ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **26**, Number 3–4, 2023, 323–338

XOR-Based Detector of Different Decisions on Anomalies in the Computer Network Traffic

Danijela PROTIC[1,*] and Miomir STANKOVIC[2]

[1]Center for Applied Mathematics and Electronics, Vojvode Stepe 445, 11000 Belgrade, Serbia
[2]Mathemathical Institute of SASA, Knez Mihajlova 36, 11000 Belgrade, Serbia
E-mails: danijelaprotic318@gmail.com*, miomir.stankovic@gmail.com
* Corresponding author

Abstract. Anomaly-based intrusion detection systems are designed to scan computer network traffic for abnormal behavior. Binary classifiers based on supervised machine learning have proven to be highly accurate tools for classifying instances as normal or abnormal. Main disadvantages of supervised machine learning are the long processing time and large amount of training data required to ensure accurate results. Two preprocessing steps to reduce data sets are feature selection and feature scaling. In this article, we present a new hyperbolic tangent feature scaling approach based on the linearization of the tangent hyperbolic function and the damping strategy of the Levenberg-Marquardt algorithm. Experiments performed on the Kyoto 2006+ dataset used four high-precision binary classifiers: weighted k-nearest neighbors, decision tree, feedforward neural networks, and support vector machine. It is shown that hyperbolic tangent scaling reduces processing time by more than twofold. An XOR-based detector is proposed to determine conflicting decisions about anomalies. The decisions of the FNN and wk-NN models are compared. It is shown that decisions sometimes turn out differently. The percentage of the opposite decisions has been shown to vary and is not affected by dataset size.

Key-words: Anomaly-based intrusion detection; binary classification; machine learning.

1. Introduction

The rapid development of computer networks over the past few decades has contributed to an explosion in the number of malicious attacks on sensitive data. As a result, *intrusion detection systems* (IDS) have become indispensable tools for protecting computer networks from malicious attacks, human error and anomalies. IDS are generally classified into three installation types: host-based, network-based, and hybrid. Furthermore, the network-based intrusion detection systems can be divided into signature-based and anomaly-based systems, both of which are inspired by an adaptive *human immune system* (HIS) [1, 2]. The signature-based system derives

from humoral immunity, which is based on B cells that protect the body from pathogens coming from outside the body. B cells produce antibodies to kill pathogens to remember pathogen signatures. The anomaly-based system is motivated by the negative selection of T cells, which protects the body from self-cells that deviate from normal body cells. T cells secrete cytokines that cause apoptosis (cell death) of cells that lead to cancer or autoimmune diseases [3]. Signature-based IDS compare unknown network traffic against a database of known attack signatures. They are relatively fast but cannot detect unknown malicious attacks. Anomaly-based intrusion detection systems identify deviation from normal network behavior. The main advantage of anomaly-based IDS is the detection of unknown attacks. The main problem with anomaly-based detection is the time and disk space required to develop a statistical model of normal network behavior.

If abnormal network behavior is detected, binary classifiers based on supervised ML are ideal candidates for anomaly-based intrusion detection systems [4–6]. Reinforcement learning can also be used to train ML-based classifiers, but while the results of classification can be very accurate, convergence can be slowed down. In [7] the authors present algorithms to improve reinforcement learning. The authors of [8] discuss a variety of potential benefits of reinforcement learning on classification problems. Unsupervised learning can also be used for classification, but it is beyond the scope of this paper since unsupervised ML models deal with the unlabeled data sets. However, methods like k-means clustering offer many possibilities to extract hidden information from the data, which can be used in the field of big data analysis and processing, as discussed in [9, 10]. In [11] the author discusses a variety of techniques to improve the performance metrics of functional verifications for all three ML techniques. The size of the data used to train the models is the main issue with supervised ML. Datasets with multiple features of different sizes, ranges and units are challenges for supervised ML algorithms. Two preprocessing steps can be performed before classification: feature selection and feature scaling, which remove irrelevant features and scale the relevant features in the same range. While increasing classification accuracy, this reduces model complexity and processing time. It should be noted that in some cases denormalization should be performed as a postprocessing step. The impact of feature preprocessing on the accuracy and processing time of *Feedforward Neural Network* (FNN), *weighted k-Nearest Neighbor* (wk-NN), *Decision Tree* (DT), and *Support Vector Machine* (SVM) are discussed in this article. The impact of min-max normalization [12] and novel *hyperbolic tangent* (TH) scaling on model processing time and accuracy is discussed.

In the previous work, the authors have shown positive properties of the hyperbolic-tangent scaling, and the goal of improvements based on the Levenberg-Marquardt algorithm. The authors have also shown the benefits of the XOR operation to the detection of conflicting decisions [13-15]. The main idea of this research is that with the XOR detection of contradictory decisions on anomalies, it is possible to increase the detection level of malware and bugs in computer networks, improved by the use of accumulator register that deals with triggered outputs of the classifiers. The Levenberg-Marquardt algorithm is now described in details and the pseudo-code is given. In accordance with main idea, two hypotheses are derived: (1) by choosing hyperbolic tangent-type scaling and the damping strategy of the Levenberg-Marquardt algorithm, it is possible to reduce the processing time by more than twofold without significantly degrading accuracy of binary classifiers which work in parallel and monitor network traffic and (2) nonlinear binary classifiers with weights show the best properties for application in XOR detection of conflicting decisions about anomalies in the computer networks. The results show that the TH scaling has a clearly positive influence on all presented models. The FNN model is the fastest while the wk-NN model is the most accurate. These two models are used to make different decisions about

anomalies based on a binary XOR logic operation to compare the detection quality. The authors of [16] discuss the concept of two unequal sensors working in parallel and describe a decentralized detection system with parallel topology. This concept is used to detect different decisions based on an XOR operation performed on the outputs of two classifiers. To detect an anomaly, the classifiers monitor the behavior of the computer network. However, the classifier decisions sometimes differ. In their (independent) decisions, the XOR merger rule becomes relevant. If either output is zero, but not both, the XOR operation produces a logical true. Otherwise the result is false. The results passed to the accumulator register, which is initially cleared to zero and holds the results of XOR operation. The results show that the percentage of different decisions is not related to the size of the data sets. It is shown that the detector can remove decision uncertainties but cannot predict the probability of a given event occurring.

