Performance evaluation of unsupervised techniques in cyber-attack anomaly detection

Article *in* Journal of Ambient Intelligence and Humanized Computing · November 2020

DOI: 10.1007/s12652-019-01417-9

Jorge Meira
University of A Coruña
17 PUBLICATIONS **59** CITATIONS
SEE PROFILE

Rui Andrade
Instituto Superior de Engenharia do Porto
13 PUBLICATIONS **57** CITATIONS
SEE PROFILE

Isabel Praça
Instituto Superior de Engenharia do Porto
201 PUBLICATIONS **2,176** CITATIONS
SEE PROFILE

João Carneiro
Polytechnic Institute of Porto
58 PUBLICATIONS **387** CITATIONS
SEE PROFILE

Performance Evaluation of Unsupervised Techniques in Cyber-Attack Anomaly Detection

Jorge Meira[2], Rui Andrade[1], Isabel Praça[1], João Carneiro[1], Verónica Bolón-Canedo[2], Amparo Alonso-Betanzos[2], Goreti Marreiros[1]

{rfaar, icp, jomrc, mgt}@isep.ipp.pt

[1] GECAD – Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development, Institute of Engineering, Polytechnic of Porto (ISEP/IPP), Portugal

{j.a.meira, veronica.bolon, ciamparo}@udc.es

[2] LIDIA – Laboratory for Research and Development in Artificial Intelligence, Department of Computer Science, University of A Coruña, Spain

Abstract

Cyber security is a critical area in computer systems especially when dealing with sensitive data. At present, it is becoming increasingly important to assure that computer systems are secured from attacks due to modern society dependence from those systems. To prevent these attacks, nowadays most organizations make use of anomaly-based Intrusion Detection Systems (IDS). Usually, IDS contain machine learning algorithms which aid in predicting or detecting anomalous patterns in computer systems. Most of these algorithms are supervised techniques, which contain gaps in the detection of unknown patterns or Zero-day exploits, since these are not present in the algorithm learning phase. To address this problem, we present in this paper an empirical study of several unsupervised learning algorithms used in the detection of unknown attacks. In this study we evaluated and compared the performance of different types of anomaly detection techniques in two public available datasets: the NSL-KDD and the ISCX. The aim of this evaluation allows us to understand the behavior of these techniques and understand how they could be fitted in an IDS to fill the mentioned flaw. Also, the present evaluation could be used in the future, as a comparison of results with other unsupervised algorithms applied in the cybersecurity field. The results obtained show that the techniques used are capable of carrying out anomaly detection with an acceptable performance and thus making them suitable candidates for future integration in Intrusion Detection tools.

Keywords: Anomaly detection, One-class classification, intrusion detection, unsupervised learning.

1 Introduction

Computer systems play a major role in modern everyday life. Almost everything from personal calendars to financial records and e-commerce operations are done with resource to a computing device with a network connection. Important information is stored and sent in all sorts of devices, from small low power smartwatches to huge datacenters. This creates an extensive attack vector that intended individuals and/or organizations may try to outbreak. Attackers use a variety of different techniques to try to exploit safety flaws in systems. This may result in sensible data breaches, stolen user accounts or taking control over the system.
To combat these attacks, system administrators and security experts often need to use safety measures to eliminate these attacks or at least mitigate their effects. One of these safety measures are Intrusion Detection Systems (IDS). These systems perform cyber-attack detection, using a variety of techniques to discover failures and malicious activity in computer systems. IDS tend to follow one of two different approaches: (a) signature based, or (b) anomaly based. Signature-based detection requires prior knowledge of an attack before being able to identify it; on the other hand, techniques based on anomaly detection work by acquiring knowledge of the patterns that represent "normal" or "attack" data and then classify new data accordingly to their resemblance to those patterns. This later approach gives the IDS the possibility of detecting attacks, even if the attack is not currently known (a zero-day attack, that is, an attack that is unknown or unaddressed yet, and thus can be exploited to adversely affect the computer or network), because these new attacks may present more similarities to other previous attacks rather than to "normal" data.
Within anomaly-based approach IDSs, different algorithms may be used. Supervised learning algorithms are suitable for problems in which a set of already existing and previously classified samples can be used as a training dataset. On the other hand, when novel vulnerabilities and attacks are involved, there are no classified examples for a supervised algorithm to learn from it. One possibility in order to deal with this problem is the use of

unsupervised learning algorithms. Unsupervised learning techniques can learn what is normal for a given set of data and then are capable of finding deviations in new unclassified data, which in this scenario would indicate a possible attack that until now was unknown.
The motivation for this study comes from the SASSI – "Sistema de Apoio à decisão de Segurança em Sistemas Informáticos" (Decision Support System for Security in Computer System) project, which objective is the development of an Intelligent Decision Support System that centralizes, structures and allows the visualization of information regarding the activity of computer networks and the individual machines in given networks, allowing the automatic detection, prediction and prevention of anomalies, cyber-attacks and possible security risks. This platform aims to support computer network administrators who are increasingly faced with critical decision-making tasks regarding security problems that cannot be detected by typical anti-malware protection systems.
This paper focusses on cyber-attack and anomaly detection using unsupervised learning algorithms, and explores six of these algorithms: Autoencoder, One-Class Nearest Neighbor, Isolation Forest, One-Class K-Means, One-Class Scaled Convex Hull and One-Class Support Vector Machines, over two different public datasets the NSL-KDD (Tavallaee et al. 2009) and the ISCX datasets (Shiravi et al. 2012).
Our results show that the techniques used are capable of archiving high-performance results in the classification tasks tested in our case study and consequently are candidates for future implementation in an IDS.
This paper has the following structure: Section 2 presents some related work on this topic, Section 3 describes the workflow used including a description of the datasets and pre-processing techniques applied in our approach, Section 4 describes all of the unsupervised algorithms tested, how they work and which parameters where used in our application, Section 5 presents a comparative evaluation of the results, and finally Section 6 draws the conclusion and ideas for future work.

