Afyon Kocatepe Üniversitesi Fen ve Mühendislik Bilimleri Dergisi
Afyon Kocatepe University – Journal of Science and Engineering
https://dergipark.org.tr/tr/pub/akufemubid

Araştırma Makalesi / Research Article
DOI: https://doi.org/10.35414/akufemubid.1455995
AKÜ FEMÜBİD 24 (2024) 055104 (1166-1180)
AKU J. Sci. Eng. 24 (2024) 055104 (1166-1180)

Intrusion Detection System Application with Machine Learning

*Makale Bilgisi / Article Info
Alındı/Received: 20.03.2024
Kabul/Accepted: 11.07.2024
Yayımlandı/Published: xx.xx.xxxx

Makine Öğrenmesi ile Saldırı Tespit Sistemi Uygulaması

Mehmet HACIBEYOĞLU *, **Ferda Nur ARICI**, **Muhammed KARAALTUN**

Necmettin Erbakan Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Konya, Türkiye

Abstract

Information security holds paramount importance for organizations and users alike, safeguarding against unauthorized access to sensitive data. Daily usage of the internet amplifies the importance of security measures and the detection of malicious activities. Cyber-attacks, as these malicious activities are commonly known, are continually evolving with advancements in hardware, software, and complex network algorithms. Intrusion Detection Systems play a crucial role in shielding data and information from cyberattacks. The rapid progression in machine learning and deep learning, two popular methodologies in data mining, has found applications in various fields, including security. This study focuses on the use of machine learning and deep learning methods to design an intelligent intrusion detection system. For the development of this smart intrusion detection system, two well-established datasets, NSL-KDD and Kyoto 2006+, were employed. Machine learning methods were implemented utilizing the classification algorithms available in the WEKA data mining tool. The results obtained from these classification algorithms were compared with the deep learning model designed within the scope of the study. Consequently, a detailed analysis of machine learning and deep learning methods on the NSL-KDD and Kyoto 2006+ datasets for an intelligent intrusion detection system was conducted, and suggestions were proposed for further research endeavors.

Keywords: *Deep learning, Intrusion detection system, Kyoto 2006+, Machine learning, NSL-KDD*

Öz

Bilgi güvenliği, her organizasyon ve kullanıcı için bilgilere yetkisiz erişime karşı koruma sağlamak açısından son derece önemlidir. İnternet, her gün geniş bir alanda kullanılmaktadır. Bu kullanım arttıkça, güvenlik ve kötü niyetli faaliyetleri tespit etmenin önemi de artmaktadır. Bu kötü niyetli faaliyetler, siber saldırılar olarak adlandırdığımız, donanım, yazılım ve karmaşık ağ algoritmalarının gelişimiyle gün geçtikçe değişmekte ve gelişmektedir. Saldırı tespit sistemleri, verileri veya bilgiyi siber saldırılardan korumada önemli bir rol oynamaktadır. Makine öğrenimi ve derin öğrenmedeki hızlı ilerlemeler, veri madenciliğinde popüler olan bu iki yöntemin güvenlik dâhil birçok alanda kullanılmasına neden olmaktadır. Bu çalışmada, akıllı bir saldırı tespit sistemi tasarımı için makine öğrenimi ve derin öğrenme yöntemleri üzerinde çalışılmıştır. Akıllı saldırı tespit sistemi tasarımı için literatürde iyi bilinen NSL-KDD ve Kyoto 2006+ olmak üzere iki veri seti kullanılmıştır. Makine öğrenimi yöntemleri, WEKA veri madenciliği aracındaki sınıflandırma algoritmaları kullanılarak gerçekleştirilmiştir. Sınıflandırma algoritmalarından elde edilen sonuçlar, çalışmanın kapsamında tasarlanan derin öğrenme modeli ile karşılaştırılmıştır. Böylece, makine öğrenimi ve derin öğrenme yöntemleri, akıllı bir saldırı tespit sistemi için NSL-KDD ve Kyoto 2006+ veri setleri üzerinde detaylı bir şekilde analiz edilmiş ve ileri çalışmalar için önerilerde bulunulmuştur.

Anahtar Kelimeler: *Derin öğrenme, Saldırı tespit sistemi, Kyoto 2006+, Makine öğrenmesi, NSL-KDD*

1. Introduction

Nowadays, ensuring a high level of security in information and communications technology (ICT) systems and networks is crucial. This ensures that users and organizations can communicate in a safe and trustworthy environment. ICT systems and networks are continually susceptible to various security vulnerabilities, connection attempts, penetration attacks, and other similar intrusions by unauthorized and malicious users (Gurung et al. 2019). Any breach or intrusion in the security of ICT systems and networks poses a critical problem, as these systems process a variety of sensitive user data. Internal and external hacker attacks, which often advance in obscurity to evade detection, can be executed manually or through computer-based methods against ICT systems and networks (Vinayakumar et al. 2019). Day by day, cyberattacks are continually evolving alongside advancements in hardware, software, and network topologies featuring extremely complex algorithms.

