Utrecht University

Bachelor Thesis
Bsc Artificial Intelligence

Isolation Forest

The accuracy of isolation forest and the possible effects of misclassified anomalies

---

Author:
Hidde Goossens

Supervisor:
Dr. M. (Thijs) van Ommen

Second Assessor:
Dr. Marijn Schraagen

A 7.5 ECT Bachelor thesis for the degree of Bachelor of Science

in

Artificial Intelligence

at the

Faculty of Humanities

June 26, 2020

Abstract

Anomaly detection is a topic in data science that is receiving more and more attention. Not only for its benefits in the business world, but in all sorts of different areas (e.g. cybersecurity, health care, behaviour etc.). The purpose of this thesis is to answer whether isolation forest (a machine learning anomaly detection algorithm) is accurate in classifying these anomalies and what the effects in a particular area (internet traffic) can be. The way to achieve this, is by taking a labelled data set and applying the algorithm to it, to see if it can find all the labelled anomalies. For the effects of misclassification, this paper will be looking at a specific area/data set and discuss all possible outcomes with its chances of happening. Isolation forest proves to be a valuable algorithm that can minimize risk and maximize benefits.

Contents

Chapter 1

Introduction

"Information is the oil of the 21st century, and analytics is the combustion engine"

Peter Sondergaard, Senior Vice President, Gartner

To state that a data point is an anomaly, some requirements are needed. First of all, anomalies only arise sporadically in data. Second, the properties are notably different from the normal data. Typically, anomalies in data will be translated into some kind of issue depending on the data set. Issues like medical illness[14], credit card fraud[2] or even network traffic patterns that could indicate unauthorised access[6]. For all of this to work, anomaly detection systems need algorithms with a high detection rate, low misclassification rate and a fast execution.

Most other existing approaches to anomaly detection are based on a modeled data set and start with building a profile of what a normal data point should look like. Then it identifies the points that do not fit these criteria. Next a ranking system is formed and rates which data points are most likely to be anomalies and which are not. The distance and density of the data points are the most commonly used forms of basic ranking. Many different methods, such as Local outlier Factor[5], classification-based methods[1] and clustering-based methods[9], have been created with these properties in mind. Two significant disadvantages of these methods are that they are optimized on finding perfect normal data points instead of focusing on the optimisation of finding anomalies, and that most algorithms can only handle low dimensional data, because of their high computational complexity.

This thesis, however, presents a different kind of model-based method that is focused on isolating anomalies rather than creating these normal data profiles. To accomplish this, the idea is to capitalize on the two properties anomalies have, which have been mentioned before. With this method and these properties we should be able to isolate them rather quickly from other data points. In this paper it will be explained that a tree structure can be built effectively and efficiently while the leaves of the tree are all the data points. The closer they are to the root of the tree, the quicker they are isolated from the rest, meaning that the normal data points will be deeper at the end of the tree. This isolation property the tree has, will form our foundation of the method we will use to detect anomalies. In the rest of the paper this tree will be called Isolation tree or iTree.

Isolation forest or iForest, the method we will investigate in this thesis, can be built from multiple iTrees from the same data set. The data points that have the shortest average path length are the anomalies we are looking for. Which leads me to my research question: "What is the accuracy of isolation forest and what are the effects of misclassified anomalies?".

1.1 Artificial intelligence and societal relevance

While other already existing methods for anomaly detection only focus on which points can be considered normal and not divergent, this machine learning based algorithm is actually more efficient and can operate in higher complexities making it more useful and applicable to more sorts of data sets. These data sets can contain all sorts of data, making it advantageous in all different sectors.

1.2 Structure of this thesis

In Chapter 2, the data from the data set and the reason why this particular data set was chosen will be explained. In Chapter 3, the algorithm will be explained and discussed. In Chapter 4 the results of the algorithm will be applied to the data sets. Chapter 5 will discuss the effects of the misclassification of anomalies. Chapter 6 analyzes the results and Chapter 7 will be used to reflect upon the utility of isolation forest and whether it should or should not be used more often, while considering the risk of its effects.

Chapter 2

Data

The data sets we use come from a bigger data set: "KDDCUP99", from the UCI machine learning repository[8]. The subsets we use are called Http (KDDCUP99) and Smtp (KDDCUP99)[12]. These are constructed by splitting the original data set using the service attribute, the four of which are: service, duration, src_bytes and dst_bytes. The service attribute is then divided into http, smtp, ftp, ftp_data and others subsets of which http and smtp are used. Both data sets have an X_file and a y_file, in which the X_file is a list of multi-dimensional points of data and the y_file labels the data points as either either 1 or 0, respectively an outlier or an inlier.

The data sets Http and Smtp contain respectively 567,497 and 95,156 data points of which 2211 (0.4%) and 30 (0.3%) are outliers. After the algorithm has been completed the data points will be classified in four different groups: True positives (TP), False positives (FP), True Negatives (TN) and False Negatives (FN). A data point is a TP when the algorithm classifies the point as an outlier and it is in fact an outlier. FP is when it is classified as an outlier, but it is actually not one. FN are outliers that are not classified as such and TN are inliers classified as inliers[13].

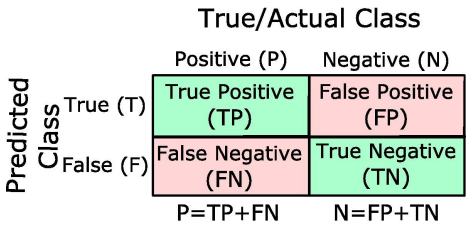


Figure 2.1: Classification of data in P/N and T/F

With the data classified in these classes, we can calculate the Precision, Recall and F-measure. They respectively mean the percentage TP of all the positive selected data. The percentage TP of TP and FN combined, and F-measure is the harmonic mean of the precision and recall. These measurements give us the possibility to compare and look at the accuracy in general.

