Pune Institute of Computer Technology Dhankawadi, Pune

A SEMINAR REPORT ON

ANOMALY DETECTION USING MACHINE LEARNING

SUBMITTED BY

Name : Nirmayi Kelkar
Roll No. : 31141 Class: TE-1

Under the guidance of
Prof. Pranjali Joshi

DEPARTMENT OF COMPUTER ENGINEERING

Academic Year 2020-21

DEPARTMENT OF COMPUTER ENGINEERING

Pune Institute of Computer Technology Dhankawadi, Pune-43

CERTIFICATE

This is to certify that the Seminar report entitled

"ANOMALY DETECTION USING MACHINE LEARNING"

Submitted by

Nirmayi Kelkar Roll No. : 31141

has satisfactorily completed a seminar report under the guidance of Prof. Pranjali Joshi towards the partial fulfillment of third year Computer Engineering Semester II, Academic Year 2020-21 of Savitribai Phule Pune University.

Prof. Pranjali Joshi
Internal Guide
Department of Computer Engineering

Prof. M.S.Takalikar
Head

Place: Pune
Date: 08/11/2021

ACKNOWLEDGEMENT

It is my pleasure to present this report on “Anomaly detection using Machine Learning”.

I would like to express my gratitude to my guide, Prof. Pranjali Joshi, Department of Computer Engineering for her guidance and help. She has constantly supported me and has played a crucial role in the completion of this report. Her motivation and encouragement from the beginning till the end has enabled me to make this seminar in its present form.

I would like to thank our seminar coordinator, Prof. B.D. Zope for regular interactions and advice to improve the seminar.

I would also like to thank our Head of Department, Prof. M.S.Takalikar and Principal, Prof. R. Shreemati for their encouragement and support.

Last but not the least I would thank my teachers, parents and friends who have helped me and guided me at various stages of this process.

TABLE OF CONTENTS

ABSTRACT

Anomaly detection is a process that finds the outliers of a dataset i.e. those items that most probably do not belong to the dataset. These anomalies might point to unusual network traffic, uncover a defective sensor on the network, or simply identify data for pre-processing before analysis. Outlier detection is important in a variety of scenarios such as credit card fraud, cyber security and to detect faulty equipment in systems. Anomaly detection helps detect outliers and may be used in informing the responsible parties to take appropriate measures. Machine learning techniques have been used in the past for anomaly detection. In this seminar, we give a brief survey of the main techniques used for anomaly detection. A recent technique that is quickly gaining popularity is the Isolation tree algorithm for anomaly detection. This algorithm has been described, implemented and results shown for a standard data set that is widely available.

KEYWORDS

Anomaly detection, isolation forest, isolation tree, anomaly score, aberration.

1. INTRODUCTION

Data sets sometimes contain values that are significantly different from other values in the data set. Such values are commonly known as anomalies, outliers, aberrations or contaminants in different application domains. Anomaly detection is the technique of recognizing an outlier or event that does not follow a particular trend or behavior. Anomaly detection, thus, refers to the problem of finding abnormal patterns in a set of data.

Anomaly detection finds extensive use in a wide variety of applications. They are important for detecting fraudulent credit card activities, fault detection in critical machinery, intrusion detection in cyber security, different patterns in health data (e.g., heart rate anomalies) to name just a few. Further applications of anomaly detection are provided in the Section 2.

In Section 3, we will provide a brief literature survey of the gamut of techniques used by researchers for anomaly detection. We will see that one of the more recent and popular methods for anomaly detection is the Isolation Forest method.

Section 4 describes the Isolation Forest technique in detail along with the concepts of Isolation tree, Isolation Forest and Anomaly score while the detailed algorithm used for implementing this technique is provided in Section 5.

Section 6 describes the implementation of the algorithm using Python's scikit library and the use of a common (IRIS) dataset. Results of this implementation are also provided.

Finally, Section 7 concludes the report and identifies some promising avenues for further work.

2. MOTIVATION

The importance of detecting anomalies in a data set translates to significant and often critically actionable information in a wide variety of application domains. A recent financial report (Teddar and Buzzard, 2020) estimates that the financial loss due to fraudulent transactions has reached around US $ 17 billion annually with approximately 5% of users having experienced fraud incidents of some kind.