2 Related Work

As IDS's classification problems are a frequent topic of study in the literature, many authors have proposed and studied interesting techniques to deal with the problem of unknown attacks. The task of identifying if a new instance belongs to the class of the data that has been used for training the classifier, or whether it is an outlier, is known as one-class classification. This means that the classifier only learns the data patterns of one class (target class) in the training phase. There are other names called to this field like novelty or outlier detection, and concept learning (Khan and Madden 2014). One-class algorithms were proven to be an important tool for several domains as in disease detection (Gardner et al. 2006), intrusion detection (Giacinto et al. 2008), text/document classification (Manevitz and Yousef 2001), or predictive maintenance (Shin, Eom, and Kim 2005).
Fernández-Francos et al. (Fernández-Francos, Fontenla-Romero, and Alonso-Betanzos 2017) presented a novel One-Class classification algorithm purposed for targeting distributed environments called One-Class Convex Hull-Based Algorithm. Their results showed that this method was accurate in one-class classification problems and efficient in big data scenarios due to the distributed nature of the approach. Castillo et al. (Castillo et al. 2015) proposed a Distributed One-Class Support Vector Machine (DOC-SVM) method for classification problems. They experimented with different datasets and their results demonstrated that the proposed DOC-SVM was able to achieve accurate results and with a reduction in the necessary training time when compared to other classifiers known in the literature. Chen et al. (Chen et al. 2017) introduced the autoencoder ensembles for unsupervised outlier detection. They presented the random edge sampling technique in which it randomly drops connections in a neural network retaining a certain level of control on the connection density between several layers, so in this way they can create various models with different types of density. The mentioned method was used in conjunction with adaptive data sampling approach where the authors applied the RMSprop (Tieleman and Hinton 2012) optimization method to speed up the learning process. Their method, named as RandNet, which stands for Randomized Neural Network for Outlier Detection, showed robustness avoiding the overfitting problem, and it was competitive with respect to other neural network techniques.
In the intrusion detection field, Goldstein and Uchida (Goldstein and Uchida 2016) presented a comparative evaluation of unsupervised algorithms used in the context of Anomaly Detection. The algorithms were applied to a group of different datasets, one of each was the KDD 99, described in Section 3.2, however the analyses only used part of the dataset regarding HTTP traffic. It is important to note that an improved version of this dataset called NSL-KDD is presented and used in this paper.
Aleroud and Karabatis (Aleroud and Karabatis 2013) explored the detection of zero-day attacks, with an approach that combines already existing methods with linear data transformation techniques such as discriminant functions that separated the data in normal patterns from attack patterns, and anomaly detection techniques using the One Class Nearest Neighbor algorithm (1-NN) to identify the zero-day attacks. Their approach consisted in a system of several static components and processes. The first component was the network data repository where they used the NSL-KDD dataset. The second component represented the pre-processing methods applied in the NSL-KDD dataset, where they converted numeric features into bins. The third module, Misuse detection, consisted in identifying attacks that are relevant to a particular context and also identifying normal activities in the network to

reduce the false positives alerts. This module used conditional entropy to create known attacks context profiles using patterns from historical data. Finally, the last component represented the anomaly detection module where it used the 1-NN algorithm to detect deviation from normal activity and also used the Singular Value Decomposition (SVD) technique to reduce the data dimensionality. They showed good performance in their approach, detecting zero-day attacks with a low false positive rate.
Casas and Mazel (Casas, Mazel, and Owezarski 2012) presented the concept of an Unsupervised Network Intrusion Detection System (UNIDS), using Sub-Space Clustering and Multiple Evidence Accumulation techniques for outlier detection. Their unsupervised security system consisted in analyzing packets captured in continuous times slots of fixed length running in three consecutive steps. In the first step it was performed the clustering analysis to detect anomalous time slots. The second step used a multi-clustering algorithm based on a combination of several techniques (Parsons, Haque, and Liu 2004, Ester et al. 1996, Fred and Jain 2005) to rank the degree of abnormality of all the identified outlying flows. The third step used a simple threshold detection technique to flag the top-ranked outlying flows as anomalies. Their evaluation of this system included its application to the KDD 99 dataset.
Noto et al. (Noto, Brodley, and Slonim 2012) studied anomaly detection using an approach called FRaC, feature regression and classification. The FRaC technique built a model of normal data and the distances of its features and used the learnt model to detect when an anomaly occurred. They also compared their approach with other commonly used techniques, such as, Local Outlier Factor (LOF), One class support vector machines (One class SVM) and Cross-feature analysis (CFA).
Our work intends to show and compare the behavior of several one-class classification algorithms (some of them already mentioned in this section) and apply them in two recent intrusion datasets with the purpose of identifying if these techniques could be integrated in an IDS inside the SASSI project.

3 Anomaly Detection Methodology

In our work, we will study the behavior of several unsupervised algorithms based in one class classification, in order to verify if these techniques are a viable solution to discover and detect unknown attacks. In this section, we describe the network anomaly detection methodology, as shown in Fig. *1*. We describe the datasets used and the pre-processing techniques applied to them before feeding the algorithms, as well as the unsupervised techniques employed. In the next section (Section 4) we will explain how the anomaly detection algorithms work.

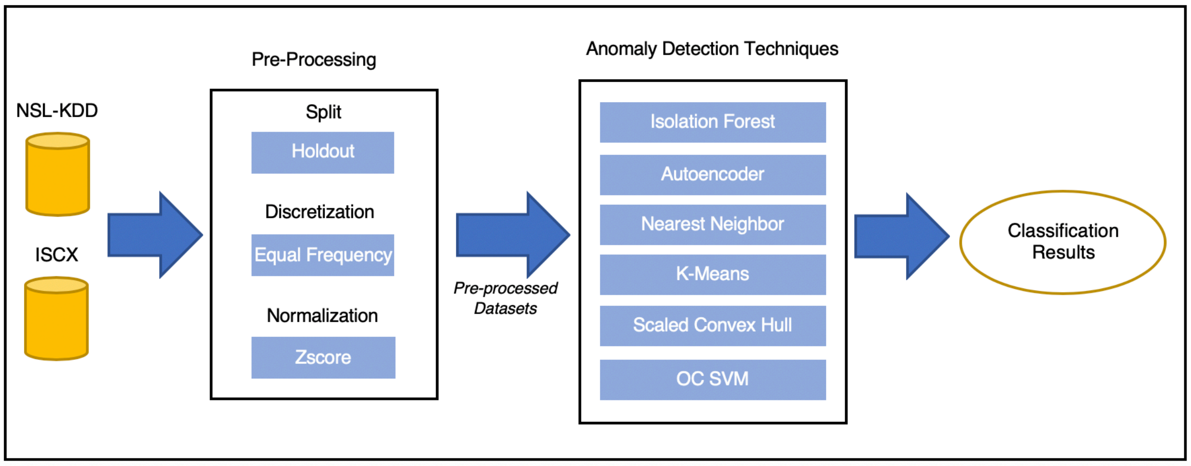

Fig. 1 - Anomaly detection methodology – The datasets were splitted, normalized and discretized through some pre-processing techniques before being applied in the algorithms learning and testing phase.

In our exploration, we analyzed the NSL-KDD (Tavallaee et al. 2009) and the ISCX datasets (Shiravi et al. 2012). These datasets contain samples from normal activity and from simulated attacks in computer systems and are commonly used in the literature. Before using the learning algorithms, we have employed some pre-processing methods to prepare the data.