Chapter 3

Algorithm

This chapter provides an overview about all the elements of isolation forest. How an iForest is constructed (section 3.1), the iTrees it consists of (section 3.2), the path length from the root to a datapoint (section 3.3) and the way the anomaly score is calculated (section 3.4). But first, what does it mean for a data point to be isolated?

A data point is isolated when it is separated from the other data points. Usually there are only a few anomalies in a data set with different characteristics. This gives them a higher chance to be isolated when recursively, uniform partitioned. This process continues until all data points are isolated. So when an iTree is constructed and all data points are secluded from the others in the tree, the data points with the shortest path to the root of the tree are the ones which are most likely to be an outlier. This is shown in figure 3.1.

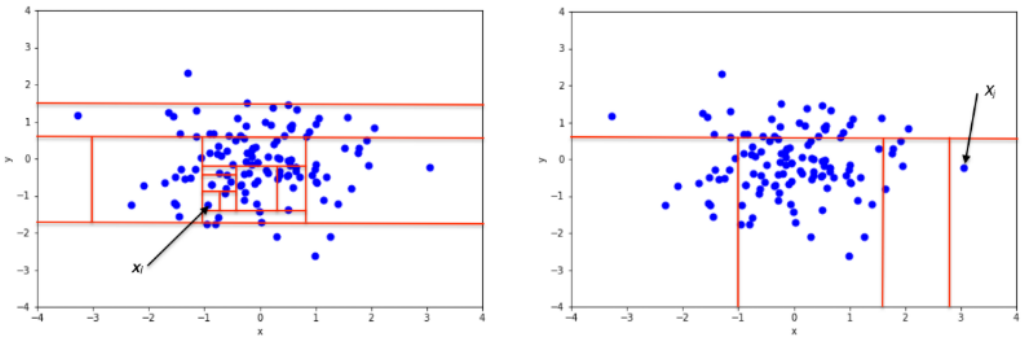
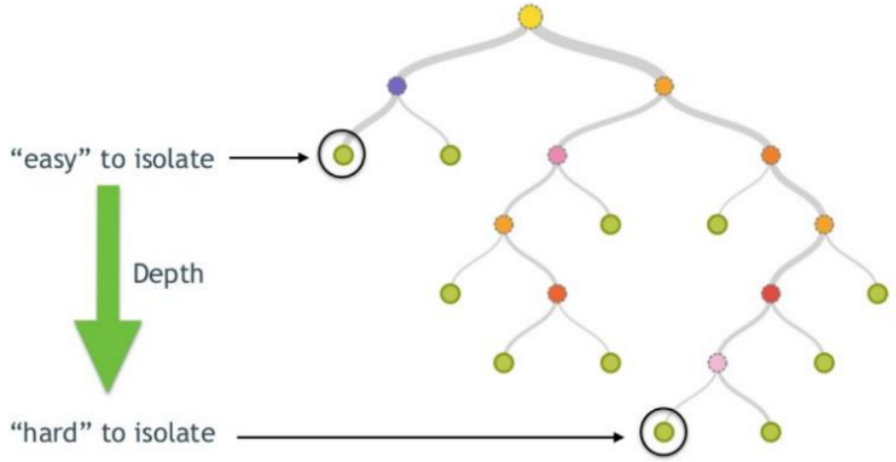


Figure 3.1: Difference in amount of separation needed to be isolated



3.1 iForest

To be able to use isolation forest and detect anomalies, we first have to train the algorithm. After training it, we can test it and get a certain anomaly score (section 3.4) for each data point. In figure 3.2 is the foundation of the iForest algorithm in detail. On input X, number of trees t, and sub-sampling size ψ a set of t iTrees are formed. Input X can be any multidimensional database. The amount of iTrees needed to calculate the average path length for each data point is usually quite low, since it converges rather quickly (appendix A.1). How the iTrees are built is discussed in section 3.2. The union of all the iTrees that are made, is called the iForest.

Algorithm 1 : $iForest(X, t, \psi)$

Inputs: X - input data, t - number of trees, ψ - sub-sampling size
Output: a set of t *iTrees*

1: **Initialize** $Forest$
2: set height limit $l = ceiling(\log_2 \psi)$
3: **for** $i = 1$ to t **do**
4: $X' \leftarrow sample(X, \psi)$
5: $Forest \leftarrow Forest \cup iTree(X', 0, l)$
6: **end for**
7: **return** $Forest$

Figure 3.2: iForest algorithm[11]

3.2 iTree

iForest generates t iTrees. The iTrees are where the actual data is stored and divided. As long as there are data points left that are not isolated, it continues to divide the data in two. It selects a random attribute in the data set X, and splits the data set for a random split point between the minimal and maximal values of the attribute. When it splits, it stores the one half in the left side of the iTree and the other in the right half. For both those sides it continues splitting when not isolated. In figure 3.3 a more detailed way is depicted. ExNode stands for nodes that are isolated. InNode are nodes that are not isolated yet. An iTree is a proper binary tree, meaning that each node has either zero daughters (in other words ExNode) or two daughters (making it an InNode). With the iTree done, the only remaining part is the task of detecting the anomalies itself. The way they are detected in iTrees is by looking at their path lengths to the root in the iTree, or by looking at their anomaly score. The anomalies are the ones at the top of the list. The next section 3.3 will define and calculate the path length.

