

Outlier detection method and metric for the outliers	**Mistakes made in detecting the 5 most outlying points**	**Mistakes made in detecting the 10 most outlying points**
IF (1000 trees) and l_2-distance to the mean.	1 (1)	1 (5)
IF (1000 trees) and kNN-distance with $k = 50$.	1 (1)	1 (5)
IF (1000 trees) and PDF of the distribution.	1 (1)	1 (5)
$E[RS]$ **of** $l_1(x_i, CM)$ **and l_2-distance to the mean.**	1 (1)	2 (5)
$E[RS]$ **of** $l_1(x_i, CM)$ **and kNN-distance with $k = 50$.**	1 (1)	2 (5)
$E[RS]$ **of** $l_1(x_i, CM)$ **and PDF of the distribution.**	1 (1)	2 (5)
$E[RS]$ **of** $l_2(x_i, CM)$ **and l_2-distance to the mean.**	1 (1)	1 (5)
$E[RS]$ **of** $l_2(x_i, CM)$ **and kNN-distance with $k = 50$.**	1 (1)	1 (5)
$E[RS]$ **of** $l_2(x_i, CM)$ **and PDF of the distribution.**	1 (1)	1 (5)
$E[RS]$ **of** $l_{max}(x_i, CM)$ **and l_2-distance to the mean.**	0 (0)	1 (1)
$E[RS]$ **of** $l_{max}(x_i, CM)$ **and kNN-distance with $k = 50$.**	0 (0)	1 (1)
$E[RS]$ **of** $l_{max}(x_i, CM)$ **and PDF of the distribution.**	0 (0)	1 (1)
$E[RS]$ **of kNN-distance with $k = 5$ and l_2-distance to the mean.**	1 (1)	1 (6)
$E[RS]$ **of kNN-distance with $k = 5$ and kNN-distance with $k = 50$.**	1 (1)	1 (6)
$E[RS]$ **of kNN-distance with $k = 5$ and PDF of the distribution.**	1 (1)	1 (6)
$E[RS]$ **of kNN-distance with $k = 10$ and l_2-distance to the mean.**	1 (1)	1 (6)
$E[RS]$ **of kNN-distance with $k = 10$ and kNN-distance with $k = 50$.**	1 (1)	1 (6)
$E[RS]$ **of kNN-distance with $k = 10$ and PDF of the distribution.**	1 (1)	1 (6)

From Table 3.23, we can conclude that all the new transformation methods give us very accurate results for this specific dataset. Almost all outliers are detected by all the methods. When looking at the 5 most outlying points, the new transformation methods also make almost no ordering mistakes. In this case, the new transformation method in combination with the l_{max}-norm performs the best of all the methods. The 5 most outlying points are detected perfectly and in detecting the 10 most outlying points this method makes only 1 mistake.

3.5.6 Outlier Detection in Low-Dimensional Subspaces

In this subsection, we are going to explore a specific dataset and use the original IF and Transformed IF on this dataset to try to find a specific outlying point. This is done in order to see how robust our methods are.

The dataset we are using is a ten-dimensional normally distributed set of 101 points with correlation between the dimensions. The variance is equal to 1 in all the dimensions. However, the mean is not the same in every dimension. It is namely equal to 0 in i dimensions and equal to 4 in $10 - i$ dimensions for $i = 1, ..., 9$. The point we are considering is $(4, 4...., 4)$, which is in this case an outlier in i dimensions and an inlier in $10 - i$ dimensions. For different values of i and different values of the correlation, we are going to explore which methods detect this point as one of the 5 most outlying points in the set. The results of these experiments are summarized in Table 3.24.

In Table 3.24, we see that if the correlation increases the transformation method in combination with the l_1-distance has the most difficulties with detecting the point $(4, 4, ..., 4)$ as one of the 5 most outlying points. The values of i for which it is detected for this norm decreases when

Table 3.24: Values of i for which the outlier given by $(4, 4, ..., 4)$ is detected in a ten-dimensional normally distributed dataset for different values of correlation.

Outlier detection method	**Values of i for which the outlier is detected (Correlation = 0.25)**	**Values of i for which the outlier is detected (Correlation = 0.50)**	**Values of i for which the outlier is detected (Correlation = 0.75)**
IF (1000 trees)	2-9	2-9	3-9
$E[RS]$ **of** $l_1(x_i, CM)$	3-9	4-9	5-9
$E[RS]$ **of** $l_2(x_i, CM)$	2-9	2-9	3-9
$E[RS]$ **of** $l_{max}(x_i, CM)$	1-9	1-9	1-9
$E[RS]$ **of kNN-distance with** $k = 5$	2-9	1-9	1-9
$E[RS]$ **of kNN-distance with** $k = 10$	2-9	1-9	1-9

the correlation increases. The transformation method in combination with the l_2-distance did not obtain the best results of all the methods, but it still performs reasonably well. The same holds for the original IF method with 1000 trees. For this dataset, the transformation method in combination with the l_{max}- and kNN-distance gives us the most robust results. The l_{max}-distance detects the outlier in all the cases we considered. Almost the same holds for the kNN-distance with the only difference for a correlation of 0.25.

3.6 Summary

Summarizing, we have developed theoretical formulas for the fringe and interior points of one-dimensional datasets based on the number of random splits that the original Isolation Forest algorithm needs to isolate a point. We started with recursion formulas, which are computationally very demanding, and therefore we looked for expressions faster to implement. In the one-dimensional case, we have found these expressions for both the fringe and interior points. However, in the two-dimensional case only expressions for fringe points in both directions were found. After the development of these theoretical formulas, we have numerically validated them in Section 3.3 and Section 3.4. This was done by comparing the theoretical probabilities and numerical probabilities coming from the IF algorithm of multiple datasets. The IF algorithm was tested with and without pruning. The conclusion that can be drawn from this validation is that the theoretical and numerical probabilities were approximately equal when we considered fully grown trees for the specific datasets we tested. For the pruned trees, there are some differences between the theoretical and numerical probabilities. This is because the trees have been cut off at a certain height. The formulas developed in Section 3.3 were for the points in the one-dimensional case. The effort to expand this to a two-dimensional dataset turned out to be cumbersome. Therefore, in Section 3.5 new outlier detection methods were tested based on a transformation from multi-dimensional datasets to one-dimensional ones. On these new one-dimensional sets, we could then use the formulas for the one-dimensional case. We have looked at the number of mistakes made in detecting the most outlying points for multiple datasets. The TIF in combination with the global transformations, based on the distance to the CM of every point, made still some mistakes in detecting the outliers. However, the performances of the new global transformation method in combination with the l_1- and l_2-norms gave us the most accurate results of the three norms we looked at. A dataset that was already introduced in Subsection 2.3.4 was also used to test the new methods on. The TIF in combination with the three global transformations did not detect any of the four clear outliers in this set. The

last transformation we have considered is a new local transformation based on the kNN-distance of every point. The TIF in combination with this local transformation detected all clear outliers. Moreover, this transformation method was also tested on the ten-dimensional standard normally distributed dataset. It gave us the same results as the new transformation method in combination with the l_2-norm for this specific dataset. The numerical IF still gave the most accurate results of all the methods considered for all the datasets, except the t-distribution in Subsection 3.5.5, we considered in this chapter. This ten-dimensional t-distribution with 1 degree of freedom had very fat tails and therefore very clear outliers. Almost all of these outliers were detected by the new transformation methods and thus every method performed very well on this specific dataset. Finally, in Subsection 3.5.6, we explored the robustness of our methods. We did this by experimenting on a ten-dimensional normally distributed set of 101 points with correlation between the dimensions and variance 1 in every dimension. The mean was equal to 0 in i dimensions and equal to 4 in $10 - i$ dimensions for $i = 1, ..., 9$. The point we were considering was $(4, 4...., 4)$, which was in this case an outlier in i dimensions and an inlier in $10 - i$ dimensions. For different values of the correlation, we explored the values of i for which the methods detected this outlying point. The result of that was that the transformation method in combination with the l_1-distance had the worst performance of all the methods considered. The l_{max}- and kNN-distance gave us the most accurate results and were thus the most robust methods for this specific dataset. The overall conclusion that can be drawn from all the experiments done with these new transformation methods in this chapter is that the transformation method in combination with the l_2- and kNN-distance performs the best overall. In some experiments, these norms gave us the best performance of all norms and in the experiments where they weren't the best norms they still gave us reasonably accurate results. With the l_2-norm the cost of the outliers increase exponentially and the main advantage of the kNN-distance is that it considers the direct neighbourhood of every datapoint. These statements could be explanations for the very accurate performances of these two distances in combination with the Transformed IF method.

4. Further Rigorous Testing of Isolation Forest and Transformed Isolation Forest

In this chapter, the original Isolation Forest method and the Transformed Isolation Forest method are used for further testing and more experiments. In Section 4.1, multiple components of the Isolation Forest algorithm are rigorously tested. This is done to explore the impact of each component on the performance of the IF algorithm. In Section 4.2, more advanced components of the IF algorithm are tested: different score functions for the IF method are considered in Subsection 4.2.1 and the performances of the different outlier detection methods are compared in Subsection 4.2.2.

4.1 Tests with Classical Isolation Forest Components

The original Isolation Forest algorithm, as described in Section 2.2, has multiple components that can be changed to improve the results. In this section, three of these components are tested. In Section 3.1, multiple theoretical formulas were developed for this algorithm and the computing times of the implementations of these formulas are therefore compared in Subsection 4.1.1. The next component that is tested is the number of isolation trees that the algorithm uses. This is done in Subsection 4.1.2. Finally, the impact of pruning of the isolation trees is tested in Subsection 4.1.3.

4.1.1 Computing Times

In the one-dimensional case, all the theoretical probabilities for the interior points that have been calculated up until now, come from the first implementation in Python. This implementation is based on the first theoretical formulas of the interior points, which are given by (3.9). This works properly, but only for small datasets. For larger datasets, it is extremely slow due to the recursion formulas. The solution for this is found in the simplified formulas for the interior points, which are given by Equation (3.14). These formulas also have been implemented in Python and the result is a function that performs much better than the first implementation based on Equation (3.9). It has been tested on the same two datasets as the first implementation and that resulted in the same theoretical probabilities for both implementations. This means that both implementations are correct, but the major difference is that the second one is much faster. On the standard normally distributed dataset used in Section 3.3, both implementations have calculated all the theoretical probabilities for four different interior points. This has been done hundred times by both implementations. The results of this experiment are summarized in Table 4.1.

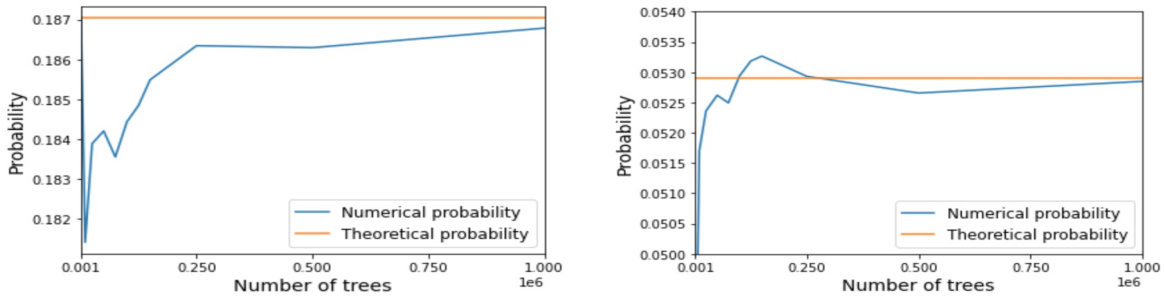
From Table 4.1, it can be concluded that the implementation based on Equation (3.14) is much faster than the implementation based on Equation (3.9) for the specific set-up we used.

Table 4.1: Computation times of two different implementations of the theoretical formulas of the interior points used on a standard normally distributed dataset (average of 100 runs).

Point	**Computation time** (3.9)	**Computation time** (3.14)
Interior point 1	44 s	0.17 s
Interior point 2	29.2 s	0.18 s
Interior point 3	53.2 s	0.18 s
Interior point 4	54.2 s	0.17 s

4.1.2 Number of Trees

In the Isolation Forest algorithm there are multiple variables that can be changed in order to improve the results. One of them is the number of trees: increasing this number will lead to better results, but also to a higher computation time. The goal is to find the number of trees that leads to accurate results with a computation time that is not too high and thus a balance between the two. In order to find this, the standard normally distributed dataset used in Section 3.3 with 10 datapoints is used again. The left fringe point is first explored: the numerical probabilities and the theory are compared for an increasing number of trees. In Figure 4.1, we see the numerical probabilities converging to the theoretical probabilities when the number of trees increases.



(a) 4 Random splits

(b) 5 Random splits

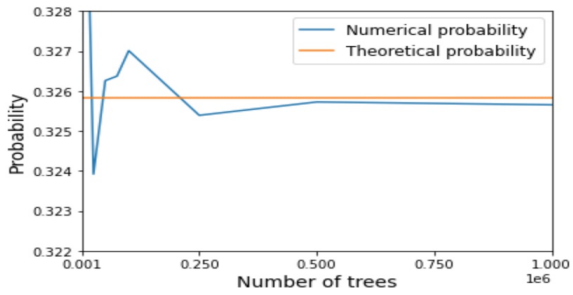
Figure 4.1: Probability that the left fringe point gets isolated in 4 and 5 random splits for a different number of trees.

This is also done with an interior point of this dataset. The corresponding results of that can be seen in Figure 4.2 for 4 and 5 random splits.