3.1 NSL-KDD

In 1999, in the third international competition in the conference Knowledge Discovery and Data Mining (KDD), the KDD 99 dataset (UCI Machine Learning Repository 2015)[1] was presented to the scientific community. This dataset is frequently used in the literature of IDS evaluation, and contains simulated network activity samples, corresponding to normal and abnormal activity divided in five categories:
• **Denial of Service (DoS):** An intruder tries to make a service unavailable (contains 9 types of DoS attacks);
• **Remote to Local (R2L)**: An intruder tries to obtain remote access to the victim's machine (contains 15 types of R2L attacks);
• **User to Root (U2R)**: An intruder with physical access to the victim's machine tries to gain super-user privileges (contains 8 types of U2R attacks);
• **Probe**: An intruder tries to get information about a victim's machine (contains 6 types of Probe attacks);
• **Normal**: It constitutes the normal operations or activities in the network.
Tavallaee et al. (Tavallaee et al. 2009) made some improvements on KDD 99 dataset and the result was the NSL-KDD dataset. This dataset is already organized in two subsets: one to train the algorithms, and another one to test them. Each data sample contains 43 features where four of them are nominal type, six are binary and the rest of them are numerical type. As we are testing one class classification algorithms, it was selected a portion of normal data from the training set and a portion of both normal and attack data from the test set, where the attack data contains all four attack categories and represents 10% of the test set.
Some pre-treatment techniques were applied to the dataset before performing the discretization and normalization operations, as shown in Fig. ***1***. Some features were removed namely: 'Wrong_fragment', 'Num_outbound_cmds', 'Is_hot_login', 'Land' and 'level_difficulty' because they have redundant values in at least one of the subsets. In the case of the 'level_difficulty' feature, it represents the level of difficulty of attacks' detection by learning algorithms. This feature was removed because its information is not relevant in a real-world anomaly detection problem. Another pre-treatment operation to the data was the conversion of nominal features to numerical features, since the algorithms to be employed afterwards cannot handle non-numerical data. After performing the cleaning of the subsets, two different pre-processing techniques were applied to the data. First, the data with continuous features was discretized with the equal frequency technique. With this technique, the values of the features were divided in k bins in the way that each bin contains approximately the same number of samples. Thus, each bin has $\frac{n}{k}$ adjacent values. The value of k is a user defined parameter, and to obtain this value we used the heuristic n where n is the number of samples. This discretization technique can provide better accuracy and fast learning in certain anomaly detection algorithms, since the range of values is smaller (H. Liu et al. 2002).
The second pre-processing technique was the data normalization, to have all the features within the same scale. This operation prevents some classification algorithms to give more importance to features with large numeric values. Once the features are all on the same scale, the classifiers assign the same weight to each attribute. The Z-Score and MinMax were the normalization techniques applied to the data. The Z-score technique transforms the input, so the mean is zero and the standard deviation is one. On the other hand, the MinMax transform the original input data to a new specific set where the values range are between 0 to 1. We tested the algorithms with each pre-processing technique and with both combined to evaluate which techniques improve the performance of the algorithms. Then we made 5 experiences with each algorithm with the best pre-processing techniques and calculate all the performance metrics mean to compare their results.

3.2 ISCX

ISCX is a dataset developed by Shiravi et al. (Shiravi et al. 2012) at the Canadian cybersecurity institute. This dataset is based on the concept of profiles that contain detailed descriptions of abstract intrusions and distributions for applications, protocols, services, and low-level network entities. To create this dataset, real network communications were analyzed to create profiles for agents that generate real traffic for HTTP, SMTP, SSH, IMAP, POP3 and FTP protocols. In this regard, a set of guidelines have been established to delineate a valid dataset that establishes the basis for profiling. These guidelines are vital to the effectiveness of the dataset in terms of realism, total capture, integrity, and malicious activity (Shiravi et al. 2012). Each data sample in the ISCX dataset contains 21 attributes. There is a total of 7 days of network traffic captured with 4 different attack types shown in Table 1.

[1] https://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data

Table 1. ISCX captured activity. The attacks were captured along with normal network activity. To distinguish between a normal observation and an abnormal one it is presented in the ISCX dataset an attribute called "label" where value 1 represents an attack and value 0 represents normal activity.

Capturing date	Network activity
11/06/2010	Normal
12/06/2010	Normal
13/06/2010	Normal + Internal infiltration into the network
14/06/2010	Normal + HTTP Denial of Service
15/06/2010	Normal + Distributed Denial of Service using a Botnet IRC
16/06/2010	Normal
17/06/2010	Normal + Brute Force SSH

For this dataset, we did the following changes before applying the pre-processing techniques shown in Fig. ***1*** and described in the NSL-KDD dataset:

- All nominal features where converted to numerical – the algorithms used cannot handle non-numeric features;
- All "Payload" features were removed – These are string features, so it is not possible to train and test the algorithms with these features;
- The source and destination IP address features were removed – There is no interest in training the algorithms with these features, since the IP addresses are constantly changing;
- A new feature was created to represent the time interval of an operation on the network, defined as the difference between the features "stop date time" and "start date time".

3.3 Unsupervised learning algorithms

Unsupervised learning algorithms are suitable for scenarios where the objective is to perform outlier detection to a dataset. Some of these algorithms follow the basic idea of learning from a training dataset that only contains normal samples, and in the classification the output is either "normal" if it resembles the learnt set or "outlier" if it does not. These algorithms are named as one-class classification methods and appear to be good candidates for the problems of discovering unknown attacks, since every attack can be considered an outlier. In this work, we applied a set of 6 different one-class algorithms, namely Autoencoder, Nearest Neighbor, K-Means, Isolation Forest, Support Vector Machines and Scaled Convex Hull, which performance is to be evaluated over the NSL-KDD and ISCX datasets.

3.3.1 Autoencoder

An Autoencoder is a neural network that is part of a sub-area of machine learning called Deep learning (sets of algorithms with several layers of processing which are used to model high-level abstractions of data (Deng, Yu, and others 2014)). Neural Networks are interconnected processing units that are organized by one or more layers which can be used in the implementation of a complex functional mapping between input and output variables (Bishop 1995) . They can perform linear or non-linear transformations through the processing of the units in the different layers (Mazhelis 2016).
The autoencoder, also known as autoassociator, is a kind of neural network that is trained to make the input features the same or very similar to the output features (Japkowicz 1999). In the classification task, the autoencoder can reproduce accurately only the vectors whose structure is similar to the structure learned by the neural network.
As a neural network, the autoencoders are sensitive to outliers since they contribute to the minimization of the error function. The disadvantage of this method is the need of employing a number of parameters that have to be specified by the user (Tax 2001). These parameters consist of selecting a number of hidden layers, a number of hidden units in each layer, the type of transformation function, the learning rate and the stopping rule. In addition to these parameters, it is necessary to estimate a number of weights (usually equal to the number of hidden units and input units) for the training set. A large amount of data is essential for an accurate estimation of weights. The computational resources for this algorithm are considerably high, since the learning process is iterative, being repeated several times throughout the training set, until the stop rule is satisfied.

For the application of this algorithm, the H2O[2] package was used. This package is an open source mathematical engine for big data processing machine learning algorithms, such as generalized linear models, gradient boosting, Random Forests and neural networks (deep learning) in several cluster environments (Stadler 2011).
After loading the datasets (NSL-KDD and ISCX) already pre-processed to this engine, the *h2o.deeplearning* function was used to train the autoencoder algorithm. Regarding the parameters to be used, we tested several, but we find that Glander's[3] approach, applying a bottleneck architecture used to fraud detection in credit card transactions, presented better results. When tuning the autoencoder, we used a separated validation set, and we found that using a bottleneck architecture, where the number of neurons in the middle layer are lower than the first and last layers, it presented better classification results. The Area Under the Curve metric was employed to compare the performance of the algorithm with different hyperparameters values. The hyperparameters values used were:

- Hidden c (50,5,50) - defines the number of hidden layers and units of the neural network, in this case the vector c (50, 5, 50) contains 3 values and each value corresponds to number of neurons per layer;
- Activation: Than - we define the activation function hyperbolic tangent;
- Epochs = 20 - Specify the number of times to iterate the dataset.