The remainder of the paper is structured as follows. Section 2 is devoted to the related work. A binary classification scheme is described in Section 3. Feature selection and feature scaling methodologies as well as the design of the XOR detector are described in details. Section 4 includes observations on the experimental setup and results. Section 5 concludes the paper.

2. Related Work

Many research difficulties in data analysis, visualization, and understanding are related to the availability and usability of the multidimensional data, as features affect anomaly detection [17]. Dimensionality reduction and feature scaling support binary classification to reduce processing time and improve ML model accuracy. There are many numerous detection and classification techniques that have been extensively studied in the literature dealing with feature-based classification [18-21]. The feature selection approach to save storage space and speed up the classification algorithms is proposed in [19]. When the authors compared neural network, k-NN, and SVM models, and found that many of them had over 50% faster processing times with 90% accuracy when using only half of features. A study on the performance of feature selection optimization is presented in [20]. The authors show a positive impact on the accuracy of the ML model, reducing performance latency and reducing computational complexity. In [21] the authors summarize the research on DT, SVM, FNN and nearest neighbour models. They emphasize that proper feature selection can reduce processing time and improve classification performance. They also discuss the effects of feature normalization in the range ± 1. In addition to feature selection, feature scaling is often used. Without feature scaling, ML algorithms tend to weight higher values and treat smaller values as lower values regardless of their units. Features with different scales can cause the model to diverge, overestimate, underestimate, or ignore some parameters and decrease the estimation efficiency. Normalization is a feature scaling technique that ensures each data point has the same scale. It is useful when dataset does not contain outliers and it is known the distribution of the dataset is non-Gaussian. The authors of [22] describe the Min-Max normalization for network intrusion detection using ML models on the selected Kyoto 2006+ dataset. To avoid bias from outliers in the unbalanced dataset, normalization is used before splitting and after the dataset is balanced. Various supervised ML algorithms are widely used for anomaly-based classification of computer network traffic. The SVM model uses the hyperplane in the n-dimensional space to classify instances [23, 24]. The instances on either side of the hyperplane are classified differently. The k-NN algorithm identifies a sample based on its k neighbours and calculates the distances between them. The parameter k affects the performance of the classifier. When k is small, the model tends to overfit. A large value of k can lead to misclassification of the instance [25]. The wk-NN model extends the k-NN model in such a way that the instance from

the training set that is closest to the new instance has a higher weight in the decision than those that are further away [26, 27]. In [28] the authors discuss the complexity of cognitive systems, which is revealed in that the achievements of such system advantages form the machine learning algorithms, including neural networks. The output of the neural network is determined by the prediction probability and the classification threshold [29]. The computation in the network is performed by passing the input data to compute the outputs and backward propagating the error of the cost function to adjust the weights [30]. The DT model predicts the class label in the input data following decision from the root to the leaf nodes. The branching conditions associated with each node divide the possibility space into subsets [6].

A number of anomaly-based intrusion detection systems have been studied over the years using various datasets, most of which are simulations of computer network behavior. The authors examined, described and compared data from ADFA-LF/WD, AWID, CAIDA, CICIDS2017, CSE-CIC-2018, ISCX2012, KDD CUP '99, Kyoto 2006+, NSL-KDD, UNSW-NB15 and many other datasets [22, 31–33]. The Kyoto 2006+ dataset is only one collected over real network traffic and is intended solely for anomaly-based intrusion detection. The feature *Label* determines whether the anomaly will be detected or not. As a result, the dataset serves as the basis for the research presented in this paper. In this way, the binary logical operation XOR can be used to determine the relationship between classifier outputs. The performance of the XOR rule is shown in [16]. The authors describe the one-bit quantified data sensors that send the data to a fusion center that uses the XOR rule to make a final decision based on two sensors working in parallel at the same time.

3. Classification Scheme

A binary classification scheme presented in this paper consists of four steps. Three of these are well-known preprocessing, classification, and postprocessing. In addition, this scheme contains an additional XOR block for comparing the outputs of the classifiers.

3.1. Data collection

To conduct the experiments, we used the Kyoto 2006+ dataset as it was developed for evaluating of the network-based IDSs. The dataset includes daily records of real network traffic data collected from $\sim$350 honeypots, including two dark web sensors with $\sim$300 unused IP addresses and other IDSs installed on five different computer networks inside and outside Kyoto University [34]. It consists of 24 statistical, numerical and categorical features, 14 of which are derived from the KDD Cup '99 dataset and another 10 features used exclusively for the assessment and further analysis of anomaly-based IDS [22, 26, 35] (see Table 1). The first part of the Kyoto 2006+ dataset contains $\sim$90 million instances recorded between 2006 and 2009. The second part, covering the period from November 2009 to December 2015, contains another 20 GB of data [36]. The dataset includes DoS, exploits, malware, port scans, and shell code attacks on honeypots, but does not provide information on specific attacks. Instead, the feature *Label* is used to indicate whether the attack exists or not [37]. The original data is marked with three labels: 1 for normal sessions, –1 for known attacks, and –2 for unknown attacks. However, because unknown attacks are very rare in a dataset ($<1\%$), we gave known and unknown attacks the same label (–1). The IDS Bro was used to convert data from packet-based traffic into a session format. The IDS Bro is a signature and behavior-based analysis framework that provides information about communication protocols and unusual network behavior [32]. The Bro event engine receives

Internet Protocol (IP) packets and converts them into events that are targeted to the policy script interpreter, which generates outputs. The problem with the Kyoto 2006+ is its size. In this study, the problem is solved in a preprocessing step that removes all irrelevant features and scales the relevant ones.

3.2. Feature selection

Feature selection is used to remove any extraneous features that might interfere with the classification process. For that reason, all redundant and irrelevant data are eliminated [38].