Intrusion Detection Systems (IDS) are essential for identifying and stopping constantly changing and advancing hostile attacks. An IDS is a security tool that analyzes network traffic to analyze the system for detecting suspicious activity and notifying the system or network administrator (Vasilomanolakis et al. 2015). IDS

*Sorumlu Yazar/Corresponding Author: Mehmet HACIBEYOĞLU e-posta/e-mail: hacibeyoglu@erbakan.edu.tr

is divided into two classes: Host-Based Intrusion Detection System (HIDS) and Network-Based Intrusion Detection System (NIDS). HIDS monitors the host computer for suspicious behaviors, including altering or removing system files, executing unauthorized system calls, or making undesirable configuration adjustments, and notifies the user accordingly (Vinayakumar et al. 2019). HIDS uses log file data, including sensor logs, system logs, software logs, file systems, disk assets, user account information, and other pertinent data for each system. NIDS is typically positioned at network points like gateways, switches, and routers to identify attacks and potential threats within network traffic (Puzis et al. 2008).

Network traffic analysis and attack detection are primarily carried out through two methods: misuse detection and anomaly detection. Misuse detection relies on predefined signatures and attack models to identify attacks. Information indicating that each pattern represents a specific type of attack is stored in the signature database, which is regularly updated with newly discovered attack types. During attack detection, the system searches for patterns similar to known attack patterns in the signature database (Kim et al. 2014). As a result, while this method excels at accurately detecting known attacks, identifying new attacks proves challenging. Anomaly detection, on the other hand, relies on heuristic mechanisms to identify unknown malicious activities. These mechanisms learn the normal behavior and properties of the network to generate a learning model. Any traffic flow that significantly deviates from the learning model is considered an intrusion or attack (Qassim et al. 2016). While such an intrusion detection system can identify new and unknown attacks, it often comes with a high false positive rate due to the challenge of distinguishing between normal and abnormal network behavior. The classification of IDS by deployment and detection techniques is presented in Figure 1 (Ahmad et al. 2021).

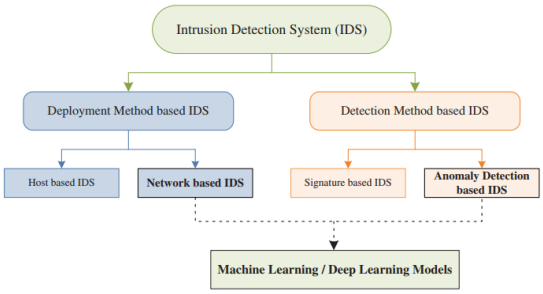

Figure 1. The classification of IDS

This article's primary contribution is to assess the impact of data preprocessing techniques on the effectiveness of IDS and to compare the success of the proposed deep learning (DL) model with existing machine learning (ML) algorithms.

2. Related Works

This section provides a summary of studies on the ML and DL approaches used in the development of NIDS and HIDS. As sub-branches of artificial intelligence (AI), ML and DL are effective methods utilized in the development of IDS by learning from big data (Prasad and Rohokale 2020). In recent years, these techniques have gained significant popularity in the field of network security, particularly with the advent of powerful graphics processing units. DL-based techniques, characterized by deep structures, have proven to be more effective in learning complex information from raw data compared to ML-based techniques (Dong and Wang 2016). Diro and Chilamkurti (2018) used DL as a novel intrusion detection technique with promising outcomes for attacks on IoT devices. According to the authors, the addition of various protocols resulted in the emergence of thousands of zero-day attacks, many of which were minor variants of previously known cyberattacks. This situation highlighted the difficulty even traditional ML systems face in detecting these minor attacks over time (Diro and Chilamkurti 2018). Khraisat et al. (2018) examined various data mining techniques that may reduce the number of false negatives and false positives in anomaly intrusion detection systems. The study used the NSL_KDD dataset and found that the intrusion detection system created with the C5 decision tree classifier worked very well and had few false alarms (Khraisat et al. 2018). Shone et al. (2018) presented a novel DL technique for intrusion detection. The authors proposed a nonsymmetric deep auto-encoder for unsupervised feature learning and stacked it with the random forest algorithm classifier. The proposed model was implemented on a TensorFlow-enabled graphics processing unit and evaluated using the KDD Cup '99 and NSL-KDD cyberattack datasets (Shone, Ngoc et al. 2018). Duan et al. (2019) proposed an IDS based on the improved artificial bee colony with elite-guided search equations. The authors employed an enhanced artificial bee colony algorithm to optimize the initial weights of the neural network, preventing the model from converging to a local optimum and enhancing training speed. The developed model demonstrated strong classification capabilities, achieving a high detection rate for attacks (Duan et al. 2019). In a study by Sahu and Mehtre (2015), a multi-class classification model was presented using the J48 Decision Tree algorithm on the Kyoto 2006+ dataset. They demonstrated that their proposed model achieved an accuracy of 97.2% (Sahu and

