DETECTION OF ANOMALIES IN THE COMPUTER NETWORKS: FEATURE SCALING METHODOLOGY

Danijela D. Protic[1], Miomir S. Stankovic[2]

1 Centre for Applied Mathematics and Electronics, Vojvode Stepe 443, 11000 Belgrade, Serbia
e-mail: danijelaprotic318@gmail.com
[2] Mathematical Institute of SASA, Knez Mihajlova 35, 11000 Belgrade, Serbia
e-mail: miomir.stankovic@gmail.com

Abstract:

In order to detect unusual network activity, anomaly-based intrusion detection systems monitor computer network behaviour. The main challenge in detecting anomalies is determining the notion of normality. Binary classifiers can be trained to categorize unknown data as normal or abnormal. Supervised machine learning has increased the efficiency and precision in detecting anomalies. The main advantage of anomaly-based classification is that unknown attacks can be detected. The disadvantage is that developing a network behaviour model takes time since a large amount of training data is required to ensure reliable results. In binary classification, it is critical to reduce the size of the dataset. Feature scaling is widely used for reducing model complexity. This paper describes a new scaling technique based on tangent hyperbolic min-max normalization and the principle of the Levenberg-Marquardt (LM) algorithm. The experiments are carried out on the feedforward neural network (FNN), k-nearest neighbour (k-NN), weighted k-NN (wk-NN) and support vector machine (SVM) models trained on daily records from the Kyoto 2006+ dataset. All models are shown to be highly accurate. The tangent-hyperbolic min-max feature scaling outperforms the Z-score standardization and Min-Max normalization in terms of the processing time.

Key words: intrusion detection, machine learning, feature scaling, binary classification, feedforward neural networks, Kyoto 2006+ dataset

1. Introduction

Anomaly-based intrusion detection systems (IDSs) monitor behaviour of the computer network and generate alerts when unusual activity is detected [1]. The main challenge in detecting anomalies is developing a statistical model that determines the notion of normality and triggers an alarm when abnormal behaviour is detected. The main difficulties in detecting anomalies are determining 'normality' and learning distinctions between normal and abnormal network behaviour [2]. The main advantage of anomaly-based IDS is that it can detect unknown attacks. The disadvantage, on the other hand, is that building a network behaviour model takes time. The basic idea behind anomaly detection is to build a high-precision classifier that is trained on a small number of instances, runs in real time, and detects deviations from normal network behaviour [3]. Binary classifiers based on supervised machine learning (ML) can be trained to classify unknown network traffic into normal or abnormal data categories [4]. The k-NN models are the simplest classifiers with the performance dependent on the number of neighbours, distance metrics and the decision rule. The k-NN algorithm's disadvantages include a costly test phase and high reliance on the dataset size

[5]. The wk-NN algorithm extends the k-NN algorithm so that the instances within the learning set that are especially close to the new instance should be weighted more heavily in the decision [1]. If the interpretability of the model is not a primary concern, FNNs are widely used in classification due to their universal approximation property. One of the fastest FNNs updates the weights using the LM optimization algorithm, which combines the gradient descent (GD) and Gauss-Newton (GN) algorithms to perform a complex training process [6]. To classify the instances, the SVM algorithm uses a hyperplane in *n*-dimensional space [7]. The instances on opposite sides of the hyperplane are assigned to different classes. However, a large amount of training data is required to that all classifiers produce reliable results. In binary classification, reducing the size of the data set is an important task. As a result, feature scaling is frequently used to reduce model complexity [8]. This paper examines the impact of feature scaling on classification using the k-NN, wk-NN, FNN and SVM models. Daily records from the Kyoto 2006+ dataset are used to train classifiers [9]. The idea was to go into the dimensionality reduction to redefine the feature vector before it was used for training [10]. The Z-score standardization and Min-Max normalization have a negative impact on training in terms of processing time and number of false positives. Tangent hyperbolic min-max feature scaling speeds up training and leads to faster convergence of the classification algorithms, according to results.