We used the *h2o.anomaly* function after the model finalized its training process. This function is intended to detect anomalies in a dataset. The function reconstructs the original dataset using the training model and calculates the MSE for each point in the test set.
Then we created a graphic that represents the reconstruction of the mean square error as shown in Fig. ***2***. This graphic represents an example of a test made on a test sample of the ISCX dataset. It turns out that at a certain point the MSE increases. This means that the model could not correctly identify these records, which could be considered an anomaly. So, we drawn a threshold, in this case equal to 0.002, where all records above this threshold are treated as anomalies.

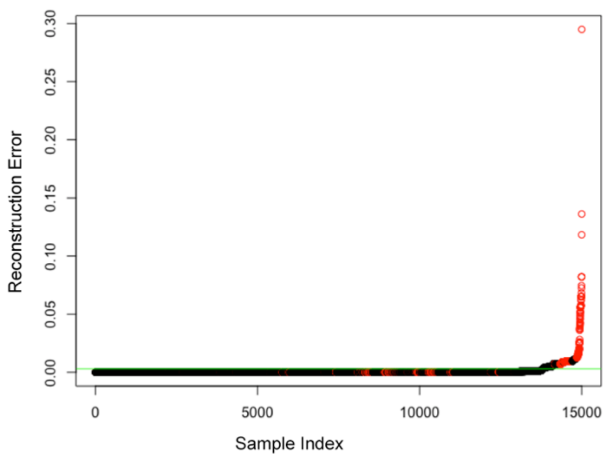

Fig. 2 - Reconstruction of the mean square error.

3.3.2 One-class k Nearest Neighbor

The One-Class Nearest Neighbor (OCNN) is an adaptation of the original K-Nearest Neighbor supervised algorithm using distances between neighbors to classify the data. In the training phase the OCNN algorithm starts by memorizing the observations. The observations are objects where each one of them represents a point in space defined by the features. In the training process all observations are only from the class that represents normal activity in the network. After memorizing all the objects, in the test phase the algorithm will use a metric to calculate distances between points. As distance metric, the Euclidean distances given by the equation 1 were used, where x_i' and x_j' are two objects represented by vectors in space $\mathbb{R}d$.

[2] https://cran.r-project.org/web/packages/h2o/index.html

$$d(X_i, X_j) = \sqrt{\sum_{i=1}^{d} (x_i' - x_j^i)^2} \tag{1}$$

In this phase, the distances between the test object and its first nearest neighbor (if k is equal to 1) from the training set are calculated. Then the distance values are used to identify a maximum distance and define a threshold. If that distance is higher than the maximum distance threshold, then the sample in question is classified as an outlier. For anomaly detection in the NSL-KDD and ICSCX datasets, and for the OCNN algorithm the value of k was chosen to be equal to 1 because after testing with different values, this value obtained the highest performance in the data classification.

3.3.3 One-Class K-Means

The K-Means is a clustering algorithm, which is the process of partitioning a set of data (or objects) into smaller subclasses with common characteristics. The number of clusters is defined initially and remains fixed throughout the process. The goal of this algorithm is to find different groups in data defined by the variable k. Based on the data features, this algorithm works iteratively to assign each observation or object to one of the k groups. The algorithms start by estimating the k centroids that are the centers of each cluster. In this step, each object is assigned to its nearest centroid, based on the squared Euclidean distance. As shown in the equation 2, being c_i the collection of centroids in set C, each object x is assigned to a cluster based on where $dist(\cdot)$ is the standard Euclidean distance.

$$\arg min_{c_i \in C}\ dist(c_i, x)^2 \tag{2}$$

Then subsequently the centroids are recomputed. This process is done by taking the mean of all objects assigned to that centroid's cluster. In the equation 3 the set of data point assignments for each *i*th cluster centroid is S_i.

$$c_i = \frac{1}{|S_i|} \sum\nolimits_{x_i \in s_i} x_i \tag{3}$$

To apply this algorithm as one-class classification, the building process of clusters should only use normal data examples. In the classification process, the algorithm calculates all the test data points distance to the closest cluster. Then a threshold is defined and if the calculated distance to each object is higher than the threshold value, the sample is classified as an anomaly. It was used the silhouette analysis that measures how close each point in one cluster is to points in the neighboring clusters. This measure gives us information about the best parameter (number of clusters) to apply. In both the NSL-KDD and ISCX datasets the ideal number of clusters was set to 4.

3.3.4 Isolation Forest

Isolation Forest (F. T. Liu, Ting, and Zhou 2012) is a method for outlier detection that uses data structures called trees, such as binary trees. Each tree is created by partitioning the instances recursively, by randomly selecting an attribute and a split value between the maximum and minimum values of the selected attribute (F. T. Liu, Ting, and Zhou 2012). Being T an external node of a tree with no child or an internal-node designated by test with exactly two daughter nodes (T_l, T_r). A test is an attribute q with a split value p, where $q < p$ meaning the data points will be divided into T_l, T_r.
To build an insolation tree, the data $X = \{x_1, \dots, x_n\}$ will be recursively divided by randomly selecting an attribute q and a split value p, until it reaches three conditions (F. T. Liu, Ting, and Zhou 2012):

- The tree reaches a height limit;
- $|X| = 1$;
- All data in X have the same values;

To detect anomalies, the observations are sorted according to their path lengths or anomaly scores. The path length $h(x)$ represents the number of edges x that go through an isolation tree from the root node until the traversal is terminated at an external node. To calculate the anomaly score, Lui et al. (F. T. Liu, Ting, and Zhou 2012) used the analysis from Binary Search Tree (BST) to estimate the average path length of an isolation tree represented in equation 4:

$$c(n) = 2H(n-1) - (\frac{2(n-1)}{n}) \tag{4}$$

The observations from the dataset are represented by n and $H(i)$ is a harmonic number and can be estimated by the Euler's constant. The parameter $c(n)$ was used to normalize $h(x)$ since it represents the average of given n. The equation 5 represents the anomaly score s of an observation x :

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{5}$$

$E(h(x))$ is the average of $h(x)$ from a collection of isolation trees. Using the anomaly score Lui et al. (F. T. Liu, Ting, and Zhou 2012) verified that observations with a s value much smaller than 0.5 are quite safe to be regarded as normal observations, while observations with a s value very close to 1 are definitely anomalies, and observations that return $s \approx 0.5$ don't really mean any distinct anomaly.
In our tests, we used the default algorithm parameter of 100 trees in both datasets, since experimentally the variation of this parameter did not show any substantial impact in the performance.