Algorithm 2 : $iTree(X, e, l)$

Inputs: X - input data, e - current tree height, l - height limit
Output: an iTree

1: **if** $e \geq l$ or $|X| \leq 1$ **then**
2: return *exNode*$\{Size \leftarrow |X|\}$
3: **else**
4: let Q be a list of attributes in X
5: randomly select an attribute $q \in Q$
6: randomly select a split point p from max and min values of attribute q in X
7: $X_l \leftarrow filter(X, q < p)$
8: $X_r \leftarrow filter(X, q \geq p)$
9: return *inNode*$\{Left \leftarrow iTree(X_l, e+1, l)$,
10: $Right \leftarrow iTree(X_r, e+1, l)$,
11: $SplitAtt \leftarrow q$,
12: $SplitValue \leftarrow p\}$
13: **end if**

Figure 3.3: iTree build algorithm[11]

3.3 Path Length

The path length $h(x)$ of a data point x is calculated by the number of edges traveled, from the root node of the iTree to the external node x. As seen in figure 3.4, by using recursion the algorithm finds its path from the root down to node x based on the data stored in x. The average path length for each data point over t amount of iTrees will be used to calculate the anomaly score , which will be explained in section 3.4.

Algorithm 3 : $PathLength(x, T, e)$
Inputs : x - an instance, T - an iTree, e - current path length; to be initialized to zero when first called
Output: path length of x
1: **if** T is an external node **then**
2: return $e + c(T.size)$ {$c(.)$ is defined in Equation 1}
3: **end if**
4: $a \leftarrow T.splitAtt$
5: **if** $x_a < T.splitValue$ **then**
6: return $PathLength(x, T.left, e + 1)$
7: **else** $\{x_a \geq T.splitValue\}$
8: return $PathLength(x, T.right, e + 1)$
9: **end if**

Figure 3.4: iForest algorithm[11]

3.4 Anomaly score

An anomaly score is needed in every anomaly detection algorithm. It is used to rank the data and based on that rank, the algorithm decides which data entries are most likely to be anomalies compared to the others. The anomaly score computation is based on the observation of the structure of iTrees, which resembles Binary Search Trees(BST)[7]. When the path length to an external node terminates, this results in an unsuccessful search in the BST and the following formula 3.1[11] is applied.

$$c(m) = \begin{cases} 2H(m-1) - \frac{2(m-1)}{n} & \text{for } m > 2 \\ 1 & \text{for } m = 2 \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

In the formula above n is the testing data size and m the size of the sample set. The latter which can be of equal size when the same data set is used for training. H is the harmonic number and can be calculated by $H(i) = ln(i)+\gamma$, where $\gamma = 0.5772136649$ (the Euler–Mascheroni constant[4]). As $c(m)$ is the average of $h(x)$ given m, we can use this to normalise $h(x)$. The anomaly score s of a data point x can be defined with the following formula 3.2[11]:

$$s(x, m) = 2^{-\frac{E(h(x))}{c(m)}} \tag{3.2}$$

$E(h(x))$ is the average of $h(x)$ from a set of iTrees. With this formula, we can conclude the following three points:

- When $E(h(x))$ approaches $c(m)$, then $s(x, m) = 2^{-\frac{c(m)}{c(m)}}$. Resulting in s approaching 0.5.
- When $E(h(x))$ approaches 0, meaning it is close to the root, it results in s approaching 1.
- When $E(h(x))$ approaches $m-1$, the end of the tree, it results in s approaching 0.

With these previous three points in mind, we can infer the following three as well.

- If the data point has an anomaly score of s close to 1, it is very likely to be an anomaly.
- If the data point has an anomaly score of s smaller than 0.5, it is very likely to be an inlier.
- If all the data points return an anomaly score of around 0.5, then it is very likely that there is no anomaly in the data.

Chapter 4

Results

At first the only data set researched was the HTTP data set. The first table made is table A.2. Here, the only variable is contamination, which stands for the percentage of anomalies expected in the data set. This can be preset before running the code, functioning as a threshold. The data set contained labels saying whether each data point is an outlier or inlier as well. With this information it is possible to check how accurate isolation forest when applied to the data sets.

Figure 4.1 is a graph made from the data in appendix A.2. Figure 4.2 is zoomed in on the lower 5000 points of figure 4.1. Almost 99% of the anomalies were hidden within the 1.3% and 1.4% contamination frame. While the contamination is only 0.4%, the algorithm classifies too many inliers as outliers. What the cause is of this result will be further explained in chapter 5.

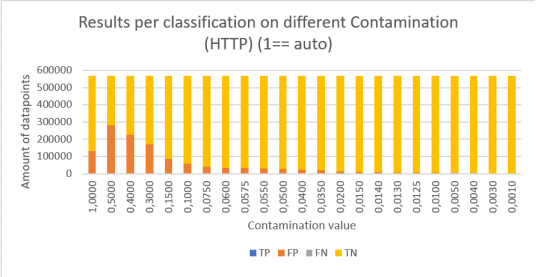


Figure 4.1: HTTP classification at different contamination rates

With the result that all data labeled as anomalies were very similar, the algorithm was applied to a second data set SMTP to determine whether the results would be different. The results can be found in table A.3, and the graphs in appendix A.6 and appendix A.7. On this data set, isolation forest does not display results as accurately as predicted. At a contamination rate of 38%, all anomalies were found. Lower contamination rates resulted in at least three or more missed anomalies. While the amount of anomalies in SMTP is 0.3%.

