Neural Computing and Applications (2020) 32:3475–3501
https://doi.org/10.1007/s00521-019-04396-2

ORIGINAL ARTICLE

NSNAD: negative selection-based network anomaly detection approach with relevant feature subset

Naila Belhadj aissa[1] · Mohamed Guerroumi[1] · Abdelouahid Derhab[2]

Received: 19 January 2018 / Accepted: 18 July 2019 / Published online: 6 August 2019

Abstract
Intrusion detection systems are one of the security tools widely deployed in network architectures in order to monitor, detect and eventually respond to any suspicious activity in the network. However, the constantly growing complexity of networks and the virulence of new attacks require more adaptive approaches for optimal responses. In this work, we propose a semi-supervised approach for network anomaly detection inspired from the biological negative selection process. Based on a reduced dataset with a filter/ranking feature selection technique, our algorithm, namely negative selection for network anomaly detection (NSNAD), generates a set of detectors and uses them to classify events as anomaly. Otherwise, they are matched against an Artificial Human Leukocyte Antigen in order to be classified as normal. The accuracy and the computational time of NSNAD are tested under three intrusion detection datasets: NSL-KDD, Kyoto2006+ and UNSW-NB15. We compare the performance of NSNAD against a fully supervised algorithm (*Naïve Bayes*), an unsupervised clustering algorithm (*K-means*) and a semi-supervised algorithm (*One-class SVM*) with respect to multiple accuracy metrics. We also compare the time incurred by each algorithm in training and classification stages.

Keywords Intrusion detection system (IDS) · Anomaly detection · Feature selection · Artificial immune system (AIS) · Negative selection · NSL-KDD dataset · Kyoto2006+ dataset · UNSW-NB15 dataset

1 Introduction

In recent years, security threats, attacks and intrusions in network infrastructures have become one of the major causes of great losses and massive sensitive data leaks. A countless number of mechanisms are used to minimize, detect and counter these security issues.

Anomaly detection is one of the techniques proposed to ensure the integrity and the confidentiality of data. In general, anomaly/outlier detection can be seen as a normal/anomaly classification problem [32]. Several modern techniques exist in the literature addressing this issue using neural network [78], Bayesian network [17], clique clustering approaches [10, 38, 67], bandit clustering [26, 29, 35, 52, 55], support vector machine (SVM) [65], fuzzy logic [82], graph theory [37, 54, 69], decision tree [56], genetic programming [21, 34], artificial immune systems (AIS) [11, 16, 79, 80] and more.

The biological immune system (BIS) has several properties such as robustness, error tolerance, decentralization, recognition of foreigners, adaptive learning and memory which makes it a very complex and promising source of inspiration for several domains. Artificial immune system, which is the field that tries to mimic the complex mechanisms of the BIS, is the focus of much research since the early 1990s [22] to tackle with complex engineering problems. Theories and algorithms were proposed and exploited for pattern recognition, data mining, optimization, machine learning and anomaly detection to name only a few [3].

✉ Naila Belhadj aissa
nbelhadj@usthb.dz

Mohamed Guerroumi
guerroumi@gmail.com

Abdelouahid Derhab
d.ouahid@gmail.com

1 Faculty of Electronic and Computer Science, University of Sciences and Technology Houari Boumediene, Algiers, Algeria

2 Center of Excellence in Information Assurance (CoEIA), King Saud University, Riyadh, Kingdom of Saudi Arabia

Indeed, anomaly detection approach [5, 8] in network security relies on building normal models or profiles and discovering variation/deviation from the model in the observed data. This process is strongly similar to the main objective of the biological immune system. Several models were proposed that imitate BIS mechanisms such as clonal selection, negative and positive selections and immune cells network [95].

In this paper, we propose a negative selection algorithm, namely *Negative Selection for Network Anomaly Detection (NSNAD)* which includes the following outlined contributions:

(a) We propose a filter/ranking-based feature selection using the Coefficient of Variation. The advantage of this statistical metric is twofold: First, it is independent from the class, and second, it can be measured regardless of the attribute's scale and unit.

(b) Most of the previous works dealt with nominal attributes by coding them with iterative integers or binarizing them. The first approach depends on the categories' order of the nominal feature, which means that different orderings will yield to different numerical values and thus a biased classification. The second requires more computational time and memory resources as each value will be represented by an additional binary attribute. In our work, we replace the nominal attributes by their occurrence probabilities when statistical operations are carried out (feature selection phase). Otherwise, we handle them as strings.

(c) We noticed that traditional negative selection implementations, usually, generate detectors as random binary sequences with an R-chunk (*r* consecutive bytes) matching against binary strings representing the self. We consider, in our work, both the real and string representations of all attributes in the dataset. We randomly choose instances from the unlabeled training data to be detectors, and we validate them against self-data in each dimension. Unlabeled training data contain both normal and anomaly records.

(d) Our detector radius is border-based, which means that every detector has its own range of activation and it corresponds to the distance between the detector and the border instances in self-data.

(e) Furthermore, in order to identify new attacks, we optimize the detection phase with an additional verification against self-space. Inspired from the biological Human Leukocyte Antigen (HLA), we define an Artificial HLA as the volume of self-space and ensure that the incoming instance is not actually a "self" before classifying it into anomaly.

(f) Finally, we evaluate our approach not only under NSL-KDD dataset, which has been for the last decades the most used benchmark for the test and evaluation of IDS, but also using two more up-to-date datasets: Kyoto2006+ and UNSW-NB15.

The remainder of this paper is organized as follows: Section 2 presents some background and related work regarding biological and artificial immune systems as well as feature selection techniques. Section 3 details each phase of our proposed algorithm. Section 4 presents an experimental design. The results, analysis and discussion are provided in Sect. 5. An extensive comparison between AIS-based intrusion detection techniques is given in Sect. 6. We finally draw some concluding remarks in Sect. 7.

2 Background and related works

In this section, we present some background on biological and artificial immune systems; we briefly explain the biological immune response and point out the artificial theories inspired from each step of this process. We also review some previous work done in the field of intrusion detection using artificial immune system (AIS) approaches. We discuss feature selection algorithms and their classification, and finally, we provide a brief comparison between our algorithm and other related work.

2.1 Biological and artificial immune systems

The biological immune system (BIS) responds to an intrusion or any pathogen[1] through two types of immunity: innate and adaptive immunity [85] (Fig. 1).

The *innate immunity*, also known as *non-specific immunity*, is considered as the first line of defence. It consists in four categories of barriers: (1) *Anatomical*, which includes the skin, the mucus, etc., (2) *Physiological* like the temperature and the pH, (3) *Phagocytic* such as the macrophages and the polymorph nuclear and (4) *Inflammatory* as an antibacterial activity. The phagocytic and cytotoxic cells, known as Natural Killer cells (NK), are the key agents that ensure the pathogen's termination. If the innate immunity fails to destroy the pathogen, it triggers a specific immune response.

The adaptive immunity, on the other hand, is said to be specific because it only responds to a particular pathogen. Each pathogen caries a certain shaped protein called antigen. The lymphocytes recognize each cell of its own body as self through Human Leukocyte Antigen (HLA_1 and

[1] Any disease-producing agent, especially a virus, bacterium, or other microorganism.

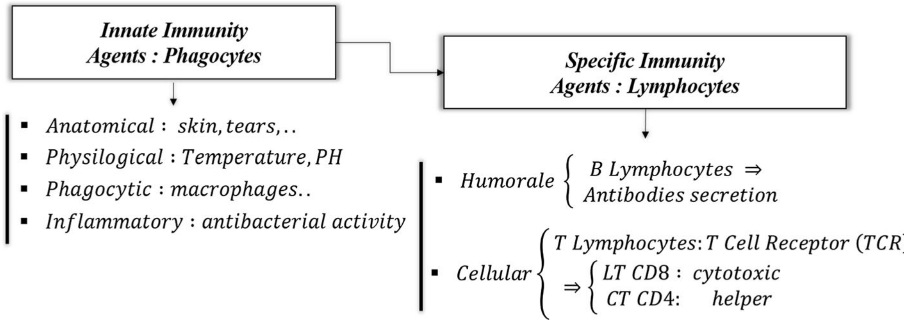

Fig. 1 Biological immune system components

HLA_2). The first class of HLA (HLA_1) has a ubiquitous expression. It is expressed on the surface of all nucleated cells including the Antigen Presenting Cells (APC), whereas the second class: HLA_2 is expressed only by the Antigen Presenting Cells, such as dendritic cells, macrophage, and Lymphocytes B (LB) cells [70].

Any other antigen is flagged as foreign and has to be destroyed. To do so, the immune response produces killer lymphocytes (B and T) and antibodies to target this one particular foreign antigen as well as memory cells in order to enhance the immune response in case of a second exposure to the same pathogen [72].

The BIS is a very complex system, and its interaction and defence mechanisms are in constant discovery. Indeed, immune cells interactions during the specific immune response, their proliferation and their maturation process have led to the definition of the main artificial immune theories and models, namely clonal selection theory, negative and positive selection theories, immune network theory and Danger theory [23]. Moreover, BIS features have been a great source of inspiration for many researchers in several fields such as pattern recognition, optimization and anomaly detection. These features can be summarized as follows:

– *The self/nonself-discrimination* During its maturation, the T cells precursor can either turn into LT4 or into LT8 through the positive selection process. Based on a survival signal delivered to the lymphocytes that can identify with a small affinity one of the HLA classes, it becomes an LT4 cell if it recognizes the HLA class II (HLA_2) or an LT8 cell if it recognizes a HLA class I (HLA_1).

  Thereafter, the LT cells, which recognize "too well" the HLA paired with a self-peptide (self-reactive T cells), must be eliminated. This elimination (apoptosis) process is called negative selection and has been an inspiration for several classification models [24].

  This maturation process leads to the generation of two types of mature LT cells (LT4 and LT8) capable of discriminating between the self-antigens, through HLA, and the foreign antigens during the immune response.

– *Immune response* APCs are cells that present antigenic peptides on their surface along with HLA_1 or HLA_2 to recruit LT_8 or LT_4 cells, respectively. The LT_8 cells become LT cytotoxic, and the LT_4 cells become T-helpers. Those T-helpers are involved in the clonal selection process [77] and the maturation of LB cells to plasmocytes in order to produce antibodies.

  Several algorithms were proposed for classification, clustering and pattern recognition that are inspired from the biological clonal selection [15].

– *Memorization and distribution* Some immune cells become memory cells for a specific foreign antigen once an immune response is activated from its first exposure. They ensure quicker and more effective immune responses of the BIS without going through the recognition and affinity maturation process.

In addition to all the above cited characteristics, self-regulation, decentralized functioning, immune response adaptation, cell proliferation are some other properties that inspired several models as solutions to real and complex problems [12]. Table 1 summarizes the immunological concepts, the models inspired therefrom and their use in computational problems.

2.2 AIS in intrusion detection

Several researchers exploited adaptive biological immunity mechanisms, mimicked them and applied them to solve several real-world and engineering problems. The ability of the human body to automatically distinguish between self-cells and nonself-cells in order to protect itself from harmful pathogens is highly consistent with the concept of intrusion detection. Hence, we are interested to apply this mechanism in intrusion detection field.

Forrest and her team introduced in 1994 one of the first applications of AIS in intrusion detection [28]. They identified the problem of protecting computer systems as the problem of learning to distinguish self from nonself. Their algorithm runs in two steps; (i) generation of a set of string detectors that do not match any of the protected data and (ii) monitoring of the protected data by comparing

Table 1 Immunity-based computational models and specific immunological concepts [23]

Immunological concepts and entities	Immunity-based models	Computational problem
Self/nonself-recognition, T cells	Negative selection algorithm	Fault and anomaly detection
Idiotypic network, immune memory, B cells	Immune network theory	Supervised and unsupervised learning
Clonal expension, affinity maturation, B cells	Clonal selection algorithm	Search and optimization
Innate immunity	Danger theory	Defence strategy

them to the detectors. Any changes in the data that activate the detectors are considered as potential intrusions.

Later on, they designed an artificial immune system (ARTIS) framework [40] and applied it in network intrusion detection domains by implementing ARTIS into LISYS (Lightweight Intrusion detection SYStem).

An immune-based network intrusion detection system (AINIDS) was proposed in [94]. AINIDS has two main components: an antibody generation and an antigen detection. It includes the generation of passive immune antibodies to detect known attacks and automatic immune antibodies that integrate statistic methods with fuzzy reasoning systems to detect novel attacks. Experiments were carried out under collected data from authors' LAN and DARPA dataset.

Others like Hong [41] presented a hybrid immune learning algorithm that combines the advantages of real-valued negative selection algorithm (RNSA) and a classification algorithm to mainly find a boundary between normal and anomaly classes.

More recently, Shen et al. [83] applied Rough Set Theory feature selection on KDDCup99 dataset and used a negative selection algorithm to detect anomalies.

Zhang et al. [100] proposed an integrated intrusion detection model based on artificial immunity (IIDAI), a vaccination strategy and a method to generate initial memory antibodies with Rough Set (RS). IIDAI integrates misuse detection model as well as anomaly detection model.

Seresht and Azmir [81] proposed a multi-agent AIS-based approach for a distributed intrusion detection system. Multiple functionalities of the proposed IDS (namely MAIS-IDS) were inspired from AIS paradigm such as the cloning, the mutation and the collaboration between agents. MAIS-IDS is a hybrid anomaly IDS (host and network) that uses agents in virtual machines where network traffic was being analyzed. Authors used a small portion of NSL-KDD to test the performance of their system with 19 features chosen from the literature. The results showed that the accuracy and false alarm rates reached 88% and 14%, respectively.

Mohammadi et al. [60] presented a real-time anomaly detection system based on a probabilistic artificial immune algorithm. A first version named SPAI (Simple Probabilistic Artificial Immune method) used the probability density function as self-detectors. As the computational cost seemed to be quite important, the authors proposed a second version, namely CPAI (Clustered Probabilistic Artificial Immune algorithm) where normal profile was clustered into subgroups. These subgroups were given priority values in a third version of the proposed algorithm in order to enhance the response time.

Ghanem et al. [30] presented a hybrid approach for anomaly detection using generated detectors, based on multi-start metaheuristic method and genetic algorithms. Detectors were generated using self- and nonself-training data, which was strongly inspired from negative selection-based detector generation. The approach reached 96% detection rate and 7% false positive rate when evaluated with NSL-KDD dataset.

Abas et al. [1] applied Rough Set Theory (RST) on gureKddCup6percent dataset in order to eliminate irrelevant and redundant features. Then, they tested R-chunk and negative selection algorithm on the six resulting features. The accuracy of the experimental evaluation reached 90%.

Igbe et al. [43] proposed a framework for a distributed network intrusion detection system (d-NIDS) based on negative selection algorithm (NSA). They used a genetic algorithm to generate a set of detectors that were distributed between IDS participants. The performance of this framework was evaluated using NSL-KDD dataset; an average detection rate of 98% and false positive rate of 1.77% were recorded.

Authors in [80] presented an AIS-based anomaly detection and prevention system. They introduced self-tuning and detector power to achieve dissimilarities among the detectors in order to cover as much space as possible with minimum computational cost. Experiments were carried out using different percentages of KDDCup99 training dataset. They studied and compared the effects of the dataset and the affinity threshold on detectors' generation as well as detection rate and false positive rate. The results showed a high detection rate (100%) but presented important false alarm values (68%).