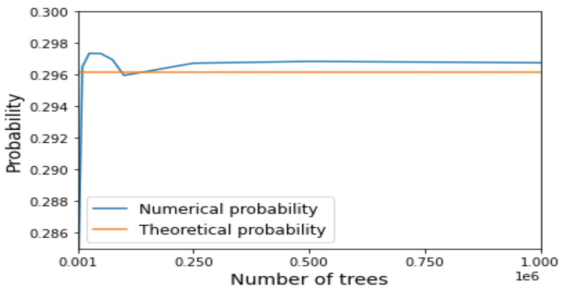
From Figure 4.1 and Figure 4.2, it can be concluded that for both points increasing the number of trees will lead to more accurate results. For these two points, the difference between the numerical and theoretical variances when increasing the number of trees has also been looked at. This has been done for multiple random seeds. First, both the numerical and theoretical variances are plotted for different numbers of trees. Next to these plots, the graph of the absolute difference between the two is shown in Figure 4.3 and Figure 4.4.

Finally, a log-log plot of the difference between the numerical and theoretical variances is created for both the left fringe point as well as for the interior point. This can be seen in Figure 4.5.

The slope of the two log-log plots in Figure 4.5 is calculated in order to find the rate of convergence. For the left fringe point, the slope is equal to -0.64. For the interior point, the slope is equal to -0.84. Note that, the slope of a log-log plot is equal to a rate of convergence of $N^{-\text{slope}}$ with N the number of trees. Hence, the numerical variance converges with approximately a

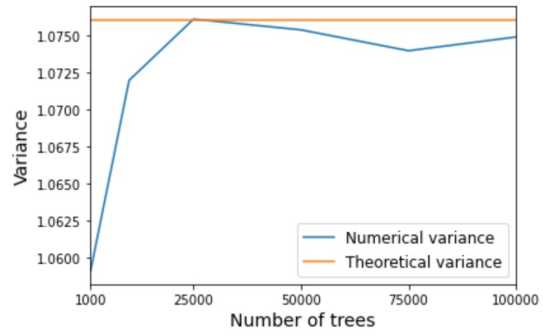


(a) 4 Random splits



(b) 5 Random splits

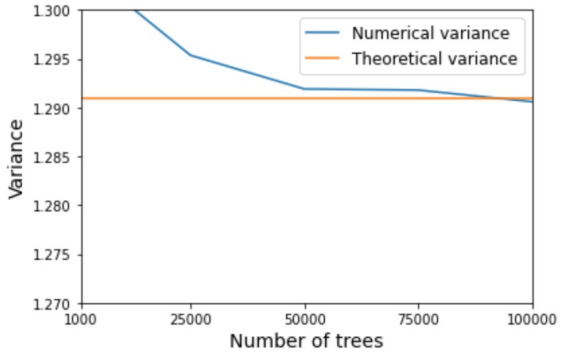
Figure 4.2: Probability that the interior point gets isolated in 4 and 5 random splits for a different number of trees.



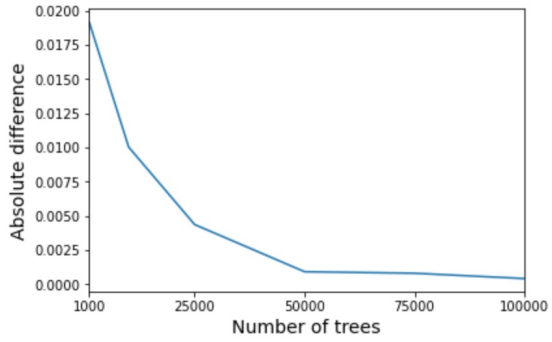
(a) Numerical and theoretical variance.

(b) Absolute difference between the numerical and theoretical variance

Figure 4.3: Variance of the number of random splits that is needed to isolate the left fringe point for a different number of trees, for multiple samples of Isolation Forest.



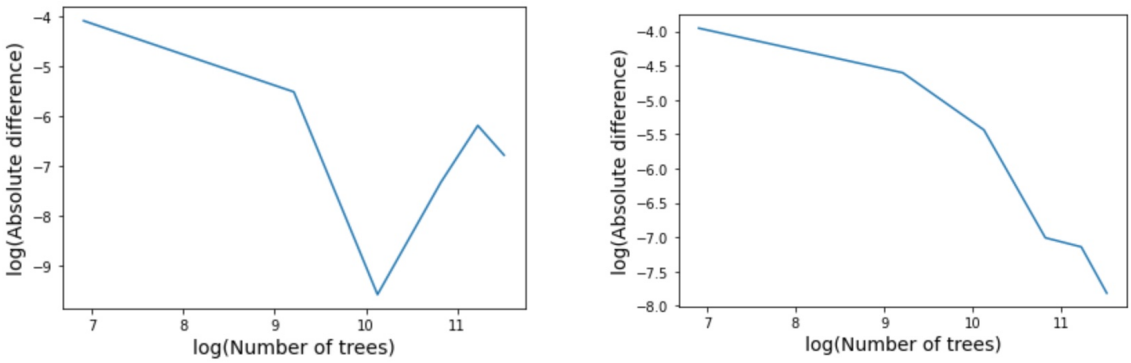
(a) Numerical and theoretical variance.



(b) Absolute difference between the numerical and theoretical variance.

Figure 4.4: Variance of the number of random splits that is needed to isolate the interior point for a different number of trees, for multiple samples of Isolation Forest.

rate of $N^{0.64}$ to the theoretical variance for the left fringe point of this specific dataset and with approximately a rate of $N^{0.84}$ for the interior point of this specific dataset.



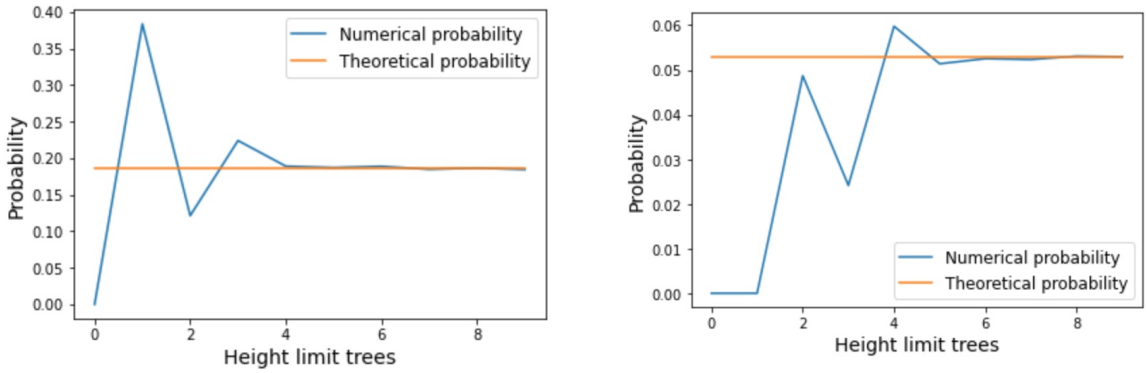
(a) Left fringe point

(b) Interior point

Figure 4.5: Log-log plot of the absolute difference between the numerical and theoretical variance for an increasing number of trees, for multiple samples of Isolation Forest.

4.1.3 Pruning

Pruning of trees in the Isolation Forest algorithm is a very interesting topic, because it saves a lot of computation time. Therefore, it would be important if the trees could be pruned as early as possible with as accurate results as the fully grown trees. To check this, the same dataset from Section 4.1.2 is used. First, the left fringe point of this dataset has been explored. The number of random splits to isolate this point is a constant here. For an increasing height limit of the trees, the difference between the theoretical probability and the numerical probability has been investigated. In Figure 4.6, this can be seen for 4 and 5 random splits for the left fringe point of this dataset:



(a) 4 Random splits

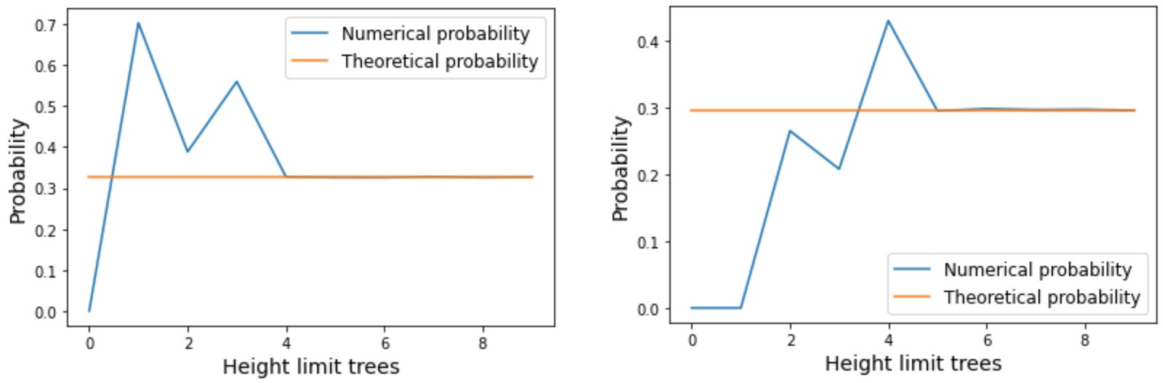
(b) 5 Random splits

Figure 4.6: Probability that the left fringe point gets isolated in 4 and 5 random splits with pruned trees.

Hence, in Figure 4.6 it is clear that for 4 random splits the numerical and theoretical probabilities are very similar for a height limit of minimal 4. For 5 random splits, this height limit has to be minimal 5, for this specific dataset.

The same experiments are done for an interior point of this dataset. The results can be seen in Figure 4.7.

In Figure 4.7, the results are similar as for the left fringe point above. Hence, it can be concluded for this set-up that for accurate numerical probabilities of a point getting isolated after s random



(a) 4 Random splits

(b) 5 Random splits

Figure 4.7: Probability that the interior point gets isolated in 4 and 5 random splits with pruned trees.

splits a height limit of s suffices.

4.2 Performance of Advanced Isolation Forest Components

The Isolation Forest algorithm also has more advanced components that have an impact on the performance of the algorithm. For example, the score function and therefore multiple different scoring functions are tested in Subsection 4.2.1. In Subsection 4.2.2, the performance of the Isolation Forest algorithm is compared with the performances of two other outlier detection methods. This is done by looking at the number of mistakes made in detecting the most outlying points for multiple different datasets.

4.2.1 Different Scoring Functions

Another interesting aspect of the IF method, is the score function and in particular how this function is defined. The article [13] proposes an enhanced score function on forest level. The usual score function makes use of the product rule and is defined as

$$\text{Score} = 2^{-\frac{\sum_{t\in\mathcal{F}} h_t(x)}{c(n)|\mathcal{F}|}} = 2^{-\frac{h_1(x)}{c(n)|\mathcal{F}|}} \cdots 2^{-\frac{h_{|\mathcal{F}|}(x)}{c(n)|\mathcal{F}|}}, \tag{4.1}$$

with $|\mathcal{F}|$ the number of trees. The authors of [13] use a different score function that makes use of the sum rule instead of product rule. This score function is defined as follows:

$$s(x) = \frac{\sum_{t\in\mathcal{F}} 2^{-\frac{h_t(x)}{c(n)}}}{|\mathcal{F}|}. \tag{4.2}$$

This score function may give a better performance, because the product rule may be inaccurate due to the fact that a single bad score can decrease the overall performance of the methodology significantly.

To check if this score function led to better results, the Isolation Forest algorithm from [8] has been extended with this new scoring function. After that, it is run on a dataset of 1000 points that is standard normally distributed. The results of this experiment are summarized in Table 4.2.

Table 4.2: Number of mistakes made on average, over multiple random datasets, in detecting the most outlying points for a dataset of 1000 points while using the code of [8] and the scoring function from [13]. Between brackets the number of mistakes if also the order is taken in consideration.

Outlier detection method	Mistakes made in detecting the 5 most outlying points	Mistakes made in detecting the 10 most outlying points	Mistakes made in detecting the 15 most outlying points	Mistakes made in detecting the 20 most outlying points
Isolation Forest (100 trees)	1.35 (3.2)	1.45 (7.15)	2.1 (11.55)	2.85 (16.2)
Isolation Forest (1000 trees)	1.25 (3.3)	1.3 (7.15)	2.25 (11.9)	2.95 (16.7)

Hence, the conclusion that can be drawn from the comparison of Table 4.2 with Table 4.7, is that this new scoring function does not lead to better results in this case. The reason why [13] got better results with this new scoring rule could be that they use 16 different datasets that encompass a wide variety of cases. They differ in size, in the number of features and in the percentage of outliers. The dataset that is used in this case is just a set with 1000 points that come from a standard normal distribution and therefore not so diverse as their data.

With the theoretical probabilities for the fringe and interior points from Section 3.1, the scores (2.11) can also be calculated and with that it is possible to look again at the ordering of the most outlying points. A mixture of three normal distributions is used. On this set, the three outlier detection methods have been tested and their performances are checked. This is done for one random seed only this time. The difference is, that the performances are now compared with the theory. This can be seen in Table 4.3.

Table 4.3: Number of mistakes made in detecting the most outlying points for a dataset of 1000 points with 980 points coming from a standard normal distribution, 10 points from a normal distribution with mean -15 and variance 1 and 10 points from a normal distribution with mean 15 and variance 1. Between brackets the number of mistakes if also the order is taken in consideration.

Outlier detection method	Mistakes made in ordering the 20 most outlying points
Local Outlier Factor	0 (18)
K-means Clustering	0 (8)
Isolation Forest (100 trees)	0 (19)
Isolation Forest (1000 trees)	0 (18)
Isolation Forest (Theory)	2 (19)

From Table 4.3, we can conclude that in theory Isolation Forest has an infinite number of trees, but this does not lead to major improvements in detecting the 20 most outlying points for this specific dataset. Moreover, in theory IF does not even detect all the 20 most outlying points.