Mehtre 2015). Swathi and Rao (2019) compared various Partial Distance Search-based (PDS) k nearest neighbor classifiers on the Kyoto 2006+ dataset for attack detection. The experimental studies indicated no significant difference between the classifiers (Swathi and Rao 2019). Park et al. (2018) analyzed the performance of Random Forest on various datasets derived from the Kyoto 2006+ dataset for attack detection. Chitrakar and Huang (2014) proposed a Candidate Support Vector based on Incremental SVM (CSV-ISVM) algorithm on the Kyoto 2006+ dataset for attack detection and analyzed the results. Kasongo (2023) proposed XGBoost long short-term memory (XGBoost-LSTM) and XGBoost Simple Recurrent Neural Networks (XGBoost-Simple-RNN) algorithms based on neural network algorithm for NSL-KDD and the UNSW-NB15 benchmark datasets. The obtained results have been compared with different types of Recurrent Neural Networks, as a result, the XGBoost-LSTM algorithm obtained the best result in the NSL-KDD on the other hand XGBoost-Simple-RNN algorithm achieved competitive results in UNSW-NB15 benchmark dataset (Kasongo 2023). Du et al. (2023) proposed a network intrusion detection classification model based on a convolutional neural network and long short-term memory algorithms (NIDS-CNNLSTM). The proposed NIDS-CNNLSTM applied n KDD CUP99, NSL_KDD, and UNSW_NB15 benchmark datasets, the outcomes of the proposed NIDS-CNNLSTM show a high detection rate and classification accuracy and a low false rate (Du et al. 2023). Zakariah et al. (2023) proposed a novel intrusion detection system based on an artificial neural network (IDS-ANN), the proposed IDS has been tested on the NSL_KDD dataset. The results of the proposed IDS have been compared with ML classifiers like k-nearest neighbors, DL, Support Vector Machine, Long Short-Term Memory, and Deep Neural Network. The performance of the proposed IDS consistently outperformed each of these ML classifiers in all evaluations (Zakariah et al. 2023). Bakro et al. (2024) present an improved cloud IDS based on the synthetic minority over-sampling technique (SMOTE), information gain (IG), chi-square (CS), particle swarm optimization (PSO), and random forest (RF). In the proposed IDS, SMOTE has been utilized to address the imbalanced data issue, IG, CS, and PSO have been used for feature selection, and finally the RF is utilized for detecting and classifying types of attacks. The proposed IDS has been verified by the UNSW-NB15 and Kyoto benchmark datasets, as a result, the proposed IDS significantly outperforms other IDSs proposed in the related work according to evaluation metrics (Bakro et al. 2024). The list of related works are shown in Table 1.

Table 1. List of related works

Reference	Algorithm	Dataset	Evaluation Metrics			Classification Type	
			TPR	FPR	AUC	Binary	Multi Class
Chitrakar and Huang (2014)	Candidate Support Vector based Incremental SVM	Kyoto 2006+	N/A	N/A	N/A	+	N/A
Sahu and Mehtre (2015)	J48 Decision Tree	Kyoto 2006+	97.20%	47.00%	97.2%	N/A	Normal Attack Unknown Attack
Park et al. (2018)	Random Forest	Kyoto 2006+	N/A	N/A	99%	N/A	Normal Attack Unknown Attack
Shone et al. (2018)	Non-symmetric Deep Auto-Encoder + Random Forest	NSL-KDD	97.73% 94.58% 3.82% 2.70% 94.67%	20.62% 1.07% 3.45% 50.00% 16.84%	N/A N/A N/A N/A N/A	N/A	Normal Dos R2L U2R Probe
Duan et al. (2019)	Artificial bee colony + BP neural networks	NSL-KDD	97.23%	N/A	98.12%	+	N/A
Swathi and Rao (2019)	Partial Distance Search-based (PDS) K-NN	Kyoto 2006+	N/A	N/A	99.28%	N/A	Normal Attack Unknown Attack
Rama Devi and Abualkibash (2019)	Logistic Regression Random Forest Stochastic Gradient Descent Naive Bayes Adaboost Multi-Layer Perceptron	NSL-KDD	N/A N/A N/A N/A N/A N/A	4.7% 3.2% 4.8% 2.1% 4.5% 3.9%	97.4% 99.7% 97.4% 89.5% 89.3% 88.9%	+	N/A

Table 1. (continued) List of related works

Reference	Algorithm	Dataset	Evaluation Metrics			Classification Type	
			TPR	FPR	AUC	Binary	Multi Class
Su et al. (2020)	Deep Learning	NSL-KDD	98.45%	16.52%	90.13%	+	N/A
		NSL-KDD	97.5%	25.7%	N/A	N/A	Normal
			87.55%	1.52%	N/A		Dos
			44.25%	0.91%	N/A		R2L
			20.95%	0.09%	N/A		U2R
			85.76%	1.15%	N/A		Probe
Choudhary and Kesswani (2020)	Deep Learning	NSL-KDD	89.14%	0.91%	96.33%	+	N/A
Kasongo (2023)	XGBoost-LSTM	NSL-KDD	N/A	N/A	99.49%	N/A	Normal Dos R2L U2R Probe
Du et al. (2023)	NIDS-CNNLSTM	NSL-KDD	N/A	0.29%	99.9%	+	Normal Abnormal
Zakariah et al. (2023)	IDS- ANN ANN	NSL-KDD	N/A	N/A	97.5%	+	Normal Abnormal
Bakro et al. (2024)	Modified RF	Kyoto 2006+	99.25%	0.008	99.25%	N/A	Normal Abnormal

TPR: True Positive Rate, FPR: False Positive Rate, AUC: Accuracy

3. Materials and Methods

3.1. Datasets

3.1.1. NSL-KDD Dataset

The KDD Cup 99 dataset, prepared by Stolfo et al. (2000), is the best known and most widely used dataset in the evaluation of anomaly detection methods in computer networks. The KDD Cup 99 dataset was created in the DARPA'98 IDS assessment program. The dataset contains 41 features in which each record is labeled as normal or attack and redundant records that complicate the classification task for ML algorithms (Revathi and Malathi 2013, Tavallaee et al. 2009). These undesirable features of the KDD Cup 99 dataset have been found by researchers to affect the detection accuracy of many IDSs, and the NSL-KDD dataset was derived to overcome these disadvantages (Tavallaee et al. 2009). The NSL-KDD dataset, which is the revised and cleaned-up version of KDD CUP 99 dataset has certain advantages. These are (Tavallaee et al. 2009, Datti and Verma 2010);

- The training set does not have duplicate entries, which aids in the learning process for classifiers.
- The test set is free of any duplicate records.
- There is an inverse relationship between the proportion of records in the original KDD dataset and the number of records selected from each difficulty level group.
- The quantity of records in both the training and test sets is adequate.