2.3 Feature selection

Feature selection is an important data preprocessing phase used to remove irrelevant, redundant and noisy data with respect to the description of the problem at hand. It aims to improve the data processing performance and reduce the computational complexity.

Feature selection has been the focus of researchers in many fields since the early nineties [51]. Various methods have been proposed, tested and discussed. They can globally be classified into two categories: filter-based and wrapper-based methods.

2.3.1 Filter-based methods

These methods are based on performance evaluation metric deduced from general characteristics of the training data (Fig. 2). They are carried out once, and the output can be provided to different classifiers. Filters are commonly independent from the learning algorithm. They are known for their low computational complexity and good generalization ability.

Some filters provide a feature ranking rather than an explicit best feature subset, and the cutoff point in the ranking is usually chosen via cross-validation. Among feature selection and feature ranking techniques, we find: correlation-based feature selection (CFS), fast correlation-based feature selection (FCBF), gain ratio attribute evaluation (GR), information gain (IG), chi-squared evaluation and others [46].

2.3.2 Wrapper-based methods

As to wrapper methods, they use the feedback of a classification algorithm to assess the quality (efficiency) of feature subsets (Fig. 3). These subsets are usually created using some search strategy. Even though the wrapper approach has the advantage of handling the possible interactions between features, it does not ensure the generalization as good as filter approach does. This is mainly due to the fact that it leans toward the specific learning algorithm used to choose the best feature subset.

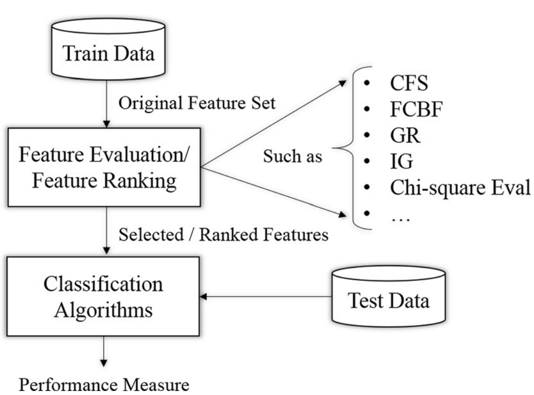

Fig. 2 Filters feature selection

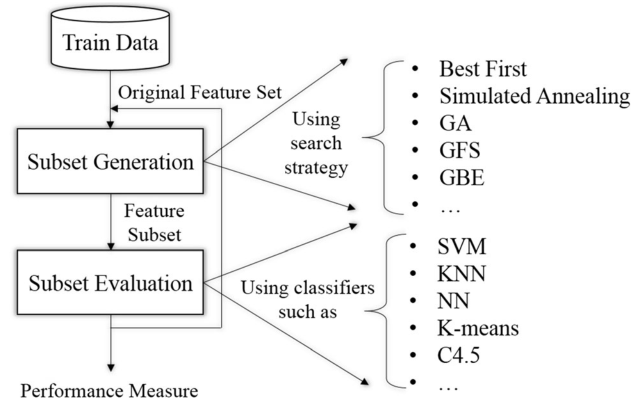

Fig. 3 Wrapper feature selection

Furthermore, wrappers are restricted by the time complexity of the learning algorithm, which increases rapidly as data get larger.

2.4 Feature selection in intrusion detection

In this section, we provide a small overview of feature selection techniques applied in intrusion detection area using different datasets, including UNSW-NB15, the latest benchmark provided by the Australian Centre for Cyber Security [63].

Kayacik et al. [47] investigate the relevance of each feature in KDD99 intrusion detection dataset to discriminate normal behavior form attacks using information gain (IG). Based on the entropy of a feature, IG measures the role of this feature in predicting the class label. It is said to be relevant if the value of IG is close to 1. Since IG is measured for discrete values, authors preprocessed continuous features by partitioning each one into equal-sized partitions using equal frequency intervals. Authors reached the conclusion that *normal*, *neptune* and *smurf* classes are highly related to certain features which make their classification easier as compared to other type of attacks that fall into U2R and R2L categories.

Gonzalez-Pino et al. [31] integrated a feature selection process based on information gain (IG) in their intrusion detection system that deploys data mining and fuzzy logic approaches. In order to efficiently detect malicious events and be able to analyze large amounts of data in real time, an IDS has to select attributes that are relevant enough for an adequate detection profile with a low FPR. To this end, authors used decision tree learning with IG algorithm to select features from DARPA99 dataset. They limited their experiments to evaluate the detection of *ipsweep* attack and compared the results with and without using IG algorithm.

The article in [66] presents a study of four filter-based feature selection methods using different classification algorithms (5-Nearest Neighbor, C4.5 decision tree and

Naïve Baye) under Kyoto2006+ dataset for intrusion detection. Authors performed ANalysis Of Variance (ANOVA) along with Tukey's Honestly Significant Difference (HSD) test to compare the performance of three feature rankers: signal-to-noise ratio, chi-squared and AUC. Authors concluded with the statement that overall filter-based rankers perform better than the feature subset evaluation method. Among the feature rankers, S2N is performing best.

In [9], authors investigate the minimal subset of the most relevant features in NSL-KDD dataset. They used Correlation Feature Selection (CFS) technique to filter attributes highly correlated with the class and uncorrelated with each other. Different search methods were used to build subsets, and Naïve Bayes algorithm was the algorithm performed to compare the results. The experiments point out 12 features from the complete set that are commonly selected by all search methods as the most important and reliable attributes. The accuracy and FPR reported under these 12 attributes regarding U2R attack category are 65.43% and 0.01%, respectively.

Another article [75] studied feature selection for machine learning algorithms under NSL-KDD dataset. They tried to find the optimal feature subset using Discretized Differential Evolution (DDE), which is a population-based search technique, and C4.5 decision tree algorithm. The classification performances under training and testing sets with tenfold cross-validation reached 99.01% and 82.37%, respectively, with an average FPR of 0.007% and 0.15%.

Authors in [48] have tried to minimize the computational time of machine learning and data mining techniques with high-dimensional data using feature selection. They investigated a wrapper approach based on a GA as search strategy and logistic regression as learning algorithm. They used various subsets of both KDD99 and UNSW-NB15 datasets for their experiments. The proposed algorithm selected an average of 18 and 20 attributes from KDD99 and UNSW-NB15 with an accuracy reaching 99.3% and 92.5%. Moreover, the same features applied to *C4.5*, *RF* and *NBTree* reached an average accuracy of 99.7%, 99.8% and 99.7% under KDD99 and 80.6%, 80.45% and 80.2%, respectively.

Janarthanan and Zargari [44] explored relevant features in both NSL-KDD and UNSW-NB15 using machine learning techniques. Since the two datasets are significantly different, authors compared their results against previous works applied on both datasets with Random Forest algorithm. They employed several feature selection algorithms implemented in Weka tool as CfsSubsetEval with GreedyStepwise method and InfoGainAttributeEval with Ranker method. Experiments were carried out using two feature subsets, one extracted from previous work ([99] for KDD99 and [62] for UNSW-NB15 datasets) and the other proposed by the authors. The results show that the second subset improves Kappa statistic, which indicates better detection rates.

2.5 Comparison with related work

Most of the related work previously cited used a binary representation of data flow without fully explaining the conversion process from raw or featured data connections into binary strings, as well as the computational cost of such an operation. Moreover, when featured connections are considered without binary transformation, authors tend to deal with nominal attributes by coding them with iterative integers. This usually leads to the biased classification results. In this work, we replace the nominal attributes by their occurrence probabilities when statistical operations are conducted (as feature selection). Otherwise, we handle them as nominal in order to gain in efficiency and make most of the information provided by this type of features.

Besides, most of the empirical analyses are performed, exclusively, under KDDCup99 dataset, which have been for the past two decades the benchmark for the evaluation of IDS and the only labeled dataset publically available even though it is largely outdated.

We detail, in this paper, the full process of our proposed algorithm based on the negative selection theory, from the feature selection phase with a multi-type representation (real, nominal), through detectors generation to the decision rules and the classification phase. We test and evaluate our approach under NSL-KDD dataset, Kyoto2006+ dataset as well as UNSW-NB15 dataset.

As for feature selection, a great amount of work has been done regarding this preprocessing phase in network intrusion detection, but almost all of the published results used DARPA 98 [39] or KDDCup99 dataset [44, 47, 48, 68] and few used NSL-KDD [9, 75] or Kyoto2006+ [4, 6, 66] datasets. An increasing number of researches have been carried out using the new benchmark data for intrusion detection, namely UNSW-NB15 developed by Moustafa and Slay [63] at the Australian Center for Cyber Security. Table 2 gives a summarized overview of feature selection techniques discussed in the related work.

We propose as feature selection module, a new filter-based algorithm that exploits the Coefficient of Variation (CV) to rank attributes of datasets. The algorithm Coefficient of Variation-based Feature Selection (CVFS) is further explained in Sect. 3.2, and the results are presented in Sect. 5.1.

Table 2 Summary feature selection related work

References	Feature selection	N	Classifier	Dataset	Attack category	Evaluation metrics
[39]	EA	1–8	RBF	DARPA 1998	Back, Dict, Guest, Nmap, Ipsweep, Portsweep, Warezclient	Classification errors, FA, MA
[47]	IG	1–11	–	KDD99	All	–
[31]	IG	3	FL	DARPA99	ipsweep	–
[68]	GeFS_CFS, GeFS_mRMR	5, 18	C4.5, SVM, BN, CART	KDDCup99	DOS, Probe	Accuracy
[66]	S2N, CS, AUC, CFS	6	C4.5, NB	Kyoto2006+	All	ROC, ANOVA, Tukey's Honestly Significant Difference (HSD)
[9]	CFS	12	NB	NSL-KDD	U2R	Accuracy, FPR
[75]	DDE	16	C4.5	NSL-KDD	All	Accuracy, Error Rate, TPR, FPR, Precision, Recall, F-measure
[48]	GA	18, 20	LR	KDD99, UNSW-NB15	All	Accuracy, Recall, FAR, AIC, McFadden R2
[44]	Cfs Subset Eval + Greedy Stepwise, IG + Ranker	10, 8	RF	KDD99, UNSW-NB15	All	Kappa stat[a], ROC value
[62]	ARM	11	NB, EM	UNSW-NB15	All	Accuracy, FAR

EA evolutionary algorithm, *IG* information gain, *GeFS* generic feature selection, *S2N* signal to noise, *CS* chi-square, *AUC* area under curve, *CFS* correlation-based feature selection, *DDE* discritized differential evolution, *GA* genetic algorithm, *ARM* association rule mining, *RBF* radial basis function, *FL* fuzzy logic, *NB* Naïve Bayes, *LR* linear regression, *RF* random forest, *EM* expectation–maximization; *FA* false alarm, *MA* missing alarm, *AIC* Akaike information criterion

[a]Kappa statistic is a measure that takes the expected figure into account by deducting it from the predictor's success and expressing the result as a proportion of the total for a perfect predictor, to yield extra success out of a possible total of predictions

3 NSNAD description

3.1 Overall architecture

The overall architecture and main components of the negative selection algorithm for network anomaly detection (NSNAD) are depicted in Fig. 4. The training dataset is passed through a filter/ranking-based feature selection, which results in a subset of relevant features with low dispersion. NSNAD is based on a semi-supervised classification, in the sense that the training data used as input contain labeled as well as unlabeled records [102]. The labeled instances represent normal or self-cells in the biological sense. These self-data are used to validate a set of detectors randomly picked from the complete unlabeled training data with both classes. A radius for each detector is computed and used to classify instances from the test set as anomaly.

NSNAD detectors can be assimilated to LT4 cells recognizing only AGs presented by APCs along with an HLA_2. An interaction with the AG will allow it to be classified as "anomaly". However, if the latter is not flagged by any detector, the possibility that it is (by analogy) a cell expressing only HLA_1 (not recognized by LT4 cells) cannot be disregarded. This led us to add a second verification based on an $Artificial-HLA_1$. It consists in comparing the structures of the AG (instance) with the self's HLA_1. If these structures are close enough, the instance is considered as "normal"; otherwise, it is classified as "anomaly".

3.2 Feature selection

In the preprocessing phase, we introduce a feature selection technique that falls into filter-based category as it uses a statistical metric to rank the features. The cutoff point in the ranking can be fixed by two different ways: either using a subjective threshold or using a cross-validation feedback. The Coefficient of Variation (CV) is the statistical metric we use to define the most relevant attributes in terms of dispersion. It is expressed as follows:

$$CV_i = \frac{\sigma_i}{\mu_i}, \quad i = \{1..p\} \qquad (1)$$

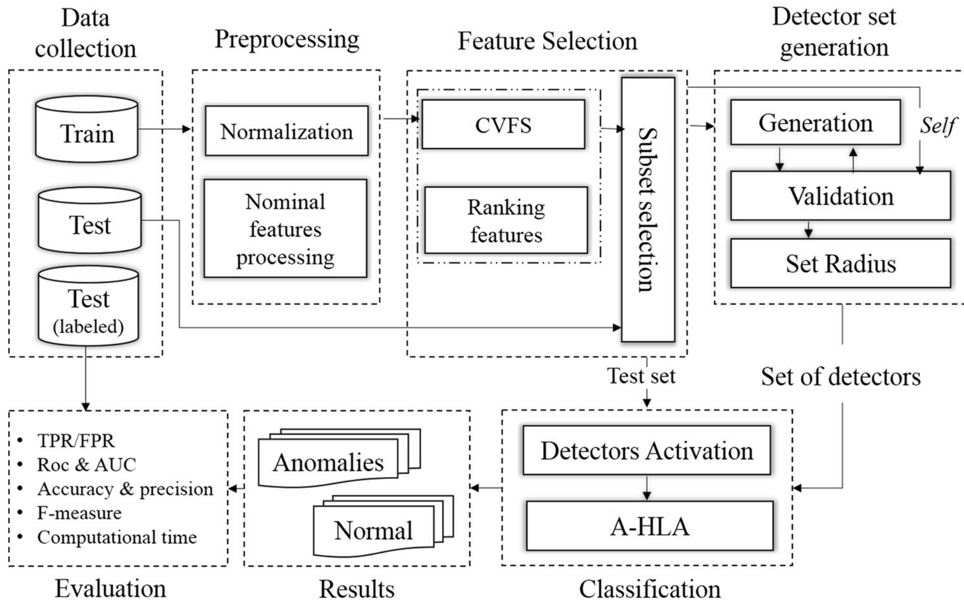

Fig. 4 Block diagram of NSNAD

where σ_i and μ_i are the standard deviation and the mean of the attribute i in the training data.

We compute the coefficient of Eq. 1 for each attribute in the training data regardless of instances' class. In fact, one of the advantages of this technique is that it does not need a label attribute in the ranking process, unlike other filter-based algorithms. As another advantage of CVFS, it makes it possible to compare attributes with different scales and units. Moreover, CVFS defines both the dispersion and the homogeneity of attributes depending on whether their values are high or low.

The choice of whether to keep features with high or low CV values depends on the field of application. For instance, in radar image processing, the CV is used as filtering criterion. A value of CV below a threshold results in the application of a filter. It is also used for values lower than a threshold to select, in the multi-temporal radar images, the targets with a stable temporal backscatter.