For the dataset used in Table 4.3, we also looked at different score functions. The goal is to get a score function that makes fewer mistakes in the ordering of the 20 most outlying points. In Chapter 3, the whole distribution of the number of random splits that was needed to isolate a point was explored. Therefore, there are now many more options for the score function such as the variance or the conditional expectation of the number of random splits that is needed to isolate a point. First, the following score function is tested:

$$s_2(x) = 2^{-Var_x(RS)}. \tag{4.3}$$

This should lead to a higher score for outlying points than for inliers, because the variance is bigger for inliers than for outliers. Finally, the following score function is tested:

$$s_3(x) = E_x(RS|RS \leq i), \tag{4.4}$$

for different values of i. The expectation of the number of random splits needed to isolate a point conditional on a few random splits should be bigger for outliers than for inliers, because outliers are isolated in a smaller number of random splits. The results can be seen in Table 4.4. From Table 4.4, it can be concluded that these two new score functions are not significantly better than the original score function in terms of ordering mistakes. The conditional expectation as a score function (4.4) does not detect all the 20 most outlying points. The variance as a score function (4.3) performs very well on this specific dataset, because it detects all the most outlying points.

Table 4.4: Number of mistakes made in detecting the most outlying points for a dataset of 1000 points with 980 points coming from a standard normal distribution, 10 points from a normal distribution with mean -15 and variance 1 and 10 points from a normal distribution with mean 15 and variance 1. Between brackets the number of mistakes if also the order is taken in consideration.

Outlier detection method	**Mistakes made in detecting the 20 most outlying points**
Local Outlier Factor	0 (18)
K-means Clustering	0 (8)
IF with score function $E_x(RS\|RS \leq 2)$	3 (18)
IF with score function $E_x(RS\|RS \leq 3)$	3 (18)
IF with score function $E_x(RS\|RS \leq 4)$	4 (18)
IF with score function $2^{-Var_x(RS)}$	0 (20)

In all the previous tables, we looked at different score functions and compared the results with the 20 most outlying points and their ordering. These points were calculated by looking at the distance to the mean, with the most outlying points having the biggest distance to the mean. Another metric to calculate these points beforehand is also used, namely the kNN-distance for different numbers of neighbours. The kNN-distance is then the mean of the distance of every point to its neighbours. This was also used as a method to detect the most outlying points in Section 3.5. This was also compared with the new score functions and the results can be seen in Table 4.5.

From Table 4.5, it can be concluded that this new metric does not lead to a better ordering of the most outlying points for this specific dataset. There are still a lot of mistakes, also in detecting the most outlying points. Moreover, the new scoring functions do not give significantly better results in comparison with the original scoring function, in this case. Again, the variance as a score function (4.3) performs very well on this specific dataset, because it detects all the most outlying points.

Another metric is explored to calculate the most outlying points of a dataset, namely the probability density function of the distribution, already introduced and discussed in Section 3.5. Thus, the probability density function is a function that provides the likelihood that the value of a random variable will fall between a certain range of values. Therefore, the lower the PDF is of a point, the more outlying it will be. In these experiments, we are using a mixture distribution consisting of three normal distributions with a total of 1000 points. The first distribution has mean -15 and variance 1 and we are giving this distribution the weight $w_1 = \frac{10}{1000}$, the second

Table 4.5: Number of mistakes made in detecting the most outlying points for a dataset of 1000 points with 980 points coming from a standard normal distribution, 10 points from a normal distribution with mean -15 and variance 1 and 10 points from a normal distribution with mean 15 and variance 1. Between brackets the number of mistakes if also the order is taken in consideration.

Outlier detection method and metric for the outliers	**Mistakes made in detecting the 20 most outlying points**
Local Outlier Factor and kNN-distance with $k = 5$**.**	5 (20)
Local Outlier Factor and kNN-distance with $k = 10$**.**	2 (20)
Local Outlier Factor and kNN-distance with $k = 50$**.**	0 (18)
K-means Clustering and kNN-distance with $k = 5$**.**	5 (18)
K-means Clustering and kNN-distance with $k = 10$**.**	2 (19)
K-means Clustering and kNN-distance with $k = 50$**.**	0 (4)
IF (1000 trees) and kNN-distance with $k = 5$**.**	5 (20)
IF (1000 trees) and kNN-distance with $k = 10$**.**	2 (19)
IF (1000 trees) and kNN-distance with $k = 50$**.**	0 (16)
IF (Theory) and kNN-distance with $k = 5$**.**	4 (16)
IF (Theory) and kNN-distance with $k = 10$**.**	1 (17)
IF (Theory) and kNN-distance with $k = 50$**.**	2 (19)
IF with score function $E_x(RS\|RS \leq 2)$ **and kNN-distance with** $k = 5$**.**	3 (18)
IF with score function $E_x(RS\|RS \leq 2)$ **and kNN-distance with** $k = 10$**.**	2 (19)
IF with score function $E_x(RS\|RS \leq 2)$ **and kNN-distance with** $k = 50$**.**	3 (18)
IF with score function $E_x(RS\|RS \leq 3)$ **and kNN-distance with** $k = 5$**.**	3 (19)
IF with score function $E_x(RS\|RS \leq 3)$ **and kNN-distance with** $k = 10$**.**	2 (18)
IF with score function $E_x(RS\|RS \leq 3)$ **and kNN-distance with** $k = 50$**.**	3 (19)
IF with score function $E_x(RS\|RS \leq 4)$ **and kNN-distance with** $k = 5$**.**	3 (18)
IF with score function $E_x(RS\|RS \leq 4)$ **and kNN-distance with** $k = 10$**.**	3 (17)
IF with score function $E_x(RS\|RS \leq 4)$ **and kNN-distance with** $k = 50$**.**	4 (19)
IF with score function $2^{-Var_x(RS)}$ **and kNN-distance with** $k = 5$**.**	5 (19)
IF with score function $2^{-Var_x(RS)}$ **and kNN-distance with** $k = 10$**.**	2 (18)
IF with score function $2^{-Var_x(RS)}$ **and kNN-distance with** $k = 50$**.**	0 (20)

distribution has mean 0 and variance 1 and we are giving this distribution the weight $w_2 = \frac{980}{1000}$ and the third distribution has mean 15 and variance 1 and we are giving this distribution the weight $w_3 = \frac{10}{1000}$. Hence, this leads to the following PDF for this dataset:

$$f(x) = w_1 \cdot f_1(x) + w_2 \cdot f_2(x) + w_3 \cdot f_3(x)$$

$$= \frac{10}{1000} \cdot \frac{1}{\sqrt{2\pi}} e^{-\frac{(x+15)^2}{2}} + \frac{980}{1000} \cdot \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} + \frac{10}{1000} \cdot \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-15)^2}{2}} \tag{4.5}$$

with $f_1(x), f_2(x)$ and $f_3(x)$ the PDFs of the three different distributions. For the mixture distribution with PDF given by (4.5), the most outlying points are also compared with those detected by the different outlier detection methods we have considered together with the new scoring functions. The results of that can be seen in Table 4.6.

Table 4.6: Number of mistakes made in detecting the most outlying points for a dataset of 1000 points coming from a mixture distribution consisting of 3 different normal distributions. Between brackets the number of mistakes if also the order is taken in consideration.

Outlier detection method	**Mistakes made in detecting the 20 most outlying points**
Local Outlier Factor	5 (20)
K-means Clustering	7 (20)
IF (1000 trees)	4 (19)
IF (Theory)	3 (20)
IF with score function $2^{-Var_x(RS)}$	7 (20)
IF with score function $E_x(RS\|RS \leq 2)$	4 (18)
IF with score function $E_x(RS\|RS \leq 3)$	3 (18)
IF with score function $E_x(RS\|RS \leq 4)$	3 (18)

From Table 4.6, it can be concluded that this new metric does not lead to major improvements for this specific dataset. If we namely compare the results of this table with the results of Table 4.3, Table 4.4 and Table 4.5, then we see that all the methods make more mistakes when using this metric for calculating the most outlying points. Moreover, the new scoring functions for IF are not outperforming the original scoring function.

4.2.2 Different Outlier Detection Methods

The goal of the computational experiments is to compare the performances of different outlier detection methods. The first method is the Isolation Forest method [4, 8, 18]. The second method is called Local Outlier Factor [15, 25]. It compares the density of each object O of a dataset X with the density of the k nearest neighbours of O, see Subsection 2.1.2. The last method we consider is K-means clustering [25], which is described in Subsection 2.1.4. This method calculates for each point in a dataset the distance to the center of the nearest cluster. The experiments are done with a one-dimensional standard normally distributed dataset. For this set of datapoints, the 5, 10, 15 and 20 most outlying points are identified by calculating their distance to the mean of the distribution. The outliers are the points with the largest distance to the mean. The goal of the experiment is to understand how well these three methods detect the most outlying points. The first experiments that are done, were with one constant random seed

for the dataset and also for the Isolation Forest method. This gave us an interesting insight in how well these methods perform, but these kinds of results won't be used. The reason for this, is that it could be the case that for this particular random seed a method performs extremely well or bad and that does not reflect the overall performance in a good way. In Figure 4.8, the distribution of two random seeds for the data is pictured. This gives a clearer view of the minor differences between them. That will be the case for all random seeds and that is why these experiments are done with multiple random seeds for the dataset and also for the IF method.

(a) Random seed 19

(b) Random seed 10

Figure 4.8: Distribution of a standard normal distribution Y with 1000 datapoints created with 2 different random seeds.

The average of this was taken and summarized in Table 4.7.

Table 4.7: Number of mistakes made on average, over multiple samples for Isolation Forest and over multiple random datasets, in detecting the most outlying points for a dataset of 1000 points. Between the brackets is the number of mistakes if also the order is taken in consideration.

Outlier detection method	Mistakes made in detecting the 5 most outlying points	Mistakes made in detecting the 10 most outlying points	Mistakes made in detecting the 15 most outlying points	Mistakes made in detecting the 20 most outlying points
Local Outlier Factor	1.15 (2.6)	1.95 (6.6)	2.8 (11.25)	3.3 (16.0)
K-means Clustering	0.2 (0.8)	0.4 (2.55)	0.45 (5.45)	0.55 (8.95)
Isolation Forest (100 trees)	0.74 (2.64)	1.0 (5.93)	1.49 (9.84)	1.15 (14.04)
Isolation Forest (1000 trees)	0.565 (1.65)	0.87 (4.50)	1.32 (8.29)	0.95 (12.29)

On average, IF and K-means clustering are very accurate in detecting the most outlying points if the order is not taken in consideration. The LOF method has the worst performance of the three, but is not the main method that is considered in this thesis. This comparison is between the Isolation Forest method, a density-based method, namely the LOF and to a distance-based method, namely the K-means clustering method. The IF method performs well in comparison to the LOF, but the K-means clustering method performs the best. The reason of this very good performance is that this algorithm works in an equivalent way as how the most outlying points have been defined. We also see that the IF method performs better if the number of trees increases. If the order of the top outlying points is also taken into account, then all three methods are not accurate in terms of the detection. To try to understand this, first a new dataset was created and this looked as follows: 980 points are coming from a standard normal distribution, 10 points from a normal distribution with mean -15 and variance 1 and 10 points from a normal distribution with mean 15 and variance 1. On this new set, all three methods are tested and their performances are checked. It was expected, that the methods would detect all 20 top outlying points and also get the order right. The first prognosis was indeed correct, but the second one didn't. This can be seen in Table 4.8.

The results in Table 4.8 are clear. All three methods detect the 20 top outlying points without

Table 4.8: Number of mistakes made on average, over multiple samples for Isolation Forest and over multiple random datasets, in detecting the most outlying points for a dataset of 1000 points with 980 points coming from a standard normal distribution, 10 points from a normal distribution with mean -15 and variance 1 and 10 points from a normal distribution with mean 15 and variance 1. Between brackets the number of mistakes if also the order is taken in consideration.

Outlier detection method	Mistakes made in detecting the 20 most outlying points
Local Outlier Factor	0.0 (17.55)
K-means Clustering	0.0 (3.15)
Isolation Forest (100 trees)	0.0 (17.32)
Isolation Forest (1000 trees)	0.0 (15.49)

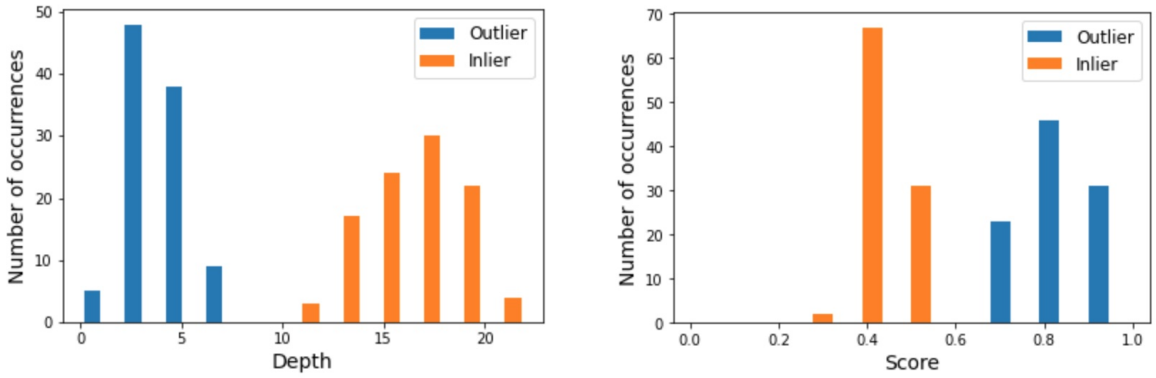
mistakes, but if the order is taken into account results are very different for two of them. The K-means clustering method is the only method of the three that still performs well if the order is taken into consideration. However, it also has some disadvantages: it requires to specify the number of clusters (k) in advance and it assumes that we deal with spherical clusters and that each cluster has roughly an equal numbers of observations. The IF method and LOF method cannot get the order right of the outliers. In fact, they both have almost the complete ordering incorrect. This is the reason why the variances of the scores are calculated, from the IF method, of these 20 most outlying points in Table 4.9. The variances of the depths of the points in the IF trees, which is just the number of RS to isolate the points, of these points are also calculated and listed in Table 4.9. The reason that we are looking at this is that if the scores or depths of the points are close to each other and the variances are large this may lead to a different ordering and thus to ordering mistakes.

Table 4.9: Variance of the scores of the 20 most outlying points for a dataset of 1000 points while using the code from [8] with 100 trees.