Table 1. The Kyoto 2006+ dataset [26]

No	Feature	Description
1	Duration – basic	The length of the connection (seconds).
2	Service – basic	The connection's server type (dns, ssh, other).
3	Source bytes – basic	The number of data bytes sent by the source IP address.
4	Destination bytes – basic	The number of data bytes sent by the destination IP address.
5	Count	The numbers of connections whose source IP address and destination IP address are the same to those of the current connection in the past two seconds.
6	Same_srv_rate	% of connections to the same service in the Count feature.
7	Serror_rate	% of connections that have 'SYN' errors in Count feature.
8	Srv_serror_rate	% of connections that have 'SYN' errors in Srv_count (% of connections whose service type is the same to that of the current connections in the past two seconds) features.
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	% of connections whose source port is the same to that of the current connection in Dst_host_count feature.
12	Dst_host_serror_rate	% of connections that have 'SYN' errors in Dst_host_count feature.
13	Dst_host_srv_serror_rate	% of connections that have 'SYN' errors in Dst_host_srv_count feature.
14	Flag	The state of the connection at the time of connection was written (tcp, udp).
15	IDS_detection	Reflects if IDS triggered an alert for the connection; '0' means any alerts were not triggered and an Arabic numeral means the different kind of alerts. Parenthesis indicates the number of the same alert.
16	Malware_detection	Indicates if malware, also known as malicious software, was observed at the connection; '0' means no malware was observed, and string indicates the corresponding malware observed at the connection. Parenthesis indicates the number of the same malware.
17	Ashula_detection.	Means if shellcodes and exploit codes were used in the connection; '0' means no shellcode or exploit code was observed, and an Arabic numeral means the different kinds of the shellcodes or exploit codes. Parenthesis indicates the number of the same shellcode or exploit code
18	Label	Indicates whether the session was attack or not; '1' means normal. '–1' means known attack was observed in the session, and '–2' means unknown attack was observed in the session.
19	Source_IP_Address	Means source IP address used in the session. The original IP address on IPv4 was sanitized to one of the Unique Local IPv6 Unicast Addresses. Also, the same private IP addresses are only valid in the same month; if two private IP addresses are the same within the same month, it means their IP addresses on IPv4 were also the same, otherwise are different.
20	Source_Port_Number	Indicates the source port number used in the session.
21	Destination_IP_Address	It was also sanitized.
22	Destination_Port_Number	Indicates the destination port number used in the session.
23	Start_Time	Indicates when the session was started.
24	Duration	Indicates how long the session was being established.

Table 2. Selected features from the Kyoto 2006+ dataset

No	Feature	No	Feature	No	Feature
1	Count	4	Srv_error_rate	7	Dst_host_same_src_port_rate
2	Same_srv_rate	5	Dst_host_count	8	Dst_host_serror_rate
3	Serror_rate	6	Dst_host_srv_count	9	Dst_host_srv_serror_rate

The nine numerical features listed in Table 2 are the key features identified to conduct the experiments presented in this article. All categorical features, connection duration features, and statistical features are removed from the Kyoto 2006+ dataset. Only the feature *Label* remains for further analysis. Network traffic is detected normally when $Label = 1$. The anomaly is detected when $Label = -1$.

The problem that remains after this is that the features are not scaled. When features are at drastically different scales, they can produce skewed and incorrect results and negatively impact model evaluation. For these reasons, feature scaling is performed.

3.3. Feature scaling

Feature scaling is a data normalization technique used prior to classification to scale the independent features in the same range of data. In this article, we present a TH scaling based on min-max normalization, hyperbolic tangent function and LM damping strategy [39], [40]. The min-max normalization scales features in the ranges [-1,1] or [0,1]. The normalization of the instance $x(i)$ over interval $[a, b], \forall a, b \in R$ is given by the expression

$$x(i)_{new} = a + \frac{(x(i) - x_{Min})(a - b)}{x_{Max} - x_{Min}}, \tag{1}$$

where $x(i)_{new}$ denotes a new instance, and x_{Max} and x_{Min} are minimum and maximum values, respectively, of unscaled feature.

The hyperbolic tangent function $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ is an *S*-shaped zero-centered function bounded in [–1,1], with $tanh(\pm 1) \approx \pm 0.7616$.The gradient $tanh(x)^{'} = 1 - tanh(x)^2$ has a narrow slope, with $tanh(x)'_{x \to 0 \pm \varepsilon | \varepsilon \to 0 \approx 1}$. The part of the $tanh$ function limited to ± 0.7616, can be approximated by a linear function $f(x) \approx x, x \in [-0.7616, 0.7616]$. This can be used to constrain the instances to the same symmetric fixed range $[-0.7616, 0.7616]$, and the features can be scaled as follows:

$$x(i)_{TH} = tanh\left(\frac{x(i) - \frac{x_{Max} + x_{Min}}{2}}{\frac{x_{Max} - x_{Min}}{2}}\right), \tag{2}$$

where $x(i)_{TH}$ represents the scaled instance. In [41] the authors explain the differences between exponential and Taylor-based calculations and provide details on calculation of the hyperbolic tangent function for matrix calculations. They have also shown an increase in accuracy when using the differential polynomial approach. Since minimum and maximum values of the features are unknown, TH normalization is used to avoid biased and incorrect results and to speed up classifier training. The main goal is to find an optimal value of the nonlinear, continuous, real-valued loss function $f : R \to R$ using the damping strategy of the LM algorithm used to alternate the *gradient descent* (GD) and the *Gauss-Newton* (GN) algorithms. The GD finds the optimal value by iterating from an initial guess to the best value based on negative gradient. As the GD reaches the optimal value, each step gets smaller and smaller. The GN algorithm is based on

a generalization of the Newton's method for solving nonlinear least-squares problems [42] in which the Hessian matrix is approximated by using the Jacobian matrix.