In a multi-class classification case with p attributes, a high CV value of an attribute, usually, reflects its important contribution in the class's discrimination. However, in anomaly detection, an intrusion is identified if it deviates from a normal behavior. Ideally, this normal behavior has to be projected as a compact sphere in the multidimensional space.

Since NSNAD is a semi-supervised algorithm, we rather seek the least disperse attributes, those that allow us to project normal instances as tightly as possible in order to reduce false negatives and optimize the detection rate.

Before applying CVFS process, we excluded some features of Kyoto2006+ dataset that, we believe, are irrelevant for the overall classification approach. We leave out the source and the destination IP addresses so that the method is independent from these particular addresses. We also omit the features related to security analysis resulting from some software (as clam anti-virus and Snort) so that our method can be generalized to other networks that might not have such software in their network.

We ordered the features in ascending order according to their CV values so that we can choose the best feature subset. In fact, there are two ways to determine the number of attributes to retain in a subset: (1) by choosing a threshold for the CV beyond which a subset of attributes is kept. In this case, the threshold is entirely subjective and may not yield the best subset for a given classifier. (2) by using another method that exploit the tested classifier feedback to choose the best attribute's subset. In our work, we adopt the second method following these steps:

- Fix the cardinality of the smallest feature subset, and let it be $|FTsubset_0|$.
  $|FTsubset_0| < |FT|$, with $|FT|$ = number of features in the data.
- Choose an evaluation metric, eg: f-measure;
- Run the algorithm using k-folds cross-validation[2] ($k = 6$) against training data;
- Add attributes incrementally with respect to the order made under CV values;
- Choose the feature subset that gives the highest f-score to carry further experiments and comparison.

[2] In k-fold cross-validation, the original sample is randomly partitioned into k equal-sized subsamples. Of the k subsamples, a single one is retained as test data, and the remaining $k-1$ subsamples are used as training data. The cross-validation process is then repeated k times, with each of the k subsamples used exactly once as test data. The k results from the folds are then averaged to produce a single estimation.

In this case, the algorithm has to be run at most $[(|FT| - |FTsubset_0|) + 1] \times 6$ times. The process's results are detailed in Sect. 5.1.

3.3 Detector set generation

NSNAD is based on the negative selection algorithm known as the principle of self/nonself-discrimination of the immune system. In the proposed algorithm, normal records from train set represent the "self". The detectors, on the other hand, represent the matured LT4 immune cells that do not match any self-data.

Along with the normal instances from the train set, the complete unlabeled training data are provided to NSNAD in order to generate the detectors. In a random manner, detector candidates are picked from the complete unlabeled dataset and validated against normal samples regarding each feature:

Let $TR \in R^p$ be the training data featured by p attributes and contain both classes (normal and anomaly).

$S \subset TR$ is a set of normal instances (represent self-antigens) and r_{self} their radius. $r_{\text{self}} = \{r_1, r_2, \ldots, r_p\}$ is a p-dimensional vector and:

$$r_i = \begin{cases} \alpha \times \sigma_{i_{\text{self}}}, & \text{if } i \text{ is numeric} \\ \text{String value with the largest number of occurences in } S & \text{if } i \text{ is nominal} \end{cases} \tag{2}$$

where α is a regulation factor $\in [1, 3]$. For an average confidence interval of 96%, we put $\alpha = 2$ (refer to three sigma rule or empirical rule [27]).

$\sigma_{i_{\text{self}}}$ is the standard deviation of the attribute i in S.

To construct a set of detectors, $DT \in R^p$, we validate each detector candidate instance $X = \{x_1, x_2, \ldots, x_p\}$, randomly picked from the non-labeled TR, versus self-data according to the nature of the attribute i as follows:

$$\text{if } i \text{ is nominal}: \; x_i \neq r_i$$
$$\text{if } i \text{ is numeric}:$$
$$\text{For } j = 1 \ldots |S|, \text{ if } \left\{ \begin{array}{lll} & |x_1 - s_{1j}| & > r_1 \\ \text{and} & |x_2 - s_{2j}| & > r_2 \\ \text{and} & \ldots & \\ \text{and} & |x_p - s_{pj}| & > r_p \end{array} \right\} \begin{array}{l} \text{then} \\ DT = \\ DT \cup X \end{array} \tag{3}$$

Which means that the distance between the instance X and all "normal" instances in S has to be greater than the self-radius r_{self} regarding each and all the attributes.

If these conditions are met, X will be added to the detector set $DT \in R^p$. Otherwise, another candidate instance is randomly picked from TR and validated with Eq. 3 as explained above. This process is repeated until DT contains a predefined number of detectors or all the instances were picked out.

Moreover, we assigned to each detector d from DT a radius rd used to test its activation in the classification phase. This radius represents the lowest Manhattan distance between d and self-instances (Eq. 4).

$$rd = \min_{j=1}^{|S|} \; dist\,(d, s_j)$$
$$\text{and}: \; \text{dist}\,(d, s_j) = \sum_{i=1}^{P} |d - s_{ij}| \tag{4}$$

Pseudocode 1 details the complete process.

Pseudo code 1 Detector set generation

Input: $S \in R^p$, $TR \in R^p$, $nbr\_detectors$, α
Output: Set of nonself detectors DT
Begin $\alpha = 2$
Compute self_radius : $r_{self} = \{\alpha \times \sigma_{1_{self}}, \; \alpha \times \sigma_{2_{self}}, .., \; \alpha \times \sigma_{i_{self}}, .., \; \alpha \times \sigma_{p_{self}}\}$
$nbr\_d = 0$
while $((nbr\_d < nbr\_detectors) or (nbr\_d = |TR|))$ **do**
  Pick a candidate detector X from TR
  // Validation
  for all $(s \in S)$ **do**
    $dist = \{|x_1 - s_1|, \; |x_2 - s_2|, .., \; |x_i - s_i|, .., \; |x_p - s_p|\}$
    if $(dist_{\{1..p\}} < r_{self_{\{1..p\}}})$ **then**
      Break //X is a self
    end if
  end for
  //Add detector and compute radius
  if (reached the last self instance) **then**
    Compute detector radius (Eq. 4)
    add X to DT
    $nbr\_d{+}{+}$
  end if
end while
End.

3.4 Classification

The classification phase is a two-stage process. First, the radius of the generated detectors is used to identify and classify an incoming instance as "anomaly". If this instance is not in the range of all the detectors' radius, it is compared to an *A-HLA class I (Artificial Human Leukocyte Antigen)* to be classified as "normal".

For an input instance a to activate the detector d, the Manhattan distance between d and a must at most be equals to the detector's radius rd (Eq. 4). In other words, for an instance a to be classified as anomaly, the inequality Eq. 5 should be satisfied for at least one detector in DT.

$$\text{dist}\,(a, d) = \sum_{i=1}^{P} |a_i - d_i| \leq rd \tag{5}$$

If none of the detectors in *DT* has been activated, the instance *a* could be classified as normal/self. However, in our work we first compare *a* to the Artificial HLA *(A-HLA)* analogously to the HLA class I (see Sect. 2.1). We perform this additional verification in order to detect any new attack that was not covered by the detectors. We identify our A-HLA (Artificial HLA) as the volume of self-data V_{self} (Eq. 6). S is the self-space and μ its mean.

$$V_{\mathrm{self}} = \left(\frac{2}{\sqrt{p}}\right)^{p} \prod_{j=1}^{P} \max\left(|s_{ij} - \mu_{ij}|\right), \quad \forall s \in S \quad \text{and} \quad i = 1 \ldots |S| \tag{6}$$

In this case, for an incoming instance *a* to be classified as normal, it should:

- not activate any detector in *D*,
- and satisfy the inequality (Eq. 7) regarding the volume of self-space.

$$\left(\frac{2}{\sqrt{p}}\right)^{p} \prod_{j=1}^{P} |a_j - \mu_j| < V_{\mathrm{self}} \tag{7}$$

Pseudocode 2 summarizes the classification process.

Pseudo code 2 Classification

```
Input:  DT ∈ R^p, TS ∈ R^p, S ∈ R^p
Output: TS labeled (normal/anomaly)
Begin
Compute self_volume V_self (Eq. 6)
for all (a ∈ TS) do
   for all (detector d ∈ DT) do
      dist = distance(d, a) (Eq. 5)
      if (dist ≤ rd) then
         classify a as anomaly
      end if
      Break
   end for
   //a is not in any detectors' radius, check A-HLA
   if (Eq. 7 satisfied) then
      classify a as normal
   else
      classify a as anomaly
   end if
end for
End.
```

4 Experimental design

Three baseline algorithms are chosen to compare against NSNAD performances: Naïve Bayes, *K*-means and One-class SVM.

Naïve Bayes is based on Bayes theorem; it uses a labeled input samples in order to build a model, wherefrom an unknown sample is classified under a likelihood probability [71]. We exploit in this work, JAVA package "Naive-Bayes" from Weka tool [36].

K-means, on the other hand, is a clustering algorithm that does not need any prior knowledge of the data distribution or a label. It rather aims to partition *n* samples into *k* groups [53]. Training set is partitioned into two groups with *k*-means; incoming samples, from the test set, are then assigned to the nearest cluster (using Euclidean distance). It should be known that this kind of algorithms make the implicit assumption that normal samples are more frequent than anomalies, which means that the biggest cluster is assumed to contain normal instances after clustering [76].

As for *One-class SVM*, it is a semi-supervised variant of the support vector machine (SVM) algorithm [20]. The principle is to learn a decision function for novelty detection classifying new data as similar or different from the training set. A kernel function is implicitly used as similarity measure. The RBF kernel is the most used function with SVM classification algorithms [92]. One of its advantages is the possibility to apply other distances than the Euclidean distance in the exponential expression [14]. Moreover, its "sigma" parameter gives more flexibility with regard to the input space dimension. We used in this work the "svm" package of Scikit-learn tool [13, 74].

4.1 Datasets

To assess the performance of any detection approach, experimentation on benchmarks or enough heterogeneous and realistic data, with up-to-date network attacks, is required [96]. In our work, we evaluated our approach with three different datasets, namely NSL-KDD, Kyoto2006+ and UNSW-NB15 datasets. A brief description of each dataset is given in this section along with test subsets used in the empirical study.

4.1.1 NSL-KDD

NSL-KDD is the refined version of KDDCup99 known for some deficiencies mentioned in [90]. It has the following advantages over the original KDD dataset [25]:

- Redundant records are removed so that the classifiers will not be biased toward more frequent records.
- A sufficient number of records are available in the train and test datasets, which make it affordable to run the experiments on the complete set. Moreover, the evaluation results of different research works will be consistent and comparable.
- The number of selected records from each difficult level group is inversely proportional to the percentage of records in the original KDD dataset.

The dataset contains a large volume of network TCP connections, the results of 5 weeks of capture in the Air Force Network. Each connection consists of 41 attributes

Table 3 NSL-KDD dataset

Dataset	Description	#Normal	#Attack
nslKDD Train+	Full NSL-KDD train set with attack-type labels and difficulty level	67,343	58,630
nslKDD Train20%	20% KDDTrain+ dataset	13,449	11,743
nslKDD Test+	Full NSL-KDD test set with attack-type labels and difficulty level	9711	12,833
nslKDD Test-21	KDDTest+ dataset with no difficulty level 21	2152	9698

Table 4 NSL-KDD test subsets

Sub.	Description (%)	#Normal	#Attack	All
1	10	7799	7052	14,851
2	20	15,477	14,226	29,703
3	30	23,221	21,334	44,555
4	40	30,909	28,497	59,406
5	50	38,586	35,672	74,258
6	60	46,312	42,798	89,110
7	70	54,055	49,906	103,961
8	80	61,880	56,933	118,813
9	90	69,585	64,080	133,665
10	100	77,054	71,463	148,517

plus a label of either normal or a type of attack. Simulated attacks fall into one of the four following categories:

- *DOS (Denial of service)* aims at making a service or a resource unavailable.
- *U2R (User to root)* A simple user tries to exploit a vulnerability in order to obtain super user or administrator privileges.
- *R2L (Remote to local)* Attacker attempts to gain access (account) locally on a machine accessible via the network.
- *PROBE* represents any attempt to collect information about the network, the users or the security policy in order to outsmart it.

NSL-KDD is actually available in the form of four datasets. Table 3 shows the distribution of normal and attacks instances in each dataset.

In our study, we use normal instances of nslKDDTrain+ dataset as *self* and we perform our tests under ten subsets from the complete NSL-KDD data (nslKDDTrain+ and nslKDDTest+ combined). Each test subset represents a percentage of the data as it is described in Table 4.

4.1.2 Kyoto2006+

Kyoto2006+ dataset is a collection of over 2.5 years of real traffic data (Nov. 2006–Aug. 2009) realised by Kyoto University for evaluating IDSs using a much recent dataset than KDDCup99, which was for a long time, the only publically available labeled dataset. Kyoto dataset was collected from 348 honeypots (Windows XP, Windows Server, Solaris...) deployed inside and outside of Kyoto University. All traffic was thoroughly inspected using three security softwares SNS7160 IDS, Clam AntiVirus, Ashula, and since Apr.2010, snort was added.

Connections are featured by 24 attributes, among which, the first 14 ones were extracted based on KDDCup99 dataset and the remaining 10 attributes were added to further analyze and evaluate network IDSs. A detailed analysis can be found in [86]. The dataset is available as text files [91] representing the daily traffic labeled as "normal" or "attack". Figure 5 shows the monthly distribution of the dataset in 2009 after removing duplicate records.

This study uses the January 1, 2009, as training set and tests the algorithms on the subsets described in Table 5.

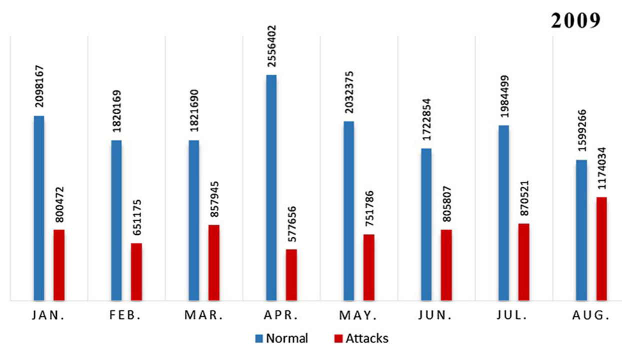

Fig. 5 Monthly distribution of Kyoto2006+ in 2009

Table 5 Kyoto2006+ test subsets

Sub.	Description	#Normal	#Attack	All
1	20090102	43,228	73,267	116,495
2	20090104	48,003	47,068	95,071
3	20090105	54,426	59,051	113,477
4	20090112	62,615	27,588	90,203
5	20090121	70,972	39,957	110,929
6	20090123	71,843	32,158	104,001
7	20090124	62,677	26,702	89,379
8	20090125	70,712	23,250	93,962
9	20090128	78,376	26,104	104,480
10	20090130	80,613	25,729	106,342

4.1.3 UNSW-NB15

UNSW-NB15 dataset is the newest benchmark dataset for IDS test and evaluation [64]. It was created by a research group at the Australian Centre for Cyber Security (ACCS). It contains both real modern normal network connections and synthetical attack traffic generated using IXIA PerfectStorm tool in the Cyber Range Lab of the ACCS. The following categories of attacks, along with normal traffic, were simulated from an updated CVE Web site:

– *Analysis* represents different attacks like port scan, spam and scripts penetration.
– *Backdoors* Bypassing normal authentication to secure unauthorized remote access to a resource.
– *DoS* A malicious attempt to make a server or a network resource unavailable.
– *Exploits* Commands or instructions that take advantage of a vulnerability in a network or program.
– *Fuzzers* Injection of massive random data into an application or a network to crash it.
– *Generic* Attack that works against all block-ciphers (with a given block and key size), without consideration about the structure of the block-cipher.
– *Reconnaissance* contains all strikes that can simulate attacks that gather information.
– *Shellcode* A small piece of code used as the payload in the exploitation of software vulnerability.
– *Worms* Attacker replicates itself in order to spread to other computers. Often, it uses a computer network to spread itself, relying on security failures on the target computer to access it.