2. Related work

Many research challenges in data analysis, visualisation, and understanding relate to the availability and usability of the multidimensional data, as the nature of trait influence anomaly detection [11]. Prior to classification, the pre-processing steps reduce and scale features, which helps improving the properties of the ML models. Dimensionality reduction, feature scaling, binary classification and the data collection are presented in the text that follows.

2.1 Dimensionality reduction and feature scaling

Dimensionality reduction is a common pre-processing step before classification that reduces the number of features to the most important ones. In [12], the authors propose the feature selection method to save storage space and speed up the classification algorithms. They compared neural network, k-NN and SVM models and showed that many of them had over 50% faster processing time with ~90% accuracy when only half of all features were used. In [13], the authors summarize the research on DT, SVM, FNN, and nearest neighbour models and describe the effect of instance normalization in the range ± 1. The authors emphasize that using feature selection correctly can significantly improve processing time and classification performance.

Prior to classification, feature scaling is frequently used in addition to dimensionality reduction. If no feature scaling is done, ML algorithms tend to weight higher values and treat smaller values as lower values regardless of their units. Normalization and standardization are the two most common methods for feature scaling. The goal of normalization is to ensure that each data point has the same scale. Features with different scales may cause model to diverge, overestimate, underestimate or ignore some parameters, and reduce the efficiency of the estimation. Normalization is useful when there are no outliers in the dataset and when the distribution of the data set is known to be non-Gaussian, which can be useful in nearest neighbour algorithms. In [14] and [15], the authors describe Min-Max normalization to detect network intrusion using ML models on the selected Kyoto 2006+ dataset. Normalization is used before splitting the dataset and after balancing to avoid bias caused by outliers in the imbalanced dataset because minimum and maximum values of the features are unknown. In contrast to normalization, the Z-score standardization focuses on values that are centred around the zero mean and have unit standard deviation. It is useful when the model follows a Gaussian distribution but does not scale features in the same range. In [8], the authors present a Z-score scaling method that can be used as a pre-processing step in anomaly-based intrusion detection.

2.2 Classification models

Supervised ML algorithms are widely used for anomaly-based classification of computer network traffic. The SVM model uses the hyperplane in *n*-dimensional space to classify instances [16-18]. The instances falling on the different sides of the hyperplane are assigned to different classes. The k-NN algorithm identifies a sample based on its *k* neighbours and compute the distances between them [19], [20]. The parameter *k* affects the performance of the classifier. The model is prone to overfitting when *k* is small. A large value of *k* may cause the instance to be misclassified [21]. In [22], the authors describe the k-NN algorithm, which assigns objects to the class that contains the majority of their nearest neighbours in the feature space. Weighted k-NN model extends k-NN so that the instance from the training set that is close to the new instance has a higher weight in the decision than those that are further away [1]. The neural network's output is determined by the prediction probability and the classification threshold [20], [23]. Calculation is carried out by forwarding the input data to calculate the outputs, and propagating the error of cost function backward to adjust the weights [24]. In [25], the authors present statistical data for publication citations from 2005 to 2020 using various ML approaches to intrusion detection. With ∼27,000 citations in research, SVM-based articles (1716 in total) have the most cited topics. More than 2,000 publications on neural network-based intrusion detection have received over 21,000 citations (252 publications and 5,496 citations). The authors also demonstrate that ML-based intrusion detection has significantly more references than statistical, knowledge-based and biotechnology-based approaches. Because of the results stated above, this article refers to the SVM, k-NN, wk-NN and FNN classifiers.