The following information outlines a few of these domains where anomaly detection plays a vital role in cost saving.

The manufacturing industry deals with heavy machinery such as pumps, pipes, and furnaces etc. which are critical assets for operation. Prediction of future anomalies in these machinery plays a vital role for saving millions of dollars on repair.

In defense and government setting, anomaly detection is best used for identifying excessive and fraudulent government spending, budgeting and audits. This can save governments a huge amount of money.

Anomalous traffic patterns in a computer network may indicate that a hacked computer is sending sensitive data to an unauthorized destination.

In the health care industry, an anomalous MRI image may indicate presence of tumors. This prompts the doctors for immediate remedial therapies.

Anomalous readings from a space craft sensor could indicate a fault in a component of the space craft. This helps the scientists to take immediate damage control actions.

The above lists the need for finding a comprehensive, fast and efficient need for anomaly detection.

3. LITERATURE SURVEY

A lot of literature already exists on the broad field of anomaly detection. Systematic reviews (e.g. Nassif et al., (2021), Chandola et al, (2009), etc) provide a good launching pad from which to explore the subject.

A study of the literature indicates that different machine learning techniques have been used to analyze the problem of anomaly detection. These may be broadly classified into two types:

(i) Supervised learning techniques

(ii) Unsupervised learning techniques.

These are described below.

(i) Supervised learning techniques

In supervised learning techniques, all instances of training data must have an attribute – normal or anomalous. Based on this attribute or label, the model will be able to predict whether a future instance of data will be normal or anomalous. Supervised learning techniques for anomaly detection are a type of binary classification problem for which there is extensive literature. An example of supervised learning techniques that has been used in the literature for anomaly detection is one class SVM.

In this technique, that can be applied to a dataset with n attributes, a set of hyperplanes is constructed in n dimensional space to separate the "normal" attributes from the "anomalous" attributes. However, this method suffers from the limitation that it is not suitable for large datasets and where the datasets have a large amount of noise.

Also, in most real life applications, data is generated in real time and there is no label of "normal" or "anomalous" on it.

(ii) Unsupervised learning techniques

Unsupervised learning techniques are those that are able to predict anomalies even in the absence of previously supplied labels such as “normal” or “anomalous”. These are of great practical importance and have been widely used in anomaly detection. Some of the most common techniques used in this case are (a) k-means (b) DBSCAN (c) Isolation forests.

These are briefly described below.

(a) K-means

In this method, data points in the set are looked at and are grouped into a predefined number of clusters, k depending on the Euclidean distance between the points. The centroid of each cluster is then calculated.

A threshold value is used by the model to detect anomalies. Any data point that lies beyond the threshold value from the centroid of all clusters is considered to be an anomaly.

However, this method suffers from the drawback that it is highly susceptible to noise and fails on non-convex clusters. It is also difficult to specify the value of k beforehand and the method is computationally intensive.

(b) Density based Spatial Clustering of Applications with Noise (DBSCAN)

DBSCAN is one of the most common clustering algorithms used in research and cited widely.

In this technique, multi-dimensional data is taken as input and clustered according to model parameters (Number of clusters, c, minimum neighbors, k and the neighborhood radius of a point, ε). If a point has greater than k neighbors within an ε distance, then it is considered a “core point” of the cluster and if it has exactly k neighbors it is called a “border point”. Points outside the clusters so created are considered anomalies.

However, this method suffers from the drawback that is difficult to specify the value of the number of clusters c beforehand since data is generated continuously on the fly.

(c) Isolation forests

This method was introduced by Liu et al., (2008) and has rapidly gained popularity being implemented in the Python scikit library. It has been successfully applied to detect anomalies in many diverse fields such as web-streaming and manufacturing (Susto et al., 2017).

It was recognized that most of anomaly detection approaches such as the ones described above, construct a profile of normal instances, then identify anomalies as those that do not conform to the normal profile. Therefore, the anomaly detection capabilities are a side-effect or by-product of the algorithm that essentially identifies what is normal (e.g. by clustering). This leads to two major drawbacks. (i) The approaches are seldom optimized to detect anomalies – as a result, the approaches often underperform, resulting in too many false alarms (ii) Many existing methods are constrained to low dimensional data and small data size.