3.3.5 One-Class Scaled Convex Hull

The Scaled Convex Hull (SCH) is an algorithm based on a previously proposed method by Casale et al. (Casale, Pujol, and Radeva 2011) that uses the geometrical structure of the Convex Hull (CH) to define the class in one-class classification problems. This algorithm uses random projections and an ensemble of CH models in two dimensions, and thus this method can be suitable for larger dimensions in an acceptable execution time (Fernández-Francos, Fontenla-Romero, and Alonso-Betanzos 2017). As we can see in equation 6.

$$CH(S) = \left\{ \sum_{i=1}^{|S|} \theta_i x_i \;\middle|\; (\forall i{:}\, \theta \geq 0) \wedge \sum_{i=1}^{|S|} \theta_i = 1, x_i \in S \right\} \tag{6}$$

the CH of a finite set of points $S \in R^d$ provides a tight approximation among several convex forms being this approximation prone to over-fitting. Fernández-Francos et al. (Fernández-Francos, Fontenla-Romero, and Alonso-Betanzos 2017) used reduced/enlarged versions of the original CH to avoid the over-fitting problem, where in the training phase an outlier can lead to shapes that do not represent the target class accurately. To resolve this problem they applied the formula presented by Liu et al. (Jaynes and Cummings 2015) to calculate the expanded polytope, where the vertices are defined with respect to the center point $c = \left(\frac{1}{|S|}\right) \sum_i x_i,\ \forall x_i \in S$ and the expansion parameter $\lambda \in [0, +\infty)$ as in equation 7:

$$v^\lambda : \{\lambda v + (1-\lambda)c | v \in CH(S)\} \tag{7}$$

The parameter λ represents a constant extension ($\lambda > 1$) or constant contraction ($0 < \lambda < 1$) of the CH regarding c.
An approximation of the decision made by the expanded CH in the original d-dimensional space by means of an ensemble of τ randomly projected decisions on 2-D spaces was proposed. In this way, the authors defined a decision rule that states that a point does not belong to the modeled class if and only if there exists at least one projection in which the point lies outside the projected CH.
To have a better understanding of this method, in Fig. ***3*** is graphically represented an example where a 3-D convex figure is approximated by three random projections in 2-D. We can observe in Fig. ***3*** that the point can be inside one or more projections but in fact that point lies outside the original geometric form.

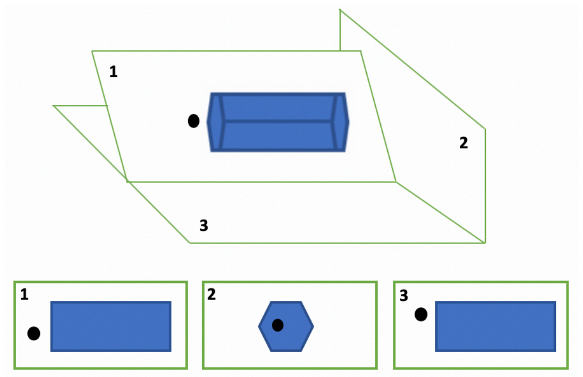

Fig. 3 - Ensemble of projected decisions on 2-D based on Fernández-Francos et al. (Fernández-Francos, Fontenla-Romero, and Alonso-Betanzos 2017)

In the SCH algorithm Fernández-Francos et al. (Fernández-Francos, Fontenla-Romero, and Alonso-Betanzos 2017) proposed three different definitions of center:

1. The average of all points in the projected space;
2. The average of the CH vertices in the projected space;
3. The average position of all the points in the projected polytope.

Each type of center leads to different decision regions (if a point belongs or not to the target class), giving more flexibility to this method.
In our experiment we found that the best parameters for this algorithm were:

- A value of $\lambda = 1{,}22$ in the NSL-KDD and a $\lambda = 1{,}11$ in the ISCX dataset;
- Around 2000 projections;
- A center type that uses the average of the CH vertices in the projected space.

3.3.6 One-Class Support Vector Machines

Support Vector Machines (SVM) have the capability to solve classification and regression problems. This algorithm focuses on the search for a hyperplane (generalization of a plane in different dimensions, for example in a two-dimensional plane is a line that separates and classifies data) that better divides a dataset into two classes. SVMs are effective in classifying linearly separable data or having an approximately linear distribution. However, there are many cases where it is not possible to properly divide the training data using a hyperplane. To solve this problem, SVMs can create a non-linear decision boundary by projecting a non-linear function ϕ to a space with a higher dimension. This means that SVMs can project the data from the training set of its original space I to a new space of greater dimension, denominated as feature space F (Hearst et al. 1998).
To calculate the scalar products between objects mapped in the new space, functions called Kernels $K(x_i, x_i) = \phi(x)^T\phi(x_i)$ are used. The usefulness of kernels lies therefore in the simplicity of their calculation and in their capacity to represent abstract spaces. The most used kernels are the polynomials, the radial base function and the sigmoidal. Each of them has parameters that must be determined by the user (Gama et al. 2015).
The hyperplane equation is represented by $w^T x + b = 0$, with $w \in F$ and $b \in R$ in two-dimensional space. The constructed hyperplane as mentioned, determines the margin between the classes. The use of slack variables ξ_i will allow some data points to lie within the margin where the constant $C > 0$ determines the trade-off between maximizing the margin and the number of training data points within the margin (Vlasveld 2013). This way they will prevent SVM from over-fitting with noisy data. The objective function of the SVM classifier is represented in the equation 8:

$$\min_{w,b,\xi_i} \frac{\|w\|^2}{2} + C\sum_{i=1}^{n} \xi_i \tag{8}$$

Subject to:

$$y_i(w^T\phi(x_i) + b) \geq 1 - \xi_i \text{ for all } i = 1, \dots, n$$
$$\xi_i \geq 0 \text{ for all } i = 1, \dots, n$$

Therefore, for anomaly detection problems the One-Class Support Vector Machines (OCSVM or ν-SVM) will only train with data from one class, in this case, the class that represents normal activity in the network. Basically, it separates all the data points from the origin and maximizes the distance from the hyperplane to the origin. This results in a binary function that captures regions in the input space where the probability density of the data lives. The minimization function is given by the equation 9 (Schölkopf et al. 2000):

$$\min_{w,\xi_i,\rho} \frac{1}{2}\|w\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n} \xi_i - \rho \tag{9}$$

Subject to:

$$\left(w \cdot \phi(x_i)\right) \geq \rho - \xi_i \text{ for all } i = 1, \dots, n$$
$$\xi_i \geq 0 \text{ for all } i = 1, \dots, n$$

As we can see in equation 9 the parameter ν characterizes the solution, instead of the C parameter that decided the smoothness in equation 8. The parameter ν sets an upper bound on the fraction outliers and a lower bound on the number of training examples used as support vectors. Due to the importance of this parameter, this approach is also known as ν-SVM.
The tests performed allowed to obtain the best following parameters in anomaly detection for the NSL-KDD and ISCX datasets:

- The radial base kernel function was used;
- The $\gamma = 0.3$ in the NSL-KDD and $\gamma = 4.2$ in the ISCX (parameter used for the radial basis kernel);
- The $\nu = 0.01$ in the NSL-KDD and $\nu = 0.005$ in the ISCX.

4 Performance Evaluation

All combinations of the pre-processing techniques with the unsupervised learning algorithms were tested and we present the results of the best techniques applied to each algorithm for NSL-KDD and ISCX datasets in table 2. To evaluate the performance of the classifiers we used several metrics as described below:

Area Under the Curve

A well-known way of comparing the classifiers performance is using the Area Under the Curve (AUC) metric. The AUC calculates the area under a Receiver Operating Characteristic curve (ROC) which is a graph showing the classification performance at various thresholds settings drawn by two parameters, the True Positive Rate (TPR) $\frac{True\ Positives(TP)}{TP+False\ Negatives(FN)}$ and the False Positive Rate (FPR) $\frac{False\ Positives\ (FP)}{FP+True\ Negatives(TN)}$. Each point on the ROC represents a TPR/FPR pair corresponding to a particular decision threshold.