Score	**Variance of the score**	**Variance of the depth**
0.831	0.00397	2.068
0.811	0.00457	2.462
0.763	0.00345	2.116
0.760	0.00365	2.320
0.738	0.00498	3.304
0.701	0.00483	3.422
0.695	0.00482	3.765
0.689	0.00465	3.670
0.679	0.00494	3.702
0.652	0.00567	4.930
0.651	0.00396	3.310
0.649	0.00564	4.942
0.643	0.00469	4.294
0.641	0.00521	4.631
0.637	0.00430	3.725
0.635	0.00483	4.077
0.634	0.00416	3.609
0.627	0.00481	4.121
0.624	0.00496	4.273
0.614	0.00588	5.418

The scores in Table 4.9 are very close to each other, but they have also very small variances. However, the variances of the depths are much larger and that may be an explanation for the mistakes in the ordering that the IF method makes.

The Isolation Forest method uses the depth of a point to calculate the score. The intuition behind this is: the longer it takes to isolate a point, the lower the score and the more inlying this point will be. Hence, the depth and the score are very important quantities to detect outlying points and to distinguish them from the inliers. This is the reason why a closer look is taken at them, which is presented in Figure 4.9.



(a) Depth

(b) Score

Figure 4.9: Methods of distinguishing the top outlier and top inlier in all the 100 trees from the Isolation Forest method.

The points in Figure 4.9 are the top outlier and top inlier and therefore the points which differ the most from each other in the dataset. From the figures, it is clear that the scores are much closer to each other than the depths for these points in this specific dataset. Hence, the depth is a better way to distinguish outliers from inliers in this case. However, both the score and the depth will be sufficient to rank these points coming from this specific dataset.

4.3 Summary

Summarizing, in Section 4.1 three of the components of the Isolation Forest algorithm that can be changed to improve the performance of the algorithm were tested. In Subsection 4.1.1, the computation times of two different implementations of the theoretical formulas for the interior points in the one-dimensional case were compared. The implementation based on Equation (3.14) was much faster than the implementation based on Equation (3.9) for the specific dataset we considered. The second variable of the IF method that was investigated more thoroughly was the number of trees. In this case, the difference between the theoretical and numerical probabilities was explored for an increasing number of trees. An increasing number of trees led to more accurate numerical probabilities. In our set-up, at least 100000 trees were needed to get accurate probabilities. It is preferable to use more, but that leads to higher computation times which is something we want to avoid. The key is to find a balance between these two. Moreover, the theoretical and numerical variances were compared for an increasing number of trees and a log-log plot was created for these differences. The slope of these plots is equal to a rate of convergence of N^{-slope} with N the number of trees. The last component of the IF method, the pruning of the isolation trees, was investigated more thoroughly in Subsection 4.1.3. For an increasing height limit, we looked at the difference between the theoretical and

numerical probabilities to isolate a datapoint in 4 or 5 random splits. This was done with a fringe point and an interior point of the dataset used in Section 3.3. For the set-up we used, the numerical probabilities for a point getting isolated after s random splits were accurate for a height limit greater equal than s. Furthermore, in Section 4.2 more advanced components that have an impact on the performance of the algorithm were investigated. In Subsection 4.2.1, different score functions for the IF method were introduced. Also, different metrics were used to calculate the most outlying points beforehand to compare this with the results of the methods. The conclusion that could be drawn is that the new scoring functions were not more accurate in terms of mistakes made in detecting the most outlying points for the specific datasets we considered. The performances of three different outlier detection methods were compared in Subsection 4.2.2. This is done by looking at the number of mistakes made in detecting the most outlying points on average over multiple random datasets. Isolation Forest and K-means clustering were very accurate in detecting the most outlying points if the order is not taken in consideration. The third method, LOF, was the least accurate of the three. All three methods were not accurate when the order was taken into consideration. However, K-means clustering was the most accurate of the three methods on average over these multiple random datasets. Another dataset was tested, which had 20 clear outliers. All these outliers were detected by the three methods, but the IF and LOF method made still lots of ordering mistakes. Therefore, the variance of the scores and variance of the depths of the IF method were calculated. The scores themselves were very close to each other, but also the variances of these scores were very small. However, the variances of the depths were much larger and that may be the reason for the ordering mistakes.

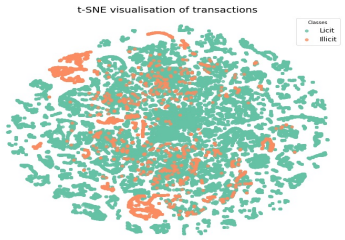
5. Real-World Financial Dataset

In this chapter, we use a real-world financial dataset to test how our methods perform on such a dataset. The dataset is introduced in Section 5.1. In Subsection 5.1.1, the comparison between the original Isolation Forest and Transformed Isolation Forest is made to see which method gives more accurate results on real-world data instead of on randomly generated small datasets. We also want to test what the impact of pruning the isolation trees, which is already introduced on small datasets in Subsection 4.1.3, is on the performance of IF on this dataset. This is done in Subsection 5.1.2. Moreover, we want to test some of the new scoring functions described in Subsection 4.2.1 on this dataset. These tests are done in Subsection 5.1.3.

5.1 Elliptic Dataset

The financial dataset that we will use for these tests is the Elliptic dataset. This dataset maps Bitcoin transactions to real entities belonging to licit categories (exchanges, wallet providers, miners, licit services, etc.) versus illicit ones (scams, malware, terrorist organizations, ransomware, etc.). It is the worlds largest labelled transaction dataset publicly available in any cryptocurrency. Moreover, it is a very imbalanced dataset with 4545 (9.76%) illicit and 42019 (90.24%) licit transactions. Every transaction in this dataset has 93 features and thus it is a 93-dimensional dataset. This cannot be visualized, but we can use t-distributed Stochastic Neighbor Embedding (t-SNE) [9] to visualize the data in 2D. This method is namely used for dimensionality reduction and that is used to visualize high-dimensional data in 2D and 3D. The t-SNE visualization of this dataset can be seen in Figure 5.1.

Figure 5.1: t-SNE visualization of the Elliptic dataset.



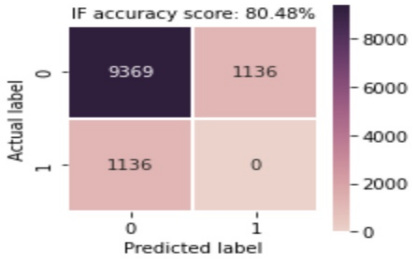
In Figure 5.1, we see that the illicit (orange) and licit (green) transactions are very mixed. There is no clear separation between the two classes and therefore we expect that our outlier detection algorithms will have problems with detecting the illicit transactions correctly. We will namely compare the outliers detected by our methods with the illicit transactions.

5.1.1 Tests with Original Isolation Forest and Transformed Isolation Forest

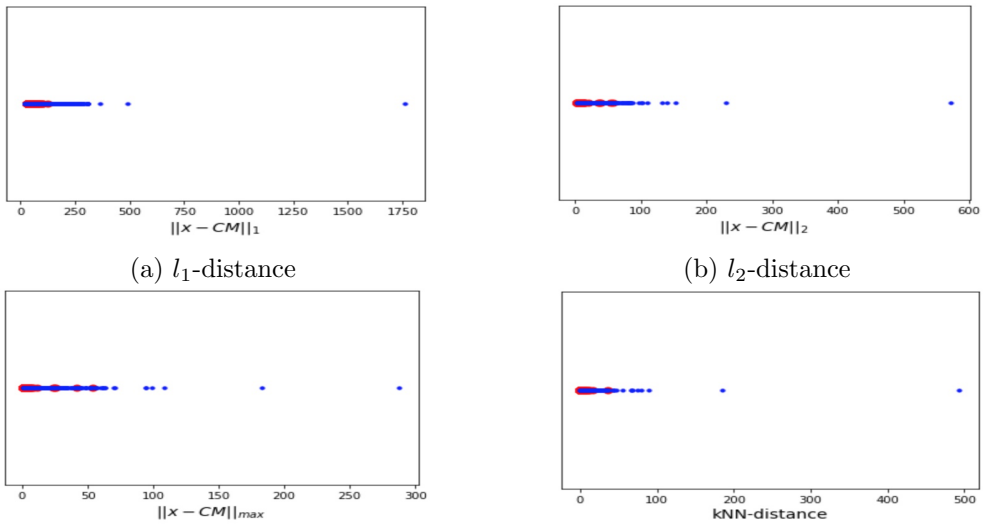
First, we have tested this dataset with the original Isolation Forest algorithm with 10000 trees, with the implementation of Scikit-learn, and we have compared the outliers detected by the IF with the transactions that are labelled as illicit. We calculated the anomaly scores (2.11) of every datapoint in the set and then we labelled the largest scores as illicit. The rest of the points we labelled as licit. We used the confusion matrix, already introduced in Subsection 2.1.5, as

a measure to assess the performance of IF. This confusion matrix can be seen in Figure 5.2. In this confusion matrix, the licit transactions are labelled as 0 and the illicit transactions are labelled as 1.

Figure 5.2: Confusion matrix of the original IF method (10000 trees) used on the Elliptic dataset.



From Figure 5.2, we can conclude that the original IF method does not detect any of the illicit transactions of the Elliptic dataset. The overall accuracy score is pretty good, but that is not the most important in this case. What we namely want and what is also very important for many financial institutions is that the illicit transactions are detected. This is far more important than detecting the licit transactions and thus we are looking for better methods in this case. Therefore, we also used the Transformed Isolation Forest on this dataset. First, we have calculated the CM of this dataset and with this we have calculated the l_1-, l_2- and l_{max}-distances between the CM and every datapoint in this set. For this set, we have also calculated the kNN-distance of every point with the number of neighbours equal to 10. These transformed datasets are shown in Figure 5.3.



(a) l_1-distance

(b) l_2-distance

(c) l_{max}-distance

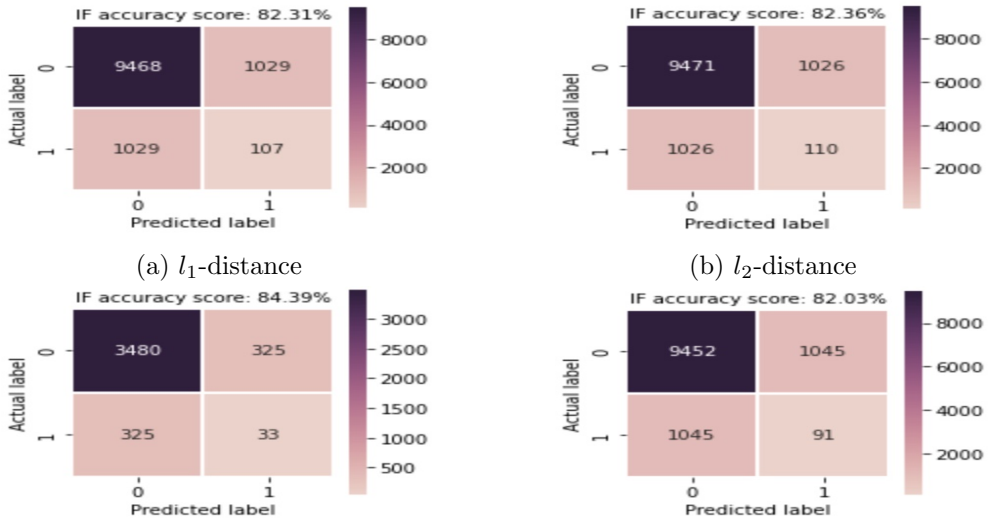
(d) kNN-distance

Figure 5.3: Transformed one-dimensional datasets of the Elliptic dataset.

In Figure 5.3, the red points are the illicit transactions and the blue points are the licit transactions and thus we see that these are mixed with each other. In these new transformed datasets, the illicit transactions are not clear outlying points and thus we do not expect the Transformed Isolation Forest to give us very accurate results in this case.

In this dataset, there are multiple points with exactly the same distances to the CM or with the same kNN-distances. Hence, there are some duplicate points in the transformed datasets of Figure 5.3. The duplicate points cannot be isolated and therefore we deleted the duplicate points

from these sets such that only one point of each duplicate remains in the set. The theoretical expectation of the number of random splits needed to isolate every datapoint is calculated and compared with the transactions that are labelled as illicit. The results of this experiment can be seen in Figure 5.4.



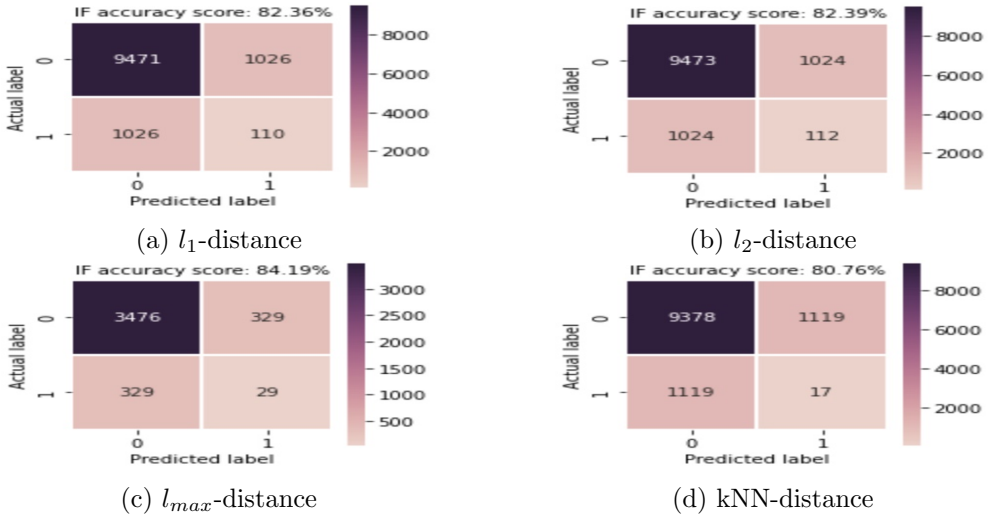
(a) l_1-distance (b) l_2-distance

(c) l_{max}-distance (d) kNN-distance

Figure 5.4: Confusion matrices of the four transformed datasets of the Elliptic dataset.