In contrast, the Isolation Forest algorithm focuses on two properties of anomalies: (i) They are the minority consisting of only a few instances (ii) They have attribute values very different from normal attribute values. Focusing on these properties makes it easier to isolate them

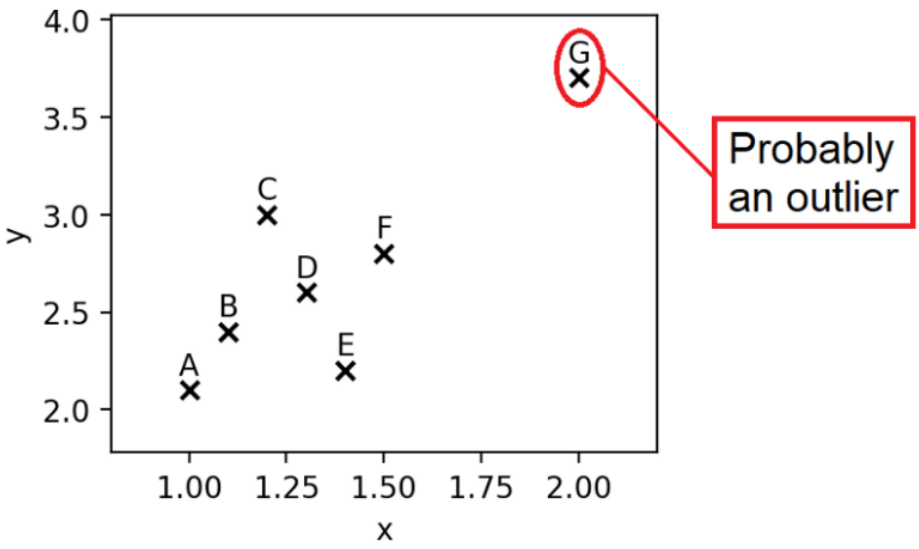
This technique has been discussed in more detail in the next section.

4. ISOLATION FOREST TECHNIQUE

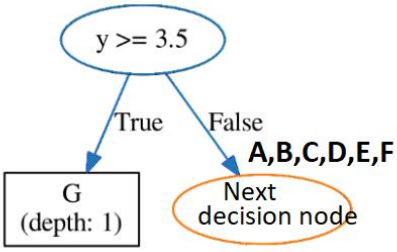
4.1 Isolation tree

As discussed in the preceding section, anomalies are few and different and therefore should be easier to isolate. The fundamental data structure used from this purpose is a binary tree structure called an isolation tree. Each node of the binary tree (except the leaf node) is a decision node that partitions the dataset into left and right sub-trees. This branching process is continued until all the data points have been isolated. This is illustrated below with an example.

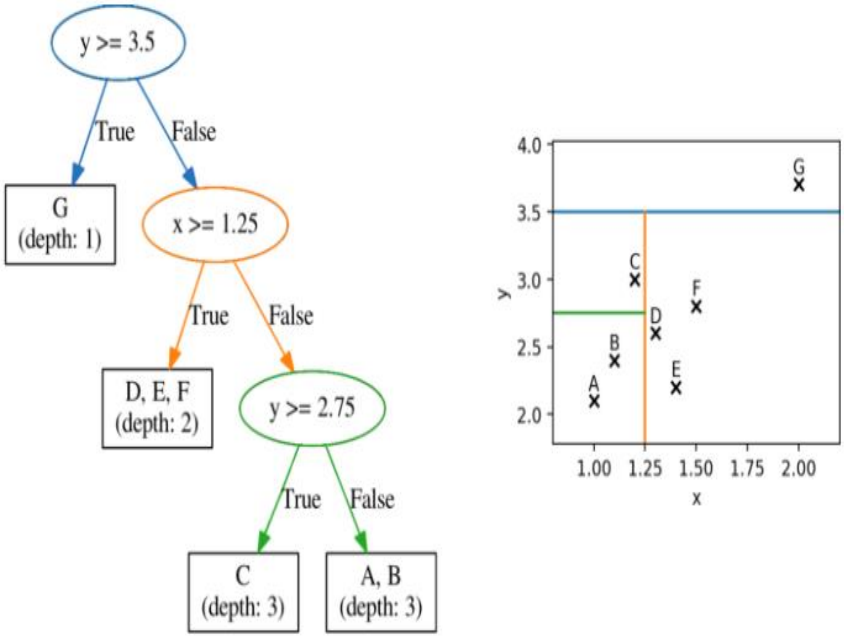
A sample data set consisting of the points A (1.0, 2.1), B (1.1, 2.4), C (1.2, 3.0), D (1.3, 2.6), E (1.4, 2.2), F (1.5, 2.8) and G (2.0, 3.7) has been plotted below.