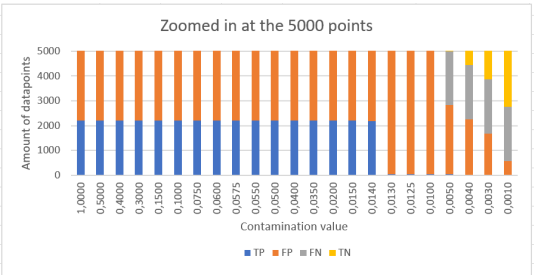


Figure 4.2: figure 4.1 zoomed in

Chapter 5

Analysis

The inaccurate values that isolation forest returned are a problem. In the HTTP data set 99% of the anomalies were clustered. This means that the values of the attributes were close together. When plotted they were so close next to each other, that it is hard for isolation forest to split these data. If an iTree were to return the height of these values they would be lower in the tree than some inliers that happen to be further off than most inliers. This results in the algorithm selecting inliers as outliers and seeing the group of anomalies as inliers. See figure 5.1.

5.1 Possible Solutions

To resolve this issue, the data points were split into subsets. With a subset it might be possible for the grouped anomalies to be more divided when in subsets, having a higher chance of being caught with isolation forest. With these subsets, two different approaches were taken to determine if the accuracy would increase: (1) training isolation forest on the original HTTP data set and testing the subsets on this trained version; represented in appendix A.4. (2) training isolation forest on the subsets and testing the subsets on this specifically trained isolation forest version appendix A.5. Unfortunately there is no difference between these approaches and the original trained data set.

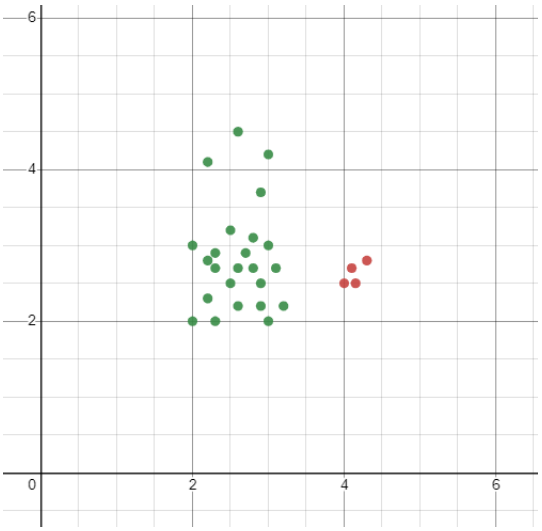


Figure 5.1: Clustered anomalies

Chapter 6

Effects

In this chapter, the effects of misclassification are discussed. The scope is cyber security. In section 6.1 incorrectly classifying too many data points as false positives is discussed. Similar to section 6.1, section 6.2 will discuss the counterpart: false negatives.

6.1 False Positives

For each data point classified as false positives, in context of the security of a website, results in having to check in case of a breach. Confirming whether (1) the information is disclosed to the intended recipient and is not available to outsiders, (2) the information is correct and cannot be modified by outsiders and (3) the information resources are always there to work with and cannot be blocked by outsiders[3]. If we assume that scanning for a breach the same is as having a breach, but without the cost repayment of the stolen records, then we can presume, that with help of the written report by IBM in 2019[10] and its data, that a breach control cost is estimated at an average of 84 thousand American dollars.

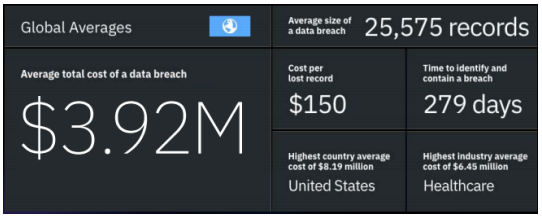


Figure 6.1: Global average breach cost 2018-2019[10]

6.2 False Negatives

Data points that are classified as false negatives, can be seen as a breaches in the website. These breaches can have numerous effects. The costs to solve issues like these are on average 47 times as big opposed to resolving a false positive. The estimated global average cost would be 3.92 million American dollars, see figure 6.1. If we were to plot the costs for every different contamination rate of table A.2. With a FP costing 84 thousand and a FN costing 3.92 million. The following graphs A.12 and A.13 would appear. The minimum of the graph is at a contamination of 0.014 classifying 5749 FP and 23 FN.

In the end minimizing both would be perfect. However, it will always be the case that FN are more costly. So if by increasing the contamination rate the amount of FN can significantly be lowered but accepting the increase in FP, then that should be prioritized.

Chapter 7

Discussion & Conclusion

7.1 Discussion

General discussion: The algorithm & results

Although the algorithm was not as accurate on the two chosen data sets, there is a possible issue that could be the cause. The algorithm does have a lot of potential and further research into this algorithm might resolve this anomaly grouping issue. Personally I thought with dividing the data in subsets and training it on the two possible training sets could have solved this issue. Another strategy might be to subset the data. Train the algorithm with the different subsets and then test with the full dataset.

7.2 Conclusion

The question remains on how effective isolation forest can be. Not all data sets contain grouped anomalies, because grouped data points usually means that they are inliers. Alas, it did not seem to be the case with the data in this paper. If it were to be explored on data sets outside of the KDDCUP99, it might deliver better results. For what to prioritize in the algorithm however, would definitely be to accept more false positives if it means preventing isolation forest from classifying false negatives. Which can be accomplished by increasing the contamination rate.

Bibliography

[1] Naoki Abe, Bianca Zadrozny, and John Langford. Outlier detection by active learning. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 504–509, 2006.