2.3 Data collection

The researchers have conducted experiments on various data sets, over the years, including AFDA-LF/WD, AWID, CAIDA, CIC-IDS-2017, CSE-CIC-2018, ISCX2012, KDD CUP '99, NSL-KDD and UNSW-NB15 [8], [14], [25-28]. However, those datasets are simulations of real network traffic and neither is solely used for experiments involving anomaly-based intrusion detection. On the other hand, the Kyoto 2006+ dataset contains the actual data that is collected from ∼350 honeypots, including two darknet sensors with ∼300 unused IP addresses and other IDSs installed on five different computer networks [14], [26], [29]. The first part of the Kyoto 2006+ dataset, consisting ∼90 million instances with 24 features, was collected between 2006 and 2009. Fourteen statistical features were derived from the KDD-Cup '99 dataset. The authors ignored all features with redundant and duplicate records and added ten new features that were used to detect the anomalies [30]. Furthermore, ∼20 GB of data, collected from November 2009 to December 2015, was added to the original dataset. The IDS Bro was used to convert packet-based traffic data to a session-based format [4]. The dataset contains DoS, exploits, malware, port scans and shell code attacks against honeypots, but does not contain any information about specific attacks. Instead, the feature Label is used to indicate whether or not the attack is present [31]. The original data set is labelled with three types of labels: 1 for normal sessions, -1 for known attacks, and -2 for unknown attacks. However, because unknown attacks are extremely rare in the dataset (∼0.7%), for the purpose of the experiments we assigned the same label (-1) to both known and unknown attacks.

3. Tangent-hyperbolic min-max feature scaling

Various feature scales can lead to biased and incorrect results for classifiers that make decisions based on the Euclidean distance between two points (such as nearest neighbour models). Also, when the GD is an optimization algorithm (classifiers such as neural networks), it can update a specific weight faster than others, so the feature scaling can assist the GD to converge faster. Standardization and normalization are two types of scaling that help to optimize classifiers' optimization.

The Z-score standardization strategy rescales the features to have a zero mean and a standard deviation of one [32]. Consider the feature vector **X** containing n instances $x(i)$, where i=1, …, n. The Z-score standardization rescales features using Eq. 1

$$x(i)_{Z-score} = \frac{x(i)-mean(\mathbf{X})}{std(\mathbf{X})}, \quad (1)$$

where $x(i)_{Z-score}$ denotes the standardized instance and *mean*(**X**) and *std*(**X**) represent the mean value and standard deviation of **X**, respectively.

Min-max normalization is a type of features scaling in the range [0,1] (Eq. 2) or [-1,1] (Eq. 3) that helps to reduce the effects of the outliers by providing lower standard deviations in the output [12], [33].

$$x(i)_{[0,1]} = \frac{x(i)-x_{min}}{x_{max}-x_{min}}, \quad (2)$$

$$x(i)_{[-1,1]} = \frac{x(i)-\frac{x_{max}+x_{min}}{2}}{\frac{x_{max}-x_{min}}{2}}, \quad (3)$$

where $x(i)_{[0,1]}$ represents an instance normalized into the range [0,1], $x(i)_{[-1,1]}$ denotes an instance normalized into the range [-1,1], while x_{max} and x_{min} represent maximum and minimum of the feature **X**, respectively. Min-max normalization is often performed on features as a pre-processing step to the classification when the distances should be calculated, or in regression algorithms when the coefficients should be prepared. When the neural networks are used for labelled data classification, it is one of the most commonly applied feature scaling methodology.

In this paper we propose a novel tangent hyperbolic min-max normalization (THN) approach based on the principle of the LM algorithm used to update FNN weights, which ensures scaling the features into a fixed range, avoids zigzagging by changing the direction of the updates based on gradients, and speeds up the training [2], [34], [35]. In [31], the authors discuss faster convergence of the zero-centred, *S*-shaped function when parameter movements over a surface area are in a specific direction and average of the input variable over the training set is close to 0.

Since the tangent hyperbolic function $tanh(x) = \frac{e^x-e^{-x}}{e^x+e^{-x}}$ is the *S*-shaped function that is centred around zero and bounded in the range [-1,1], it is applied here. It should be noted that the first derivative of the $tanh(x)$ is $tanh(x)' = \frac{d}{dx}(tanh(x)) = 1 - tanh(x)^2$. Fig. 1 depicts both $tanh(x)$ and $tanh(x)'$.