An attribute (X or Y) is chosen at random (say Y) and a split value is also chosen at random (e.g. Y=3.5). A root decision node may be made as $Y \geq 3.5$ which will result in the data set splitting as shown below.



This process is continued further as shown (typically until all the nodes have been isolated or the maximum pre-specified height of the tree has been reached). This is shown below.

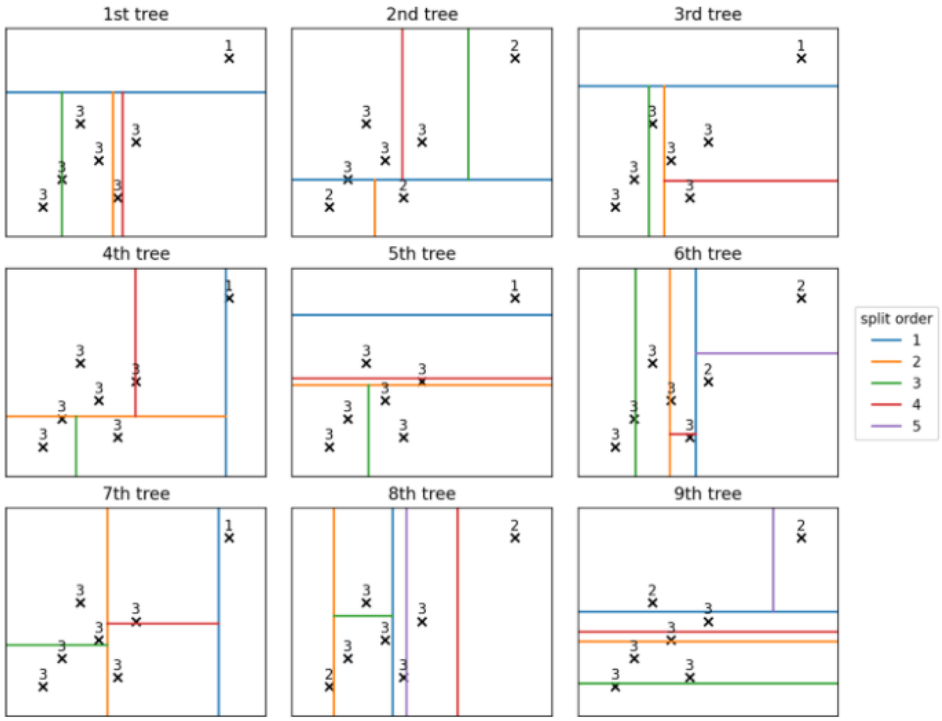


A significant finding of this step is that on average, outliers (such as G in the example above) will be closer to the root node (i.e. have a smaller depth) than a "normal" data point such as B or D.

4.2 Isolation forest

An important finding of the previous section was that on average an outlier data point lies closer to the root node (has smaller depth) than a normal point. In order to obtain a proper average, a large number of trees, $t = 100$ say are built with the random algorithm described in Section 4.1. The large number of trees together make up what is known as "Isolation forest". For the sample data considered, this is shown

below.



4.3 Anomaly score

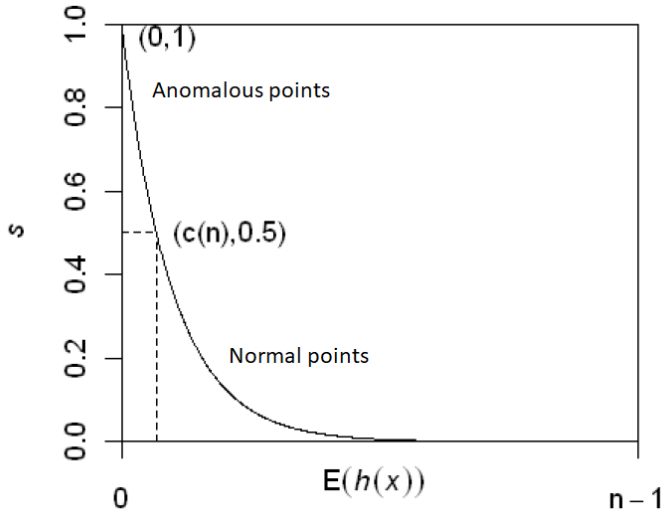
Each of the data points is passed through the ensemble of isolation trees (i.e. isolation forest) created in order to obtain its average depth, $E[h(x)]$. This is normalized with the average height of the tree of ψ nodes that is given by