[2] Emin Aleskerov, Bernd Freisleben, and Bharat Rao. Cardwatch: A neural network based database mining system for credit card fraud detection. In *Proceedings of the IEEE/IAFE 1997 computational intelligence for financial engineering (CIFEr)*, pages 220–226. IEEE, 1997.

[3] Jason Andress. *The basics of information security: understanding the fundamentals of InfoSec in theory and practice*, page 240. Syngress, 2014.

[4] Car Ant Bretschneider. Theoriae logarithmi integralis lineamenta nova. *Journal für die reine und angewandte Mathematik*, 1837(17):257–285, 1837.

[5] 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*, pages 93–104, 2000.

[6] Varun Chandola. Minnesota intrusion detection system (minds). pages 1–2, 2009.

[7] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*, pages 286–299. MIT press, 2009.

[8] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[9] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):1641–1650, 2003.

[10] IBM. Cost of a data breach study, 2019 annual report. https://www.ibm.com/security/data-breach, December 2019.

[11] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[12] Shebuti Rayana. ODDS library, 2016.

[13] Alaa Tharwat. Classification assessment methods. *Applied Computing and Informatics*, 2018.

[14] Weng-Keen Wong, Andrew W Moore, Gregory F Cooper, and Michael M Wagner. Bayesian network anomaly pattern detection for disease outbreaks. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 808–815, 2003.

Appendix A

Appendix Graphs & Tables

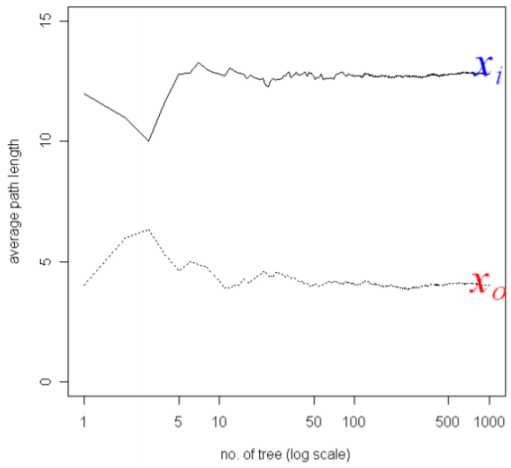


Figure A.1: Average path lengths converge, with X_0 as outlier and X_1 as inlier[11]

Contamination	TP	FP	FN	TN	Total	Precision	Recall	Fmeasure	Averages
1,0000	2211	128609	0	436677	567497	0,01690109	1	0,03324	0,082513845
0,5000	2211	281534	0	283752	567497	0,00779221	1	0,015464	0,049392387
0,4000	2211	224755	0	340531	567497	0,00974155	1	0,019295	0,057282929
0,3000	2211	167973	0	397313	567497	0,01299182	1	0,02565	0,069330026
0,1500	2211	82861	0	482425	567497	0,02598975	1	0,050663	0,109605006
0,1000	2211	54538	0	510748	567497	0,03896104	1	0,075	0,142965286
0,0750	2211	40331	0	524955	567497	0,05197217	1	0,098809	0,172526498
0,0600	2211	31835	0	533451	567497	0,06494155	1	0,121963	0,199334822
0,0575	2204	30427	7	534859	567497	0,06754313	0,996834	0,126514	0,204227484
0,0550	2204	28994	7	536292	567497	0,07064555	0,996834	0,13194	0,210230278
0,0500	2204	26171	7	539115	567497	0,07767401	0,996834	0,144118	0,223463102
0,0400	2204	20493	7	544793	567497	0,09710534	0,996834	0,176971	0,257784165
0,0350	2204	17659	7	547627	567497	0,11096008	0,996834	0,199692	0,280575695
0,0200	2202	9144	9	556142	567497	0,19407721	0,995929	0,324851	0,397461994
0,0150	2201	6312	10	558974	567497	0,25854575	0,995477	0,410481	0,472738146
0,0140	2188	5749	23	559537	567497	0,27567091	0,989597	0,431218	0,489984252
0,0130	47	5289	2164	559997	567497	0,0088081	0,021257	0,012455	0,013261161
0,0125	47	5289	2164	559997	567497	0,0088081	0,021257	0,012455	0,013261161
0,0100	47	5289	2164	559997	567497	0,0088081	0,021257	0,012455	0,013261161
0,0050	41	2795	2170	562491	567497	0,01445698	0,018544	0,016247	0,016331187
0,0040	31	2232	2180	563054	567497	0,01369863	0,014021	0,013858	0,013858469
0,0030	31	1659	2180	563627	567497	0,0183432	0,014021	0,015893	0,015988997
0,0010	24	544	2187	564742	567497	0,04225352	0,010855	0,017272	0,019934837

Figure A.2: HTTP

Contamination	TP	FP	FN	TN	Total	Precision	Recall	Fmeasure	Averages
1,000	23	14517	7	80609	95156	0,002	0,767	0,003	0,01564430
0,500	30	47548	0	47578	95156	0,001	1,000	0,001	0,00926251
0,400	30	38032	0	57094	95156	0,001	1,000	0,002	0,01074770
0,380	30	36128	0	58998	95156	0,001	1,000	0,002	0,01112161
0,360	27	34229	3	60897	95156	0,001	0,900	0,002	0,01037645
0,340	27	32332	3	62794	95156	0,001	0,900	0,002	0,01077794
0,320	23	30427	7	64699	95156	0,001	0,767	0,002	0,00956085
0,300	23	28521	7	66605	95156	0,001	0,767	0,002	0,00998165
0,200	23	19008	7	76118	95156	0,001	0,767	0,002	0,01307660
0,100	23	9493	7	85633	95156	0,002	0,767	0,005	0,02074621
0,050	21	4737	9	90389	95156	0,004	0,700	0,009	0,03003742
0,040	21	3786	9	91340	95156	0,006	0,700	0,011	0,03483352
0,030	21	2833	9	92293	95156	0,007	0,700	0,015	0,04217345
0,028	15	2650	15	92476	95156	0,006	0,500	0,011	0,03152406
0,026	15	2460	15	92666	95156	0,006	0,500	0,012	0,03310802
0,024	14	2244	16	92882	95156	0,006	0,467	0,012	0,03283757
0,022	5	2088	25	93038	95156	0,002	0,167	0,005	0,01233199
0,020	0	1904	30	93222	95156	0,000	0,000	0,000	0,00000000
0,010	0	943	30	94183	95156	0,000	0,000	0,000	0,00000000