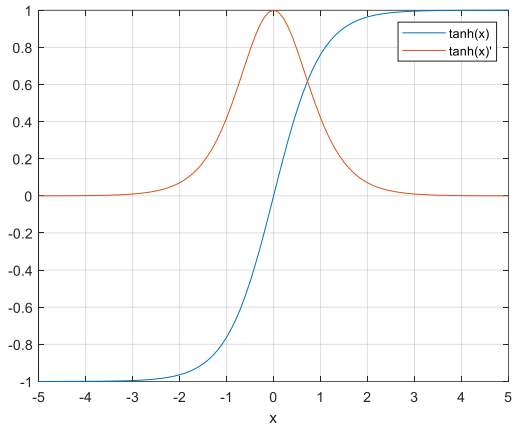

Fig. 1. Hyperbolic tangent function and its derivative

The THN scaling normalizes features according to the following formula:

$$x(i)_{THN} = tanh\left(\frac{x(i)-\frac{x_{max}+x_{min}}{2}}{\frac{x_{max}-x_{min}}{2}}\right) \quad (4)$$

where $x(i)_{THN}$ represents the instance normalized in the range $[tanh(-1), tanh(1)]$, i.e. $[-0.7616, 0.7616]$. The main idea of the THN is to use the damping strategy properties from the LM algorithm, which is explained in the text that follows. Consider the quasi-linear part of the *tanh*(*x*) in the range $[-0.7616, 0.7616]$ (see Fig. 2) and concentrate on the effects of THN scaling on gradient-based training of the classifiers. Consider a FNN with one hidden layer containing neurons with tangent hyperbolic activation functions (weights). Note that the product of $x(i)_{THN}$ and the corresponding weight can never be ± 1.

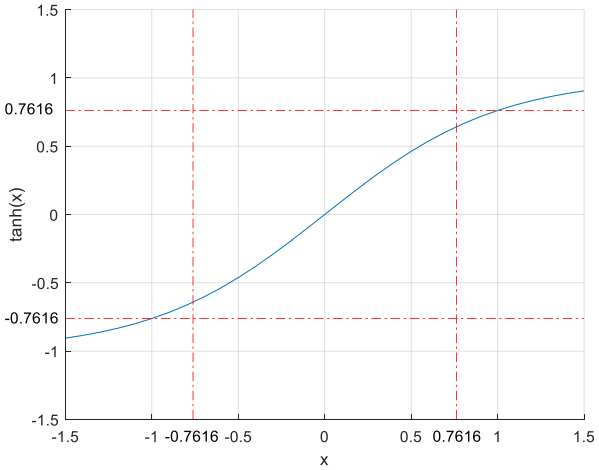

Fig. 2. Quasi-linear part of hyperbolic tangent function

The LM algorithm is a type of non-linear iterative methods that combines the GD algorithm, which minimizes the objective function based on the step length and the search direction determined by the negative of the gradient, and fast GN algorithm, which determines the function's inflection point based on the second-order derivative, which simplifies calculation of the Hessian matrix by assuming that the error function is approximately quadratic close to the optimal solution and using the introduced Jacobian matrix (**J**) [33]. The idea behind the LM algorithm is to perform an affine transformation of function $f: R \to R$ near the point *p*, using the first-order truncated Taylor formula [36] shown below

$$\hat{f}(x,p) = f(p) + Df(p)(x-p) \quad (5)$$

where $Df(p)(x-p)$ denotes the Jacobian matrix of the partial derivatives of *f*. The LM algorithm approximates the problem so that $\hat{f}(x,p) \approx f(x)$ and $x \approx p$, at the same time. Let $x^{(1)}, \cdots, x^{(k)}$ represent the iterates. The iterate $x^{(k+1)}$ should be chosen to minimize the first and second term in the expression

$$\left\|\hat{f}(x,x^{(k)})\right\|^2 + \lambda^{(k)}\left\|(x,x^{(k)})\right\|,\ \lambda^{(k)} > 0, \quad (6)$$

where $\lambda^{(k)}$is damping factor that varies with the step size. Consider the solution of the problem $x^{(k+1)}$.