$$c(\psi) = 2H(\psi - 1) - 2\frac{(\psi - 1)}{n};\ \psi > 2$$
$$= 1;\ \psi = 2$$
$$= 0; \text{otherwise}$$

This is then converted into an anomaly score given by

$$s(x, \psi) = 2^{-\frac{E[h(x)]}{c(\psi)}}$$

The anomaly scores come out to be between 0 and 1 with this formula. Anomalous points with small values of $E[h(x)]$ will have an anomaly score close to 1 while the normal points with large values of $E[h(x)]$ will have an anomaly score close to 0 as shown below.



The isolation forest may also be used to predict whether newly obtained data points that are not part of the original training data set will be normal points or anomalies.

The steps outlines above are codified into a three algorithms in the next section.

5. ALGORITHM FOR ISOLATION FOREST

The steps outlined in Section 4 have been codified in the form of three algorithms that are described in this section. The symbols used in this section are explained below.

5.1 Symbols and notation

x	a data point
X	a data set of n instances
n	number of data points in a data set, $n = \|X\|$
m	index of data point x_m, $m \in \{0, ..., n-1\}$
Q	a set of attributes
d	number of attributes, $d = \|Q\|$
q	an attribute
T	a tree or a node
t	number of trees
$h(x)$	returns the path length of x
$hlim$	evaluation height limit
ψ	subsampling size
l	a possible path length
s	an anomaly score or function which returns an anomaly score
k	the number of nearest neighbours
a	contamination level of anomalies in a data set
cl	number of anomaly clusters

5.2 Algorithm

Step 1 (Creating one isolation tree. iTree(X'))

Inputs: X′ - input data Output: an iTree

1: if X′ cannot be divided then

2: return exNode{Size ← |X′ |}

3: else

4: let Q be a list of attributes in X′

5: randomly select an attribute q ∈ Q

6: randomly select a split point p between the max and min values of attribute q in X′

7: Xl ← filter(X′ , q < p)

8: Xr ← filter(X′ , q ≥ p)

9: return inNode{Lef t ← iT ree(Xl),

10: Right ← iTree(Xr),

11: SplitAtt ← q,

12: SplitValue ← p}

13: end if

Step 2 Creating Isolation Forest (iForest (X,t,ψ)

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

1: Initialize Forest

2: for i = 1 to t do

3: X′ ← sample(X, ψ)

4: Forest ← Forest ∪ iTree(X′)

5: end for

6: return Forest

Step 3 Finding PathLength (x,T,hlim,e)

Inputs : x - an instance, T - an iTree, hlim - height limit, e - current path length; to be initialized to zero when first called Output: path length of x

1: if T is an external node or e ≥ hlim then

2: return e + c(T.size) {c(.) is defined in Equation 1}

3: end if

4: a ← T.splitAtt

5: if xa < T.splitV alue then

6: return P athLength(x, T.lef t, hlim, e + 1)

7: else {xa ≥ T.splitV alue}

8: return PathLength(x, T.right, hlim, e + 1)

9: end if

6. IMPLEMENTATION

The algorithm described above was implemented in Python using the scikit library.