Simulations were performed during 16 h on Jan 22, 2015, and 15 h on Feb 17, 2015. 100GB were captured; the first 50GB were captured with a one attack per second configuration, while the second 50GB with ten attacks per second [63]. UNSW-NB15 dataset is available as csv files at [61]. The number of records in the training and testing set as well as their distribution is given in Table 6.

Normal records in the train set are taken as "self" in this study, and test subsets are described in Table 7.

Table 6 UNSW-NB15 data distribution

Attack	Training set	Testing set
Analysis	677	2000
Backdoors	583	1746
DoS	4089	12,264
Exploits	11,132	33,393
Fuzzers	6062	18,184
Generic	18,871	40,000
Reconnaissance	3496	10,491
Shellcode	378	1133
Worms	44	130
Normal	37,000	56,000

Table 7 UNSW-NB15 test subsets

Sub.	Description (%)	#Normal	#Attack	All
1	10	5607	11,927	17,534
2	20	11,214	23,854	35,068
3	30	16,850	35,752	52,602
4	40	22,461	47,675	70,136
5	50	28,062	59,608	87,670
6	60	33,657	71,547	105,204
7	70	39,221	83,517	122,738
8	80	44,846	95,426	140,272
9	90	50,546	107,260	157,806
10	100	56,000	119,341	175,341

4.2 Input parameters

NSNAD has as input parameters: the number of detectors (*nbr_detectors*), the self-data ($S \in R^p$, $S \subset TR$), the unlabeled train ($TR \in R^p$) set and test subsets ($TS \in R^p$). The number of detectors is set to 1% testing data. Normal records of the train set represent the self. The unlabeled train data, used to generate detectors, contain both classes (normal and anomaly).

Naïve Bayes classifier constructs a probabilistic model from a set of train data and assigns class labels to problem instances using Bayes' theorem. It only takes as input a labeled train data with both classes and the unlabeled test sets to be classified.

As for *K-means* clustering algorithm, the number of clusters k is required as well as a similarity distance along with an unlabeled train and test sets. In this work, $k = 2$ and the Euclidean distance is used for similarity comparison. Once the algorithm clusters train set into two groups, we assume that the largest cluster is the "normal" traffic and assigns test instances to the nearest cluster.

Finally, the input parameters required for *One-class SVM* classifier with radial basis function (RBF) are: *gamma* that defines how far the influence of a single training example reaches. It is set to $\gamma = 1/p$, where p is the number of features in the data and *nu* which is an upper bound on the fraction of training errors and a lower bound of the fraction of support vectors [73].

All experiments are performed using Windows 8 64 bit platform with core i7 processor running at 2.40 GHz and 8 GB RAM.

Table 8 Evaluation metrics

Metric	Equation
True positive rate	$TP/(TP+FN)$
False positive rate	$FP/(FP+TN)$
Accuracy	$TP+TN/(TP+TN+FP+FN)$
Precision	$TP/(TP+FP)$
F-measure	$2\times TP/(2\times TP+FP+FN)$
Matthews correlation coefficient	$\frac{TP\times TN-FP\times FN}{\sqrt{(TP+FP)\times(TP+FN)\times(TN+FP)\times(TN+FN)}}$

4.3 Evaluation metric

Several performance metrics were used in this work. Their equations are given in Table 8 where TP (true positive) are anomalies correctly classified, TN (true negative) are normal events successfully identified as such, FP (false positive) are normal events wrongly identified as anomalies, and FN (false negative) are anomalies misclassified as normal.

5 Results and analysis

In this section, we aim to present a comprehensive comparison of two feature selection algorithms with CVFS and three different classifiers against NSNAD. We also aim to highlight the contribution of the $A-HLA_1$ step in NSNAD's overall performance optimization.

The validation process is carried out using train and test sets of each benchmark dataset. We incrementally varied the size of the provided sets in order to explore the algorithms' scalability. The distribution of the subsets is described in the previous section, Tables 4, 5 and 7.

Moreover, the results depicted in the manuscript for each sample are the mean of 10 successive runs.

5.1 Feature subset and normalization

At first, attributes of each dataset (NSL-KDD, Kyoto 2006+ and UNSW-NB15) are ranked according to their CV values (the results are given in Table 9). Then, a six-fold cross-validation is performed under training data with different feature subsets (Fig. 6). These subsets are created incrementally with respect to the order given in Table 9. The size of the outcome subsets is 29, 10 and 11 for NSL-KDD, Kyoto2006+ and UNSW-NB15 datasets, respectively. The description of each feature is detailed in Tables 15, 16 and 17 (see "Appendix").

In order to compare the relevance of our feature subset with other well-known algorithms, we applied information gain (IG) as well as Correlation Attribute Evaluation (CAE) algorithms under train set of each dataset. The ranked attributes are also reported in Table. 9. It is noteworthy to mention that both algorithms need labeled train data to evaluate the relevance of data attributes, which is not the case of CVFS.

We also present, in Fig. 7, the *f*-measure of NSNAD+HLA, NB, *K*-means as well as One-class SVM applied to the complete test sets using the same number of attributes as CVFS (29, 10 and 11 attributes for NSL-KDD, Kyoto 2006+ and UNSW-NB15 datasets, respectively). The depicted results show that NSNAD+HLA has better *f*-measure results under all datasets with both feature selection algorithms.

Furthermore, we noticed a huge difference in scale between attributes' values in the training data. This usually

Table 9 Attributes' ranking

Dataset	Algorithm	Attributes
NSL- KDD	CVFS	20, 21, 7, 22, 2, 4, 32, 29, 12, 34, 3, 33, 23, 38, 25, 26, 39, 36, 35, 40, 24, 41, 31, 28, 27, 30, 37, 1, 10, 8, 19, 14, 11, 17, 15, 18, 16, 13, 5, 9, 6
	IG	5, 3, 6, 4, 30, 29, 33, 34, 35, 38, 12, 39, 25, 23, 26, 37, 32, 36, 31, 24, 41, 2, 27, 40, 28, 1, 10, 8, 13, 16, 19, 22, 17, 15, 14, 18, 11, 7, 21, 20, 9
	CAE	29, 33, 34, 12, 39, 38, 25, 4, 26, 23, 32, 3, 28, 41, 27, 40, 35, 30, 31, 8, 36, 2, 37, 1, 22, 19, 15, 17, 14, 10, 16, 13, 18, 7, 5, 6, 11, 9, 21, 24, 20
Kyoto 2006+	CVFS	2, 14, 10, 6, 9, 15, 5, 11, 8, 13, 12, 16, 7, 1, 4, 3
	IG	16, 2, 10, 4, 3, 1, 14, 15, 9, 5, 6, 11, 8, 13, 12, 7
	CAE	10, 2, 14, 6, 9, 5, 15, 8, 13, 16, 12, 11, 7, 1, 4, 3
UNSW-NB15	CVFS	38, 43, 5, 3, 4, 11, 33, 21, 32, 42, 24, 12, 22, 23, 41, 34, 28, 37, 36, 35, 10, 27, 25, 29, 26, 13, 14, 2, 40, 30, 7, 20, 6, 17, 16, 19, 18, 39, 9, 15, 8, 31
	IG	1, 8, 28, 13, 9, 33, 29, 10, 12, 2, 18, 11, 7, 14, 26, 25, 36, 5, 17, 27, 6, 19, 3, 20, 16, 35, 42, 15, 32, 21, 4, 22, 24, 23, 37, 34, 41, 31, 43, 40, 30, 38, 39
	CAE	11, 21, 5, 36, 1, 24, 35, 10, 3, 33, 42, 32, 23, 22, 14, 37, 41, 4, 34, 29, 26, 25, 13, 17, 27, 43, 12, 40, 7, 28, 16, 18, 9, 6, 19, 20, 30, 8, 39, 38, 31, 15, 2

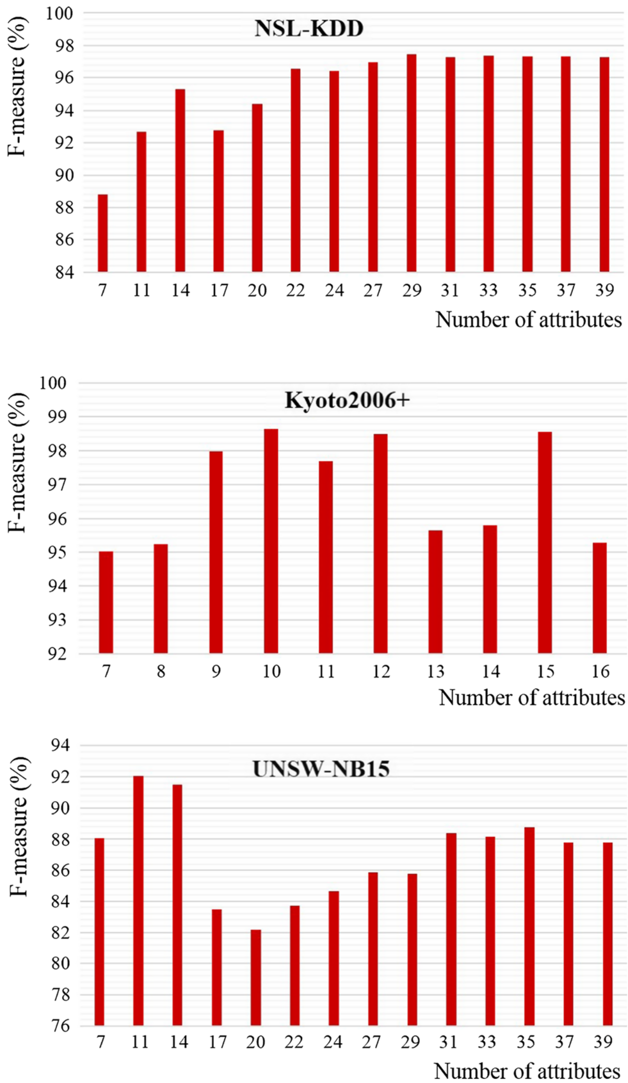

Fig. 6 NSNAD+HLA versus number of attributes

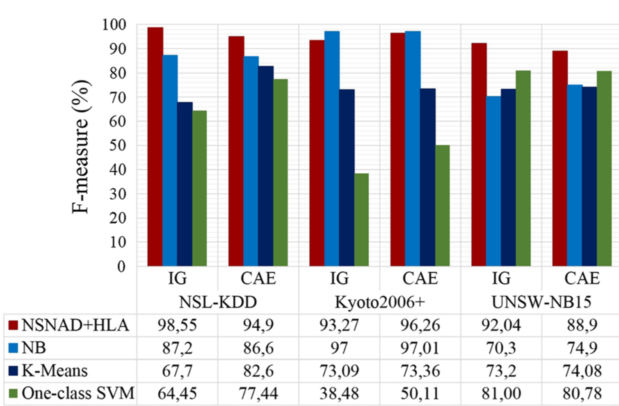

Fig. 7 Performances with other feature selection

leads to the biased results. To overcome this problem, we normalized the numerical attributes using Eq. 8.

$$x_{P\_\mathrm{norm}} = \left(\frac{x_p - \min_p}{\max_p - \min_p} \right) \tag{8}$$

where x_p is the pth attribute's value in the instance x. $\max_p$ and $\min_p$ correspond, respectively, to maximum and minimum value of the Pth attribute in the dataset.

The results of multiple experiments under various datasets are compared in terms of all the evaluation metrics mentioned in the previous subsection. Hereafter, we present, compare and discuss the results of our algorithm against Naïve Bayes classifier, K-means clustering and One-class SVM algorithm.

5.2 True positive rate (TPR)/false positive rate (FPR)

As reported in Tables 10, 11 and 12, both versions of NSNAD exhibit better performances compared to the other algorithms. Indeed, their average TPR is 93%, 94% and 86% with an average FPR of 1%, 2% and 6% when it is tested under NSL-KDD, Kyoto2006+ and UNSW-NB15 datasets, respectively. These values demonstrate the efficiency of NSNAD to detect most attacks present in the dataset. The detection rate is further improved with the A-HLA optimization, although FPR is slightly increased because of new normal behaviors (instances) that are not represented in the train.

As for Naïve Bayes classifier and K-means clustering when tested under NSL-KDD, the former displays a relatively better detection rate ($\sim$84%) than the latter ($\sim$78%). However, the FPR of K-means is much lower ($\sim$0.5%) than NB ($\sim$9%). The main reason is not only the difference in probability distribution between train and test data, but also the unbalanced type of attacks in both sets. When it comes to UNSW-NB15 dataset, the DR is almost the same ($\sim$65%), yet the FPR of K-means is nearly six times higher ($\sim$29%) than NB's ($\sim$5%), which can be explained by the supervised nature of NB classifier and the difficulty level of the dataset for unsupervised techniques. The results of NB and K-means under Kyoto2006+ dataset are thought not different. Indeed, the average TPR and the average FPR are 85% and 1.5% for both algorithms.

However, the results of One-class SVM classifier under all test subsets are very bad, especially the false positive rates that reach the 48%, 45% and 70%. This can be explained by the non-exhaustive representation of the normal class in the train data as the algorithm depends entirely on the latter for the classification.