Table 2 lists attack types of NSL-KDD datasets (Dhanabal, and Shantharajah 2015).

Table 2. Attack types of NSL-KDD dataset

Category of attack	Attack Name	Total
DoS	apache2, back, land, neptune, mailbomb, pod, processtable, smurf, teardrop, upstorm, worm	11
Probe	ipsweep, mscan, nmap, portsweep, saint, satan	6
R2L	ftp_write, guess_passwd, httptunnel, imap, multihop, named, phf, sendmail, snmpgetattack, spy, snmpguess, xclock, warezclient, warezmaster, xsnoop	15
U2R	buffer_overflow, loadmodule, perl, ps, rootkit, sqlattack, xterm	7

The detailed analysis of the NSL-KDD dataset shows the number of records in four kinds of attacks and normal traffic for training and testing is shown in Table 3.

Table 3. Number of records in NSL-KDD dataset

NSL-KDD	Number of Records					
	Dos	Probe	R2L	U2R	Normal	Total
KDD Train+	45927	11656	995	52	67343	125973
KDD Test+	7460	2421	2885	67	9711	22544
Total	53387	14077	3880	119	77054	148517

3.1.2. Kyoto 2006+ Dataset

Kyoto 2006+ is a public dataset that consists of three years of actual traffic data from November 2006 to August 2009 . The dataset has 24 features, with 14 of them being extracted from the KDD CUP'99 dataset. Ten additional elements have been included to enhance the analysis of Network-based Intrusion Detection Systems (NIDSs).

Honeypots were used to generate the Kyoto 2006+ dataset, which informs IDS researchers on the latest cyberattack trends and internet situations (Song et al. 2011). In this study, while using the Kyoto 2006+ dataset, Source_IP_Address, Source_Port_Number, Destination_IP_Address, Destination_Port_Number and Start_Time properties have been removed. Kyoto 2006+ dataset doesn't provide the attack type information. The class attribute of the dataset indicates whether the session is attack or not attack. To ensure data diversity, three different days were selected from January and May 2015, and the used dataset was created by combining them. Table 4 demonstrates the properties of the class types of the Kyoto 2006+ dataset.

Table 4. The class types of the Kyoto 2006+ dataset

	Number of Records		
Kyoto 2006+	**Normal Session**	**Known Attack**	**Unknown Attack**
Original Dataset	50,033,015	43,043,225	425,719
The Used Dataset	83,033	1,934,163	72

3.2. Feature Selection

Feature selection is the technique of selecting a subset of features from the original features. The general purpose of feature selection is to get rid of irrelevant, redundant features in the dataset in order to build robust learning models. Feature selection preserves important features, thus making the learning algorithm run faster (Kabir et al. 2010, Ladha and Deepa 2011). Also, feature selection helps visualize and understand data, reduces training times, and requires less storage (Guyon and Elisseeff 2003). The advantages of feature selection are (Ladha and Deepa 2011):

- Reduces the size of the feature set and increases the speed of the ML classification algorithm.
- Improves data quality by eliminating irrelevant and noisy data.
- Simplifies the data collection process and reduces the amount of memory needed for data storage.
- Increases the success of the classification model.

Feature selection methods can be generally evaluated in three categories. These are filter methods, wrapper methods, and hybrid methods. In filter methods, feature selection is done with functions based on statistical criteria such as distance, dependency, knowledge, and consistency measurements before any learning is studied (Budak 2018). In wrapper methods, there must be a predetermined learning model for feature selection. This method has a higher computational cost than the filter method (Guan, Liu et al. 2004). Forward selection, backward selection, and stepwise selection are examples of these methods. Hybrid methods combine filter methods and spiral methods. Decision trees and support vector machines are examples of hybrid methods (Budak 2018). In this study, wrapper methods were used as feature selection methods.

3.2.1. Wrapper Methods

Wrapper methods use a specific learning algorithm to generate a subset of features that get better solutions. Wrapper methods generally outperform filter methods in terms of prediction accuracy. However, wrapper methods are difficult to implement in high-dimensional datasets because the learning algorithms are computationally expensive (Zhu et al. 2007, El Aboudi and Benhlima 2016).

3.2.1.1. Sequential Feature Selection

The Sequential Feature Selection (SFS) is a feature selection method that initially starts with an empty set and then adds the feature that provides the best classification accuracy to the empty set. The second step involves adding the remaining features to the existing subset and evaluating the new feature subset. These steps persist until we incorporate the remaining features into the feature set and the accuracy of the classification remains unchanged. This creates a subset of the features (Whitney 1971, El Aboudi and Benhlima 2016, Yan et al. 2018). The weakness of this method is that a selected feature cannot be removed from the cluster in later steps (El Aboudi and Benhlima 2016).