The iterate $x^{(k)}$ determines stationary point when $x^{(k+1)} = x^{(k)}$. Then $x^{(k+1)}$ can be calculated as follows:

$$x^{(k+1)} = x^{(k)} - \left(Df(x^{(k)})^{\mathrm{T}} Df\left(x^{(k)}\right) + \lambda^{(k)}\mathbf{I}\right)^{-1},\ \lambda^{(k)} > 0. \quad (7)$$

The parameter **I** stands for identity matrix. The solution holds true if and only if $Df(x^{(k)})^{\mathrm{T}} Df\left(x^{(k)}\right) = 0$. The damping factor $\lambda^{(k)}$ can be adjusted as follows:

- if $\lambda^{(k)}$ is too large it should be increased because $x^{(k+1)}$ is too close to $x^{(k)}$ (usually $\lambda^{(k+1)} = 2\lambda^{(k)}$);
- if $\lambda^{(k)}$ is too small it should be decreased because $x^{(k+1)}$ is too far from $x^{(k)}$ and the approximation is poor (usually $\lambda^{(k+1)} = 0.5\lambda^{(k)}$).

The iterate $x^{(k+1)}$ can be determined using the LM algorithm as follows

$$x^{(k+1)} = x^{(k)} - \left(\mathbf{H} + \lambda^{(k)}\mathbf{I}\right)^{-1}\mathbf{J}^{\mathrm{T}} f\left(x^{(k)}\right) = x^{(k)} - \left(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \lambda^{(k)}\mathbf{I}\right)^{-1}\mathbf{J}^{\mathrm{T}} f\left(x^{(k)}\right), \quad (8)$$

where $\mathbf{H} \approx \mathbf{J}^{\mathrm{T}}\mathbf{J}$ represents approximation of the Hessian matrix. The LM algorithm switches between GD and GN algorithms (damping strategy) depending on $\lambda^{(k)}$ as follows:

- if $\lambda^{(k)} \to \infty \Rightarrow \mathbf{H} + \lambda^{(k)}\mathbf{I} \to \mathbf{I}$ and the LM algorithm behaves as the GD algorithm,
- if $\lambda^{(k)} \to 0 \Rightarrow x^{(k)}$ is close to the optimal solution and the LM algorithm behaves like the GN algorithm.

The damping strategy involves switching between the GN and GD algorithms [37]. The main idea behind THN scaling is to use a damping approach to adapt the features to the classifier inputs in order to accelerate the weight adjustment and eliminate impacts on very small and very large gradients. Note that the NTH normalization affects training of most gradient-based classification algorithms.

4. Results and discussion

The effects of THN scaling on decision about anomalies of four binary classifiers are demonstrated using MATLAB Classification Learner. The models' characteristics are as follows: (1) k-NN (k=10), (2) wk-NN (k=10; weights – squared inverse distance), (3) SVM (medium) and (4) FNN with one hidden layer (9 neurons in both input and hidden layers and one output neuron, weights – tangent hyperbolic). The experiments are carried out on ~57,000 instances of nine features from the Kyoto 2006+ dataset. The features are initially released from not-a-number (NaN) values. Then, all irrelevant features were removed (categorical features, connection duration features, statistical features and features for further analyses). Finally, nine numerical features left for training: Count, Same_srv_rate, Serror_rate, Srv_error_rate, Dst_host_count, Dst_host_srv_count, Dst_host_same_src_port_rate, Dst_host_serror_rate, and Dst_host_srv_serror_rate. Feature Label determined whether the network traffic was (1) normal or abnormal (-1). Finally, the relevant features were scaled using Z-score standardization, Min-Max normalization in [0,1] and [−1,1] ranges, and THN. 70% of instances are used to train models, while 30% are used to test them. The classification results are compared to the corresponding labels and the performance of the model is measured in terms of true positive (TP), true negative (TN), false positive (FP) and false negative (FN) results, as well as processing time (t_p). The TP denotes correctly classified normal network behaviour. TN determines the correctly identified negative results. FP denotes normal network behaviour misclassified as anomaly, whereas FN denotes abnormal network behaviour incorrectly assigned to the class 'normal'. The processing time represents time spent to training and testing the classifiers. The model accuracy (ACC) is determined as the ratio of correctly classified results to the total number of the results, as shown below

$$\mathrm{Acc} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}. \quad (9)$$

When FP and FN are both of equal importance, as in the unbalanced Kyoto 2006+ dataset, the weighted harmonic mean F_β of two variables, Precision and Recall, is used as a classification quality measure. Precision is concerned with FP and determines how many of the total predicted positives are actually positives (Eq. 10).

$$\mathrm{Precison} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}, \quad (10)$$

while Recall focuses on FN results and determines how many results out of total predicted positives are correctly predicted (Eq. 11).