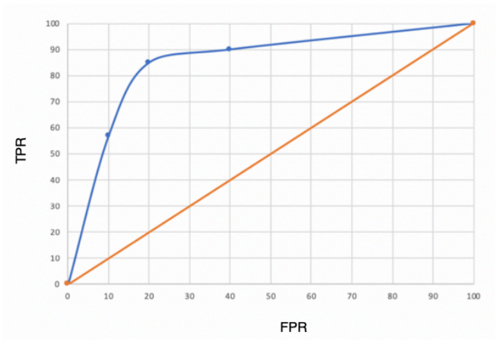

Fig. 4 - ROC curve example

Fig. ***4*** represents an example of a ROC curve, were we can see the trade-off between TPR and FPR. Those classification algorithms that have a curve close to the top-left corner indicates a better performance (as seen in

the blue line). As close the curve is to the diagonal line (marked in orange) where TPR = FPR, the less accurate the classifier is.
The AUC is a metric that can be useful to summarize the performance of each classifier providing an aggregate measure of performance across all possible classification thresholds. AUC can be seen as the probability of the model in distinguishing between positive class (anomaly) and negative class (normal activity).

Recall, Precision, F1 Score

In classification problems, metrics such as recall, precision and F1 score are mostly used to understand the amount of miss predicted observations in a specific class. Each of these metrics give us information about the different types of errors generated by the classifier. With *Recall,* also known as TPR, and *sensitivity*, presented in the AUC section, we can learn the percentage of instances from the anomaly class that are actually predicted correctly. This means that the higher the *Recall*, the smaller the false negatives (type 2 error). On the other hand, *Precision* is very similar to *Recall*, but instead of computing the false negatives it computes the false positives ($\frac{TP}{TP+FP}$). This represents the portion of anomaly class elements that were correctly predicted. So, as in *Recall,* we can say that, the higher the *Precision*, the smaller the false positives (type 1 error). When using these two metrics, there is often a tradeoff between them, so it is important to evaluate them together using another metric called F1 score, showed in equation 10:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{10}$$

As showed in equation 10*, F1 score* represents the harmonic mean which combines *Recall* and *Precision* metrics with an equal weight.

Table 2 - Comparative results using mean AUC (×100) for each algorithm using the best combination of pre-processing techniques, in NSL-KDD and ISCX datasets

Best Pre-Processing Techniques	**One-Class Algorithms**	**NSL-KDD (AUC)**	**ISCX (AUC)**
No pre-processing	Isolation Forest	81,71	90.70
Zscore	K-Means	84,76	77,06
Zscore	1-Nearest Neighbor	84,85	**95,20**
Equal Frequency + MinMax	Autoencoder	83,65	80,44
Equal Frequency + MinMax	Scaled Convex Hull	**85,30**	85,95
Equal Frequency + MinMax	Support Vector Machines	83,14	91,63

As we can see on Table 2, the algorithms One-class K-means and 1-Nearest Neighbor had the best performance applying the Z-Score techniques. The Isolation Forest algorithm had the best results without any kind of data transformation as it uses binary trees in the process of data recursive partitioning. In the case of the Autoencoder, SCH and ν-SVM had the best performances in detecting anomalies applying MinMax and Equal Frequency (EF) techniques in the pre-processing phase.
Looking at the NSL-KDD results, the SCH classifier had the best performance with an AUC value around 85. The other algorithms obtained very close results ranging between 81 and 84 AUC, where the 1-Nearest Neighbor was the second-best classifier with an AUC close to 85. Regarding the ISCX dataset, analyzing the table, we can observe that the 1-Nearest Neighbor algorithm obtained the highest AUC result, followed by ν-SVM. In this dataset the AUC results were higher compared to the NSL-KDD. One of the reasons for this is the fact that the NSL-KDD has 38 different types of attacks compared to the ISCX with only 4 different types.
To verify if there is significant difference between the classifiers performance in both datasets we applied the Nemenyi post-hoc statistical test and presented a critical difference diagram (Demšar 2006) as shown in Fig. ***5***. As we can see all the algorithms are connected to each other (thickest horizontal line underneath the critical difference scale), meaning that they are not significantly different (at level $\alpha = 0.10$).

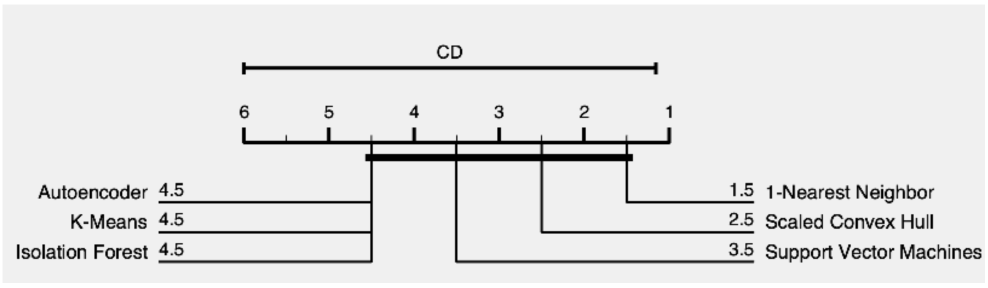

Fig. 5 – Critical difference diagram, Nemenyi post-hoc test

The non-significant difference between algorithms can be explained because we only used two datasets to test the classifiers due to the lack of good datasets in the cybersecurity field. Even though the classifiers are not significant different to each other, we can see that on average Nearest Neighbor, SCH and ν-SVM have a high score compared to the other three algorithms.
Since the test set has unbalanced classes, we plotted the performance of the algorithms using other metrics that can measure the errors more in detail. This metrics are: Recall, Precision and F1 score.