3.2.1.2. Sequential Backward Selection (SBS)

The Sequential Backward Selection (SBS) method, presented by Marill and Green (1963) (Pudil et al. 1994), works in the opposite way to the SFS method. This method starts with a set containing all the properties. The method then eliminates the feature that enhances classification accuracy by subtracting it from the feature set. These steps persist until the removal of any feature in the set fails to increase the classification accuracy. This creates a subset of the features. Similar to SFS, SBS is not guaranteed to find the optimal subset of features, but it provides rapid convergence to a solution (Marill and Green 1963, El Aboudi and Benhlima 2016).

3.3. Classification Algorithms

3.3.1. Artificial Neural Network

Artificial neural networks (ANN) are a type of supervised ML model that mimics the neural processes of the human brain. The system consists of neurons, which are processing units, and their interconnections (Krose and Smagt 1996). The threshold logic unit presented by

Warren McCulloch and Walter Pitts in 1943 and the Perceptron designed by Frank Rosenblatt in 1957 can be considered the basis of ANNs (Rojas 2013). The net input of the cell is calculated by linearly combining the inputs as a result of multiplying the n inputs applied as inputs to the artificial neuron with the relevant weight value. Then, the calculated input value is subjected to an activation function, and the output of the cell is calculated.

Layers of artificial neurons combine to form artificial neural networks. They usually consist of an input layer, one or more hidden layers, and an output layer which are called multilayer neural networks. Multi-layer neural networks are the most commonly used neural network architectures due to their simplicity. A fully connected neural network occurs when each neuron transmits all the values it generates to the subsequent neuron.

3.3.2. K-Nearest Neighbor

The K-Nearest Neighbor (K-NN) algorithm is one of the simplest supervised ML algorithms that predicts the class of the new sample based on feature similarity using all samples in the training set (Ahmad et al. 2021). This algorithm calculates the distance between the newly arrived sample and each sample in the training set and then estimates the newly arrived sample's class based on the number of classes with the smallest distance. Euclidean, Manhattan, and Minkowski distance equations in K-NN are equations used to calculate distances between two samples (Khan et al. 2002). In the K-NN algorithm, the k parameter is one of the factors affecting the model's performance. If the value of k is too small, the model may overlearn. A large k value could lead to the misclassification of the new sample (Zhang et al. 2019).

3.3.3. Decision Tree

Decision Tree (DTree) is one of the basic supervised ML algorithms used for both classification and regression, taking the rules from the class-labeled training sets (Gorunescu 2011, Chary and Rama 2017). In DTree, a pattern consists of nodes, branches, and leaves. Every node symbolizes a feature. The branch symbolizes a decision or regulation. Every leaf symbolizes a class label (Chary and Rama 2017, Sahani et al. 2018). There are different decision trees for the classification process, such as CART, C4.5, and ID3. In this study, C4.5 decision tree classifier is used. C4.5 is an entropy-based classifier that measures the uncertainty of the dataset. C4.5 uses entropy to calculate the information gain. The information gain determines the degree and importance of the attributes for generating a rule by constructing the tree structure (Gorunescu 2011). After calculating the information gain of all attributes in the dataset, the attribute with the largest information gain value is placed at the root of the tree. The remaining attributes are placed on the branches from root to tip.

3.3.4. Naïve Bayes

The Naive Bayes (NB) classification method is a probabilistic classifier based on Bayes' theorem. Bayesian classifiers determine the most likely class for a particular occurrence based on its feature vector. The NB classifier, $P(C_i \mid X)$ determines the probability that the X instance belongs to the class C_i. The sample to be classified is assigned to the class with the highest probability. The NB classifier makes learning easier by presuming that features are unrelated to the classes that are provided. Even though this assumption of independence is frequently faulty, NB frequently outperforms more sophisticated classifiers in real-world scenarios. The NB method can run faster than other classifiers and may provide higher classification accuracy when applied to large datasets (Rish 2001, Gorunescu 2011).

3.3.5. Decision Table

The Decision Table (DTable) is a learning algorithm based on schema-specific feature selection. This selection process involves identifying the optimal subset of features by evaluating the performance of learning schemas using different feature subsets. Decision tables, a type of classifier with schema-specific attribute selection, are increasingly employed across various fields (Witten and Frank 2002, Witlox et al. 2009). A decision table comprises two main components: the schema and the body. The schema represents a pre-selected set of attributes that define the data, while the body is a table of labeled data items. In this table, the attributes specified by the schema constitute the rows, and the decisions form the columns. When presented with an unlabeled sample, a decision table classifier seeks matches in the decision table using only the features in the schema. If the instance is not located, the decision table's majority class is returned. The most common class among all matching instances will be returned if there is no other outcome (Hodge et al. 2006).

3.3.6. Deep Learning

DL is one of the most popular ML techniques today. DL is a ML algorithm inspired by the human brain that mimics how neurons receive and process information through interaction. DL is defined as the use of interconnected deep networks with multiple layers to produce an output. The layers use the results from the previous stage as input

and transfer them to the next layer so that they can produce an output. Similar to the structure and depth of the human brain, DL methods learn from the low-level characteristic to the high-level concept. DL is a subset of ML activities that includes many hidden layers with deep web features. DL models are more efficient than ML approaches due to their intricate architecture and capacity to autonomously extract significant information from the dataset to provide an output (Dong and Wang 2016, Wei et al. 2018, Ahmad et al. 2021). In DL, the number of hidden layers is greater than in typical neural networks, allowing for the creation of more complicated and nonlinear interactions. Due to its good performance, DL is used in many fields in the literature, such as biomedicine, computer vision, manufacturing, agriculture, image processing, and medicine (Mohsen et al. 2018). There are many reasons behind the frequent use of DL methods today. The main ones are:

• The increase in the amount of data: The widespread use of the internet has led to the production and storage of large amounts of data in digital media. DL methods enabled the realization of this big data use.