6.1 Input Data

The IRIS data set which is a famous dataset used in Machine Learning literature was used for the implementation. The data set consists of 50 samples from each of three species of Iris family flowers (Iris Setosa, Iris Virginica and Iris Versicolor). Four attributes (sepal length, sepal width, petal length, petal width) are available for each instance. The tabulated data looks as follows

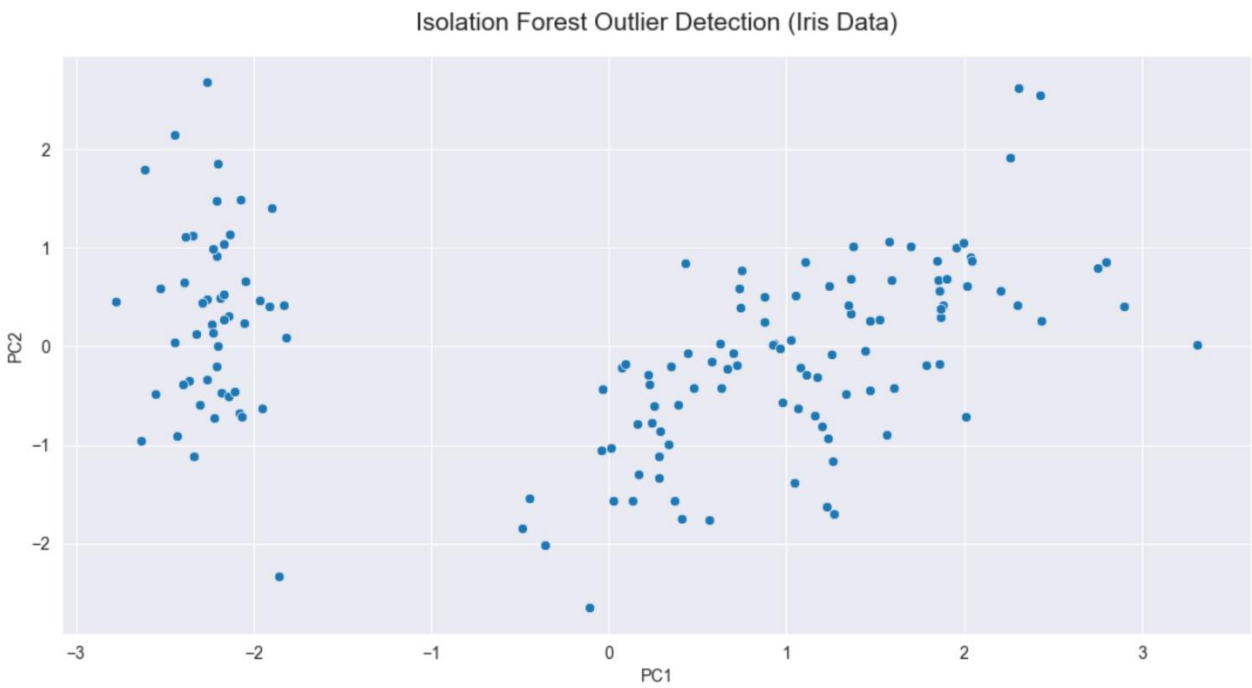
Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa
11	5.4	3.7	1.5	0.2	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3	1.4	0.1	Iris-setosa
14	4.3	3	1.1	0.1	Iris-setosa
15	5.8	4	1.2	0.2	Iris-setosa

147	6.3	2.5	5	1.9	Iris-virginica
148	6.5	3	5.2	2	Iris-virginica
149	6.2	3.4	5.4	2.3	Iris-virginica
150	5.9	3	5.1	1.8	Iris-virginica

6.2 Results

The data when subjected to a dimensionality reduction using PCA can be reduced to

two dimensions. The data when plotted out looks as follows.