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}. \tag{11}$$

The F_β is given with the following formula

$$F_\beta = \frac{1}{\beta\frac{1}{\text{Precision}}+(1-\beta)\frac{1}{\text{Recall}}}. \tag{12}$$

When $\beta = 1$, F_β is referred to as the F1-score (F1). In this case Precision and Recall are both of equal importance in the F1-score.

The Z-score standardization is used for the experiments to scale the features so that the instances are centred around zero-mean with unit standard deviation. To solve problems caused by different scales, the Min-Max normalization into the range [0,1] is used. Furthermore, the Min-Max normalization into the range ± 1 is used to resolve problems caused by very large or very small derivatives. The THN is used because of three important properties: (1) the features are scaled in the same range, (2) the *tanh* function speeds up training and (3) zigzagging is avoided. Accuracy, processing time and F1-score are given in Table 1.

	Z-standardization			**Min-Max [0,1]**			**Min-Max [-1,1]**			**THN**		
	ACC	**t_p [s]**	**F1**	**ACC**	**t_p [s]**	**F1**	**ACC**	**t_p [s]**	**F1**	**ACC**	**t_p [s]**	**F1**
k-NN	0.9943	102.35	0.99110	0.9945	107.35	0.99212	0.9941	103.2	0.99198	0.9930	56.1	0.99011
wk-NN	0.9948	103.25	0.99259	0.9948	102.73	0.99263	0.9958	105.3	0.99287	0.9940	56.34	0.99164
SVM	0.9922	36.28	0.98885	0.9915	36.99	0.98933	0.9917	43.39	0.98889	0.9910	26.89	0.98694
FNN	0.9943	12	0.99176	0.9953	11	0.99328	0.9931	12.12	0.99042	0.9936	5	0.98886

Table 1. Accuracy, processing time and F1-score

The models presented in the Table 1 are proven to be highly accurate, with ACC $\geq$ 99.1%, resulting from both feature selection and feature scaling. It should be noted that the selection of numerical features is critical in the classification. The results show that the THN scaling has a clear positive effect on processing time (see Fig. 3) for all the models, at the cost of slight degradation of the classifier accuracy for ~0.1% and F1-score (< 0.5%).

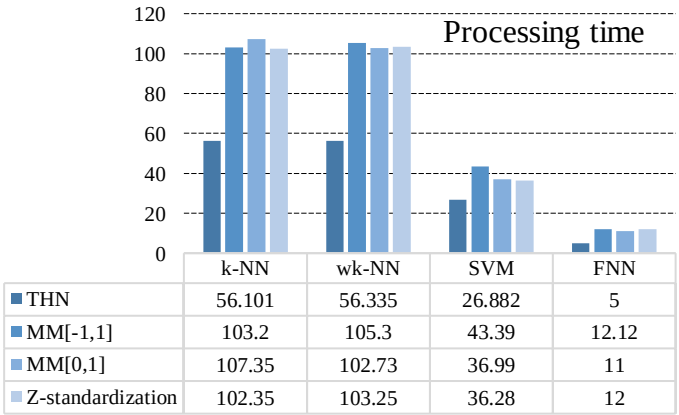

Fig. 3. Processing time

The results also support the hypothesis that it is reasonable to use the THN scaling to accelerate training because it reflects not only to the FNN but also other classifiers. In the case of nearest neighbour classifiers, which are lazy learners, THN scaling can be used to reduce the processing time for more than twice.

5. Conclusion

The impact of feature selection and scaling on binary classification using k-NN, wk-NN, SVM and FNN models is discussed in this paper. Classifiers are trained using daily records from the Kyoto 2006+ dataset. Categorical features, connection duration features, statistical features and features for further analyses are removed from the dataset. Finally, nine numerical features left to train the models. The feature Label classifies network traffic as normal or abnormal. To scale the relevant features, Z-score standardization, Min-Max normalization in [0,1] and [−1,1] ranges, and THN were used. The models are highly accurate as a result of both feature selection and feature scaling. The results show that THN scaling has a clear positive effect on processing time for all models, at the expense of slight decrease in classifier accuracy.