In our previous work we have described the LM algorithm in detail. Here we present a brief description of the pseudo-code of the update mechanism [43]. Let the input vector be $\mathbf{x} = [x_1\, x_2 \cdots x_n]^T$. The LM algorithm minimizes the Euclidean distance $||f(x)||^2$. The gradient $\nabla\mathbf{f}(\mathbf{x})$ and the Hessian $\mathbf{H}(\mathbf{f}(\mathbf{x}))$ are vectors of partial derivatives and matrix of second partial derivatives of $\mathbf{f}(\mathbf{x})$, respectively, given as follows:

$$\mathbf{f}(\mathbf{x}) = \left[\frac{\partial\mathbf{f}(\mathbf{x})}{\partial x_1}\frac{\partial\mathbf{f}(\mathbf{x})}{\partial x_2}\cdots\frac{\partial\mathbf{f}(\mathbf{x})}{\partial x_n}\right], H(\mathbf{f}(\mathbf{x})) = \nabla^2\mathbf{f}(\mathbf{x}) = \\ = \begin{bmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial x_1^2} & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_2^2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_n} \\ \vdots & & \ddots & \vdots \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_n^2} \end{bmatrix}_{n\times m}. \tag{3}$$

For any partial derivative $g_i = \frac{\partial f_j(\mathbf{x})}{\partial x_i}(i = 1\ldots n,\ j = 1\ldots m)$ in high-dimensional space the gradient-matrix of partial derivatives with respect to each dimension is $g_j(\mathbf{x}) = \nabla f_j(\mathbf{x}) = [g_{1,j}(\mathbf{x})\, g_{2,j}(\mathbf{x}) \cdots g_{n,j}(\mathbf{x})]^T$ and the Jacobian matrix $\mathbf{J}(\mathbf{f}(\mathbf{x}))$ contains all first-order partial derivatives of $\mathbf{f}$, so that

$$\mathbf{J}(\mathbf{f}(\mathbf{x})) = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_n} \\ \vdots & & \ddots & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1} & \frac{\partial f_m(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_m(\mathbf{x})}{\partial x_n} \end{bmatrix}_{n\times m}, \tag{4}$$

and the Hessian matrix is then determined as $\mathbf{H}(\mathbf{f}(\mathbf{x})) = \mathbf{J}(\nabla\mathbf{f}(\mathbf{x}))^T$.

The LM algorithm uses Taylor's truncated formula to determine the iterate $x^{(k+1)}$, based on k previous iterates $x^{(1)}, x^{(2)}, \ldots, x^{(k)}$. The main idea of the algorithm is to find the optimal solution $x^{(k+1)} \approx x_{opt}$ and $f(x, x_{opt}) \approx \hat{f}(x)$, *i.e.* to minimize both the first and second part of the expression

$$||\hat{f}(x, x^{(k)})||^2 + \lambda^{(k)}||x - x^{(k)}||, \lambda^{(k)} > 0, \tag{5}$$

where $\lambda^{(k)}$ denotes the damping factor, which varies with the step size. The LM algorithm is the iterative algorithm that finds the optimal value iterate $x^{(k+1)}$ of the function f in terms of

$$x^{(k+1)} = x^{(k)} - (\, H + \lambda^{(k)}\mathbf{I})^{-1}\mathbf{J}^T f(x^{(k)}) = x^{(k)} - (\mathbf{J}^T\mathbf{J} + \lambda^{(k)}\mathbf{I})^{-1}\mathbf{J}^T f(x^{(k)}), \tag{6}$$

where $\mathbf{I}$ stands for the identity matrix. The point $x^{(k)}$ is considered stationary point if and only if $x^{(k+1)} \approx x^{(k)}$. The LM algorithm starts at initial point $x^{(0)}$ and the initial damping factor $\lambda^{(0)}$. During the iterative procedure $\lambda^{(k)}$ is adjusted. If $\lambda^{(k)}$ is too big $x^{(k+1)}$ is too close to $x^{(k)}$), and progress is slow. Otherwise $x^{(k+1)}$ is far from $x^{(k)}$ and approximation is poor. The LM algorithm acts as the GD algorithm if $\lambda^{(k)} \to \infty$, because $\mathbf{H} + \lambda^{(k)}\mathbf{I} \to \mathbf{I}$, and as the

GN algorithm if $\lambda^{(k)} \rightarrow 0$, because the iterate $x^{(k)} \rightarrow x_{opt}$. The authors show in [41] that the approaches based on Taylor series are accurate but computationally intensive. The update mechanism for adjusting the damping parameter $\lambda^{(k)}$ is given by the following pseudo-code:

Pseudo-code of the LM algorithm update mechanism

1 if $||f(x^{(k+1)})||^2 < ||f(x^k)||^2$, $f(x^{(k+1)})$ is better than the current objective, accept new $x^{(k+1)}$ and reduce $\lambda^{(k)}$,
2 otherwise increase $\lambda^{(k)}$ and do not update $x^{(k)}$; $x^{(k+1)} = x^{(k)}$.

3.4. Binary classification

In supervised ML, binary classification is a process that divides a data set into two distinct classes, one normal and one abnormal, by measuring a set of attributes. Table 3 shows the properties of the classifiers used for the experiments presented in this article: learning algorithm, stopping criteria, prediction speed, memory usage, interpretability, and model flexibility. MATLAB 2020a Classification Learner is used as a tool to train and test the classifiers [44]. The gallery contains tunable models, trained with hyperparameter optimization. Depending on the hyperparameters, ML models automatically optimize their parameter values using Bayeisan optimization, which minimizes model loss based on the selected validation options. The DT is one of the most efficient classifiers, but almost unaffected by feature scaling. Optimizable hyperparameters of the DT model are maximum number of splits and split criterion. In our case, the software searches among logarithmic-scaled integers in the range $[1, max(2, n-1)]$,where n is the number of instances, and also searches under the Gini's diversity index, Twoing's rule, and the maximum deviation reduction. In this case, maximum number of splits is 20. Nearest neighbor classifiers are called lazy learners. The k-NN is one of the simplest classification algorithms, identifying a sample based on k neighbors; when k is small, the model tends to overfit; if k is large, the instance may be misclassified. The wk-NN algorithm introduced by Dudani [45] is a modified version of the k-NN algorithm. In the decision, the instance of the first neighbor gets the highest weight (1), while the k^{th} instance gets the lowest weight (0). In the experiments presented here, the hyperparameters that can be tuned are number of neighbors, distance metric, distance weight, and binary split (true or false). The default number of neighbors in the experiments presented here $k \leq 10$. The software searches among logarithmic-scaled integers in the range $[1, max(2, round(n/2))]$,where n is the number of instances. The distance metric is Euclidean and the weights are calculated as squared inverse distances. The Gaussian SVM presented in this paper offers high speed classification. The hyperparameters used for optimization are: kernel function, kernel scale, multiclass method and data standardization. In our work we used the Gaussian kernel function and software search among positive logarithmic-scaled values in the range $[0.001, 1000]$. The One-vs-One multiclass method is used and data is standardized by searching between true and false. The number of kernels is $\sqrt{P}/4$, where P is the number of predictors.