5.3 ROC and AUC

Roc (Receiver Operating Characteristic) curves in Fig. 8 depict the relationship and visualize the detection rates

Table 10 TPR/FPR results under NSL-KDD dataset

Sub.	NSNAD without HLA		NSNAD with HLA		NB		*K*-means		One-class SVM	
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
1	84.11	0.19	94.37	1.42	82.7	9.7	77.96	0.29	98.71	48.63
2	84.21	0.21	95.33	1.46	83.3	9.6	78.3	0.3	98.73	48.83
3	87.4	0.37	95.56	1.62	83.4	9.5	78.32	0.31	98.67	48.27
4	91.32	0.9	96.16	1.7	83.5	9.4	78.35	0.33	98.67	48.11
5	91.6	1.01	96.37	1.72	83.4	9.4	78.39	0.39	98.67	48.34
6	93.02	1.2	96.43	1.78	83.5	9.3	78.39	0.42	98.66	48.43
7	93.1	1.26	96.45	1.82	83.5	9.2	78.41	0.44	98.65	48.38
8	93.31	1.43	96.5	1.83	83.5	9.1	78.46	0.5	98.62	48.43
9	93.7	1.77	96.58	1.91	83.5	9.1	78.49	0.54	98.62	48.49
10	93.09	1.92	96.59	2.09	83.6	9	78.54	0.58	98.57	48.30

Table 11 TPR/FPR results under Kyoto2006+ dataset

Sub.	NSNAD without HLA		NSNAD with HLA		NB		*K*-means		One-class SVM	
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
1	97.42	4.74	99.04	3.5	66.56	1.73	50.56	0.73	100	48.78
2	88.55	0.31	98.78	2.84	71.24	1.78	61.24	0.78	100	46.17
3	92.22	4.74	99.4	3.8	73.46	1.82	63.46	0.82	100	48.41
4	96.3	0.55	98.79	2.91	81.66	1.94	81.66	1.74	100	44.45
5	94.31	2.43	98.69	2.69	99.52	3.42	90.19	2	100	44.52
6	92.35	0.59	98.1	1.47	99.33	3	86.23	1.9	100	41.99
7	95.29	0.58	98.39	1.7	98	2.4	68.31	0.99	100	45.40
8	90.39	0.34	97.35	1.32	99.15	2.8	72.93	1.3	100	41.60
9	94.5	1.06	98.75	2.76	99.24	2.95	82.9	1.78	100	40.04
10	96.98	0.78	98.59	2.56	99.46	3.24	85.38	1.83	100	39.42

Table 12 TPR/FPR results under UNSW-NB15 dataset

Sub.	NSNAD without HLA		NSNAD with HLA		NB		*K*-means		One-class SVM	
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
1	69.8	3.2	85.6	5.7	65.5	5.1	66.43	29.39	85.41	75.12
2	72.4	3.45	90.18	8.2	65.1	5.2	65.96	29.57	84.80	75.56
3	76.3	4.11	90.5	8.51	64.8	5.3	65.64	29.74	84.69	75.71
4	78.66	5.25	91.7	9.2	64.9	5.3	65.72	29.67	84.76	75.70
5	82.17	5.9	91.8	10.01	64.9	5.4	65.66	29.74	84.90	75.78
6	83.63	6.31	91.9	10.3	65	5.5	65.71	29.75	84.91	75.79
7	86.51	7.06	92.01	10.6	65	5.5	65.72	29.86	84.86	75.87
8	87.32	7.87	92.5	11.02	65	5.5	65.73	29.99	84.90	75.83
9	87.6	8.08	93.4	11.8	65	5.5	65.75	29.96	84.91	75.76
10	87.82	8.5	93.8	12.1	65	5.7	65.79	30.14	84.92	75.60

versus the false positive rates as the classifier decision threshold is varied. The test sets used in the evaluation for each dataset are: the full NSL-KDD dataset; the records of January 30, 2009, of Kyoto2006+ dataset; and the complete test set of UNSW-NB15.

Meanwhile, the area under the ROC curve (AUC) metric demonstrates the trade-offs between TPR and FPR. A higher AUC value indicates a high TPR and low FPR, which is the goal of intrusion detection algorithms.

AUC values reported in the plots highlight the algorithm that performs the best. Again, NSNAD with A-HLA optimization displays the best results with up to 0.995 on average. The worst values are recorded by One-class SVM classifier, especially under UNSW-NB15 since it is the

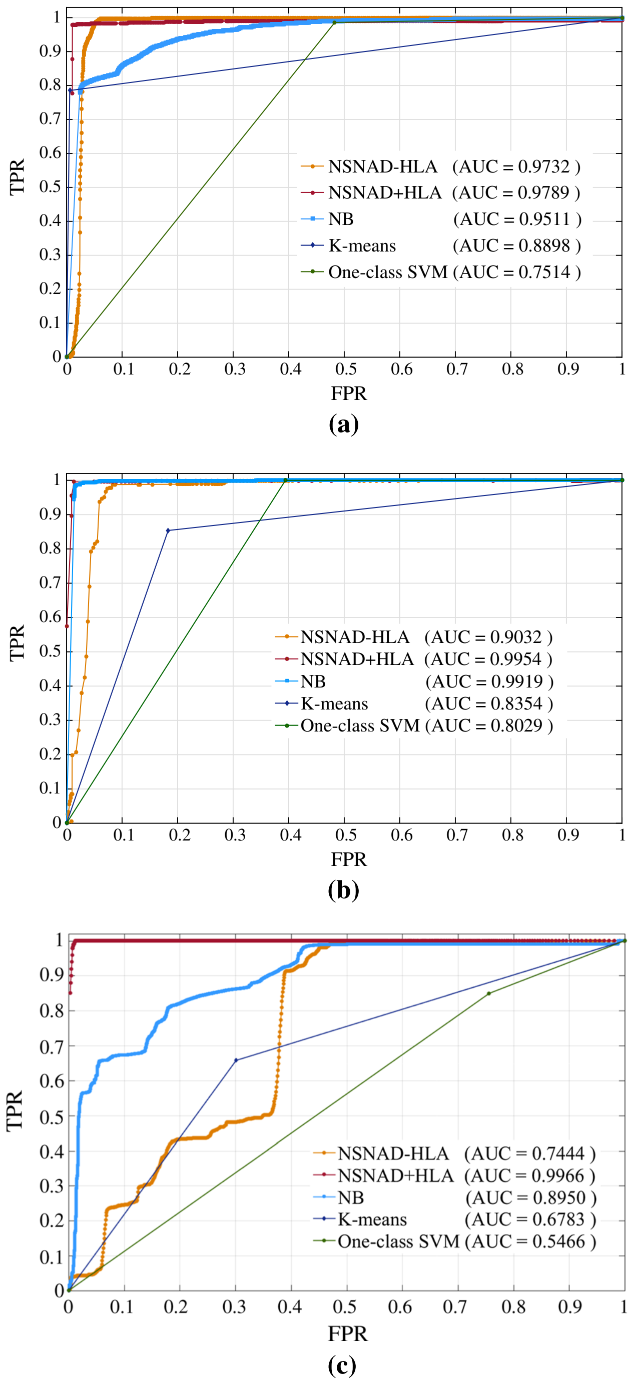

Fig. 8 ROC results under **a** NSL-KDD, **b** Kyoto2006+ and **c** UNSW-NB15 datasets

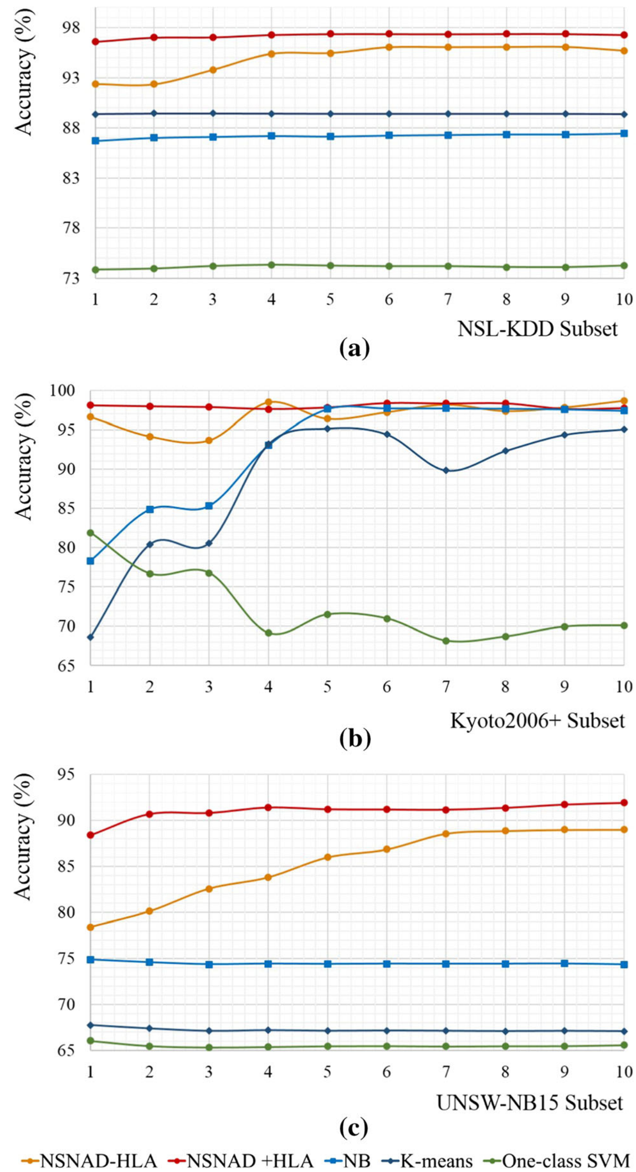

Fig. 9 Accuracy results under **a** NSL-KDD, **b** Kyoto2006+ and **c** UNSW-NB15 datasets

most challenging dataset. AUCs of NB and *K*-means algorithms reach around 0.90 and around 0.8, respectively.

5.4 Accuracy and precision results

The accuracy of a classifier is a global measure that reflects the probability of correctly classified records over the complete data, regardless of a specific class. The precision, on the other hand, is the probability of correctly detected anomalies over the original amount of anomalies in the data. A classifier is said to be good if it is accurate and precise.

As shown in Fig. 9, NSNAD+HLA is the most accurate algorithm among the five (NSNAD without HLA included). Indeed, the rate of correctly classified samples is up to 97% with a precision as high as around 98% (Fig. 10). As it appears in the graphs, the A-HLA optimization enhanced the accuracy since it performs an additional verification before the final classification. The improvement is even more pronounced under NSL-KDD and UNSW-NB15 datasets.

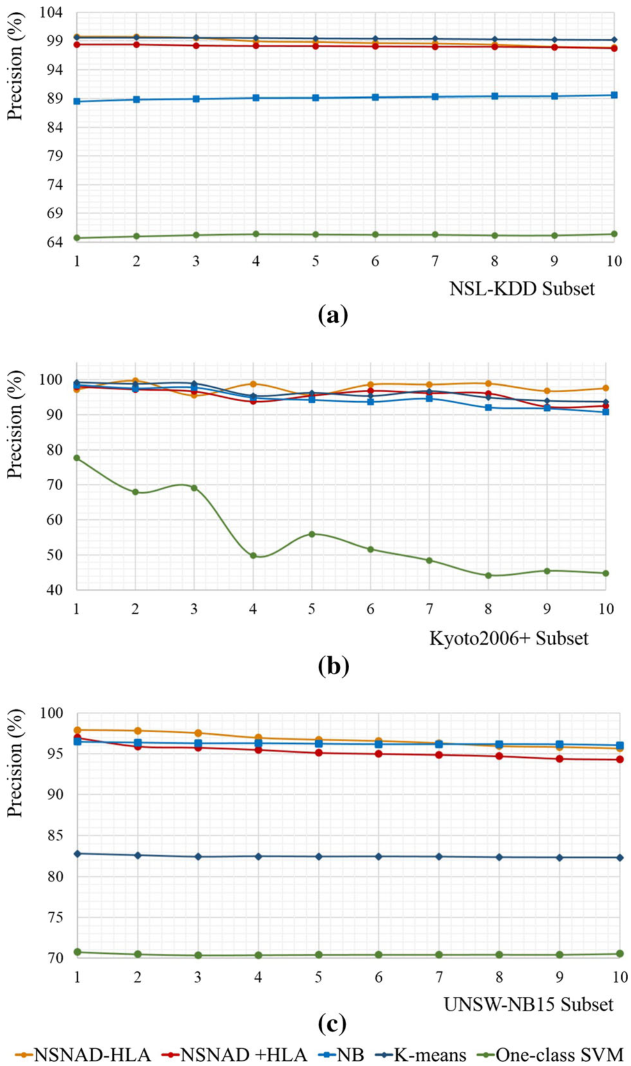

Fig. 10 Precision results under **a** NSL-KDD, **b** Kyoto2006+ and **c** UNSW-NB15 datasets

One-class SVM, on the other hand, is the least accurate algorithm, especially under NSL-KDD and Kyoto2006+ datasets; its rate of accuracy and precision are around 70%.

K-means and NB perform broadly the same when it comes to accuracy. Yet, *K*-means exhibits a better precision than the other algorithms under NSL-KDD and Kyoto2006+ datasets due to its low FPR when tested under these datasets.

5.5 *F*-measure results

This metric considers both precision and TPR. It reflects the overall performance of the classifier to correctly identify anomalies. Figure 11 depicts the *F*-measure results of the four compared algorithms (NSNAD, NB, *K*-means and One-class SVM) plus the version of our proposed algorithm without the A-HLA step. It clearly shows that not only our algorithm outperforms the others with values up to 97%, but also point out the contribution of A-HLA step in achieving better results.

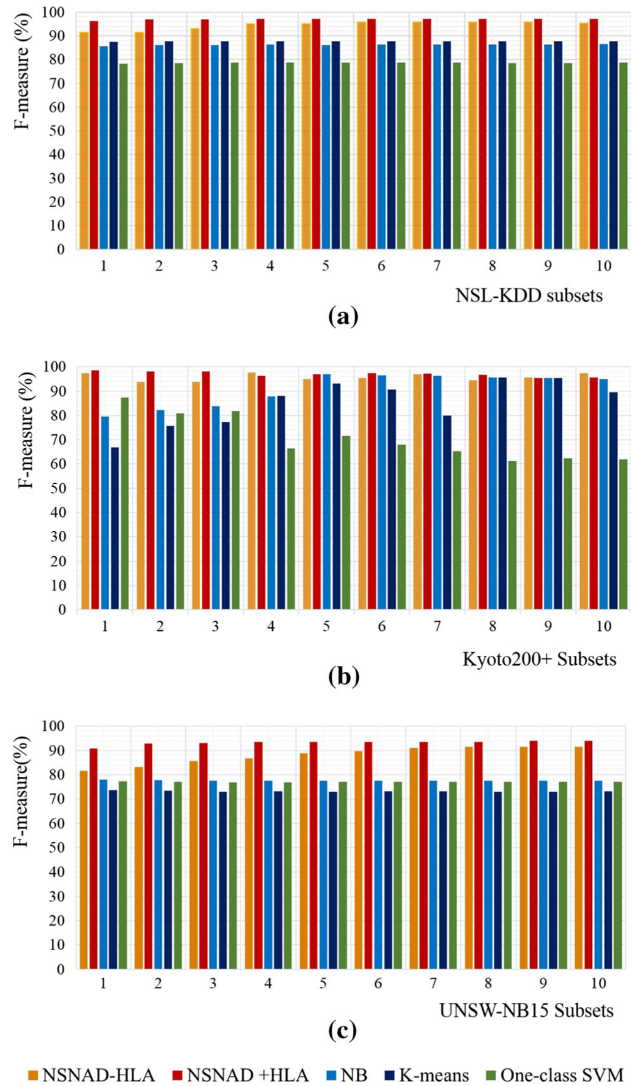

Fig. 11 *F*-measure results under **a** NSL-KDD, **b** Kyoto2006+ and **c** UNSW-NB15 datasets

5.6 Matthews correlation coefficient (MCC)

This coefficient was introduced by Brian W. Matthews in 1975 [59] to evaluate the quality of a binary classification. Contrary to *F*-measure, MCC takes into account TP, FP, TN and FN. Indeed, the *f*-measure results are dependent on one class (in this case anomaly). If we consider normal class, the f-measures' values change accordingly. MCC considers the two cases at the same time and involves all the classifier outputs. The comparative results given in Fig. 12 show MCC values regarding each algorithm under the chosen benchmark dataset. Here again, NSNAD results outperform those of Naive Bayes, *K*-means and One-class