From Figure 5.4, we can conclude that the Transformed IF gives more accurate results than the original IF in this case. The overall accuracy scores are higher and most importantly more illicit transactions are detected. Moreover, we notice that the dataset of Figure 5.4c is much smaller than the other datasets and than the original dataset. This dataset has namely a lot of duplicate points and those points were deleted.

Finally, we also used the original numerical Isolation Forest algorithm on these 4 transformed datasets to get a better comparison between the methods. The confusion matrices of these tests are shown in Figure 5.5.



(a) l_1-distance (b) l_2-distance

(c) l_{max}-distance (d) kNN-distance

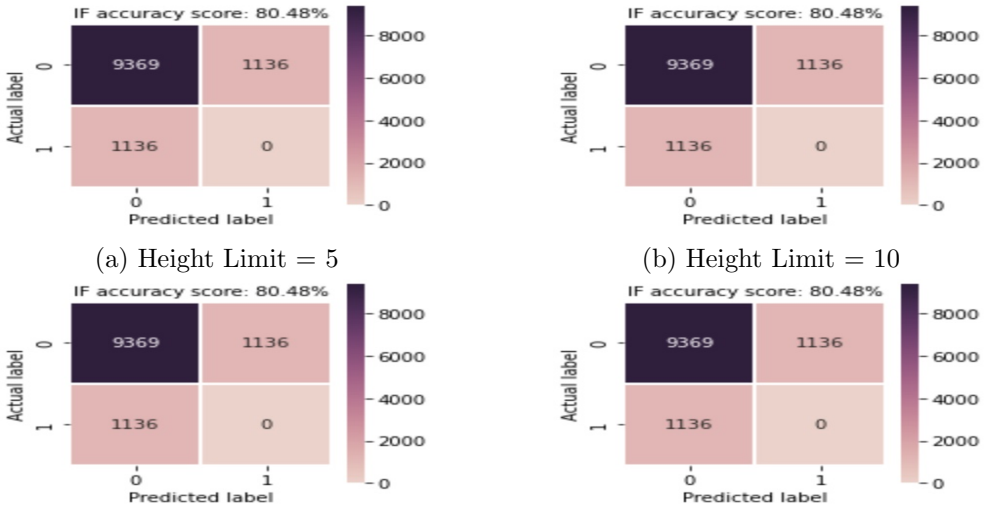
Figure 5.5: Confusion matrices of the original IF method (10000 trees) used on the four transformed datasets of the Elliptic dataset.

From Figure 5.5, we can conclude that for the transformation methods based on the distance to the CM of the whole dataset the results are approximately the same as the results of the original IF used on the same datasets. However, the results are more accurate than the results of the original IF used on the original dataset with 93 features. Thus, the transformation to a one-dimensional dataset gives us more accurate results in this case. The transformation method based on the kNN-distance of every point gives more accurate results than the original IF method used on the same dataset. The original IF method namely detects less illicit transactions than the transformation method. Again, the results are more accurate than the results of the original IF used on the original dataset with 93 features. Hence, also in this case the transformation to a one-dimensional dataset gives us more accurate results.

5.1.2 Pruning

The next component of the original IF that we will test with this real-world dataset is the impact of the pruning of the isolation trees. This is done in this subsection.

For different height limits of the isolation trees, we will run the original IF method with 1000 trees, this time with the implementation developed in [8] instead of the Scikit-learn implementation, on the dataset with 93 dimensions. The anomaly scores (2.11) of every datapoint are calculated and with these scores we determined the illicit points. The points with the largest scores are namely labelled as illicit. The illicit points detected by the original IF method are again compared with the transactions that are labelled as illicit beforehand. The most important aspect of this experiment is to explore what the impact is of pruning the trees on the performance of the original IF method as an outlier detection method. The results of these experiments are shown in Figure 5.6.



(a) Height Limit = 5

(b) Height Limit = 10

(c) Height Limit = 20

(d) Height Limit = 50

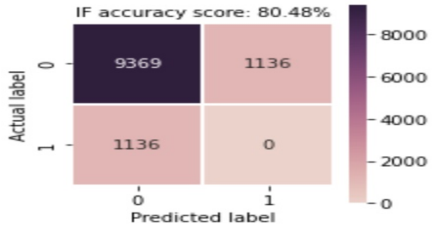
Figure 5.6: Confusion matrices of the original IF method (1000 trees) used on the Elliptic dataset for different height limits of the trees.

We can conclude from Figure 5.6 that pruning of the isolation trees has no impact on the results of these experiments. For different height limits, there are no differences in the number of illicit transactions that are detected and also no differences in the overall accuracy scores.

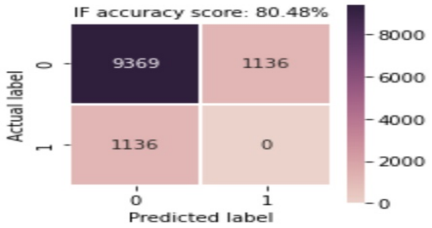
5.1.3 Different Scoring Functions

In Subsection 4.2.1, different scoring functions for the Isolation Forest method are introduced and experimented with. The goal of this subsection is to compare the results of the original IF in combination with the original scoring function (2.11) with the results obtained with the other scoring functions described in Subsection 4.2.1 for this real-world financial dataset.

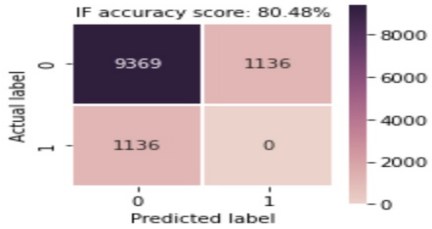
The first scoring function we will explore is given by Equation (4.2). The same experiments as in the previous subsection are done with this score functions, but now with a constant height limit and also with the implementation developed in [8]. These results are compared with the original scoring function given by Equation (2.11). This comparison is shown in Figure 5.7.



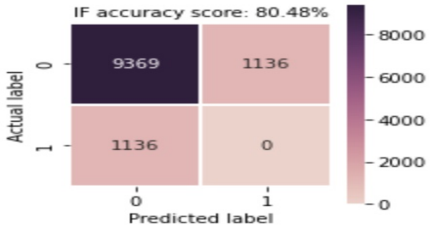
(a) Score function (2.11) (height limit = 20)



(b) Score function (4.2) (height limit = 20)



(c) Score function (2.11) (height limit = 50)



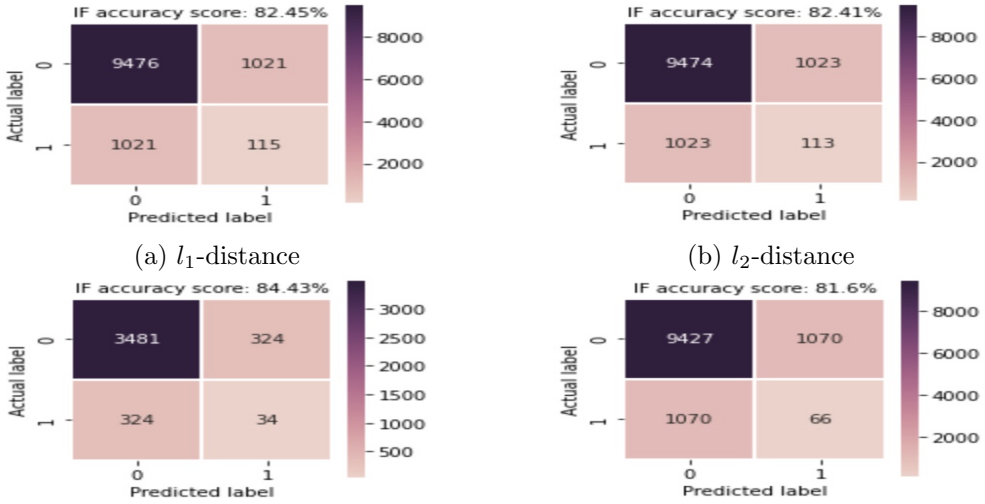
(d) Score function (4.2) (height limit = 50)

Figure 5.7: Confusion matrices of the original IF method (1000 trees) used with two different scoring functions for two different height limits.

From Figure 5.7, we can conclude that the new scoring function defined by Equation (4.2) does not give more accurate results than the original scoring function defined by Equation (2.11). In fact, both scoring functions give us exactly the same results for this specific dataset.

The next scoring functions we are going to test on this dataset are also already introduced in Subsection 4.2.1 and are given by Equation (4.3) and Equation (4.4). These are the theoretical variance and conditional expectation of the number of random splits needed to isolate a point. The formulas of these quantities are developed in Section 3.1 for one-dimensional datasets only. Hence, these scoring functions can only be used on one-dimensional datasets and therefore we will use the four transformed one-dimensional datasets of the Elliptic dataset, shown in Figure 5.3, to test these new scoring functions. The first new scoring function that is tested on these four transformed datasets (without duplicates) is given by Equation (4.3) and the same experiments are performed as done in Subsection 5.1.1. Thus, the points with the largest scores are labelled as illicit and are compared with the transactions that are labelled as illicit beforehand. The results of these experiments are shown in Figure 5.4.

If we compare Figure 5.8 with Figure 5.2, we see that the Transformed IF with this new scoring function given by Equation (4.3) gives more accurate results than the original IF for this specific dataset. This was also the case for the Transformed IF with the original scoring function as we can see in Figure 5.4. The results of the Transformed IF with the new scoring function are not more accurate than those of the Transformed IF with the original scoring function, which can



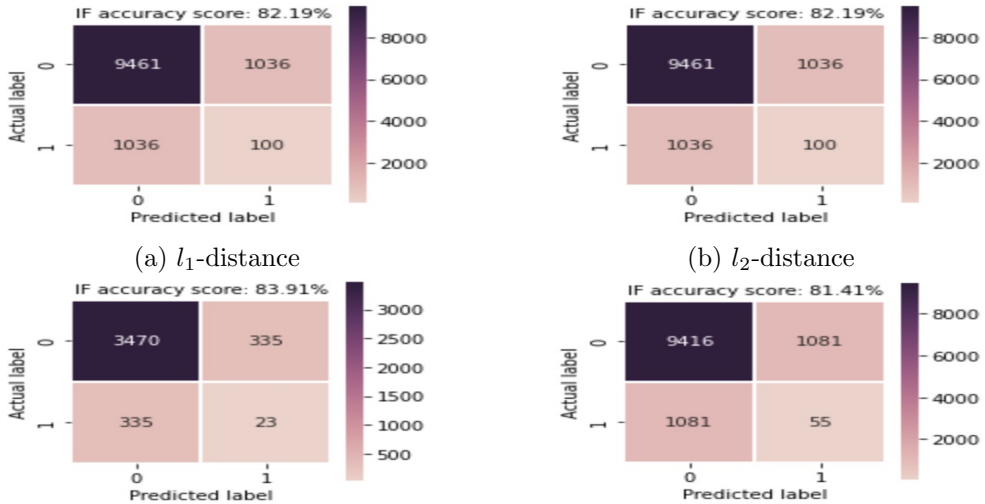
(a) l_1-distance (b) l_2-distance

(c) l_{max}-distance (d) kNN-distance

Figure 5.8: Confusion matrices of the Transformed IF method with the new scoring function given by (4.3) used on the four transformed datasets of the Elliptic dataset.

be concluded from the comparison of Figure 5.8 with Figure 5.4.

The last new scoring function that is tested in combination with the Transformed IF method is given by Equation (4.4) with $i = 3$. The same experiments are done with this scoring function as we have done with the previous scoring functions. The results of these experiments are given in Figure 5.9.



(a) l_1-distance (b) l_2-distance

(c) l_{max}-distance (d) kNN-distance

Figure 5.9: Confusion matrices of the Transformed IF method with the new scoring function given by (4.4) with $i = 3$ used on the four transformed datasets of the Elliptic dataset.

If we compare Figure 5.9 with Figure 5.4 and Figure 5.8, we can conclude that the Transformed IF with the scoring function given by Equation (4.4) gives the least accurate results of the three scoring functions in combination with the Transformed IF. However, it still performs much better than the original IF in this case.

5.2 Summary

Summarizing, in Section 5.1 we introduced the real-world financial dataset that was used in this chapter for all the experiments and tests. This dataset was the Elliptic dataset and it mapped Bitcoin transactions to real entities belonging to licit categories versus illicit ones and thus it was a labelled dataset. In Subsection 5.1.1, we compared the performances of the original IF tested on this dataset and the Transformed IF tested on the transformed one-dimensional datasets belonging to this dataset. The datapoints with the largest scores were labelled as illicit and these points were compared with the transactions that were labelled as illicit beforehand. The original IF method didn't detect any of the illicit transactions of the Elliptic dataset. However, the Transformed IF in combination with the l_1-, l_2-, l_{max}- and kNN-distance gave more accurate results. More illicit transactions were detected by this method and the overall accuracy scores were also higher. In Subsection 5.1.2, the impact of pruning of the isolation trees on the results of the original IF as an outlier detection algorithm was tested. We used the implementation of the original IF method developed in [8] instead of the Scikit-learn implementation, which was used in Subsection 5.1.1. The same experiments were done as in the previous subsection for different height limits of the isolation trees. The conclusion that was drawn from these experiments was that pruning of the isolation trees had no impact on the results. For different height limits, there were no differences in the results. Finally, in the Subsection 5.1.3, different scoring functions were tested for this specific financial dataset. These scoring functions were already introduced in Subsection 4.2.1. First, the scoring function given by Equation (4.2) was compared with the original scoring function of the original IF algorithm. The result of that comparison was that both scoring functions gave us exactly the same results. Moreover, the two scoring functions given by Equation (4.3) and Equation (4.4) were tested in combination with the Transformed IF method. These two new scoring functions in combination with the Transformed IF gave more accurate results than the original IF method, just like the Transformed IF did in combination with the original scoring function. The scoring function given by Equation (4.3) and the original scoring function performed the best of these three scoring functions. The reason for the improvement of the performances when using the Transformed IF method in comparison with the original IF method could be the transformation from 93 dimensions to 1 dimension. That could make it easier to detect the outliers, because the methods only have to consider 1 feature for each datapoint. This dataset has so many dimensions that it could be the case that some of these dimensions are not important for outlier detection and are then just noise for our methods.