A neural network is a black box-like structure that processes inputs to produce useful outputs based on prediction probability and classification threshold. The calculation is performed by propagating input data forward to calculate outputs and then propagating errors backward to adjust weights. The main problem with neural networks is that for drastically different scales of input data, the classification model diverges, underestimates, overestimates, or ignores some parameters. In the experiments, the backpropagation-based FNN with one hidden layer is used.

Table 3. Properties of the classifiers

	DT	**SVM**	**wk-NN**	**FNN**
Learning algorithm	Medium tree	Medium Gaussian	Euclidean, inverse squared distance	LM
Stopping criteria	Maximum splits = 20	sqrt(P)/4	k=10, No. of instances in the training set	min MSE
Prediction speed	Fast	Fast	Medium	Fast
Memory usage	Low	Medium	Medium	Medium/High
Interpretability	Easy	Hard	Hard	Hard
Flexibility	Medium	Medium	Hard	Medium/High

The FNN contains nine inputs, nine neurons in the hidden layer, and one neuron at the output (9-9-1), with hyperbolic tangent transfer functions applied to all neurons. The results are generated based on the Mean Squared Errors (MSE) algorithm with the following default values: maximum number of iterations=1000, magnitude of the gradient $\leq 10^{-5}$, and validation checks = 6. By default, to train and test the models, MATLAB splits the data set into three subsets for training (70%), validating (15%) and testing (15%) the models. Suggested default values are used to evaluate the classifiers.

3.5. XOR detector of different decisions

In [16] the authors examine the performance of the XOR rule in detecting presence or absence of a deterministic signal. The authors present two sensors assumed to be unequal with identical marginal distribution for the noise components. This concept, shown in Fig. 1, as well as the main principles of parallel computing [46] are used to detect different decisions. The aim is to compare the classification ability of two high-precision binary classifiers working in parallel at the same place in the computer network. This is a decentralized detection system with a parallel topology [47], in which both the wk-NN and the FNN serve as sensors. To classify network traffic as normal or abnormal, classifiers detect an anomaly or consider network behavior normal. However, every now and then one of the classifiers detected an anomaly while the other determined that the network traffic is normal, and vice versa. For this reason, classifier results are subjected to the XOR operation. In the case of independent decisions, when each sensor makes its decision based on the classification rule, the XOR operation used as the fusion rule becomes meaningful. By performing the XOR operation on the predicted outputs, conflicting decisions can be detected. If one of the outputs is zero, the result is true; otherwise it is false. The outputs of the classifiers are out_{wk_NN} and out_{FNN} for wk-NN and FNN, respectively. The result of the XOR operation performed on the corresponding input data is then $out(i) = xor(out_{wk-NN}(i), out_{FNN}(i))$. The value of $out(i)$, which can be logically true (1) or false (0), is passed to the accumulator register, which is first cleared to zero and contains the sum of the results of the (p) XOR operation, where p can vary allowing for more accurate results.

The advantages of the accumulator register are shorter instructions and less memory storage since the accumulator always contains an operand. Each further output $out(i)$ is thus added to the contents of the accumulator. Another benefit is that the instruction cycle takes less time to process because the accumulator saves time fetching instructions from memory. Since the number of decisions varies and can be set by the user, the sensitivity of the additional alerts is determined not only by the detection of conflicting decisions, but also by the company's need for more granular protection mechanisms. It should be noted that while the XOR-based detector can

help in decision making, it cannot predict the likelihood of a particular event occurring. Its sole purpose is to provide additional warnings to the decision-making bodies.

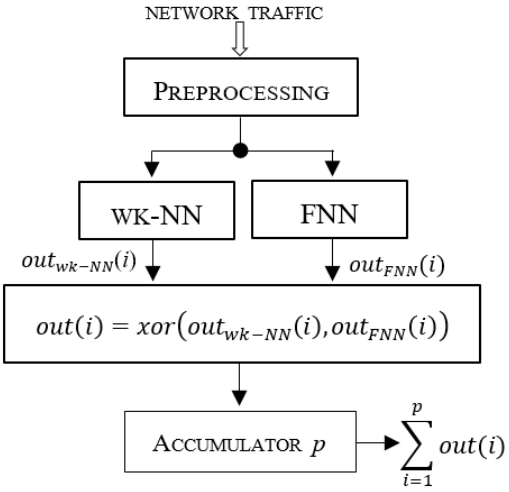

Fig. 1. Conceptual design of XOR detector.

4. Experimental Results

To compare the effects of feature selection and feature scaling on binary classification for all classifiers described, experiments are performed as follows. Models are evaluated using MATLAB Classification Learner. First, a daily record from the Kyoto 2006+ dataset with $\sim 60,000$ instances is stripped of Not-a-Number (NaN) values that MATLAB does not recognize. Then all irrelevant features are removed and $\sim 57,300$ instances is used to train and test the models; 70% is used for training, 15% is used for validation (to show if the validation performance is consistent with the training one [48]), and 15% is used for testing the models. Feature Label is used to detect anomalies; if $Label = 1$ the instance is classified as 'normal', while if $Label = -1$, the instance is classified as 'anomaly'.

Then the issue of dataset size is addressed by using feature selection to reduce the Kyoto 2006+ dataset from 24 to nine numerical features recognized as the most relevant for classifier evaluation. The literature implies that users who are knowledgeable about their dataset can select features that meet some criteria based on their knowledge and experience [49, 50]. According to this principle, feature selection proposed in this paper is performed as follows: (1) Remove all categorical features (17 features are left for model training: 1, 3-17, 24), (2) Cut out all statistical features and features for more analyses planned. Finally, features 5-13 are used to evaluate the model. Feature Label is used to indicate the presence of an attack. The original record has three labels: 1 for standard sessions, -1 for known attacks, and -2 for unknown attacks. However, since unknown attacks occur sporadically in the data set, we also assign the label -1 to unknown attacks. In our previous work, we showed the detailed description of feature selection [21]. Here we illustrate the impact on feature selection for 17 and 9 features for three-day records, recorded in 16th May 2009, 12th August 2008 and 14th February 2007, respectively (see Table 4). The

results are presented in terms of accuracy and processing time. Accuracy (ACC) is defined as the ratio of the total number of correctly classified instances to the total number of results:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}. \tag{7}$$

True Positive (TP) indicates normal network behavior has been correctly classified. *True Negative* (TN) determines the correctly classified negative results. The False Positive (FP) result determines the misclassification of normal network behavior. False Negative (FN) denotes an abnormality that has been misclassified as 'normal'. The processing time results from the sum of the training time and test time of the models ($t_p = t_{train} + t_{test}$). The accuracy and processing time for 17 and 9 features are labeled ACC-17, tp-17, ACC-9, and $t_p - 9$.