References

[1] Protic D, Stankovic M, Detection of Anomalies in the Computer Network Behaviour. European Journal of Engineering and Formal Sciences, 4(1):7-13, 2020.
[2] Omar S, Ngadi A, Jebur HH, Machine Learning Techniques for Anomaly Detection: An Overview. International Journal of Computer Applications 79(2):33-41, 2013.
[3] Alhakami W, Alerts Clustering for Intrusion Detection Systems: Overview and Machine Learning Perspectives. International Journal of Advanced Computer Science and Applications, 10(5):573-582, 2019.
[4] Zhou SK, Medical Image Recognition, Segmentation and Parsing, 2016.
[5] Mighan SN, Kahani MA, A novel scalable intrusion detection system based on deep learning. Int. J. Inf. Secur. 20:387–403, 2021. https://doi.org/10.1007/s10207-020-00508-5.
[6] Protic D, Feedforward neural networks: The Levenberg-Marquardt optimization and the optimal brain surgeon pruning. Military Technical Courier 63(3):11-28, 2015.
[7] Serkani E, Gharaee H, Mohammadzadeh N, Anomaly detection using SVM as classifier and DT for optimizing feature vectors. ISeCure 11(2):159-171, 2019.
[8] Ali PJM, Faraj RH, Data Normalization and Standardization: A Technical Report. Machine Learning Technical Reports 1(1):1-6, 2014. https://doi.org/10.13140/RG.2.2.28948.04489.
[9] Song J, Takakura H, Okabe Y, Eto M, Inoue D, Nakao K, Statistical Analysis of Honeypot Data and Building of Kyoto 2006+ Dataset for NIDS Evaluation. Proc. 1st Work-shop on Building Anal. Datasets and Gathering Experience Returns for Security, Salzburg, pp.29-36, 2011. https://doi.org/10.1145/1978672.1978676.
[10] Bhandari A, Feature Scaling for Machine Learning: Understanding the Difference Between Normalization vs. Standardization, 2020. https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/.
[11] Musheer RA, Verma CK, Srivastava N, Dimension reduction methods for microarray data: a review. AIMS Bioengineering 4(2):179-107, 2017. http://dx.doi.org/10.3934/bioeng.2017.1.179.
[12] Al-Imran M, Ripon SH, Network Intrusion Detection: An Analytical Assessment Using Deep Learning and State-of-the-Art Machine Learning Models. Int. J. Comput. Intell. Syst. 14:200, 2021. https://doi.org/10.1007/s44196-021-00047-4.
[13] Protic D, Stankovic M, Anomaly-Based Intrusion Detection: Feature Selection and Normalization Instance to the Machine Learning Model Accuracy. European Journal of Engineering and Formal Sciences 1(3):43-48, 2018.