6. Conclusion and Discussion

In this chapter, conclusions are drawn from all the experiments conducted throughout this thesis. Furthermore, recommendations for future research are provided. The research questions stated in Section 1.1 are answered in Section 6.1. In Section 6.2, recommendations for future research are discussed.

6.1 Conclusion

In this section, we will draw conclusion based on the results derived from this thesis. This is done by answering the research questions stated in Section 1.1. The first research question is given by:

Can we relate the concept of Isolation Forest with density- and distance-based outlier detection methods?

In Chapter 2, multiple definitions of an outlier together with the methods that are based on them were introduced and explained. These were existing methods coming from the literature we studied. The definitions and methods were separated based on distance, density, cluster, depth and isolation. Hence, there were distance- and density-based outlier detection methods and these were elaborated on in Chapter 2.

The distance-based methods were introduced first in Subsection 2.3.2. Multiple distance-based outlier definitions and detection methods were stated and explained. The most important definition of an outlier mentioned in this subsection was the Distance-Based (DB) outlier, given by Definition 2.2. The advantage of this definition is that it can be used in the case that the data distribution does not fit any standard distribution. However, in many cases there is a relationship between the different variables and then multivariate outlier definitions are more useful. A distance-based multivariate outlier definition is given by Definition 2.6 and is called the Mahalanobis distance. Distance-based methods are based on evaluating distances between datapoints. Anomaly scores are often based on neighbour distances. It is clear that anomalies have larger separation distances compared to their neighbours. One of the advantages of these methods is that this separation can be easily visualized.

The density-based methods were separated into non-parametric and parametric density-based methods in Subsection 2.1.2 and Subsection 2.1.3. Non-parametric density-based methods investigate specific regions of a dataset, determining the local density within these regions with the number of points and thus the dataspace is partitioned. Outliers are in this situation the points in the regions with lower local densities. These methods do not look at the specific density distribution. The Local Outlier Factor method is the most significant method that finds outliers based on non-parametric density estimates. However, the parametric density-based do need the specific density distribution of the dataset that is explored. This is also one of the major disadvantages of these methods.

In the literature, distance-based and density-based methods are often interchanged because of the similarity between the definitions. In Chapter 2, we saw that distance-based definitions of an outlier can also be interpreted as density-based definitions. Moreover, distance-based definitions can be used to create density-based methods for detecting outliers.

Another type of outlier detection methods are the cluster-based methods, which are not mentioned in the research question, but we will briefly mention them. These methods classify a

datapoint as an outlier if it is not a member of a cluster of points. The most popular clustering algorithm is K-means clustering.

The second research question is given by:

How can we mathematically define the concept of isolation? And can we use this new framework to develop better outlier detection methods?

In Section 2.2, we started by introducing and explaining definitions of an outlier and outlier detection methods based on the notion of depth. The Isolation Forest algorithm also makes use of the depth of datapoints. However, it is based on the concept of isolation and the main difference between this concept and the previous definitions of outliers is that it looks directly at the outliers, while the previous definitions are based on the normal instances. An isolation-based outlier is defined as an observation that is separated from the rest of the instances. Anomalies are points that deviate from other points, and also there are fewer outliers than inliers, therefore they are more susceptible to isolation. In isolation trees, instances are partitioned recursively until all instances are isolated. Anomalies are isolated earlier in the trees than the normal instances, because of their distinguishable attribute-values and because there are fewer outliers than normal instances.

The IF method consists of two stages. The first stage is called the training stage and in this stage the isolation trees are constructed from a sub-sample of the data. In the second stage, which is called the evaluation stage, an anomaly score s is derived from the expected path length $E(h(x))$ for each datapoint. The expected path length is derived by passing all the datapoints through each isolation tree in the isolation forest. The anomaly score s is given by Equation (2.11) and the higher the score of a point the more outlying it is. Two major advantages of this method are that the detection performance converges quickly with a very small number of trees and it only requires a small sub-sampling size to achieve high detection performance with high efficiency.

Furthermore, in Chapter 2, a new method based on the original Isolation Forest algorithm were developed to detect the most outlying points in a dataset. This method is based on a projection that first transforms a multi-dimensional dataset into a one-dimensional dataset. This one-dimensional dataset can then be used to detect the outliers in the same way as the original IF method. The first transformation is based on the distance of every point to the center of mass of the dataset. For this transformation a proper distance function is needed and therefore we introduced multiple distance functions. The second and final new transformation is based on the kNN-distance of every datapoint. This distance is derived via the k-nearest neighbors algorithm by looking at the distance of every datapoint to its k nearest neighbours. The kNN-distance is then the mean of the distances to its k nearest neighbours. The transformation based on the distance to the CM is a global transformation, but the second transformation, based on the kNN-distance, is a local transformation. It considers the neighbourhood of each datapoint separately.

The third research question is given by:

What is the theory behind the Isolation Forest method?

There is not much theory known for the original IF method, which was introduced in Section 2.2, and therefore this is explored more thoroughly. This was done in Chapter 3 and in Section 3.1 of this chapter we developed theory for one-dimensional datasets. We considered different

types of points, namely the fringe points and interior points. The theory that was developed is based on the number of random splits that the original IF algorithm needs to isolate a point.

First, theory for the fringe points was developed in Subsection 3.3.1 and that was done by considering the recursion formula given by Equation (3.1), which was the probability that the left fringe point gets isolated in $s + 1$ random splits. This formula was computationally very demanding and therefore we looked for an expression that was faster to implement. Eventually, this was found in Equation (3.6), which can be efficiently implemented in Python by making use of cumulative summations of the previous terms.

In the next subsection, theory was developed for the interior points of a dataset. This was done in the same way as done with the fringe points by first looking at the recursion formula given by Equation (3.9) and then searching for an expression that is computationally less demanding. We noticed that these formulas could be written in terms of the fringe formulas and that was done in Equation (3.14) and eventually proven in Lemma (3.1).

Finally, theory for two-dimensional datasets was partially derived in Section 3.2. In two-dimensional datasets, there are four possible types of points and we only developed theory for the fringe points in both dimensions. The formulas for the other types of points turned out to be long, involved and cumbersome.

The fourth and final research question is given by:

Confirm the theoretical findings with numerical experiments.

The theoretical formulas derived in Section 3.1 and Section 3.2 based on the number of random splits needed to isolate a point are numerically validated in Section 3.3 and Section 3.4. These formulas have been implemented in Python 3.9 and with these implementations the probabilities that a point gets isolated in s random splits are calculated. This was also calculated numerically with the Isolation Forest method. We have compared the theoretical and numerical probabilities to see if they match. The Isolation Forest algorithm was tested with and without pruning and thus with fully grown trees and trees with a height limit. For all the datasets we considered, the theoretical probabilities, expectations and variances were almost equal to the numerical ones with fully grown trees. For pruned trees, there were some differences due to the trees cut off at a certain point.

The theoretical formulas derived for the one-dimensional case were used in combination with our new transformation methods to detect the most outlying points in multi-dimensional datasets. The effort to expand these formulas to a two-dimensional dataset turned out to be cumbersome and therefore the Transformed Isolation Forest was developed. First, TIF transforms a multi-dimensional dataset into a one-dimensional set-up and on this one-dimensional set-up the theoretical formulas were applied. This was done by calculating the theoretical expectation of the number of random splits needed to isolate every datapoint in this new dataset. The lower the expectation, the more outlying a point. Outliers are more susceptible to isolation than inliers and therefore they are isolated in less random splits than inliers. Hence, their expectation will be lower than the expectation of the inliers.

With these new methods, we did multiple experiments with multiple different datasets. For the TIF in combination with the global transformation, based on the distance to the CM of the dataset, three distance norms were used: the l_1-, l_2- and l_{max}-norm. We looked at the most outlying points detected by these new methods and compared them with the most outlying points of the dataset calculated beforehand with three different metrics. The numerical IF method was also used in these experiments to get a more complete comparison. The datasets we considered

were one-, two- and three-dimensional standard normally distributed datasets. On top of that, we also looked at a two-dimensional uniform distribution. On these datasets, we only tested the new global transformation method and the result of that was that the combination of this method with the l_1- and l_2-norm gave the most accurate results. Moreover, a ten-dimensional standard normally distributed dataset was tested and on this dataset the new local method, based on the kNN-distance of every point, was also tested. On this dataset, the l_{max}-norm performed the worst of all the new methods considered. The other three norms gave us accurate results, but the numerical IF still gave us the most accurate results. Furthermore, a ten-dimensional dataset coming from a t-distribution with 1 degree of freedom was tested. Such a dataset has very clear outliers and thus we wanted to see if our methods detect all these outliers. This was almost the case and thus every method performed well on this specific dataset. Finally, the robustness of the new methods was tested on a very specific dataset. We did this by experimenting on a ten-dimensional normally distributed set of 101 points with correlation between the dimensions, variance 1 in every dimension and not the same mean in every dimension. The point that was considered was an outlier in some of the dimensions and an inlier in the other dimensions, but also a global outlier. For different values of the correlation and different set-ups of the dataset, we explored which methods detected this point in which cases. The result of that was that the transformation method in combination with the l_1-distance had the worst performance of all the methods considered. The l_{max}- and kNN-distance gave us the most accurate results and were thus the most robust methods for this specific dataset. The overall conclusion that can be drawn from all the experiments done with these new transformation methods in Chapter 3 is that the transformation method in combination with the l_2- and kNN-distance performs the best overall. In some experiments, these norms gave us the best performance of all norms and in the experiments where they weren't the best norms they still gave us reasonably accurate results. With the l_2-norm the cost of the outliers increase exponentially and the main advantage of the kNN-distance is that it considers the direct neighbourhood of every datapoint. These statements could be explanations for the very accurate performances of these two distances in combination with the Transformed IF method.

In Chapter 4, the methods were used for further testing and more experiments. The computing times of the recursion formula for the interior points in the one-dimensional case was compared with the new formula developed in Chapter 3. This new formula was much faster than the original one. The difference between the theoretical and numerical probabilities was also explored for an increasing number of trees. More trees led to more accurate numerical probabilities, but also higher computing times. The key was to find the balance between the two. Moreover, the impact of pruning on the difference between the theoretical and numerical probabilities was also explored. For the set-up we used, the numerical probabilities for a point getting isolated after s random splits were accurate for a height limit greater equal than s. In Subsection 4.2.1, different score functions for the IF method were introduced. The conclusion that could be drawn is that the new scoring functions were not more accurate in terms of mistakes made in detecting the most outlying points for the specific datasets we considered. In the final subsection of Chapter 4, the performances of three different outlier detection methods were compared. This is done by looking at the number of mistakes made in detecting the most outlying points on average over multiple random datasets. Isolation Forest and K-means clustering were very accurate in detecting the most outlying points if the order is not taken in consideration. The third method, LOF, was the least accurate of the three. All three methods were not accurate when the order was taken into consideration. However, K-means clustering was the most accurate of the three methods on average over these multiple random datasets.

Finally, we tested the Transformed IF method and the original IF method on a real-world financial dataset in Chapter 5. This was the Elliptic dataset and this dataset was labelled.

Hence, we have compared the outliers detected by our methods with the transactions that are labelled as illicit beforehand. Our new transformation methods gave more accurate results than the original IF method used on this dataset. More illicit transactions were detected and also the overall accuracy score was slightly higher. The impact of pruning on the performance of the original IF method was also tested in this case and we could conclude from this that pruning had no impact on the results. Finally, different scoring functions, which were already introduced in Chapter 4, were tested and compared with the results obtained with the original scoring function. The conclusion that was drawn from these experiments was that these new scoring functions didn't give us major improvements for this specific dataset. The reason for the improvement of the performances when using the Transformed IF method in comparison with the original IF method could be the transformation from 93 dimensions to 1 dimension. That could make it easier to detect the outliers, because the methods only have to consider 1 feature for each datapoint. This dataset has so many dimensions that it could be the case that some of these dimensions are not important for outlier detection and are then just noise for our methods.

6.2 Recommendations for Future Research

In this section, we will discuss some future research topics.

In Section 2.3, a method together with two transformations was presented that transforms multi-dimensional datasets into one-dimensional datasets and then detects outliers in the same way as the original Isolation Forest algorithm. This is not the only transformation that can be used in combination with this method. For example, every distance metric transforms a multi-dimensional dataset into a one-dimensional one. For further research, it is interesting to look at other new methods based on transformations or create a combination of two methods.

In Section 3.1, theory behind the IF algorithm was developed for one-dimensional datasets. These formulas were based on the number of random splits that the IF algorithm needs to isolate a point. In Section 3.2, the same was done for two-dimensional datasets. However, this was only done for one specific type of point, namely the fringe points in both directions. For further research, it would be interesting to develop theory for the other three types of points in the two-dimensional case. When this is done, the next step will be developing theory for even higher-dimensional cases.

In Section 3.5, the Transformed Isolation Forest was tested on multiple different datasets. All these datasets consisted of numerical data and therefore it would be interesting to test the TIF on categorical data to get an even more complete picture of the performance of the method.

In Subsection 4.2.1, we tested the original IF method with different new scoring functions and compared the results with the original scoring function. These tests didn't lead to major improvements and therefore, for future research, it would be interesting to test more new scoring functions and test them more thoroughly.

In Subsection 4.2.2, the performances of three different outlier detection methods were tested. The performance of the IF algorithm was compared with the performances of the LOF method and the K-means clustering algorithm. There are many more outlier detecting methods to consider and therefore it would be interesting to test the performances of more outlier detection methods and compare these performances with the performance of the IF algorithm.