The results show the significantly shorter processing time when nine features were used for classifier evaluation compared to the processing time when 17 features were used. At the same time, the accuracy of the DT model dropped to ~ 0.001, followed by the SVM model to ~ 0.026 and the k-NN and wk-NN models to ~ 0.029. For these reasons, nine features are believed to be sufficient to be used in experiments on the effects of feature scaling. It should be noted that the feedforward neural network was not used in this part of the experiments as it only deals with numerical features.

Three feature scaling techniques are used: TH, Min-Max in range [0,1] and Min-Max in range [–1,1]. The experiments are conducted to the instances recorded on 14$^{\text{th}}$ February 2007. The importance of FP and FN is determined by the F1-*score*, a harmonic mean of two variables: *Precision* (8), which determines how many of total predicted positive results are actually positive, and *Recall* (9), which determines how many of the total predicted positives were correctly predicted:

$$Precision = \frac{TP}{TP + FP}, \tag{8}$$

$$Recall = \frac{TP}{TP + FN}. \tag{9}$$

The F1-score can be calculated as follows:

$$F1 - score = \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}}. \tag{10}$$

Table 5 shows the classification results for all models in terms of Accuracy, processing time and F1-score.

First, Min-Max normalization in range [0,1] is applied. According to the results, the FNN is the most accurate model. The DT has the shortest processing time but it is not the most accurate classifier. In terms of accuracy, the SVM performs poorly. The wk-NN is very slow but extremely accurate model. Second, to avoid problems caused by very long or very small derivatives, Min-Max normalization in range [–1,1] is used. However, this normalization does not affect the classification results. The problem of long processing time remains. Third, TH scaling is applied. All features are scaled in the same symmetrical range of ± 0.7616. Compared to other scaling techniques, the processing time for all classifiers is more than twice as using Min-Max normalization in [0,1] and [–1,1] ranges. The results support the hypothesis that TH scaling significantly accelerates training. With the exception of SVM, all models show high decision accuracy. The F1-score indicates that although the Kyoto 2006+ dataset is unbalanced,

Table 4. Accuracy and processing time for 17 and nine features

Instances	Model	ACC-17	t_p–17 [s]	ACC-9	t_p–9[s]
128,740 (16/05/2009)	k-NN	0.986	682	0.982	194
	wk-NN	0.988	690	0.981	195
	DT	0.979	379	0.978	281
	SVM	0.998	9.5	0.972	3.3
93,999 (12/08/2008)	k-NN	0.994	354	0.965	109
	wk-NN	0.995	352	0.965	109
	DT	0.984	149	0.980	112
	SVM	0.995	7	0.975	2.3
58,317 (14/02/2007)	k-NN	0.993	134	0.991	32
	wk-NN	0.994	134	0.992	32
	DT	0.992	37	0.991	34
	SVM	0.995	4.4	0.989	1.7

this normalization improves classification in terms of both accuracy and processing time. The wk-NN model shows the best F1-score results and the highest accuracy. Finally, the results show that none of the scaling techniques have any impact on the DT model. Overall, the results show that TH scaling has a significantly positive effect on the processing time of the models, with a slight decrease in F1-score and accuracy.

Table 5. Accuracy, processing time and F1-score of binary classifiers; TH scaling, and Min-Max normalization in the ranges [0,1] and [–1,1]

	TH			Min-Max ([0,1])			Min-Max ([–1,1])		
Model	*Accuracy*	t_p *[s]*	*F1-score*	*Accuracy*	t_p *[s]*	*F1-score*	*Accuracy*	t_p *[s]*	*F1-score*
FNN	0.994	5	0.989	0.9953	11	0.993	0.993	12	0.990
wk-NN	0.994	5	0.992	0.9948	103	0.993	0.996	105	0.993
DT	0.994	2	0.992	0.9947	3	0.991	0.994	2	0.991
SVM	0.991	27	0.987	0.9916	37	0.989	0.992	43	0.989

Table 6. The results of the detection about opposing decisions about anomalies in the computer network traffic

Number of instances	Number of different decisions	Different decisions [%]
57300 (03/02/2007)	1160	2.02
57300 (27/02/2007)	460	0.80
58300 (14/02/2007)	100	0.17

According to the results, high-precision classifiers should recognize network traffic equally. To examine this expectation, the fastest FNN and the most accurate wk-NN model decisions are compared. It is shown that in some cases one classifier detects anomalies while the other classifies network traffic as normal and vice versa.

A high percentage of conflicting decisions can be very useful to trigger additional alerts against potential network attacks when sensitive data needs to be protected. This is especially

true for previously unknown attacks or malicious behavior undetected by signature-based IDS. The model used for the experiments is based on bit-by-bit XOR logic operation performed on the outputs of wk-NN and FNN operating in parallel, scanning unknown network traffic simultaneously to detect the number of opposing decisions.

To demonstrate the functionality of the presented detector, we ran experiments on three-day records of unknown data from the Kyoto 2006+ dataset, containing 57,300, 57,300, and 58,300 instances, recorded in 3rd, 14th and 27th February 2007, respectively. The results are shown in Table 6.

A higher percentage of different decisions indicates greater classification uncertainty, which is not only due to the classifiers' decisions but can also be caused by data errors, unidentified attacks, and other factors, all of which that contribute to decision uncertainty. The results show that the number of different decisions is not related to the size of the daily record. Accordingly, the detector can help resolve the uncertainty by calculating the percentage of conflicting classifier decisions. However, the decision-making bodies should decide how to react to the information about possible undetected intruders.