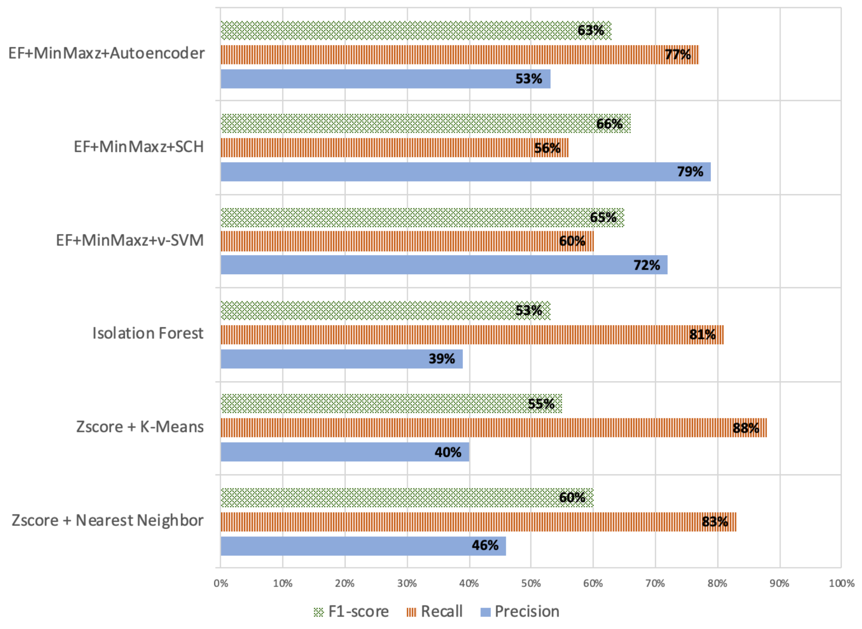

Fig. 6 - Anomaly detection results in NSL-KDD

Starting by the NSL-KDD dataset, observing the Fig. ***6***, looking at the F1 score metric as it represents the harmonic mean combining the two other metrics, we can see that all algorithms showed similar results. The isolation Forest and K-Means with 53% and 55% respectively and the others ranging between 60% to 66%, being SCH the algorithm with the highest F1 score. We can look also at precision and recall metrics as to have a better perception of the false positives and false negatives costs. Few false negatives represent a higher value of recall and vice-versa, and we can also say the same regarding precision with respect to the false positives. Observing the graphic in Fig. ***6***, all algorithms except SCH and ν-SVM had a recall value much higher than precision, so the false positives were much higher than the false negatives in these cases. In cybersecurity it is important to have a low false negative rate, since it represents the worst-case scenario, where data is predicted as normal activity, while in fact it represents malicious or abnormal activity. Regarding the SCH and ν-SVM they both had the highest F1 score compared to the other anomaly detection techniques but at the same time they had more misclassified observations that represent false negatives than misclassified observations representing false positives.

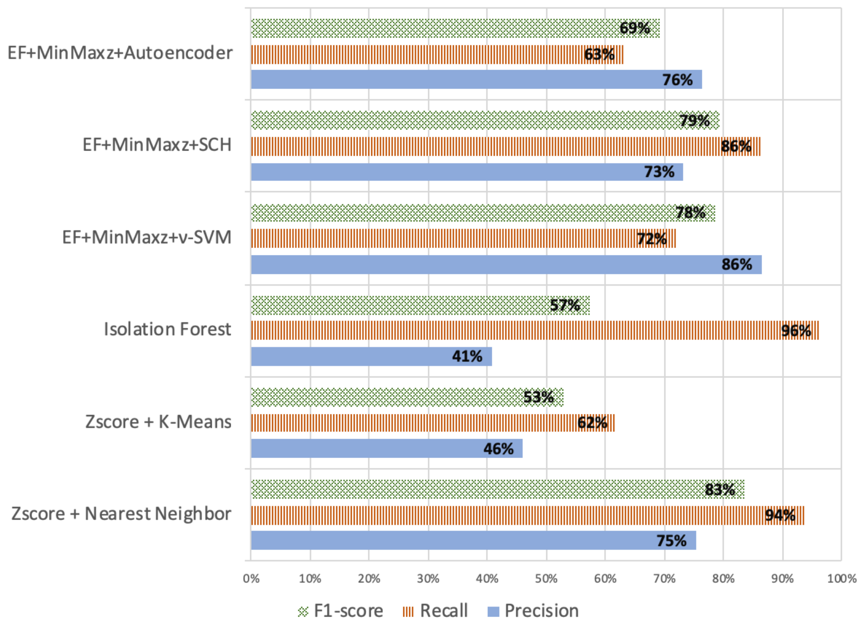

Fig. 7 - Anomaly detection results in ISCX

Analyzing Fig. *7*, concerning the ISCX dataset, we observe that Nearest Neighbor, SCH and ν-SVM have much better performance results than those obtained for the NSL-KDD. On the other hand, the Isolation Forest and K-means algorithms remained with approximately the same results as in NSL-KDD. Another fact that can be observed is that the algorithm SCH generates fewer false negatives and increases the false positives when trying to detect the four different types of attacks contained in ISCX dataset.

5 Conclusion and Future Work

Threats in information systems have become increasingly intelligent and they can deceive the basic security solutions such as firewalls and antivirus. Anomaly-based IDSs allow monitored network traffic classification or computer system calls classification in normal activity or malicious activity. The efficiency of intrusion detection depends on the techniques used in these systems. As mentioned, the work carried out in this paper was motivated by the SASSI project. The goal was to verify if any of the unsupervised techniques presented in this paper could be implemented in an IDS to support Systems administrators in decision making process of anomaly and novelty detection task. We can conclude that all algorithms could detect most of the anomalies and also showed that they have managed to separate adequately the data between classes even though they were unbalanced (to represent a more realistic environment). To choose the best method, we need to focus not only on the overall performance but also on the type of errors generated. Analyzing the performance metrics, we conclude that the 1-Nearest Neighbor, SCH and ν-SVM presented the highest results in both datasets but the SCH and ν-SVM generated more false negatives than false positives errors in the NSL-KDD dataset. Being this type of error an undesirable scenario in cybersecurity, we suggest the implementation of the 1-Nearest Neighbor since it is capable of detecting most of the anomalies and moreover it was also one of the fastest unsupervised techniques in the computing process of anomaly detection. Although unsupervised learning methods are great to generalize, detect unknown patterns and also handle the unlabeled data problem, they have also some constraints. These methods can't be too specific about the definition of the data, leading to less accuracy (generating a high number of false positives for this specific problem) compared to supervised techniques presented in the literature (Niyaz et al. 2015, Dhanabal and Shantharajah 2015,Tsai et al. 2009). Therefore, as future work, other architectures will be studied with the aim of optimizing the performance of the predictive model, like the development of a hybrid model containing unsupervised and supervised techniques to reduce the false positive rate and classify the attacks by type. Then the predictive model will be tested in a real dataset developed by the SASSI sensors[3]. After creating a consistent predictive model, the aim will be the study of action rules to aid the system administrator in the prevention of the detected attacks.

[3] Monitoring system that collects data from network communications in real time through network sensors.

Acknowledgements.
This work was supported by SASSI Project (ANI|P2020 17775) and has received funding from FEDER Funds through P2020 program and from National Funds through FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) under the project UID/EEA/00760/2019. This work has also received financial support from MINECO (grant TIN2015-65069), the Xunta de Galicia (grants ED431C 2018/34, and Centro Singular de Investigación de Galicia, accreditation 2016-2019, Ref. ED431G/01) and the European Union (European Regional Development Fund - ERDF).

References

Aleroud, Ahmed, and George Karabatis. 2013. "Toward Zero-Day Attack Identification Using Linear Data Transformation Techniques." *Proceedings - 7th International Conference on Software Security and Reliability, SERE 2013*, no. May: 159–68. https://doi.org/10.1109/SERE.2013.16.

Bishop, Christopher M. 1995. *Neural Networks for Pattern Recognition*. *Oxford University Press*. Oxford university press. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.679.1104&rep=rep1&type=pdf.

Casale, Pierluigi, Oriol Pujol, and Petia Radeva. 2011. "Approximate Convex Hulls Family for One-Class Classification." In *International Workshop on Multiple Classifier Systems*, 106–15.