Bibliography

[1] Khan Academy. *What is center of mass?* https://www.khanacademy.org/science/physics/linear-momentum/center-of-mass/a/what-is-center-of-mass.

[2] Charu C Aggarwal. "An introduction to outlier analysis". In: *Outlier analysis*. Springer, 2017, pp. 1–34.

[3] Irad Ben-Gal. "Outlier detection". In: *Data mining and knowledge discovery handbook*. Springer, 2005, pp. 131–146.

[4] Kristin Björg Bergthórsdóttir. "Local Explanation Methods for Isolation Forest: Explainable Outlier Detection in Anti-Money Laundering". MA thesis. Delft University of Technology, 2020.

[5] David Cortes. "Isolation forests: looking beyond tree depth". In: *arXiv preprint arXiv:2111.11639* (2021).

[6] Laurie Davies and Ursula Gather. "The identification of multiple outliers". In: *Journal of the American Statistical Association* 88.423 (1993), pp. 782–792.

[7] Sebastian Buschjäger, Philipp-Jan Honysz and Katharina Morik. "Randomized outlier detection with trees". In: *International Journal of Data Science and Analytics* (2020), pp. 1–14.

[8] Mark Huistra. "Locally Explainable Isolation Forest with Mixed-Attribute Data and Ternary Isolation Trees: Combatting Money Laundering with Anomaly Detection". MA thesis. Delft University of Technology, 2021.

[9] Renu Khandelwal. *T-distributed Stochastic Neighbor Embedding(t-SNE)*. https://towardsdatascience.com/t-distributed-stochastic-neighbor-embedding-t-sne-bb60ff109561. 2020.

[10] Edwin M. Knorr and Raymond T. Ng. "Algorithms for Mining Distance-Based Outliers in Large Datasets". In: *Proceedings of the international conference on very large data bases* (1998), pp. 392–403.

[11] Chi-Kwong Li. "Norms, isometries, and isometry groups". In: *The American Mathematical Monthly* 107.4 (2000), pp. 334–340.

[12] Amol Mavuduru. *How to perform anomaly detection with the Isolation Forest algorithm.* https://towardsdatascience.com/how-to-perform-anomaly-detection-with-the-isolation-forest-algorithm-e8c8372520bc. 2021.

[13] Antonella Mensi and Manuele Bicego. "Enhanced anomaly scores for isolation forests". In: *Pattern Recognition* 120 (2021), p. 108115.

[14] Bilal Mussa. *A python function to get all the possible stats from a confusion matrix.* https://towardsdev.com/a-python-function-to-get-all-the-possible-stats-from-a-confusion-matrix-e9bf8cda836a. 2011.

[15] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng and Jörg Sander. "LOF:Identifying Density-Based Local Outliers". In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data* (2000), pp. 93–104.

[16] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. "Efficient algorithms for mining outliers from large data sets". In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 2000, pp. 427–438.

[17] Ida Ruts and Peter J Rousseeuw. “Computing depth contours of bivariate point clouds”. In: *Computational statistics & data analysis* 23.1 (1996), pp. 153–168.

[18] Fei Tony Liu, Kai Ming Ting and Zhi-Hua Zhou. “Isolation Forest”. In: *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM'08)* (2008), pp. 413–422.

[19] Fei Tony Liu, Kai Ming Ting and Zhi-Hua Zhou. “Isolation-Based Anomaly Detection”. In: *ACM Transactions on Knowledge Discovery from Data(TKDD)* 6.1 (2012), pp. 31–33.

[20] Mikhail Tokovarov and Paweł Karczmarek. “A probabilistic generalization of isolation forest”. In: *Information Sciences* 584 (2022), pp. 433–449.

[21] Gayle Towell. *Center of Mass: Definition, Equation, How to Find (w/ Examples)*. `https://sciencing.com/center-of-mass-definition-equation-how-to-find-w-examples-13725851.html`. 2020.

[22] John W Tukey et al. *Exploratory data analysis*. Vol. 2. Reading, MA, 1977.

[23] Hetal Vinchhi. *Machine Learning : Introduction to K Nearest Neighbor (KNN) in Python*. `http://assignmentsolutionguru.com/article/intro-to-knn-in-python`. 2018.

[24] Mingxi Wu and Christopher Jermaine. “Outlier detection by sampling with accuracy guarantees”. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2006, pp. 767–772.

[25] Arthur Zimek and Peter Filzmoser. “There and back again: Outlier detection between statistical reasoning and data mining algorithms”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.6 (2018), e1280.

A. Additional Theoretical results

In Chapter 3, the most important theory behind the Isolation Forest algorithm has been discussed and elaborated on. Other theory can be derived from this or is very similar. This theory is also split in theory for the fringe points and theory for the interior points.

A.1 Fringe Points

The theory for the right fringe point of a dataset can be derived from the similar theoretical formulas for the left fringe point. Hence, the probabilities that the right fringe point gets isolated in s random splits for $1 \leq s \leq n-1$, which is the maximum point of the dataset, are as follows:

$$P_{x_n}^{RF}(RS = s+1) = \sum_{i=2}^{n-s} \frac{x_i - x_{i-1}}{x_n - x_1} P_{x_n}^{RF}(RS = s|(x_i, ..., x_n)). \tag{A.1}$$

After splitting between the point x_{i-1} and x_i, we only look at the remaining subset $(x_i, ..., x_n)$. This leads to the recursion formula (A.1). This formula is computationally very demanding and therefore we are looking for something faster to implement. Hence, the particular cases $RS = 1, 2, 3, 4$ are written out to look for a formula for all cases that is also fast to implement.

$$P_{x_n}^{RF}(RS = 1) = \frac{x_n - x_{n-1}}{x_n - x_1}, \tag{A.2}$$

$$\begin{aligned} P_{x_n}^{RF}(RS = 2) &= \sum_{i=2}^{n-1} \frac{x_i - x_{i-1}}{x_n - x_1} \cdot \frac{x_n - x_{n-1}}{x_n - x_i} \\ &= \frac{x_n - x_{n-1}}{x_n - x_1} \sum_{i=2}^{n-1} \frac{x_i - x_{i-1}}{x_n - x_i} = P_{x_n}^{RF}(RS = 1) \sum_{i=2}^{n-1} \frac{x_i - x_{i-1}}{x_n - x_i}, \end{aligned} \tag{A.3}$$

$$\begin{aligned} P_{x_n}^{RF}(RS = 3) &= \sum_{j=2}^{n-2} \sum_{i=j+1}^{n-1} \frac{x_j - x_{j-1}}{x_n - x_1} \cdot \frac{x_i - x_{i-1}}{x_n - x_j} \cdot \frac{x_n - x_{n-1}}{x_n - x_i} \\ &= \frac{x_n - x_{n-1}}{x_n - x_1} \sum_{j=2}^{n-2} \sum_{i=j+1}^{n-1} \frac{x_j - x_{j-1}}{x_n - x_j} \cdot \frac{x_i - x_{i-1}}{x_n - x_i} \\ &= \frac{x_n - x_{n-1}}{x_n - x_1} \sum_{j=2}^{n-2} \frac{x_j - x_{j-1}}{x_n - x_j} \sum_{i=j+1}^{n-1} \frac{x_i - x_{i-1}}{x_n - x_i} \\ &= \sum_{j=2}^{n-2} \frac{x_j - x_{j-1}}{x_n - x_j} \Big(P_{x_n}^{RF}(RS = 2) - \frac{x_n - x_{n-1}}{x_n - x_1} \sum_{i=2}^{j} \frac{x_{i+1} - x_i}{x_i - x_1} \Big), \end{aligned} \tag{A.4}$$

$$
\begin{aligned}
P_{x_n}^{RF}(RS=4) &= \sum_{k=2}^{n-3}\sum_{j=k+1}^{n-2}\sum_{i=j+1}^{n-1}\frac{x_k-x_{k-1}}{x_n-x_1}\cdot\frac{x_j-x_{j-1}}{x_n-x_k}\cdot\frac{x_i-x_{i-1}}{x_n-x_j}\cdot\frac{x_n-x_{n-1}}{x_n-x_i}\\
&= \sum_{k=2}^{n-3}\sum_{j=k+1}^{n-2}\sum_{i=j+1}^{n-1}\frac{x_k-x_{k-1}}{x_n-x_1}\cdot\frac{x_j-x_{j-1}}{x_n-x_k}\cdot\frac{x_i-x_{i-1}}{x_n-x_j}\cdot\frac{x_n-x_{n-1}}{x_n-x_i}\\
&= \frac{x_n-x_{n-1}}{x_n-x_1}\sum_{k=2}^{n-3}\frac{x_k-x_{k-1}}{x_n-x_k}\sum_{j=k+1}^{n-2}\frac{x_j-x_{j-1}}{x_n-x_j}\sum_{i=j+1}^{n-1}\frac{x_i-x_i}{x_n-x_i}\\
&= \sum_{k=2}^{n-3}\frac{x_k-x_{k-1}}{x_n-x_k}\Big(P_{x_n}^{RF}(RS=3)\\
&- \frac{x_n-x_{n-1}}{x_n-x_1}\sum_{j=2}^{k}\frac{x_j-x_{j-1}}{x_n-x_j}\sum_{i=j+1}^{n-1}\frac{x_i-x_{i-1}}{x_n-x_i}\Big).
\end{aligned}
\tag{A.5}
$$

Equation (A.5) can be expanded to a general formula for s random splits with $1 \leq s \leq n-1$:

$$
\begin{aligned}
P_{x_n}^{RF}(RS=s) = &\sum_{k=2}^{n-(s-1)}\frac{x_k-x_{k-1}}{x_n-x_k}\Big(P_{x_n}^{RF}(RS=s-1)\\
&- \frac{x_n-x_{n-1}}{x_n-x_1}\sum_{j=2}^{s}\frac{x_j-x_{j-1}}{x_n-x_j}\sum_{i=j+1}^{n-(s-3)}\frac{x_i-x_{i-1}}{x_n-x_i}\cdots\sum_{a=b+1}^{n-1}\frac{x_a-x_{a-1}}{x_n-x_a}\Big).
\end{aligned}
\tag{A.6}
$$

Equation (A.6) can be efficiently implemented in Python by making use of cumulative summations of the previous terms.

With the formulas above, the expectation and variance of the number of RS that is needed to isolate the left fringe point x_1 can also be derived. These formulas can be seen below:

$$
E_{x_n}^{RF}(RS) = \sum_{i=1}^{n-1} i P_{x_n}^{RF}(RS=i), \tag{A.7}
$$

$$
Var_{x_n}^{RF}(RS) = \sum_{i=1}^{n-1} i^2 P_{x_n}^{RF}(RS=i) - \Big(E_{x_n}^{RF}(RS)\Big)^2. \tag{A.8}
$$

A.2 Interior Points

For the interior points of a one-dimensional dataset, the probabilities given by Equation (3.9) have been written out 4 random splits and that can be seen below:

$$P_{x_j}^{IN}(RS=4) = \sum_{i=1}^{j-2} \frac{x_{i+1}-x_i}{x_n-x_1}\Big(\sum_{k=i+1}^{j-2} \frac{x_{k+1}-x_k}{x_n-x_{i+1}}\big(\frac{x_j-x_{j-1}}{x_n-x_{k+1}}\cdot\frac{x_{j+1}-x_j}{x_n-x_j} + \frac{x_{j+1}-x_j}{x_n-x_{k+1}}\cdot\frac{x_j-x_{j-1}}{x_j-x_{k+1}}\big)$$

$$+ \sum_{k=j+1}^{n-1} \frac{x_{k+1}-x_k}{x_n-x_{i+1}}\big(\frac{x_j-x_{j-1}}{x_k-x_{i+1}}\cdot\frac{x_{j+1}-x_j}{x_k-x_j} + \frac{x_{j+1}-x_j}{x_k-x_{i+1}}\cdot\frac{x_j-x_{j-1}}{x_j-x_{i+1}}\big)$$

$$+ \frac{x_j-x_{j-1}}{x_n-x_{i+1}}P_{x_j}^{LF}(RS=2|x_j,...,x_n) + \frac{x_{j+1}-x_j}{x_n-x_{i+1}}P_{x_j}^{RF}(RS=2|x_1,...,x_j)\Big)$$

$$+ \sum_{i=j+1}^{n-1} \frac{x_{i+1}-x_i}{x_n-x_1}\Big(\sum_{k=1}^{j-2} \frac{x_{k+1}-x_k}{x_i-x_1}\big(\frac{x_j-x_{j-1}}{x_i-x_{k+1}}\cdot\frac{x_{j+1}-x_j}{x_i-x_j} + \frac{x_{j+1}-x_j}{x_i-x_{k+1}}\cdot\frac{x_j-x_{j-1}}{x_j-x_{k+1}}\big)$$

$$+ \sum_{k=j+1}^{i-1} \frac{x_{k+1}-x_k}{x_i-x_1}\big(\frac{x_j-x_{j-1}}{x_k-x_1}\cdot\frac{x_{j+1}-x_j}{x_k-x_j} + \frac{x_{j+1}-x_j}{x_k-x_1}\cdot\frac{x_j-x_{j-1}}{x_j-x_1}\big)$$

$$+ \frac{x_j-x_{j-1}}{x_i-x_1}P_{x_j}^{LF}(RS=2|x_j,...,x_n) + \frac{x_{j+1}-x_j}{x_i-x_1}P_{x_j}^{RF}(RS=2|x_1,...,x_j)\Big)$$

$$+ \frac{x_j-x_{j-1}}{x_n-x_1}P_{x_j}^{LF}(RS=3|x_j,...,x_n) + \frac{x_{j+1}-x_j}{x_n-x_1}P_{x_j}^{RF}(RS=3|x_1,...,x_j). \qquad \text{(A.9)}$$