5. Conclusions

This article presents research results to improve binary classification. Feature selection and feature scaling techniques are applied to improve accuracy and reduce processing time of four ML-based binary classifiers. Three normalization techniques are applied to the relevant features from the Kyoto 2006+ dataset. All models have proven to be very accurate. Processing time is shortened by TH scaling. The XOR-based detector is used to determine the number of opposing decisions of the FNN and wk-NN models, which work in parallel and serve as anomaly sensors. The percentage of different decisions does not depend on the number of instances and ranges from 0.17 to 2.02.

References

[1] F. ALIYU, T. SHELTAMI, M. DERICHE and N. NASSER, *Human immune-based intrusion detection and prevention system for fog computing*, Journal of Network and Systems Management volume **30**, 2020, paper 11.

[2] S. SCHALLER, J. WEINBERGER, R. JIMENEZ-HERENDIA, M. DANZER and S.-M. WINKLER, *Classification of the states of human adaptive immune systems by analyzing immunoglobin and T cell receptors using ImmunExplorer*, Computer Aided Systems Theory – EUROCAST 2015, R. Moreno-Diaz, F. Pichler and A. Quesada-Arencibia, Eds. 15th International Conference, Las Palmas de Gran Canaria, Spain, February 8-13, 2015, Revised Selected Papers, Lecture Notes in Computer Science, Springer International Publishing Switzerland, 2015, pp. 302–309.

[3] M.-R. MARINESCU, M. AVRAM, C. VOITINCU, M. SAVIN, C. MIHAILESCU and L.-D. GHICULESCU, *Electrotechnical sensors with interdigitated electrodes counting T-cells*, Romanian Journal of Information Science and Technology **23**(4), 2020, pp. 368–378.

[4] S. OMAR, A. NAGADI and H.-H. JEBUR, Machine learning techniques for anomaly detection: An overview, International Journal of Computer Applications **79**(2), 2013, pp. 33–41.

[5] A. HALIMAA and K. SUNDARKANTHAM, *Machine learning based intrusion detection system*, Proceedings of 3rd International Conference on Trends in Electronics and Informatics, Tirunelveli, India, 2019, pp. 916–920.

[6] S. RUGGIERI, *Complete search for feature selection decision trees*, Journal of Machine Learning Research **20**(104), 2019, pp. 1–34.

[7] I.-A. ZAMFIRACHE, R.-E. PRECUP, R.-C. ROMAN and E.-M. PETRIU, *Neural network-based control using actor-critic reinforcement learning and grey wolf optimizer with experimental servo system validation*, Expert Systems with Applications **225**, 2023, paper 120112.

[8] M.-G. LAGOUDAKIS and R. PARR, *Reinforcement learning as classification: Leveraging modern classifiers*, Proceedings of 20th International Conference on Machine Learning, Washington DC, USA, 2003. [Online] Available https://users.cs.duke.edu/∼parr/icml03.pdf

[9] F.-H. AWAD and M.-M. HAMAD, *Improved k-means clustering algorithm for big data based on distributed smartphone Neural Engine Processor*, Electronics **11**(6), 2022, paper 883.

[10] B. NGUYEN and B. DE BAETS,*Kernel-based distance metric learning for supervised k-means clustering*, IEEE Transactions on Neural Networks and Learning Systems **30**(10), 2019, pp. 3084–3095.

[11] M.-C CRISTESCU, *Machine learning techniques for improving the performance metrics of functional verification*, Romanian Journal of Information Science and Technology **24**(1), 2021, pp. 99–116.

[12] J. WESTON, A. ELISSEFF, B. SCHOELKOPF and M. TIPPING, *Use of the zero norm with linear models and kernel methods*, Journal of Machine Learning Research **3**, 2003, pp. 1439–1416.

[13] D. PROTIC and M. STANKOVIC, *hybrid model for anomaly-based intrusion detection in complex computer networks*, Proceedings of 21st IEEE International Arab Conference on Information Technology (ACIT), Giza, Egypt, 2020, pp. 1–8.

[14] D. PROTIC, M. STANKOVIC and V. ANTIC, *WK-FNN design for detection of anomalies in the computer network traffic*, Facta Universitatis, Series: Electronics and Energetics **35**(2), 2022, pp. 269–282.

[15] D. PROTIC, L. GAUR, M. A. RAHMAN and M. STANKOVIC, *Cybersecurity in smart cities: Detection of opposing decisions of anomalies in the computer network behavior*, Electronics **11**(3718), 2022.

[16] X. SUN, S. YAGNIK, R. VISWANATHAN and L. CAO, *Performance of XOR rule for decentralized detection of deterministic signal in bivariate Gaussian noise*, IEEE Access **10**, 2022, pp. 8092–8102.

[17] R.-A. MUSHEER, C.-V. VERNA and N. SRIVASTAVA, *Dimension reduction methods for microarray data: a review*, AIMS Bioengineering **4**(2), 2017, pp. 179-187.

[18] E. ARICAN and T. AYDIN, *An RGB-D descriptor for object classification*, Romanian Journal of Information Science and Technology (ROMJIST) **25**(3–4), 2022, pp. 338–349.

[19] M. AL-IMRAN and S. H. RIPON, *Network intrusion detection: An analytical assessment using deep learning and state-of-the-art machine learning models*, International Journal of Computational Intelligence Systems **14**, 2021, paper 200.

[20] N. BINDRA and M. SOOD, *Evaluating the impact of feature selection methods on the performance of the machine learning models in detecting DDoS attacks*, Romanian Journal of Information Science and Technology **23**(3), 2020, pp. 250–261.

[21] D. PROTIC and M. STANKOVIC, *Anomaly-based intrusion detection: Feature selection and normalization influence to the machine learning model accuracy*, European Journal of Engineering and Formal Sciences **1**(3), 2018, pp. 43–48.

[22] O. OSANAIYE, O. ORGUNDILE, F. AINA and A. PERIOLA, *Feature selection for intrusion detection system in a cluster-based heterogenous wireless sensor network*, Facta Universitatis, Series: Electronics and Energetics **32**(2), 2019, pp. 315–330.