• GPUs and increased processing power: Powerful and efficient parallel calculations can be made using the GPU (Graphics Processing Unit). GPUs are used to train DL algorithms faster on large datasets.

• Increasing depth: With the increase in processing power, DL methods can be used in practice.

Solving problems with DL is equivalent to designing the multi-layer network structure in the best and most appropriate way. While designing ML models that learn with input data, there are some parameters that designers need to decide for the algorithms and techniques to be used in the model. Likewise, in DL models, the designer decides on the dropout value, the number of layers, and the number of neurons. Typically, these parameters are not exact initially but change based on the specific situation and dataset. Hyper-parameters are parameters that vary based on the specific problem and dataset. The hyper-parameters that determine the performance of the DL that need to be adjusted in DL training are as follows (Sarker 2021):

• Dataset Size: The size and variety of the dataset are one of the most important factors in DL algorithms. The larger and more diverse the dataset, the higher the learning rate and time spent learning.

• Mini Batch Size: With the large data size in DL applications, processing all data at the same time consumes time and memory. Because in each iteration of the learning, gradient descent is calculated on the network with the backpropagation process, and the weight values are updated. The larger the number of data, the longer this process will take. For this reason, the data is processed in parts. These pieces are called mini-batches.

• Learning Rate and Momentum Coefficient: The updating of parameters in DL algorithms is done during the backpropagation process. During backpropagation, the difference is calculated by computing backward derivatives using the chain rule. This difference is then scaled by the learning rate parameter and used to update the weight values. The learning rate in this process can be a constant, incremental, momentum-dependent, or adaptively learned value.

• Optimization Applications: To determine the best value in nonlinear issues, optimization techniques are applied. The optimization techniques adam, adamax, adagrad, adadelta, and stochastic gradient descent are commonly employed in DL applications. These algorithms differ in terms of speed and performance. The choice of these algorithms is also hyper-parameter.

• Number of Training Rounds (Epoch): During the training process, the model updates its weights using backpropagation after each batch of data is processed. Subsequently, the identical procedure is implemented for additional training datasets. The best suitable weight values are attempted to be calculated in each training step. An epoch refers to the number of training steps. Weight values are calculated incrementally in DL, resulting in a low success rate in the initial epochs which improves as weights are updated. Learning ceases after reaching a specific stage.

• Activation Function: Activation functions convert the output values to non-linear values after weight calculation in multilayer artificial neural networks. Nonlinearity, which is a feature of DL methods, is due to the nonlinearity of the activation functions and is used in solving nonlinear problems. Sigmoid, Tangent Hyperbolic, Relu and Softmax are frequently used activation functions.

• Dropout: In fully connected layer networks, dilution of nodes below a certain threshold increases the success.

• Number of Layers and Hidden Layers: The most important feature that distinguishes DL applications from other artificial neural networks is the number of layers. Layers and hidden layers create a depth, and as the depth increases, the learning rate and rate increase. The number of layers varies according to the design of the

model. The success rate in DL depends on both the number of layers and hyper-parameters.

There are many DL architectures developed in the literature until today. Examples are Convolutional Neural Networks (CNN), Deep Neural Networks (DNN), Recurrent Neural Networks (RNN), AutoEncoder (AE), and Deep Belief Network (DBN). In this study, DNN architecture was designed and used. A basic DL architecture known as DNN allows the model to learn across several levels. For ML tasks like regression and classification, DNNs work incredibly well. There are numerous hidden layers in addition to the input and output layers. Multiple hidden layers in DNNs are used to transport the input data from the input layer to the output layer. DNN is used to model complex nonlinear functions. DNNs, unlike traditional networks, contain multiple hidden layers that use specially designed mathematical operations and activation functions. Thus, the increasing number of hidden layers increases the abstraction level of the model to increase its capacity (Anuse and Vyas 2016, Dong 2018). There is a common problem in the training phase of traditional networks: overfitting. In the overfitting problem, the network learns too much of some examples in the training set. Thus, the network may not learn from other samples in the training set or samples that are not seen in the test set. Dropout has been proposed to overcome the overfitting problem. The dropout operation randomly selects some of the nodes from the network and does not use them in the training process (Srivastava, Hinton et al. 2014).

3.4. Dataset Splitting

In this study, K-fold Cross-validation method is used to split the dataset. The K-fold Cross validation technique is one of the most used approaches for model selection, model parameters, and error prediction for classifiers. Cross-validation is used to evaluate the generalization ability of models and to prevent overfitting. K-fold cross-validation involves randomly dividing the dataset into k subsets, where one subset is used as the test set and the remaining k-1 subsets are used as the training set in each iteration. These operations are repeated for the number of sub-sets by changing the test set each time. Here "fold" refers to the number of sub-sets. The performance of the model is the average of the performances of k clusters. In this method, fold selection is important. Figure 2 shows an example of 10-fold cross-validation (Zhang and Liu 2023).

3.5. Data Normalization

Data normalization is data preprocessing techniques and means converting all the variables in the data to the same range. Data normalization can improve the accuracy and efficiency of ML algorithms. Decimal Scaling, Min-Max Normalization, and Z-Score Standardization are the frequently used normalization techniques in the literature (Al Shalabi and Shaaban 2006).