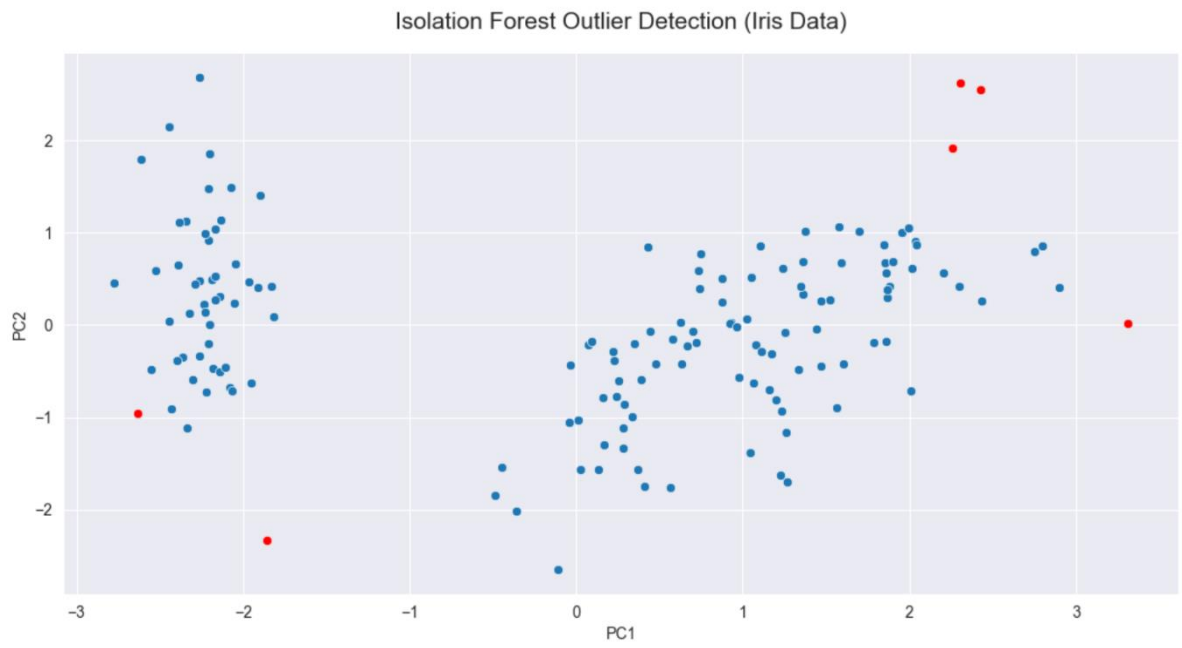


The Isolation Forest Algorithm available in the scikit sklearn.ensemble was used on this data with the following parameters:

(i) Number of isolation trees in the forest, $\boldsymbol{t = 100}$

(ii) Contamination parameter (% of outliers expected = 4%)

With these parameters, the six outliers were returned as part of the dataset. The outliers returned have been plotted in red in the diagram below.



It may be seen that visually the algorithm and the code is able to pick out the outliers in a real world data set such as the Iris dataset.

7. CONCLUSION AND FUTURE WORK

In this report, anomaly detections using Machine Learning was studied. A study of the past literature shows that practical data often requires the use of unsupervised techniques for anomaly detection. One of the most promising algorithms for detecting outliers in a given data set is the Isolation forest algorithm (Liu et al, 2008)

The algorithm was studied and implemented using Python's scikit library and a widely used and commonly available IRIS dataset. Results of this implementation showed that outliers are properly detected using the Isolation Forest algorithm.

Further work may be performed by using the Isolation Forest algorithm with different datasets.

A comparative study of the Isolation Forest algorithm along with some of the other algorithms mentioned (such as DBSCAN and k-Means) in terms of accuracy of the algorithms and speed of computation may also be performed.

REFERENCES

1. Ali Bou Nassif, Manar Abu Talib “Machine Learning for Anomaly Detection: A Systematic Review,”.Publisher-IEEE .doi:: 10.1109/ACCESS.2021.3083060. Date of Publication: 25 May 2021.
2. Chandola V., Banerjee, A., Kumar, V., (2009), “Anomaly Detection: A Survey”, ACM Computing Surveys, vol 41, no. 3, September 2009, Article 15.
3. Nassif A.B., Talib, M. A., Nasir, Q., Dakalbab, F. M., (2021), “Machine Learning for Anomaly Detection: A Systematic Review”, IEEE Access, Computer Science, June 2021, DOI 10.1109/ACCESS.2021.3083060
4. Teddar, K and Buzzard, J., “2020 Identity Fraud Study: genesis of the Identity Fraud Crisis” April 2020, Javelin Strategy, Internal Report.
5. Liu, F. T., Ting, K. M. and Zhou, Z. H., “Isolation forest”, in Proc. 8th IEEE Int. Conf. Data Mining, 2008, pp 413-422.
6. Susto, G.A., Beghi, A and McLoone S., “ Anomaly detection through an on-line Isolation forest : An application to plasma etching”, in Proc. 40th Int. Conv. Inf. Communication Technology, Electronics, Microelectronics (MIPRO), May 2017, pp 89-94.