Figure A.3: SMTP

Subset	TP	FP	FN	TN	Total	Precision	Recall	Fmeasure	Averages
1	71	590	1	18466	19128	0,10741	0,98611	0,19372	0,27377
2	60	569	0	18314	18943	0,09539	1,00000	0,17417	0,25516
3	82	580	0	18121	18783	0,12387	1,00000	0,22043	0,30112
4	77	598	1	18384	19060	0,11407	0,98718	0,20452	0,28451
5	68	580	1	18525	19174	0,10494	0,98551	0,18968	0,26969
6	67	578	0	18257	18902	0,10388	1,00000	0,18820	0,26939
7	72	587	0	18308	18967	0,10926	1,00000	0,19699	0,27816
8	56	611	0	18450	19117	0,08396	1,00000	0,15491	0,23517
9	76	595	0	18466	19137	0,11326	1,00000	0,20348	0,28458
10	73	607	0	18212	18892	0,10735	1,00000	0,19389	0,27508
11	70	587	0	18108	18765	0,10654	1,00000	0,19257	0,27376
12	58	602	0	18097	18757	0,08788	1,00000	0,16156	0,24214
13	66	613	1	18102	18782	0,09720	0,98507	0,17694	0,25684
14	71	570	0	18273	18914	0,11076	1,00000	0,19944	0,28059
15	82	620	1	18356	19059	0,11681	0,98795	0,20892	0,28889
16	70	565	0	18312	18947	0,11024	1,00000	0,19858	0,27974
17	85	621	0	18134	18840	0,12040	1,00000	0,21492	0,29578
18	74	604	1	18406	19085	0,10914	0,98667	0,19655	0,27662
19	69	575	0	18422	19066	0,10714	1,00000	0,19355	0,27474
20	64	598	0	17879	18541	0,09668	1,00000	0,17631	0,25735
21	78	578	0	18127	18783	0,11890	1,00000	0,21253	0,29345
22	81	607	0	18404	19092	0,11773	1,00000	0,21066	0,29163
23	84	609	0	18396	19089	0,12121	1,00000	0,21622	0,29704
24	78	584	0	18098	18760	0,11782	1,00000	0,21081	0,29177
25	82	573	0	18183	18838	0,12519	1,00000	0,22252	0,30314
26	84	567	1	18344	18996	0,12903	0,98824	0,22826	0,30761
27	92	583	0	18166	18841	0,13630	1,00000	0,23990	0,31977
28	77	601	0	18164	18842	0,11357	1,00000	0,20397	0,28507
29	61	548	0	18089	18698	0,10016	1,00000	0,18209	0,26323
30	76	559	0	18064	18699	0,11969	1,00000	0,21378	0,29467
Average for subset	72,95	588,34	0,23	18253,58	18915,91	0,11024	0,99687	0,19843	0,27938
Total	2204	17659	7	547627	567497				

Figure A.4: HTTP divided into 30 subsets with Contamination = 0,035 trained on HTTP set

Subset	TP	FP	FN	TN	Total	Precision	Recall	Fmeasure	Averages
1	68	590	0	18131	18789	0,10334	1,00000	0,18733	0,26851
2	68	578	0	17798	18444	0,10526	1,00000	0,19048	0,27167
3	71	598	0	18419	19088	0,10613	1,00000	0,19189	0,27308
4	72	591	0	18272	18935	0,10860	1,00000	0,19592	0,27710
5	87	571	0	18138	18796	0,13222	1,00000	0,23356	0,31373
6	74	601	0	18584	19259	0,10963	1,00000	0,19760	0,27876
7	72	582	0	18022	18676	0,11009	1,00000	0,19835	0,27951
8	82	584	0	18370	19036	0,12312	1,00000	0,21925	0,29998
9	64	597	1	18202	18864	0,09682	0,98462	0,17631	0,25616
10	79	578	0	18159	18816	0,12024	1,00000	0,21467	0,29554
11	74	590	0	18280	18944	0,11145	1,00000	0,20054	0,28168
12	59	610	0	18424	19093	0,08819	1,00000	0,16209	0,24269
13	70	587	1	18105	18763	0,10654	0,98592	0,19231	0,27235
14	81	576	0	18103	18760	0,12329	1,00000	0,21951	0,30023
15	78	581	0	18233	18892	0,11836	1,00000	0,21167	0,29261
16	66	593	0	18232	18891	0,10015	1,00000	0,18207	0,26321
17	65	588	0	17987	18640	0,09954	1,00000	0,18106	0,26218
18	82	582	1	18282	18947	0,12349	0,98795	0,21954	0,29920
19	74	585	1	18182	18842	0,11229	0,98667	0,20163	0,28164
20	83	585	0	18425	19093	0,12425	1,00000	0,22104	0,30171
21	64	597	0	18211	18872	0,09682	1,00000	0,17655	0,25760
22	82	589	0	18475	19146	0,12221	1,00000	0,21780	0,29857
23	66	597	1	18281	18945	0,09955	0,98507	0,18082	0,26077
24	74	591	0	18310	18975	0,11128	1,00000	0,20027	0,28141
25	74	586	1	18178	18839	0,11212	0,98667	0,20136	0,28137
26	73	591	0	18308	18972	0,10994	1,00000	0,19810	0,27926
27	84	585	0	18434	19103	0,12556	1,00000	0,22311	0,30371
28	76	591	0	18378	19045	0,11394	1,00000	0,20458	0,28566
29	67	600	1	18366	19034	0,10045	0,98529	0,18231	0,26229
30	75	590	0	18333	18998	0,11278	1,00000	0,20270	0,28381
Average for subset	73,14	588,74	0,23	18253,37	18915,84	0,11045	0,99672	0,19879	0,27971
Total	2204	17664	7	547622	567497				