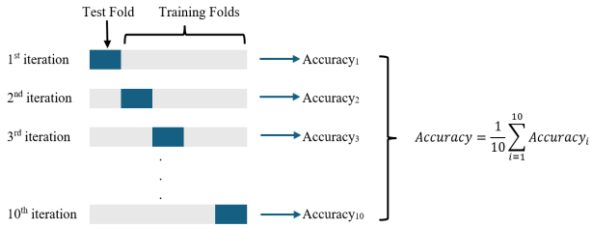

Figure 2. 10-fold cross-validation

In the study, the Min-Max normalization method is used. The Min-Max normalization is a method that provides linear conversion from a predefined range to a newly defined range (Patro and Sahu 2015). The formula of the Min-Max normalization for an attribute A is shown in Equation 1.

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A \quad (1)$$

Here, v is the value to normalize, v' is the new normalized value, min_A is the minimum value of attribute A, max_A is the maximum value of attribute A, new_min_A and new_max_A are the new minimum and maximum limits of attribute A to scale. The normalized features in the study are scaled to the range of 0 and 1. In the study, the features applied to the Min-Max normalization process in the NSL-KDD dataset and the Kyoto 2006+ dataset are shown in Tables 5 and 6.

Table 5. Normalized features in the NSL-KDD dataset

Feature Names
duration, src_bytes, dst_bytes, land, wrong_fragment, urgent, num_failed_logins, logged_in, dst_host_count, su_attempted, num_access_files, count, num_file_creations, num_shells, num_compromised, num_root, hot, num_outbound_cmds, dst_host_srv_count, is_host_login, is_guest_login, srv_count, dst_host_count, root_shell

Table 6. Normalized features in the Kyoto 2006+ dataset

Feature Names
source_bytes, destination_bytes, IDS_detection, count, dst_host_count, dst_host_srv_count, malware_detection, ashula_detection

3.6. Encoding

For the machines to make modelling on the datasets, the dataset must be of numerical type in a way that the machine can understand. The conversion of non-digital data into digital is called Encoding (Oğuzlar 2003). In this study, coding process was applied by starting from scratch while encoding process was being done. For example, the protocol_type property in the NSL-KDD dataset takes the values of "icmp, tcp, "udp" before encoding, and "0, 1, 2" after encoding, respectively. In this study, encoding process was applied to the properties named protocol_type, service, flag, and class in the NSL-KDD dataset. In the Kyoto 2006+ dataset, the coding process has been applied to the features named protocol_type, service, and flag.

3.7. Evaluation Criteria

The following performance evaluation criteria were used to evaluate the performance of the DL model and ML methods designed in this study; accuracy, precision, recall, F-measure, and error rate. These metrics are extracted from the two-dimensional confusion matrix, which provides information about the Actual and Predicted class (Ahmad et al. 2021) shown in Table 7.

Table 7. Confusion matrix

		Predicted Class	
		Attack	**Normal**
Actual Class	**Attack**	True Positive	False Negative
	Normal	False Positive	True Negative

In Table 7, True Positive (TP) values are attack samples that are correctly classified by the classifier. True Negative (TN) values are normal samples that are correctly classified by the classifier. False Positive (FP) values are normal samples that are misclassified by the classifier. False Negative (FN) values are attack samples that are misclassified by the classifier. Using all these values, the performance evaluation metrics used in the study are calculated as (2), (3), (4) and (5):

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (2)$$

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

$$Recall = \frac{TP}{TP+FN} \quad (4)$$

$$F-Measure = 2\left(\frac{Precision\ x\ Recall}{Precision\ x\ Recall}\right) \quad (5)$$

Here, accuracy is the ratio of correctly predicted samples to all samples. This value represents model prediction accuracy. Precision shows how well the model predicts attack patterns. Recall is the proportion of accurately classified attack instances to all real attack cases. The F-Measure is calculated as the harmonic mean of Precision and Recall. This value measures the accuracy of a system, taking into account both the precision and recall of the system (Ahmad 2021).

3.8. WEKA

WEKA was developed in 1997 at the University of Waikato for data mining and ML tasks. WEKA got its name from the initials of the words Waikato Environment for Knowledge Analysis. WEKA is a set of ML and data mining algorithms. WEKA has a GUI interface and is programmed in JAVA. The file format or extension used to store data in WEKA is ARFF. WEKA has tools for visualization. Besides, it has the ability to expand and include new algorithms (Meena and Choudhary 2017). In this study, ML algorithms were tested on WEKA.

4. Experimental Results

In this study, the datasets were tested with the frequently used machine learning algorithms for classification processes in the literature on the WEKA program after applying the data preprocessing steps. The designed DL approach is given in Figure 3. In Table 8, the parameters of DL and the algorithms used are given.