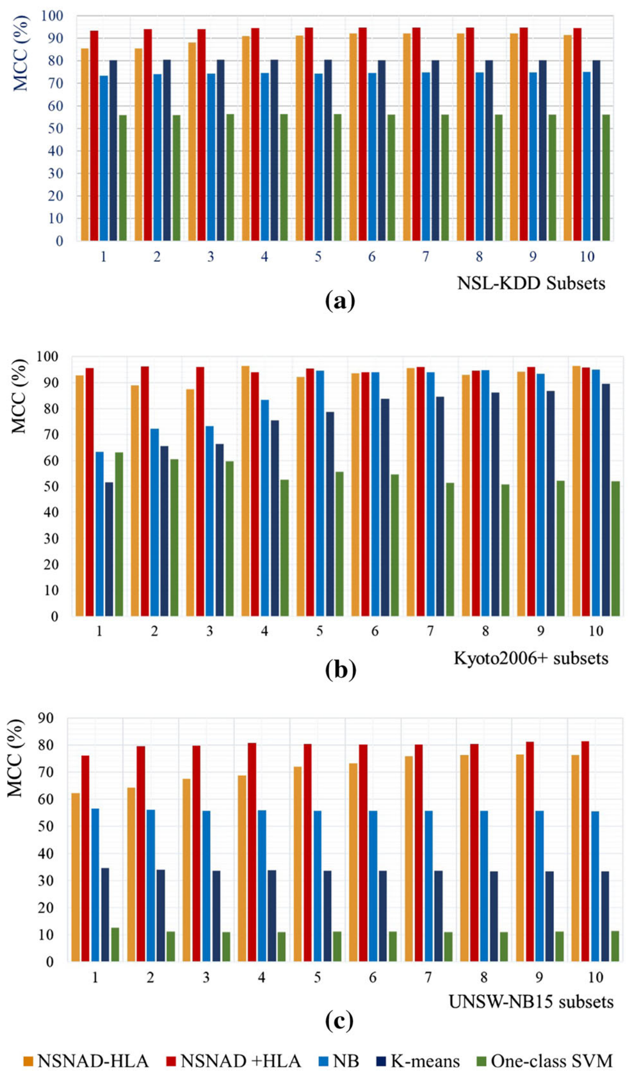

Fig. 12 MCC results under **a** NSL-KDD, **b** Kyoto2006+ and **c** UNSW-NB15 datasets

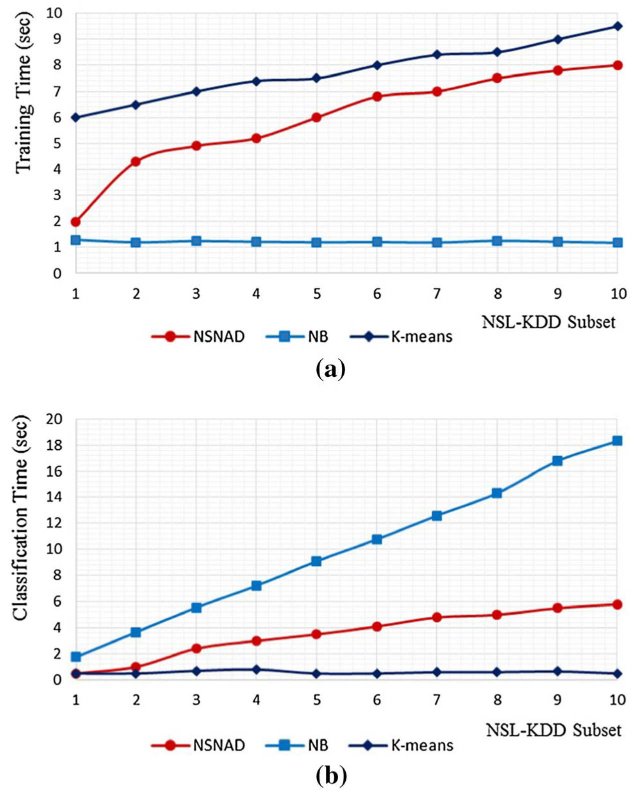

Fig. 13 Computational time of **a** training and **b** classification under NSL-KDD

SVM algorithms under every dataset. It also highlight that NSNAD with HLA verification gives slightly better results compared to its version without optimization.

5.7 Computational time

As for the overall computational time of the proposed approach, it varies from 1 to 16 s ($T(train) + T(classif)$). The training phase involves the creation and the validation of the detectors as well as the computation of their radius. The classification phase includes the activation of the detectors and the A-HLA verification. The results, as depicted in Figs. 13, 14 and 15, represent the average time incurred by each algorithm in both training and classification stages after 10 runs.

The overall observation that we could point out from these figures is that the computational time of our approach slightly increases along with the size of the test data. Indeed, as the number of detectors depends on the size of test data, the time spent during the training is more important when the test data are large. Nevertheless, the mentioned time does not exceed 8 s, 2.8 s and 9 s under NSL-KDD, Kyoto2006+ and UNSW-NB15 datasets, respectively. The computational time of the classification step, on the other hand, depends on the detectors' representation of the nonself-space. In fact, the best case (for each instance) is that the incoming instance activates the first detector to be classified as anomaly which, more likely, has been the case under Kyoto2006+ dataset with a maximum classification time of 4 s. The worst case is that after looping over all the detectors, none of them was activated so A-HLA verification is executed. This worst case have likely been seen under UNSW-NB15 dataset with maximum classification time of 7 s.

The computational time of One-class SVM has not been reported in the graphics due to their high values. Indeed, the mean time that this algorithm spends on the training/testing phases are about 16mn/19mn, 5mn/9mn and 3mn/7mn.

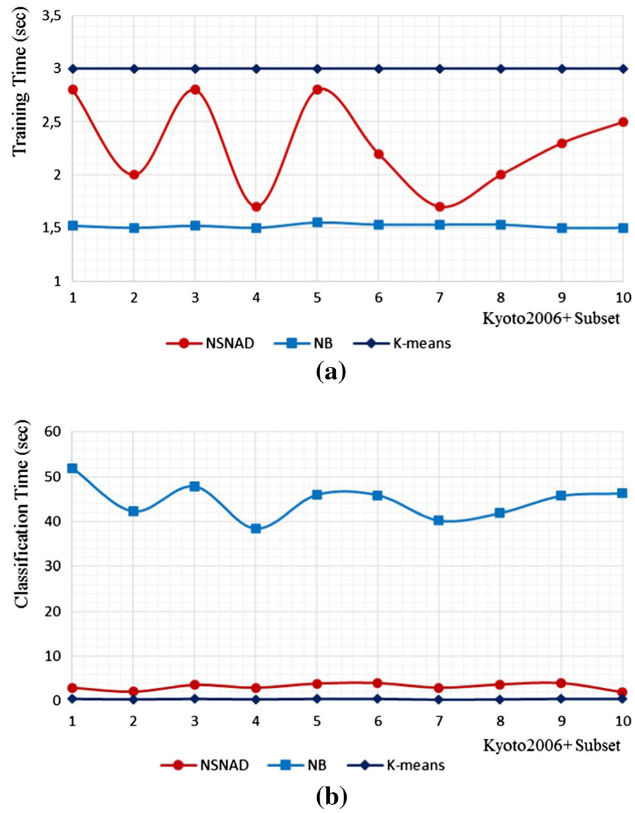

Fig. 14 Computational time of **a** training and **b** classification under Kyoto2006+

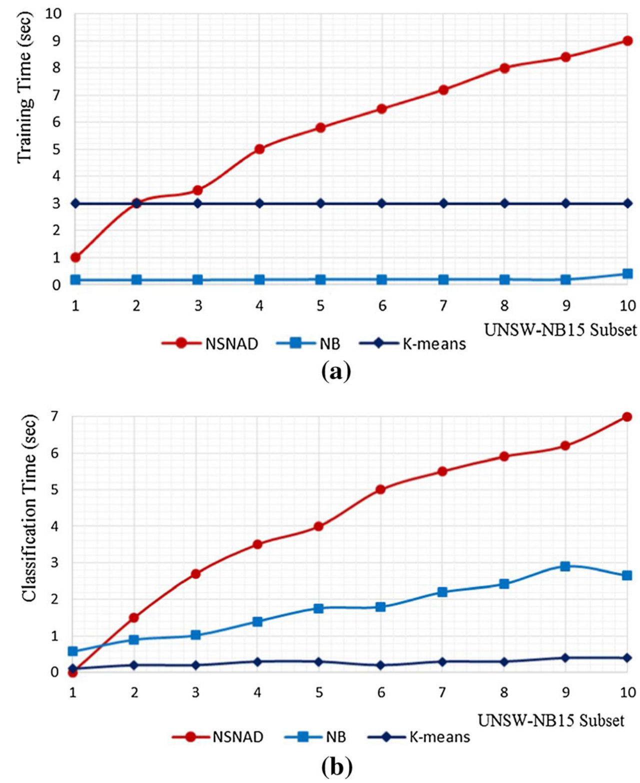

Fig. 15 Computational time of **a** training and **b** classification under UNSW-NB15

5.8 Summary of results

In order to summarize the overall results of NSNAD compared with the baseline algorithms, Table 13 provides the figures of each metric for algorithm under the benchmark datasets we used in our experiments. The results are for the complete NSL-KDD dataset, January 30, 2009, of Kyoto2006+ dataset and the complete test set of UNSW-NB15.

6 Comparison with AIS-based intrusion detection techniques

For comparison purposes, Table 14 summarizes the performance of some recent AIS-based intrusion detection techniques. As can be seen, a large number of these researches used KDDCup99 dataset for their experiments [18, 19, 60, 79, 83, 89–89, 93, 97, 98], few of them used NSL-KDD dataset [30, 43, 60] and others used private [60] or non-networ*K*-related [19, 49, 50] datasets. To the best of our knowledge, our work is the only artificial immune-based approach that uses UNSW-NB15 dataset for test experiments. In [33], UNSW-NB15 is used to test their approach based on artificial neural network to detect the attack in cloud infrastructures. In [7], these data are used to evaluate an ontology-based multi-agent model IDS for detection Web service attacks. Others like [2] used it along with KDDCup99 and NSL-KDD to test data mining techniques for intrusion detection in network traffic.

The true positive rate of most techniques under KDDCup99 dataset reaches high values (from 71.5 to 99.2%) with low values of false positive rate (from 0 to 3%). However, these figures no longer reflect the detection in nowadays networks because the database used is nearly 20 years old. It sure gives a comparative point of view against early papers, but researchers should validate their approaches with more recent datasets in addition to the old benchmarks.

Another important fact to point out from the table is the lack of techniques' computational time or time complexity reported in a large number of papers. For instance, in [98], authors reported only the modeling time of their improved clonal algorithm before and after the feature selection without mentioning the settings of the machine they experimented on.

The distributed network intrusion detection system presented in [43] uses a genetic algorithm to generate detectors, yet authors did not present an estimation for time complexity knowing that GA is usually time-consuming algorithms as many previous works mentioned [42, 58, 84].

Table 13 Summary of results

Dataset	Algorithm	TPR (%)	FPR (%)	AUC (%)	Acc (%)	Prec (%)	*F*-meas(%)	MCC (%)	Computational time (s)	
									Training	Classification
NSL-KDD (Full dataset)	NSNAD+HLA	96.59	2.09	97.89	97.27	97.72	97.15	94.54	8	5.8
	NB	83.6	9	95.11	87.44	89.6	86.49	74.93	1.18	18.3
	K-means	78.54	0.58	88.98	89.37	99.21	87.67	80.21	9.5	0.5
	One-class SVM	98.57	48.3	74.17	74.25	65.43	78.65	56.24	927.17	1818.97
Kyoto 2006+ (30 Jan 2009)	NSNAD+HLA	98.59	2.56	99.54	97.71	92.47	95.43	95.76	2.5	2
	NB	99.46	3.24	99.19	97.41	90.74	94.89	95.03	1.5	46.27
	K-means	85.38	1.83	83.54	95.07	93.70	89.35	89.52	3	0.5
	One-class SVM	100	39.42	80.29	70.12	44.74	61.82	52.06	304.3	554.21
UNSW-NB15 (Test set)	NSNAD+HLA	93.8	12.1	99.66	91.91	94.29	94.04	81.46	9	7
	NB	65	5.7	89.50	74.35	96.04	77.53	55.46	0.4	2.65
	K-means	65.79	30.14	67.83	67.09	82.30	73.13	33.37	3	0.4
	One-class SVM	84.91	75.60	54.66	65.58	70.53	77.06	11.28	152.79	713.67

Another technique combining GA and AIS is presented in [88] without time complexity reported. The authors used genetic search for correlation-based feature selection and have proposed an AIS-based classifier to classify attacks over the selected features. Both papers did not mention the CPU/RAM settings of the machine used in experiments.

Authors in [1] claimed that using RST to reduce features of GureKDDcup dataset would enhance the computational time of the algorithm, but they did not present any comparative results to support their claim, where others like [93] state that their clonal selection-based method achieves high detection rate with low complexity time without showing any results regarding the second criterion. Another clonal selection-based technique was proposed in [49] where authors experimented on three datasets varying detectors sample size and antigen sample size to investigate their effects on TPR and FPR but not on the computational time.

Experiments of the probabilistic AIS-based IDS of [60] were conducted under six different datasets, from which two were generated by the authors. We reported in the table the last version's results in terms of TPR, FPR and computational time. It is worth noting that the computational time is based on 2 s analysis, which means that the table's figures correspond to the time that the algorithm spends on classifying 2sec of monitored traffic. The figures mentioned in the table are the algorithm's time to classify 2 s of monitored traffic. In addition, authors fail to mention the actual time cost of their algorithm under KddCup99 and NSL-KDD datasets. They rather present a speedup ratio against that of SPAI, the first version of their algorithm. Although, they stated that the computational cost of SPAI is six times higher than that of RVNS (real-valued negative selection) and four times higher than that of PS (positive selection).

In this work, we evaluated our method under old benchmarks (NSL-KDD, Kyoto2006+) as well as the newest one, namely UNSW-NB15. We discuss and compare the process against two other algorithms regarding multiple metrics including TPR and FPR whose values are as good as other papers summarized in the table. Furthermore, we report training and classification computational time of each algorithm compared in this paper under each dataset.