Figure A.5: HTTP divided into 30 subsets with Contamination = 0,035 trained on the subset itself

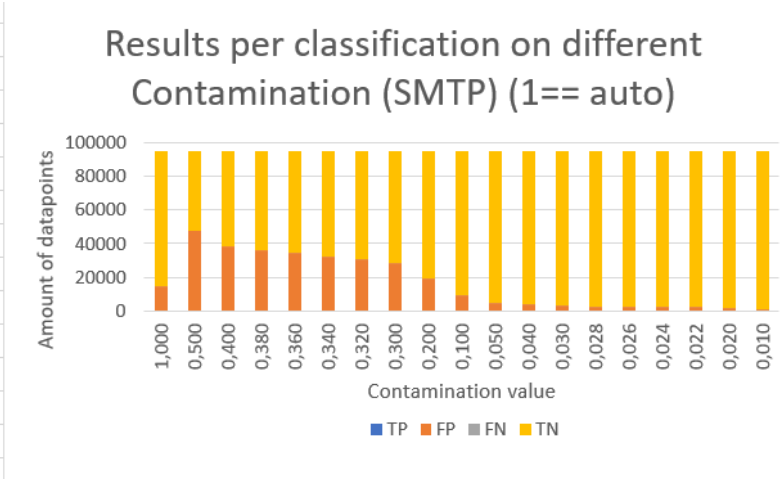


Figure A.6: SMTP classification zoomed in

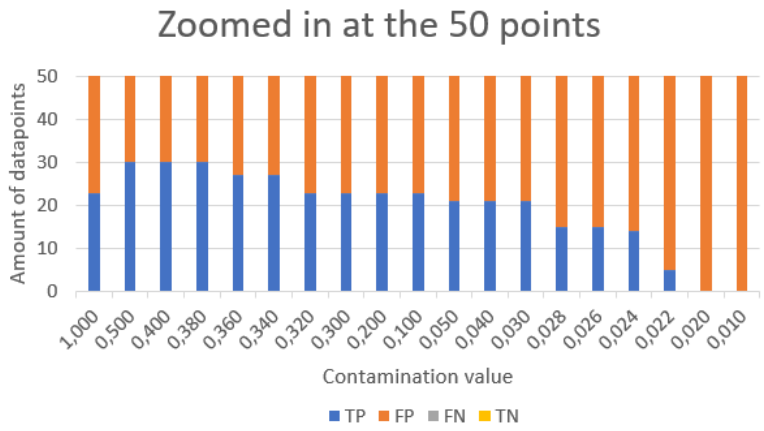


Figure A.7: SMTP classification zoomed in

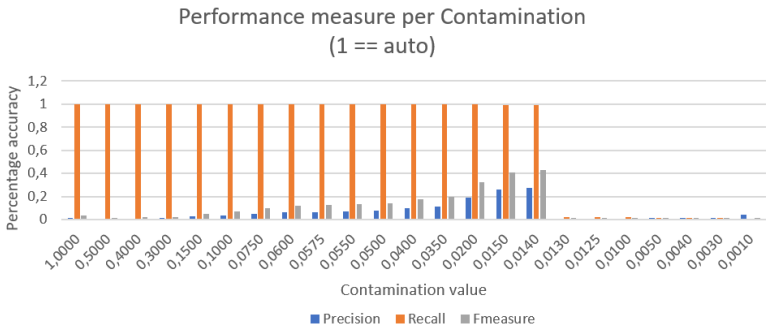


Figure A.8: HTTP performance

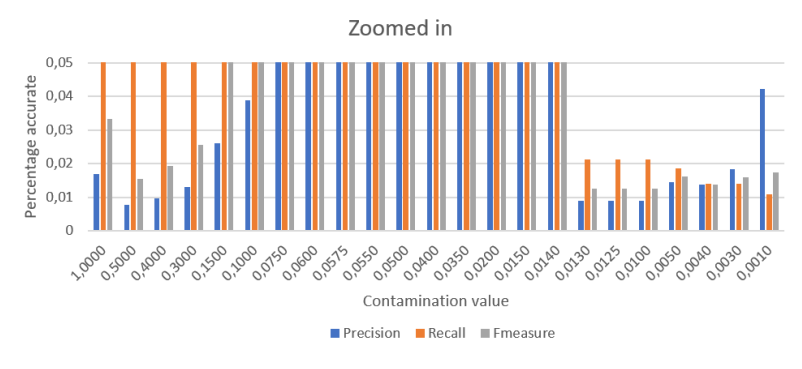


Figure A.9: HTTP performance zoomed in

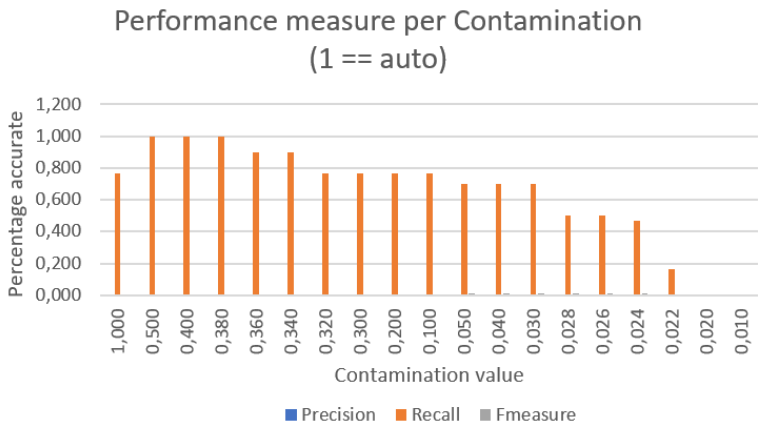


Figure A.10: SMTP performance

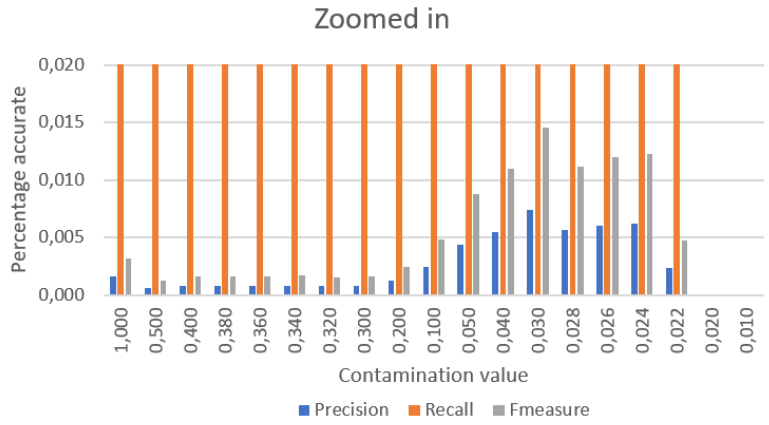


Figure A.11: SMTP performance zoomed in