[23] C. JIE, L. JIAWEI, W. SHULIN and Y. SHENG, *Feature selection in machine learning: A new perspective*, Neurocomputing **300**(26), 2018, pp. 70–79.

[24] I. AHMED, H. SHIN and M. HONG, *Fast content-based file type identification*, Advances in Digital Forensics VII, 2011, pp. 65–75.

[25] M. RING, S. WUNDERLICH, D. SCHEURING, D. LANDES and A. HOTHO, *A survey of network-based intrusion detection data sets*, arXiv:1903.02460v2 [cs.CR], 2019, pp. 1–17.

[26] D. PROTIC and M. STANKOVIC, *Detection of anomalies in the computer network behaviour*, European Journal of Engineering and Formal Sciences **4**(1), 2020, pp. 7–13.

[27] R.-E. PRECUP, G DUCA, S. TRAVIN and I. ZINICOVSCAIA, *Processing, neural network-based modelling of biomonitoring studies data and validation of Republic of Moldova data*, Proceedings of the Romanian Academy Series A: Mathematics, Physics, Technical Sciences, Information Science **23**(4), 2022, pp. 403–410.

[28] C. POZNA and R.-E. PRECUP, *Aspects concerning the observation process modelling in the framework of cognition process*, Acta Polytechnica Hungarica **9**(1), 2012, pp. 203–223.

[29] B. BOHARA, J. BHUYAN, F. WU and J. DING, *A survey on the use of data clustering for intrusion detection system in cybersecurity*, International Journal of Network Security & Its Applications **12**(1), 2020, pp. 1–18.

[30] T. NGUYEN and G. ARMITAGE, *A survey of techniques for Internet traffic classification using machine learning*, IEEE Communications Surveys & Tutorials **10**(4), 2008, pp. 56–76.

[31] D. PEREZ, S. ALONSO, A. MORAN, M.-A PRADA, J.-J. FUENTES AND M. DOMINGEZ, *Comparison of network intrusion detection performance using feature representation*. In: J. Macintyre, L. Illadis, I. Maglogoiannis and C. (eds.) Engineering Applications of Neural Networks. EANN 2019. Communications in Computer and Information Science **1000**, 2019, pp. pp 463-475.

[32] L. HARDESTY, *Explained: neural networks*, MIT News on campus and around the world, 2017. [Online] Available: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414.

[33] K. DEMERTZIS, *The BRO intrusion detection system*, Project: Machine Learning to Cyber Security, 2018.

[34] Y. LECUN, L. BOTTOU, G.-B. ORR and K.-R. MULLER, *Efficient BackProp*, Neural Computation **4**, 1992, pp. 141–166.

[35] J. SONG, H. TAKAKURA, Y. OKABE, M. ETO, D. INOUE and K. NAKAO, *Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation*, Proceedings of 1st Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, Salzburg, Austria, 2011, pp. 29–36.

[36] R. SINGH, H. KUMAR and R.-K. SINGLA, *An intrusion detection system using network traffic profiling and online sequential learning machine*, Expert Systems with Applications **42**(2), 2015, pp. 8609–8624.

[37] K. PARK, Y. SONG and Y. CHEONG, *Classification of attack types for intrusion detection systems using a machine learning algorithm*, Proceedings of 2018 IEEE 4th International Conference on Big Data Computing Service and Applications, Bamberg, Germany, 2018, pp. 282–286.

[38] S. KHALID, T. KHALIL and S. NASREEN, *A survey of feature selection and feature extraction techniques in machine learning*, Proceedings of 2014 Science and Information Conference, London, UK, 2014, pp. 372–378.

[39] K. LEVENBERG, *A method for the solution of certain problems in least squares*, Quarterly of Applied Mathematics **5**, 1944, pp. 164–168.

[40] D. MARQUARDT, *An algorithm for least-squares estimation of nonlinear parameters*, SIAM Journal in Applied Mathematics **11**(2), 1963, pp. 431–441.

[41] J. IBANEZ, J.-M. ALONSO, J. SASTRE, E. DEFEZ and P. ALONSO-JORDA, *Advances in the approximation of the matrix hyperbolic tangent*, Mathematics **9**, 2021. paper 1219.

[42] W. HARPER, *Newton's methodology*. In Quantum Reality, Relativistic Causality and Closing the Epistemic Circle, The Western Ontario Series in Philosophy of Science **73**, Springer Dordrecht, 2009, pp. 43–61.

[43] D. PROTIC and M. STANKOVIC, *The q-Levenberg-Marquardt method for unconstrained nonlinear optimization*, pp. 1–5, 2021. [Online] Available: http://arxiv.org/abs/2107.03304.

[44] Classification learner. Accessed November 28, 2022. [Online] Available: https://uk.mathworks.com/help/stats/classification-learner-app.html

[45] S.-A. DUDANI, *The distance-weighted k-Nearest-Neighbour rule*, IEEE Transactions on Systems, Man, and Cybernetics **SMC–6**(4), 1976, pp. 325–327.

[46] G.-M. STEFAN and M. MALITA, *Can one-chip parallel computing be liberated from ad hock solutions? A computational model based approach and its implementation*, Advances in Information Science and Applications **2**, 2014, pp. 582-597.

[47] H.-N. L. TEODORESCU, *Sensors based on nonlinear dynamic systems – A survey*, Proceedings of 2017 International Conference on Applied Electronics, Pilsen, Czech Republic, 2017, pp. 1–10.

[48] R.-E. PRECUP, C.-A. BOJAN-DRAGOS, E.-L. HEDREA, R.-C. ROMAN and E.-M. PETRIU, *Evolving fuzzy models of shape memory alloy wire actuators*, Romanian Journal of Information Science and Technology **24**(4), 2021, pp. 353–365.

[49] A. RAHMAN and Z. ISLAM, *AWST: A novel attribute weight selection technique for data clustering*, Proceedings of 13th Australasian Data Mining Conference, Sidney, Australia, 2015, pp. 51–58.

[50] M. A. RAHMAN and M. Z. CRUDAW, *A novel fuzzy technique for clustering records following user defined attribute weights*, Proceedings of 10th Australasian Data Mining Conference, Sydney Australia, 2012, pp. 27–42.