7 Conclusion and future work

The results presented in this paper clearly outline the efficiency provided by the artificial immune system to the field of intrusion detection. Indeed, we proposed a classification process based on the negative selection theory. This process begins with a reduction in the space through feature selection using the Coefficient of Variation to determine the least dispersed attributes. Detectors are subsequently generated and validated with respect to the normal (self-) data to cover the nonself-space via their radius. The classification we proposed exploits, at first, the radius of these detectors to classify an incoming instance as "anomaly". Then, an additional verification with the volume of "self" is performed in order to classify the instance as "normal". We called this latter, an Artificial-HLA optimization. The validation experiments of our process using old benchmark as well as up-to-date and challenging

Table 14 Comparison with AIS-based intrusion detection techniques

References	Tech.	TPR (%)	FPR (%)	Dataset	CT (Sec)		Settings	
					Tr.	Cl.	CPU	RAM
Our method (NSNAD)	NS and CV-based FS	96	1.7	NSL-KDD	8	5.8	2.40 GHz Intel® CoreTM i7	8 GB
		98.5	2.5	Kyoto2006+	2.8	4		
		91.5	9	UNSW-NB15	9	7		
[91]	NS, PS	99.1	1.99	KDDCup99	N/A	N/A	N/A	N/A
[80]	CIA, CS	69.5	N/A	KDDCup99	N/A	N/A	N/A	N/A
[99]	CS	98.7	0.15	KDDCup99	13.84	N/A	N/A	N/A
[89]	DCA and IG FS	71.5	0.0	KDDCup99	900–1200		N/A	N/A
[44]	NS, GA and IG FS	98.9	1.7	NSL-KDD(Tr.20%)	N/A	4	N/A	N/A
[21]	AIS, PBIL and CF	93.24		KDDCup99	N/A	N/A	N/A	N/A
		86.23		Australian credit card	N/A	N/A		
[98]	CS	99.2	0.2	KDDCup99	N/A	N/A	N/A	N/A
[31]	NS	96.2	7.5	NSL-KDD	1100–1700	N/A	3.0 GHz Intel® Core™ i5	4 GB
[3]	AIS and RS	90	N/A	GureKDDCup	N/A	N/A	N/A	N/A
[101]	Vaccination and RS	97.07	1.19	KDDCup99	$O(N_a N_s L_1)$[a]		N/A	N/A
[61]	Self/nonself and GMM	97	20	Darpa	N/A	3 ms	PentiumIV 2 GHz	1 GB
		98	3	KDDCup99	N/A	$T_1/2$[b]		
		95	30	NSL-KDD	N/A	$T_1/5$		
		96	20	DataSetMe	N/A	$50\mu s$		
		98	10	IUSTSip	N/A	0.7 ms		
		98	10	INRIASip	N/A	0.4 ms		
[20]	CS	94.6	5.4	KDDCup99	N/A	N/A	2.13 GHz Intel® Core™i3	N/A
[90]	AIS and GA	98.6	1.3	KDDCup99	N/A	N/A	N/A	N/A
[84]	NS and RS FS	98.25	1	KDDCup99	N/A	N/A	N/A	N/A
[94]	CS and fuzzy clustering	90	9	KDDCup99	N/A	N/A	N/A	N/A
[33]	NS	98	5	DARPA98	N/A	N/A	N/A	N/A
		95	2	DARPA99	N/A	N/A		
[51]	CS	90	5	Breast cancer	N/A	N/A	N/A	N/A
[50]	CS and NS	95	5	Cancer	N/A	N/A	N/A	N/A
		92	3	Vote	N/A	N/A		
		95	1	Iris	N/A	N/A		

Tech. Technique, *CT.* computational time, *Tr.* training, *Cl.* classification, *NS* negative selection, *PS* positive selection, *CIA* coevolutionary immune algorithm, *CS* clonal selection, *CV* coefficient of variation, *FS* feature selection, *IG* info gain, *RS* rough set, *DCA* dendritic cell algorithm, *PBIL* population-based incremental learning, *CF* collaborative filtering, *GMM* Gaussian mixture model, *N/A* not available

[a]The complexity time of the algorithm with: N_a number of antigens, N_s, number of self-antibodies, L_1 length of detectors.

[b]T_1 = the response time of SPAI, the first version of the authors' algorithm

datasets show a promising tradeoff between detection rate, false positive rate and computational time compared to other algorithms.

There are many potential and interesting future works, including:

- Improve the detectors generation process and their distribution over the nonself-space in order to balance between the detectors' number, their coverage and minimize the overlapping of their radius.
- Optimize computational time through parallelism.
- Compare NSNAD to other methods as: subspace clustering [57, 101].
- Analyze the resulting clusters through postprocessing techniques, such as quantification-based technique for cluster analysis [45].

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Appendix: Feature description

See Tables 15, 16 and 17.

Table 15 NSL-KDD attributes

n	Attribute	Description
1	Duration	Length (number of s) of the connection
2	protocol_type	Type of the protocol
3	service	Network service on the destination
4	flag	Status of the connection – Normal or Error
5	src_bytes	Number of data bytes from source to destination
6	dst_bytes	Number of data bytes from destination to source
7	land	If source and destination IP addresses and port numbers are equal, then this variable takes value 1, 0 otherwise
8	wrong_fragment	Number of wrongn fragments
9	urgent	Number of urgent packets
10	hot	Number of "hot" indicators
11	num_failed_login	Logins number of failed logins
12	logged_in	Takes 1 if successfully logged in, 0 otherwise
13	num_compromised	Number of "compromised" conditions
14	root_shell	Number of "root" accesses
15	su_attempted	Takes 1 if "su root" command attempted; 0 otherwise
16	num_root	Number of "root" accesses
17	num_file_creations	Number of file creation operations.
18	num_shell	Number of shell prompts
19	num_access_files	Number of operations on access control files.
20	num_outbound_cmds	number of outbound commands in an ftp session
21	is_host_login	Takes 1 if the login belongs to the root or admin list, 0 otherwise
22	is_guest_login	Takes 1 if the login is a "guest" login, 0 otherwise
23	Count	Number of connections to the same destination host as the current connection in the past 2 s
24	srv_count	Number of connections to the same service as the current connection in the past 2 s.
25	Serror_rate	The percentage of connections that have activated the flag s0, s1, s2 or s3, among the connections aggregated in count
26	srv_serror_rate	The percentage of connections that have activated the flag s0, s1, s2 or s3, among the connections aggregated in srv_count
27	rerror_rate	The percentage of connections that have activated the flag REJ, among the connections aggregated in count
28	srv_rerror_rate	The percentage of connections that have activated the flag REJ, among the connections aggregated in srv_count
29	same_srv_rate	The percentage of connections that were to the same service
30	diff_srv_rate	The percentage of connections that were to different services, among the connections aggregated in count
31	srv_diff_host_rate	The percentage of connections that were to different destination machines among the connections aggregated in srv_count
32	dst_host_count	Number of connections having the same destination host IP address
33	dst_host_srv_count	Number of connections having the same port number.
34	dst_host_same_srv_rate	The percentage of connections that were to the same service, among the connections aggregated in dst_host_count
35	dst_host_diff_srv_rate	The percentage of connections that were to different services, among the connections aggregated in dst_host_count
36	dst_host_same_src_port_rate	The percentage of connections that were to the same source port, among the connections aggregated in dst_host_srv_count
37	dst_host_srv_diff_host_rate	The percentage of connections that were to different destination machines, among the connections aggregated in dst_host_srv_count

Table 15 (continued)

n	Attribute	Description
38	dst_host_serror_rate	The percentage of connections that have activated the flag s0, s1, s2 or s3, among the connections aggregated in dst_host_count
39	dst_host_srv_serror_rate	The percent of connections that have activated the flag s0, s1, s2 or s3, among the connections aggregated in dst_host_srv_count
40	dst_host_rerror_rate	The percentage of connections that have activated the flag REJ, among the connections aggregated in dst_host_count
41	dst_host_srv_rerror_rate	The percentage of connections that have activated the flag REJ, among the connections aggregated in dst_host_srv_count
42	class	Indicates if the connection is "normal" or "attack"

Table 16 Kyoto2006+ attributes

n	Attribute	Description
1	duration	The length (number of s) of the connection
2	service	The connection's service type
3	src_bytes	The number of data bytes sent by the source IP address
4	dst_bytes	The number of data bytes sent by the destination IP address
5	Count	The number of connections that their source IP and destination IP address are the same to those of the current connection in the past 2 s
6	same_srv_rate	The percentage of connections to the same service in Count feature
7	serror_rate	The percentage of connections that have "SYN" errors in Count feature
8	srv_serror-rate	The percentage of connections that have "SYN" error in Srv_count feature
9	dst_host_count	Among the past 100 connections whose destination IP address is the same to that of the current connection, the number of connections whose source IP address is also the same to that of the current connection
10	dst_host_srv_Count	Among the past 100 connections whose destination IP address is the same to that of the current connection, the number of connections whose service type is also the same to that of the current connection
11	dst_host_same_src_port_rate	The percentage of connections whose source port is the same to that of the current connection in dst_host_count feature
12	dst_host_count	Among the past 100 connections whose destination IP address is the same to that of the current connection, the number of connections whose source IP address is also the same to that of the current connection
13	dst_host_srv_serror_rate	The percentage of connections that have "SYN" errors in Dst_host_srv_count
14	flag	The state of the connection at the time it was written
15	source_port_number	The source port number used in the session
16	destination_port_number	Indicates the destination port number used in the session
17	class	Indicates whether the session was attack ('1') or not ('0')

Table 17 UNSW-NB15 attributes

n	Attribute	Description
1	id	Connection identification
2	dur	Record total duration
3	proto	Transaction protocol
4	service	http, ftp, ssh, dns ..,else (–)
5	state	The state and its dependent protocol, e.g., ACC, CLO, else (–)
6	spkts	Source to destination packet count
7	dpkts	Destination to source packet count
8	sbytes	Source to destination bytes
9	dbytes	Destination to source bytes
10	rate	
11	sttl	Source to destination time to live
12	dttl	Destination to source time to live
13	sload	Source bits per second
14	dload	Destination bits per second
15	sloss	Source packets transmitted or dropped
16	dloss	Destination packets transmitted or dropped
17	sintpkt	Source inter-packet arrival time (ms)
18	dintpkt	Destination inter-packet arrival time (ms)
19	sjit	Source jitter (ms)
20	djit	Destination jitter (ms)
21	swin	Source TCP window advertisement
22	stcpb	Source TCP sequence number
23	dtcpb	Destination TCP sequence number
24	dwin	Destination TCP window advertisement
25	tcprtt	The sum of "synack" and "ackdat" of the TCP
26	synack	The time between the SYN and the SYN_ACK packets of the TCP
27	ackdat	The time between the SYN_ACK and the ACK packets of the TCP
28	smeansz	Mean of the flow packet size transmitted by the src
29	dmeansz	Mean of the flow packet size transmitted by the dst
30	trans_depth	the depth into the connection of http request/response transaction
31	response_body_len	The content size of the data transferred from the server's http service
32	ct_srv_src	No. of connections that contain the same service and source address in 100 connections according to the last time
33	ct_state_ttl	No. for each state according to specific range of values for source/destination time to live
34	ct_dst_ltm	No. of connections of the same destination address in 100 connections according to the last time
35	ct_src_dport_ltm	No of connections of the same source address and the destination port in 100 connections according to the last time
36	ct_dst_sport_ltm	No of connections of the same destination address and the source port in 100 connections according to the last time
37	ct_dst_src_ltm	No of connections of the same source and the destination address in 100 connections according to the last time
38	is_ftp_login	If the ftp session is accessed by user and password then 1, 0 otherwise
39	ct_ftp_cmd	No of flows that has a command in ftp session
40	ct_flw_http_mthd	No. of flows that has methods such as Get and Post in http service
41	ct_src_ltm	No. of connections of the same source address in 100 connections according to the last time
42	ct_srv_dst	No. of connections that contain the same service and destination address in 100 connections according to the last time
43	is_sm_ips_ports	If source IP equals to destination IP and port numbers are equal, this variable takes value 1; 0 otherwise
44	class	The name of each attack category. In this dataset, nine categories (e.g., Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms)
45	label	0 for normal and 1 for attack records

References

1. Abas EAER, Abdelkader H, Keshk A (2015) Artificial immune system based intrusion detection. In: 2015 IEEE seventh international conference on intelligent computing and information systems (ICICIS), pp 542–546. Institute of Electrical & Electronics Engineers (IEEE). https://doi.org/10.1109/intelcis.2015.7397274
2. Agrawal A, Mohammed S, Fiaidhi J (2016) Developing data mining techniques for intruder detection in network traffic. Int J Secur Appl 10(8):335–342. https://doi.org/10.14257/ijsia.2016.10.8.29
3. Al-Enezi J, Abbod M, Alsharhan S (2010) Artificial immune systems-models, algorithms and applications. http://www.arpapress.com/Volumes/Vol3Issue2/IJRRAS_3_2_01.pdf
4. Ambusaidi MA, He X, Nanda P, Tan Z (2016) Building an intrusion detection system using a filter-based feature selection algorithm. IEEE Trans Comput 65(10):2986–2998. https://doi.org/10.1109/TC.2016.2519914
5. Amer SH, Hamilton J (2010) Intrusion detection systems (ids) taxonomy-a short review. Def Cyber Secur 13(2):23–30
6. Ammar A (2015) Comparison of feature reduction techniques for the binominal classification of network traffic. J Data Anal Inf Process 3(02):11. https://doi.org/10.4236/jdaip.2015.32002
7. Anusha K, Sathiyamoorthy E (2016) Omamids: ontology based multi-agent model intrusion detection system for detecting web service attacks. J Appl Secur Res 11(4):489–508. https://doi.org/10.1080/19361610.2016.1211847