[14] Osanaiye O, Ogundile O, Aina F, Periola A, Feature selection for intrusion detection system in a cluster-based heterogeneous wireless sensor network, Facta Universitatis, Series: Electronics and Energetics 32(2):315-330, 2019. https://doi.org/10.2298/FUEE19023150.
[15] Zhao Z, Liu H, Semi-supervised feature selection via spectral analysis. Proceedings of SIAM International Conference on Data Mining, 2007.
[16] Jie C, Jiawei L, Shulin W, Sheng Y, Feature selection in machine learning: A new perspective. Neurocomputing, 300, 26:70-79, 2018. https://doi.org/10.1016/j.neucom.2017.11.077.
[17] Ahmed I, Shin H, Hong M, Fast Content-Based File Type Identification. Project Digital Forensics, 2011. http://doi,org/10.1007/978-3-642-24212-0_5.
[18] Khraisat A, Gondal I, Vamplew P, Kamruzzaman J, Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity 2-20, 2019.
[19] Parmigiani G, International Encyclopedia of the Social & Behavioral Sciences, 2001.
[20] Bohara B, Bhuyan J, Wu F, Ding J, A Survey on the Use of Data Clustering for Intrusion Detection System in Cybersecurity. Int. J. Netw. Secur. Appl., 12(1):1-18, 2020.
[21] Ring M, Wunderlich S, Scheuring D, Landes D, Hotho A, A Survey of Network-based Intrusion Detection Data Sets, arXiv:1903.02460v2 [cs.CR]:1-17, 2019.
[22] Biswas SK, Bordoloi M, Purkayastha B, Review of Feature Selection and Classification using Neuro-Fuzzy Approaches. International Journal of Applied Evolutionary Computation 7(4):28-44, 2016. http://dx.doi.org/10.4018/IJAEC.2016100102.
[23] Bhati BS, Rai CS, Analysis of Support Vector Machine-based Intrusion Detection Techniques, Arab. J. Sci. Eng. 45:2371–2383, 2020. https://doi.org/10.1007/s13369-019-03970-z.
[24] Nguyen T, Armitage G, A Survey of Techniques for Internet Traffic Classification using Machine Learning. IEEE Commun. Surveys Tutorials 10(4):56-76, 2008.
[25] Perez D, Alonso S, Moran A, Prada MA, Fuentes JJ, Domingez M, Comparison of Network Intrusion Detection Performance Using Feature Representation. In: Macintyre J., Illadis L., Maglogoiannis I. and Jayne C. (eds.) Engineering Applications of Neural Networks. EANN 2019. Communications in Computer and Information Science, 1000, 2019. https://doi.org/10.1007/978-3-030-20257-6_40.
[26] Hardesty L, Explained: Neural Networks, MIT News on campus and around the world, 2017. https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414.
[27] Demertzis K (2018) The Bro Intrusion Detection System, Project: Machine Learning to Cyber Security, 2018. https://doi.org/10.31140/RG.2.2.35333.40168.
[28] LeCun Y, Bottou L, Orr GB, Muller KR, Efficient BackProp. Neural Computation 4:141-166, 1992.
[29] Wang Y, Liu D, Huang TS, Chapter 6: Signal Processing, in: Zhangyang Wang, Yun Fu, Thomas S. Huang (eds.) Computer Vision and Pattern Recognition, Deep Learning Through Sparse and Low-Rank Modeling, Academic Press, 121-142, 2019. https://doi.org/10.1016/B978-0-12-813659-1.00006-8.
[30] SIGKDD - KDD Cup, KDD Cup 1999: Computer network intrusion detection, 2018. www.kdd.org.
[31] Lai KK, Mishra SK, Ram B, On q-Quasi-Newton's Method for Unconstrained Multiobjective Optimization Problems. Mathematics 8(616):1-14, 2020. https://doi.org/10.3390/math8040616.
[32] Kumar YV, Kamatchi K, Anomaly Based Network Intrusion Detection Using Ensemble Machine Learning Technique. International Journal of Research in Engineering, Science and Management 3(4):290-297, 2020.
[33] Garcia S, Luengo J, Herera F, Data Preparation Basic Models. In: Data Preprocessing in Data Mining. Intelligent System Reference Library 72:39-57, 2015. https://doi.org/10.1007/978-3-319-10247-4_3.
[34] Obaid HS, Dheyab SA, Sabry SS, The Impact of Data Pre-Processing Techniques and Dimensionality Reduction on the Accuracy of Machine Learning, 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON): 279-283, 2019. https://doi.org/10.1109/IEMECONX.2019.8877011.
[35] Thakkar A, Lohiya R, A Review of the Advancement in the Intrusion Detection Datasets. International Conference on Computational Intelligence and Data Science (ICCIDS 2019). Procedia Computer Science 167:636-645, 2020.

[36] Lampton M, Damping-undamping strategies for Levenberg-Marquardt least-squares method, Computers in Physics 11(1):110-115, 2019.
[37] Bidgoli AA, Ebrahimpour-komleh H, Rahnamayan S, A novel binary many-objective feature selection algorithm for multi-label data classification. International Journal of Machine Learning and Cybernetics 12:2041-2057, 2012. https://doi.org/10.1007/s13042-021-01291-y.