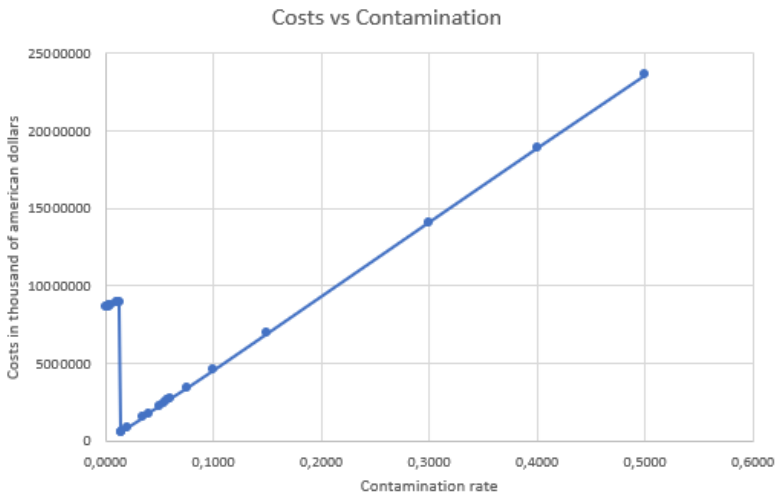


Figure A.12: Cost vs Contamination

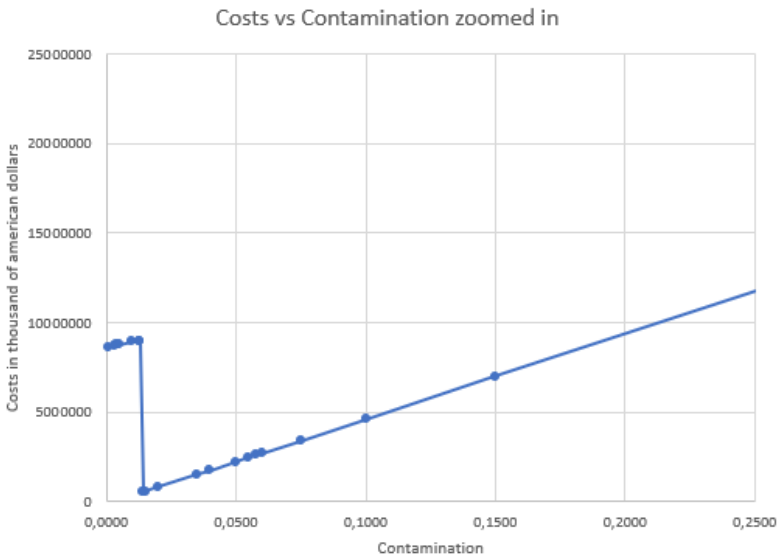


Figure A.13: Cost vs Contamination (zoomed in at the bottom)

Appendix B

Appendix Code

Github with full Python code of the Isolation_Forest_Accuracy can be found at:
https://github.com/GoossensH/BachelorThesis

Sklearn isolation forest class can be found at:
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

Sklearn isolation forest algorithm in code can be found at:
https://github.com/scikit-learn/scikit-learn/blob/fd237278e/sklearn/ensemble/_iforest.py#L27