This formula can also be written in terms of the fringe formulas:

$$P_{x_j}^{IN}(RS=4) = \sum_{i=1}^{j-2} \frac{x_{i+1}-x_i}{x_n-x_1}P_{x_j}^{IN}(RS=3|(x_{i+1},...,x_n)) + \sum_{i=j+1}^{n-1} \frac{x_{i+1}-x_i}{x_n-x_1}P_{x_j}^{IN}(RS=3|(x_1,...,x_i))$$

$$+ \frac{x_j-x_{j-1}}{x_n-x_1}P_{x_j}^{LF}(RS=3|(x_j,...,x_n)) + \frac{x_{j+1}-x_j}{x_n-x_1}P_{x_j}^{RF}(RS=3|(x_1,...,x_j))$$

$$= \sum_{i=1}^{j-2} \frac{x_{i+1}-x_i}{x_n-x_1}\frac{(x_{j+1}-x_j)(x_j-x_{j-1})}{(x_n-x_j)(x_j-x_{i+1})}\Big(\sum_{k=i+1}^{j-2} \frac{x_{k+1}-x_k}{x_j-x_{k+1}} + \sum_{k=j+1}^{n-1} \frac{x_{k+1}-x_k}{x_k-x_j}\Big)$$

$$+ \sum_{i=j+1}^{n-1} \frac{x_{i+1}-x_i}{x_n-x_1}\frac{(x_{j+1}-x_j)(x_j-x_{j-1})}{(x_i-x_j)(x_j-x_1)}\Big(\sum_{k=1}^{j-2} \frac{x_{k+1}-x_k}{x_j-x_{k+1}} + \sum_{k=j+1}^{i-1} \frac{x_{k+1}-x_k}{x_k-x_j}\Big)$$

$$+\frac{x_j-x_{j-1}}{x_n-x_1}\sum_{i=j+2}^{n-1}\frac{x_{i+1}-x_i}{x_n-x_j}\sum_{k=j+1}^{i-1}\frac{x_{k+1}-x_k}{x_i-x_j}\cdot\frac{x_{j+1}-x_j}{x_k-x_j}$$

$$+\frac{x_{j+1}-x_j}{x_n-x_1}\sum_{i=1}^{j-3}\frac{x_{i+1}-x_i}{x_j-x_1}\sum_{k=i+1}^{j-2}\frac{x_{k+1}-x_k}{x_j-x_{i+1}}\cdot\frac{x_j-x_{j-1}}{x_j-x_{k+1}}$$

$$=\frac{(x_{j+1}-x_j)(x_j-x_{j-1})}{(x_n-x_j)(x_j-x_1)}\Bigg(\sum_{i=1}^{j-2}\frac{x_{i+1}-x_i}{x_j-x_{i+1}}\sum_{k=i+1}^{j-2}\frac{x_{k+1}-x_k}{x_j-x_{k+1}}$$

$$+\sum_{i=j+1}^{n-1}\frac{x_{i+1}-x_i}{x_i-x_j}\sum_{k=j+1}^{i-1}\frac{x_{k+1}-x_k}{x_k-x_j}+\Big(\sum_{i=1}^{j-2}\frac{x_{i+1}-x_i}{x_j-x_{i+1}}\Big)\Big(\sum_{k=j+1}^{n-1}\frac{x_{k+1}-x_k}{x_k-x_j}\Big)\Bigg)$$

$$=\frac{x_{j+1}-x_j}{x_n-x_j}P^{RF}_{x_j}(RS=3|(x_1,...,x_j))+\frac{x_j-x_{j-1}}{x_j-x_1}P^{LF}_{x_j}(RS=3|(x_j,...,x_n))$$

$$+P^{LF}_{x_j}(RS=2|(x_j,...,x_n))P^{RF}_{x_j}(RS=2|(x_1,...,x_j))$$

$$=P^{LF}_{x_j}(RS=1|(x_j,...,x_n))P^{RF}_{x_j}(RS=3|(x_1,...,x_j))$$

$$+P^{RF}_{x_j}(RS=1|(x_1,...,x_j))P^{LF}_{x_j}(RS=3|(x_j,...,x_n))$$

$$+P^{LF}_{x_j}(RS=2|(x_j,...,x_n))P^{RF}_{x_j}(RS=2|(x_1,...,x_j)). \tag{A.10}$$

A.3 Two-Dimensional Case

In Section 3.4, formulas for the fringe point in both directions are derived. The next step is to look at one of the other four possibilities for the type of point we deal with. The point that is explored next is a point that is an interior point in the x-direction and a fringe point in the y-direction. In particular, a left fringe point in the y-direction. Thus, this point is (x_j, y_1). Such a point may not always exist, but in this dataset it is assumed that it does. Recall that for every split it holds that the probability that the split will be in one direction or the other is equal to $\frac{1}{2}$. The same assumptions for the sorting with the corresponding mappings are used as with the fringe points in both directions. This leads to the following recursion formula:

$$P^{IN,LF}_{(x_j,y_1)}(RS=s)=\frac{1}{2}\cdot\Big(\sum_{i=1}^{j-2}\frac{x_{i+1}-x_i}{x_n-x_1}P^{IN,LF}_{(x_j,y_1)}(RS=s-1|((x_j,y_1),...,(x_i,\tilde{y}_i)))$$

$$+\sum_{i=j+1}^{n-1}\frac{x_{i+1}-x_i}{x_n-x_1}P^{IN,LF}_{(x_j,y_1)}(RS=s-1|((x_1,\tilde{y}_1),...,(x_j,y_1)))$$

$$+\frac{x_j-x_{j-1}}{x_n-x_1}P^{LF,LF}_{x_j,y_1}(RS=s-1|((x_j,y_1),...,(x_n,\tilde{y}_n)))$$

$$+\frac{x_{j+1}-x_j}{x_n-x_1}P^{RF,LF}_{x_j,y_1}(RS=s-1|((x_1,\tilde{y}_1),...,(x_j,y_1)))\Big)$$

$$+\frac{1}{2}\cdot\sum_{i=s}^{n-1}\frac{y_{i+1}-y_i}{y_n-y_1}P^{IN,LF}_{(x_j,y_1)}(RS=s-1|((x_j,y_1),...,(\tilde{x}_i,y_i))). \tag{A.11}$$

This recursive formula is written out for the particular cases $RS=1,2$, as follows:

$$P^{IN,LF}_{(x_j,y_1)}(RS=1)=\frac{1}{2}\cdot P^{IN}_{x_j}(RS=1)+\frac{1}{2}\cdot P^{LF}_{y_1}(RS=1)$$

$$=\frac{1}{2}\cdot 0+\frac{1}{2}\cdot P^{LF}_{y_1}(RS=1)=\frac{1}{2}\cdot\frac{y_2-y_1}{y_n-y_1}, \tag{A.12}$$

$$P^{IN,LF}_{(x_j,y_1)}(RS=2)=\frac{1}{4}\cdot P^{IN}_{x_j}(RS=2)+\frac{1}{4}\cdot P^{LF}_{y_1}(RS=2)$$

$$+\frac{1}{4}\cdot\Big(\sum_{i=1}^{j-1}\frac{x_{i+1}-x_i}{x_n-x_1}P^{LF}_{y_1}(RS=1|((x_j,y_1),...,(x_i,\tilde{y}_i)))$$

$$+\sum_{i=j}^{n-1}\frac{x_{i+1}-x_i}{x_n-x_1}P^{LF}_{y_1}(RS=1|((x_1,\tilde{y}_1),...,(x_j,y_1)))\Big)$$

$$+\frac{1}{4}\cdot\sum_{i=2}^{n-1}\frac{y_{i+1}-y_i}{y_n-y_1}P^{IN}_{x_j}(RS=1|((x_j,y_1),...,(\tilde{x}_i,y_i)))$$

$$=\frac{1}{4}\cdot\frac{(x_{j+1}-x_j)(x_j-x_{j-1})}{(x_n-x_j)(x_j-x_1)}+\frac{1}{4}\cdot\frac{y_2-y_1}{y_n-y_1}\sum_{i=2}^{n-1}\frac{y_{i+1}-y_i}{y_i-y_1}$$

$$+\frac{1}{4}\cdot\Big(\sum_{i=1}^{j-1}\frac{x_{i+1}-x_i}{x_n-x_1}\frac{y_{f(2)}-y_1}{y_{f(n)}-y_1}+\sum_{i=j}^{n-1}\frac{x_{i+1}-x_i}{x_n-x_1}\frac{y_{f(2)}-y_1}{y_{f(i)}-y_1}\Big)$$

$$+\frac{1}{4}\cdot 0. \tag{A.13}$$

This formulas can be rewritten in terms of the fringe formulas in the 2D case:

$$P^{IN,LF}_{(x_j,y_1)}(RS=s) = \frac{1}{2}\cdot\sum_{i=1}^{s-1} P^{LF,LF}_{x_j,y_1}(RS=i|((x_j,y_1),...,(x_n,\tilde{y}_n)))$$

$$P^{RF,LF}_{x_j,y_1}(RS=s-i|((x_1,\tilde{y}_1),...,(x_j,y_1)))$$

$$+\frac{1}{2}\cdot\sum_{i=s}^{n-1}\frac{y_{i+1}-y_i}{y_n-y_1}P^{IN,LF}_{(x_j,y_1)}(RS=s-1|((x_j,y_1),...,(\tilde{x}_i,y_i))). \tag{A.14}$$

With the formulas above, the expectation and variance of the number of RS that is needed to isolate the interior point in the x-direction and the left fringe point in the y-direction (x_j, y_1) can also be derived. These formulas can be seen below:

$$E^{IN,LF}_{(x_j,y_1)}(RS) = \sum_{i=1}^{n-1} iP^{IN,LF}_{(x_j,y_1)}(RS=i) \tag{A.15}$$

$$Var^{IN,LF}_{(x_j,y_1)}(RS) = \sum_{i=1}^{n-1} i^2P^{IN,LF}_{(x_j,y_1)}(RS=i) - \left(E^{IN,LF}_{(x_j,y_1)}(RS)\right)^2. \tag{A.16}$$

A.4 Pruning

Outliers have shorter path lengths than inliers, so that means that they are isolated earlier in the trees. With this in mind, a height limit is introduced for the isolation trees in the IF algorithm. This leads to a reduction of the computational complexity, because the trees are not grown to completion. The most common height limit that is used in this algorithm is $l = \lceil \log_2(\psi) \rceil$ with ψ the sub-sampling size. The sub-sampling size is the minimum of 256 and the size of the dataset. Trees that are not grown to completion are called pruned trees. This will lead to different probabilities as we already saw in the tables above. To understand this more clearly, it is preferable to have some theory in the case of pruned trees. In this subsection, we will elaborate on that.

There are two possibilities for a point x in the case of pruned trees:

1. The point x is isolated before the height limit of the tree is reached. The path length or depth of this point is then equal to the number of edges from the root node to the external node.

2. The external node is terminated before the point x is fully isolated. That means that the external node contains more points than only the point x. Hence, the size of this node is > 1. The path length of the point x is then equal to the number of edges from the root node to the external node with an extra term c(Size) added. This term is the average depth of the leaves in a fully grown isolation tree given that the number of datapoints is equal to the size of the external node.

Hence, the following holds for the depth D:

$$E[D|D \leq l] = \sum_{i=1}^{l} iP(RS = i) \tag{A.17}$$

$$E[D|D > l] = l + E[c(S)] = l + \sum_{S} c(S)P(S|\text{after } l \text{ RS}) \tag{A.18}$$

with S the size of the node.

In order to work this expectation out, the term $P(S|\text{after } l \text{ RS})$ needs to be calculated. This has been done here below for a dataset with n points:

$$\begin{aligned}
P(S = k|\text{after 1 RS}) &= \frac{x_{k+1} - x_k}{x_n - x_1}, \\
P(S = k|\text{after 2 RS}) &= \sum_{i=k+1}^{n-1} \frac{x_{i+1} - x_i}{x_n - x_1} \frac{x_{k+1} - x_k}{x_i - x_1} = \frac{x_{k+1} - x_k}{x_n - x_1} \sum_{i=k+1}^{n-1} \frac{x_{i+1} - x_i}{x_i - x_1} \\
&= P(S = k|\text{after 1 RS}) \sum_{i=k+1}^{n-1} \frac{x_{i+1} - x_i}{x_i - x_1}, \\
P(S = k|\text{after 3 RS}) &= \sum_{j=k+2}^{n-1} \sum_{i=k+1}^{j-1} \frac{x_{j+1} - x_j}{x_n - x_1} \frac{x_{i+1} - x_i}{x_j - x_1} \frac{x_{k+1} - x_k}{x_i - x_1} \\
&= \frac{x_{k+1} - x_k}{x_n - x_1} \sum_{j=k+2}^{n-1} \frac{x_{j+1} - x_j}{x_j - x_1} \sum_{i=k+1}^{j-1} \frac{x_{i+1} - x_i}{x_i - x_1} \\
&= \sum_{j=k+2}^{n-1} \frac{x_{j+1} - x_j}{x_j - x_1} \Big(P(S = k|\text{after 2 RS}) - \frac{x_{k+1} - x_k}{x_n - x_1} \sum_{i=j}^{n-1} \frac{x_{i+1} - x_i}{x_i - x_1} \Big), \\
P(S = k|\text{after } l \text{ RS}) &= \sum_{j=k+l-1}^{n-1} \frac{x_{j+1} - x_j}{x_j - x_1} \Big(P(S = k|\text{after } l - 1 \text{ RS}) \\
&\quad - \frac{x_{k+1} - x_k}{x_n - x_1} \sum_{i=j}^{n-1} \frac{x_{i+1} - x_i}{x_i - x_1} \sum_{c=k+l-3}^{i-1} \frac{x_{c+1} - x_c}{x_c - x_1} \cdots \sum_{a=k+1}^{b-1} \frac{x_{a+1} - x_a}{x_a - x_1} \Big).
\end{aligned} \tag{A.19}$$

These formulas are only applicable for the leftmost node of the tree after l RS. It looks similar to the formulas of the fringe points with the difference that there are now k points left instead of 1.