8. Axelsson S (2000) Intrusion detection systems: a survey and taxonomy. Report, Technical report
9. Bahl S, Sharma SK (2016) A minimal subset of features using correlation feature selection model for intrusion detection system. In: Proceedings of the second international conference on computer and communication technologies, pp 337–346. Springer. https://doi.org/10.1007/978-81-322-2523-2_32
10. Bethi SK, Phoha VV, Reddy YM (2004) Clique clustering approach to detect denial-of-service attacks. In: Proceedings from the fifth annual IEEE SMC information assurance workshop 2004, pp 447–448. https://doi.org/10.1109/iaw.2004.1437856
11. Bhuyan M, Bhattacharyya D, Kalita J (2014) Network anomaly detection: methods, systems and tools. Commun Surv Tutor IEEE 16(1):1–34
12. Brownlee J (2011) Clever algorithms: nature-inspired programming recipes. Jason Brownlee
13. Buitinck L, Louppe G, Blondel M, Pedregosa F, Mueller A, Grisel O, Niculae V, Prettenhofer P, Gramfort A, Grobler J, Layton R, VanderPlas J, Joly A, Holt B, Varoquaux G (2013) API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD workshop: languages for data mining and machine learning, pp 108–122
14. Burges CJ (1998) A tutorial on support vector machines for pattern recognition. Data Min Knowl Disc 2(2):121–167. https://doi.org/10.1023/a:1009715923555
15. de Castro L, Zuben FV (2002) Learning and optimization using the clonal selection principle. IEEE Trans Evol Comput 6(3):239–251. https://doi.org/10.1109/tevc.2002.1011539
16. de Castro LN, Timmis JI (2003) Artificial immune systems as a novel soft computing paradigm. Soft Comput 7(8):526–544
17. Cemerlic A, Yang L, Kizza JM (2008) Network intrusion detection based on bayesian networks. In: SEKE, pp 791–794
18. Chan FT, Prakash A, Tibrewal R, Tiwari M (2013) Clonal selection approach for network intrusion detection. In: Proceedings of the 3rd international conference on intelligent computational systems (ICICS'2013), Singapore, pp 1–5
19. Chen MH, Chang PC, Wu JL (2016) A population-based incremental learning approach with artificial immune system for network intrusion detection. Eng Appl Artif Intell 51:171–181. https://doi.org/10.1016/j.engappai.2016.01.020
20. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297. https://doi.org/10.1007/bf00994018
21. Crosbie M, Spafford G (1995) Applying genetic programming to intrusion detection. In: Working notes for the AAAI symposium on genetic programming, pp 1–8. MIT Press, Cambridge
22. DasGupta D (1993) An overview of artificial immune systems and their applications. In: Artificial immune systems and their applications, pp 3–21. Springer
23. Dasgupta D, Nino F (2008) Immunological computation: theory and applications. CRC Press, Boca Raton
24. Dasgupta D, Yu S, Nino F (2011) Recent advances in artificial immune systems: models and applications. Appl Soft Comput 11(2):1574–1587. https://doi.org/10.1016/j.asoc.2010.08.024
25. Dhanabal L, Shantharajah S (2015) A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. Int J Adv Res Comput Commun Eng 4(6):446–452
26. Ding K, Li J, Liu H (2019) Interactive anomaly detection on attributed networks. In: In the twelfth ACM international conference on web search and data mining (WSDM '19). https://doi.org/10.1145/3289600.3290964
27. Empirical rule: What is it? (2017). http://www.statisticshowto.com/empirical-rule-2/
28. Forrest S, Perelson A, Allen L, Cherukuri R (1994) Self-nonself discrimination in a computer. In: Proceedings of 1994 IEEE computer society symposium on research in security and privacy, p 202. Institute of Electrical & Electronics Engineers (IEEE). https://doi.org/10.1109/risp.1994.296580
29. Gentile C, Li S, Kar P, Karatzoglou A, Zappella G, Etrue E (2017) On context-dependent clustering of bandits. In: Precup D, Teh YW (eds) Proceedings of the 34th international conference on machine learning, proceedings of machine learning research, vol 70, pp 1253–1262. PMLR, International Convention Centre, Sydney, Australia. http://proceedings.mlr.press/v70/gentile17a.html
30. Ghanem TF, Elkilani WS, Abdul-kader HM (2015) A hybrid approach for efficient anomaly detection using metaheuristic methods. J Adv Res 6(4):609–619. https://doi.org/10.1016/j.jare.2014.02.009
31. González-Pino J, Edmonds J, Papa M (2006) Attribute selection using information gain for a fuzzy logic intrusion detection system. In: Defense and security symposium, pp 62410D–62410D. International society for optics and photonics
32. González FA, Dasgupta D (2003) Anomaly detection using real-valued negative selection. Genet Program Evolvable Mach 4(4):383–403
33. Guha S, Yau SS, Buduru AB (2016) Attack detection in cloud infrastructures using artificial neural network with genetic feature selection. In: Dependable, autonomic and secure computing, 14th International conference on pervasive intelligence and computing, 2nd International conf on big data intelligence and computing and cyber science and technology congress (DASC/PiCom/DataCom/CyberSciTech), 2016 IEEE 14th Intl C, pp 414–419. IEEE
34. Guo H, Feng Y, Hao F, Zhong S, Li S (2014) Dynamic fuzzy logic control of genetic algorithm probabilities. J Comput 9(1):22–27. https://doi.org/10.4304/jcp.9.1.22-27
35. Gutierrez MP, Kiekintveld C (2016) Bandits for cybersecurity: adaptive intrusion detection using honeypots. In: AAAI Workshop: Artificial Intelligence for Cyber Security
36. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The WEKA data mining software. SIGKDD Explor Newsl 11(1):10. https://doi.org/10.1145/1656274.1656278
37. Hao F, Li S, Min G, Kim HC, Yau SS, Yang LT (2015) An efficient approach to generating location-sensitive recommendations in ad-hoc social network environments. IEEE Trans Serv Comput 8(3):520–533. https://doi.org/10.1109/tsc.2015.2401833
38. Hao F, Park DS, Li S, Lee HM (2016) Mining λ-maximal cliques from a fuzzy graph. Sustainability 8(6):553
39. Hofmann A, Horeis T, Sick B (2004) Feature selection for intrusion detection: an evolutionary wrapper approach. In: 2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541), vol 2, pp 1563–1568. Institute of Electrical & Electronics Engineers (IEEE). https://doi.org/10.1109/ijcnn.2004.1380189
40. Hofmeyr SA, Forrest S (2000) Architecture for an artificial immune system. Evol Comput 8(4):443–473. https://doi.org/10.1162/106365600568257
41. Hong L (2008) Artificial immune system for anomaly detection. In: 2008 IEEE international symposium on knowledge acquisition and modeling workshop, pp 340–343. Institute of Electrical & Electronics Engineers (IEEE). https://doi.org/10.1109/kamw.2008.4810493
42. Hoque MS, Mukit M, Bikas M, Naser A, et al. (2012) An implementation of intrusion detection system using genetic algorithm. arXiv preprint arXiv:1204.1336
43. Igbe O, Darwish I, Saadawi T (2016) Distributed network intrusion detection systems: an artificial immune system approach. In: Connected health: applications, systems and engineering technologies (CHASE), 2016 IEEE First International Conference on, pp 101–106. IEEE

44. Janarthanan T, Zargari S (2017) Feature selection in unsw-nb15 and kddcup'99 datasets. In: 2017 IEEE 26th international symposium on industrial electronics (ISIE), pp 1881–1886. IEEE
45. Kar P, Li S, Narasimhan H, Chawla S, Sebastiani F (2016) Online optimization methods for the quantification problem. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 1625–1634. ACM
46. Karegowda AG, Manjunath A, Jayaram M (2010) Comparative study of attribute selection using gain ratio and correlation based feature selection. Int J Inf Technol Knowl Manag 2(2):271–277
47. Kayacik HG, Zincir-Heywood AN, Heywood MI (2005) Selecting features for intrusion detection: A feature relevance analysis on kdd 99 intrusion detection datasets. In: Proceedings of the third annual conference on privacy, security and trust
48. Khammassi C, Krichen S (2017) A GA-LR wrapper approach for feature selection in network intrusion detection. Comput Secur 70:255–277
49. Kim J, Bentley PJ (2001) Towards an artificial immune system for network intrusion detection: An investigation of clonal selection with a negative selection operator. In: Proceedings of the 2001 congress on evolutionary computation, 2001. vol 2, pp 1244–1252. IEEE
50. Kim J, Bentley PJ (2002) Towards an artificial immune system for network intrusion detection: an investigation of dynamic clonal selection. In: Proceedings of the 2002 congress on evolutionary computation, 2002. CEC'02., vol 2, pp 1015–1020. IEEE
51. Kira K, Rendell LA (1992) A practical approach to feature selection. In: Proceedings of the ninth international workshop on Machine learning, pp 249–256
52. Korda N, Szörényi B, Shuai L (2016) Distributed clustering of linear bandits in peer to peer networks. In: Journal of machine learning research workshop and conference proceedings, vol 48, pp 1301–1309. International Machine Learning Society
53. Kumar V, Chauhan H, Panwar D (2013) *K*-means clustering approach to analyze NSL-KDD intrusion detection dataset. International Journal of Soft Computing and Engineering (IJSCE) ISSN, pp 2231–2307
54. Li S, Hao F, Li M, Kim HC (2013) Medicine rating prediction and recommendation in mobile social networks. In: International conference on grid and pervasive computing, pp 216–223. Springer
55. Li S, Karatzoglou A, Gentile C: Collaborative filtering bandits. In: Proceedings of the 39th international ACM SIGIR conference on research and development in information retrieval
56. Li X, Ye N (2001) Decision tree classifiers for computer intrusion detection. J Parallel Distrib Comput Pract 4(2):179–190
57. Lu C, Feng J, Lin Z, Mei T, Yan S (2018) Subspace clustering by block diagonal representation. IEEE Transactions on Pattern Analysis and Machine Intelligence pp 1–1. https://doi.org/10.1109/tpami.2018.2794348
58. Lu W, Traore I (2004) Detecting new forms of network intrusion using genetic programming. Comput Intell 20(3):475–494
59. Matthews BW (1975) Comparison of the predicted and observed secondary structure of t4 phage lysozyme. Biochimica et Biophysica Acta (BBA)-Protein Structure 405(2):442–451
60. Mohammadi M, Akbari A, Raahemi B, Nassersharif B, Asgharian H (2014) A fast anomaly detection system using probabilistic artificial immune algorithm capable of learning new attacks. Evol Intel 6(3):135–156. https://doi.org/10.1007/s12065-013-0101-3
61. Moustafa JSN (2016) The unsw-nb15 data set description. https://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-NB15-Datasets/
62. Moustafa N, Slay J (2015) The significant features of the unsw-nb15 and the kdd99 data sets for network intrusion detection systems. Unpublished. https://doi.org/10.13140/RG.2.1.2264.4883
63. Moustafa N, Slay J (2015) UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 Military communications and information systems conference (MilCIS), pp 1–6. IEEE. https://doi.org/10.1109/milcis.2015.7348942
64. Moustafa N, Slay J (2016) The evaluation of network anomaly detection systems: statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set. Inf Secur J Global Perspect 25:1–3. https://doi.org/10.1080/19393555.2015.1125974
65. Mukkamala S, Janoski G, Sung A (2002) Intrusion detection using neural networks and support vector machines. In: Neural Networks, 2002. IJCNN'02. In: Proceedings of the 2002 international joint conference on, vol 2, pp 1702–1707. IEEE
66. Najafabadi MM, Khoshgoftaar TM, Seliya N (2016) Evaluating feature selection methods for network intrusion detection with kyoto data. Int J Reliab Qual Saf Eng 23(01):1650001. https://doi.org/10.1142/s0218539316500017
67. Nastaiinullah, N., Adiwijaya, Kurniati, AP (2014) Anomaly detection on intrusion detection system using CLIQUE partitioning. In: 2014 2nd International conference on information and communication technology (ICoICT). IEEE. https://doi.org/10.1109/icoict.2014.6914031
68. Nguyen HT, Petrović S, Franke K (2010) A comparison of feature-selection methods for intrusion detection, pp 242–255. Springer. https://doi.org/10.1007/978-3-642-14706-7_19
69. Noble CC, Cook DJ (2003) Graph-based anomaly detection. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM Press. https://doi.org/10.1145/956750.956831
70. Owen JA, Punt J, Stranford SA et al (2013) Kuby immunology. WH Freeman, New York
71. Panda M, Patra MR (2007) Network intrusion detection using naive bayes. Int J Comput Sci Netw Secur 7(12):258–263
72. Parham P (2015) The immune system, 4th edn. Garland Science, New York City
73. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2007–2017) Scikit-learn tool. http://scikit-learn.org
74. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830
75. Popoola E, Adewumi AO (2017) Efficient feature selection technique for network intrusion detection system using discrete differential evolution and decision. IJ Netw Secur 19(5):660–669
76. Portnoy L (2000) Intrusion detection with unlabeled data using clustering
77. Rathore H (2016) Mapping biological systems to network systems
78. Ryan J, Lin MJ, Miikkulainen R (1998) Intrusion detection with neural networks. In: Proceedings of the advances in neural information processing systems 10: annual conference on neural information processing systems 1997, NeurIPS 1977, Denver,

Colorado, USA, 1997. The MIT Press 1998, ISBN 0-262-10076-2
79. Salamatova T, Zhukov V (2017) Network intrusion detection by the coevolutionary immune algorithm of artificial immune systems with clonal selection. IOP Conf Ser Mater Sci Eng 173(1):012016
80. Saurabh P, Verma B (2016) An efficient proactive artificial immune system based anomaly detection and prevention system. Expert Syst Appl 60:311–320
81. Seresht NA, Azmi R (2014) MAIS-IDS: a distributed intrusion detection system using multi-agent ais approach. Eng Appl Artif Intell 35:286–298
82. Shanmugavadivu R, Nagarajan N (2011) Network intrusion detection system using fuzzy logic. Indian J Comput Sci Eng (IJCSE) 2(1):101–111
83. Shen J, Wang J, Ai H (2012) An improved artificial immune system-based network intrusion detection by using rough set. CN 04(01):41–47. https://doi.org/10.4236/cn.2012.41006
84. Shon T, Moon J (2007) A hybrid machine learning approach to network anomaly detection. Inf Sci 177(18):3799–3821
85. Sompayrac LM (2016) How the immune system works. The how it works series, 5ed edn. Wiley, Hoboken
86. Song J, Takakura H, Okabe Y, Eto M, Inoue D, Nakao K (2011) Statistical analysis of honeypot data and building of kyoto 2006+ dataset for NIDS evaluation. In: Proceedings of the first workshop on building analysis datasets and gathering experience returns for security, pp 29–36. ACM. https://doi.org/10.1145/1978672.1978676
87. Souici-Meslati L, Zekri M (2016) Immunological approach for intrusion detection. REVUE AFRICAINE DE LA RECHERCHE EN INFORMATIQUE ET MATHÉMATIQUES APPLIQUÉES 17:
88. Sridevi R, Chattemvelli R (2012) Genetic algorithm and artificial immune systems: a combinational approach for network intrusion detection. In: 2012 International Conference on Advances in Engineering, Science and Management (ICAESM), pp 494–498. IEEE
89. Tabatabaefar M, Miriestahbanati M, Grégoire JC (2017) Network intrusion detection through artificial immune system. In: Systems Conference (SysCon), 2017 Annual IEEE International, pp 1–6. IEEE
90. Tavallaee M, Bagheri E, Lu W, Ghorbani AA (2009) A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE symposium on computational intelligence for security and defense applications. Institute of Electrical & Electronics Engineers (IEEE). https://doi.org/10.1109/cisda.2009.5356528
91. Traffic data from kyoto university's honeypots. http://www.takakura.com/Kyoto_data/data/
92. Vapnik VN (2000) The nature of statistical learning theory. Springer, New York. https://doi.org/10.1007/978-1-4757-3264-1
93. Xian JQ, Lang FH, Tang XL (2005) A novel intrusion detection method based on clonal selection clustering algorithm. In: 2005 International conference on machine learning and cybernetics, vol 6, pp 3905–3910. IEEE. https://doi.org/10.1109/icmlc.2005.1527620
94. Yan Q, Yu J (2006) Ainids: an immune-based network intrusion detection system. In: Defense and security symposium, pp 62410U–62410U. International Society for Optics and Photonics
95. Yang H, Li T, Hu X, Wang F, Zou Y (2014) A survey of artificial immune system based intrusion detection. Sci World J 2014:1–11. https://doi.org/10.1155/2014/156790
96. Yasir H, Balasaraswathi VR, Journaux L, Sugumaran M (2018) Benchmark datasets for network intrusion detection: a review. Int J Netw Secur 20:645–654
97. Yin C, Ma L, Feng L (2015) Towards accurate intrusion detection based on improved clonal selection algorithm. Multimed Tools Appl 76:1–14. https://doi.org/10.1007/s11042-015-3117-0
98. Yin C, Ma L, Feng L (2016) A feature selection method for improved clonal algorithm towards intrusion detection. Int J Pattern Recognit Artif Intell 30(05):1659013
99. Zargari S, Voorhis D (2012) Feature selection in the corrected KDD-dataset. In: 2012 Third international conference on emerging intelligent data and web technologies. IEEE. https://doi.org/10.1109/eidwt.2012.10
100. Zhang L, ying BAI Z, long LU Y, xing ZHA Y, wen LI Z (2014) Integrated intrusion detection model based on artificial immune. J China Univ Posts Telecommun 21(2):83–90
101. Zhao X, Wang G, Li Z (2016) Unsupervised network anomaly detection based on abnormality weights and subspace clustering. In: 2016 Sixth international conference on information science and technology (ICIST). IEEE. https://doi.org/10.1109/icist.2016.7483462
102. Zhu X (2005) Semi-supervised learning literature survey. Technical Report 1530, Department of Computer Sciences, University of Wisconsin, Madison