Casas, Pedro, Johan Mazel, and Philippe Owezarski. 2012. "Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge." *Computer Communications* 35 (7): 772–83. https://doi.org/10.1016/j.comcom.2012.01.016.

Castillo, Enrique, Diego Peteiro-Barral, Bertha Guijarro Berdiñas, and Oscar Fontenla-Romero. 2015. "Distributed One-Class Support Vector Machine." *International Journal of Neural Systems* 25 (07): 1550029. https://doi.org/10.1142/S012906571550029X.

Chen, Jinghui, Saket Sathe, Charu Aggarwal, and Deepak Turaga. 2017. "Outlier Detection with Autoencoder Ensembles." In *Proceedings of the 2017 SIAM International Conference on Data Mining*, 90–98. https://doi.org/10.1137/1.9781611974973.11.

Demšar, Janez. 2006. "Statistical Comparisons of Classifiers over Multiple Data Sets." *Journal of Machine Learning Research* 7 (Jan): 1–30.

Deng, Li, Dong Yu, and others. 2014. "Deep Learning: Methods and Applications." *Foundations and Trends®in Signal Processing* 7 (3--4): 197–387.

Dhanabal, L, and S P Shantharajah. 2015. "A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms." *International Journal of Advanced Research in Computer and Communication Engineering* 4. https://doi.org/10.17148/IJARCCE.2015.4696.

Ester, Martin, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, and others. 1996. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." In *Kdd*, 96:226–31.

Fernández-Francos, Diego, Óscar Fontenla-Romero, and Amparo Alonso-Betanzos. 2017. "One-Class Convex Hull-Based Algorithm for Classification in Distributed Environments." *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

Fred, Ana L N, and Anil K Jain. 2005. "Combining Multiple Clusterings Using Evidence Accumulation." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27 (6): 835–50.

Gama, João, André Ponce de Leon Carvalho, Katti Faceli, Ana Carolina Lorena, and Márcia Oliveira. 2015. *Extração de Conhecimento de Dados*.

Gardner, Andrew B, Abba M Krieger, George Vachtsevanos, and Brian Litt. 2006. "One-Class Novelty Detection for Seizure Analysis from Intracranial EEG." *Journal of Machine Learning Research* 7 (Jun): 1025–44.

Giacinto, Giorgio, Roberto Perdisci, Mauro Del Rio, and Fabio Roli. 2008. "Intrusion Detection in Computer Networks by a Modular Ensemble of One-Class Classifiers." *Information Fusion* 9 (1): 69–82.

Goldstein, Markus, and Seiichi Uchida. 2016. "A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data." *PLoS ONE*, no. April: 1–31. https://doi.org/10.7910/DVN/OPQMVF.

Hearst, Marti A, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. 1998. "Support Vector Machines." *IEEE Intelligent Systems and Their Applications* 13 (4): 18–28.

Japkowicz, Nathalie. 1999. *Concept-Learning in the Absence of Counter-Examples: An Autoassociation-Based Approach to Classification*. Rutgers University.

Jaynes, E.T., and F.W. Cummings. 2015. "A Novel Geometric Approach to Binary Classification Base." *Proceedings of the IEEE*. https://doi.org/10.1109/SSP.2012.6319793.

Khan, Shehroz S, and Michael G Madden. 2014. “One-Class Classification: Taxonomy of Study and Review of Techniques.” *The Knowledge Engineering Review* 29 (3): 345–74.

Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou. 2012. “Isolation-Based Anomaly Detection.” *ACM Trans. Knowl. Discov. Data* 6 (1): 3:1--3:39. https://doi.org/10.1145/2133360.2133363.

Liu, Huan, Farhad Hussain, Chew L I M Tan, and Manoranjan Dash. 2002. “Discretization : An Enabling Technique,” 393–423. https://pdfs.semanticscholar.org/2d18/73800b294a104a836168ac5bba11edeadc7f.pdf.

Manevitz, Larry M, and Malik Yousef. 2001. “One-Class SVMs for Document Classification.” *Journal of Machine Learning Research* 2 (Dec): 139–54.

Mazhelis, Oleksiy. 2016. “One-Class Classifiers : A Review and Analysis of Suitability in the Context of Mobile-Masquerader Detection Oleksiy Mazhelis To Cite This Version : HAL Id : Hal-01262354 One-Class Classifiers : A Review and Analysis of Suitability in the Context of Mobile.”

Niyaz, Quamar, Weiqing Sun, Ahmad Y Javaid, and Mansoor Alam. 2015. “A Deep Learning Approach for Network Intrusion Detection System.” *Proceedings of the 9th EAI International Conference on Bio-Inspired Information and Communications Technologies*. https://doi.org/10.4108/eai.3-12-2015.2262516.

Noto, Keith, Carla Brodley, and Donna Slonim. 2012. “FRaC: A Feature-Modeling Approach for Semi-Supervised and Unsupervised Anomaly Detection.” *Data Mining and Knowledge Discovery* 25 (1): 109–33. https://doi.org/10.1007/s10618-011-0234-x.

Parsons, Lance, Ehtesham Haque, and Huan Liu. 2004. “Subspace Clustering for High Dimensional Data: A Review.” *Acm Sigkdd Explorations Newsletter* 6 (1): 90–105.

Schölkopf, Bernhard, Robert Williamson, Alex Smola, John Shawe-Taylor, and John Platt. 2000. “Support Vector Method for Novelty Detection.” *Advances In Neural Information Processing Systems 12* 12: 582–88. https://doi.org/10.1.1.71.4642.

Shin, Hyun Joon, Dong-Hwan Eom, and Sung-Shick Kim. 2005. “One-Class Support Vector Machines—an Application in Machine Fault Detection and Classification.” *Computers & Industrial Engineering* 48 (2): 395–408.

Shiravi, Ali, Hadi Shiravi, Mahbod Tavallaee, and Ali A. Ghorbani. 2012. “Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection.” *Computers and Security* 31 (3): 357–74. https://doi.org/10.1016/j.cose.2011.12.012.

Stadler, T. 2011. “R Topics Documented.” *Package ‚ÄòTreePar‚Äô*, 2. https://doi.org/10.2307/2533043>.

Tavallaee, Mahbod, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. 2009. “A Detailed Analysis of the KDD CUP 99 Data Set.” *IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009*, no. Cisda: 1–6. https://doi.org/10.1109/CISDA.2009.5356528.

Tax, David Martinus Johannes. 2001. “One-Class Classification: Concept Learning in the Absence of Counter-Examples.” http://homepage.tudelft.nl/n9d04/thesis.pdf.

Tieleman, Tijmen, and Geoffrey Hinton. 2012. “Lecture 6.5-Rmsprop: Divide the Gradient by a Running Average of Its Recent Magnitude.” *COURSERA: Neural Networks for Machine Learning* 4 (2): 26–31.

Tsai, Chih Fong, Yu Feng Hsu, Chia Ying Lin, and Wei Yang Lin. 2009. “Intrusion Detection by Machine Learning: A Review.” *Expert Systems with Applications* 36 (10): 11994–0. https://doi.org/10.1016/j.eswa.2009.05.029.

Vlasveld, Roemer. 2013. “Introduction to One-Class Support Vector Machines.” Introduction to One-Class Support Vector Machines. 2013.