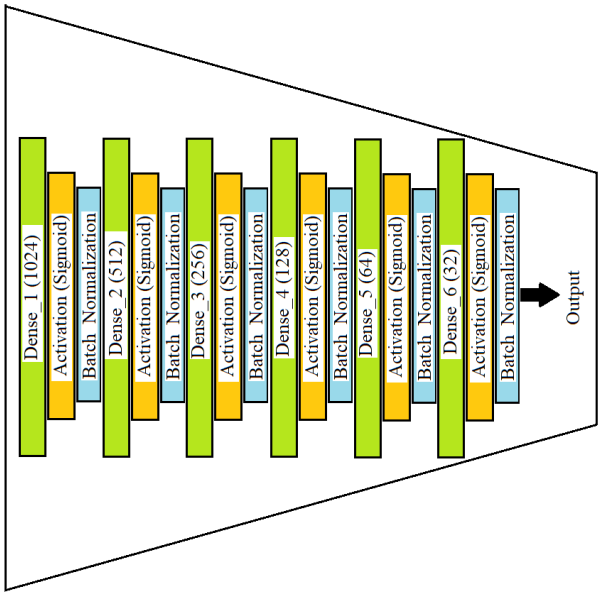

Figure 3. Deep learning model structure

Table 8. Parameters of the algorithms used

Algorithm	Parameters
ANN	batch size=128, epoch=1000
K-NN	K=5, batch size=128
DTree	batch size=128
DTable	batch size=128
DL	batch size = 128, iteration = 1000

.

Table 9. Results for the original NSL-KDD and preprocessed NSL-KDD datasets

Method	Original NSL-KDD Dataset				Preprocessed NSL-KDD Dataset			
	Accuracy	Precision	Recall	F-measure	Accuracy	Precision	Recall	F-measure
ANN	0,976	0,977	0,977	0,977	0,988	0,988	0,988	0,988
K-NN	0,991	0,991	0,991	0,991	0,990	0,990	0,990	0,990
DTree	**0,995**	**0,996**	**0,996**	**0,996**	0,992	**0,993**	**0,993**	**0,993**
NB	0,872	0,875	0,873	0,872	0,864	0,865	0,865	0,865
DTable	0,987	0,987	0,987	0,987	0,975	0,975	0,975	0,975
DL	0,994	0,994	0,994	0,994	**0,993**	**0,993**	**0,993**	**0,993**

4.1. Results in the NSL-KDD Dataset

The NSL-KDD dataset was analyzed as four separate sets as original, preprocessed, SFS feature selection applied (NSL-KDD-SFS) and SBS) feature selection applied (NSL-KDD-SBS). The results of four separate sets are given in Table 9. When the original NLS-KDD results are analyzed, The DTree outperforms others, achieving a remarkable accuracy of 99.5% and precision, recall, and F-measure all at 99.6%. On the other hand, the DL model displays high performance across the board, with an accuracy of 99.4% and precision, recall, and F-measure all at 99.4%.According to the results of preprocessed NSL-KDD; ANN, K-NN, and DL models achieve exceptionally high accuracy of 98.8%, 99.0%, and 99.3%, respectively. Precision, recall, and F-measure values are consistently strong for all models, with DTree leading with 99.3% across these metrics. NB, while having a lower accuracy of 86.4%, maintains a balanced precision and recall around 86.5%. The DTable model also performs well, with an accuracy of 97.5%. Overall, these results underscore the effectiveness of the models in capturing patterns within the data, and the choice among them should consider specific task requirements and trade-offs between precision and recall.

Table 10. Results for the NSL-KDD-SFS and NSL-KDD-SBS datasets

Method	NSL-KDD-SFS Dataset				NSL-KDD-SBS Dataset			
	Accuracy	Precision	Recall	F-measure	Accuracy	Precision	Recall	F-measure
ANN	0.987	0.988	0.988	0.988	0.988	0.988	0.988	0.988
K-NN	0.990	0.990	0.990	0.990	0.989	0.990	0.990	0.990
DTree	**0.992**	**0.993**	**0.993**	**0.993**	**0.992**	**0.993**	**0.993**	**0.993**
NB	0.864	0.865	0.865	0.864	0.857	0.858	0.857	0.857
DTable	0.975	0.975	0.975	0.975	0.975	0.975	0.975	0.975
DL	0.991	0.991	0.991	0.991	0.991	0.991	0.991	0.991

After SFS feature selection is made on the NSL-KDD dataset; ANN, K-NN, DTree, NB, DTable, and DL models all exhibit commendable performance. ANN, K-NN, and DL showcase consistently high accuracy of 98.7%, 99.0%, and 99.1%, respectively, along with matching precision, recall, and F-measure values around 98.8%, 99.0%, and 99.1%. DTree performs exceptionally well with an accuracy of 99.2% and precision, recall, and F-measure all at 99.3%. NB trails behind with an accuracy of 86.4%, while DTable maintains a solid accuracy of 97.5%.

According to the results of NSL-KDD-SBS DTree, K-NN, and DL models consistently achieve accuracy levels above 99%, showcasing their exceptional predictive capabilities. Naive Bayes lags slightly behind with an accuracy of 85.7%, suggesting it may not perform as well on this task. However, all models, including Naive Bayes, maintain balanced precision, recall, and F-measure values.

4.2. Results in the Kyoto 2006+ Dataset

The Kyoto 2006+ dataset was analyzed as three separate sets: original, preprocessed, and SBS feature selection applied. Since all features were selected during feature selection using the SFS method, no experimental study was conducted with the Kyoto 2006+ SFS dataset. The results are given in the table below. When the original Kyoto2006+ results are examined ANN and K-NN achieve high accuracy levels of 97.7% and 98.6%, respectively, with balanced precision, recall, and F-measure values around 97.5-98.6%. Decision Tree excels with an accuracy of 99.5% and near-perfect precision, recall, and F-measure scores at 99.6%. Naive Bayes, while displaying a

lower accuracy of 69.3%, exhibits remarkably high precision at 95.1%, although with a lower recall and F-measure. Decision Table performs well with an accuracy of 98.9% and consistent precision, recall, and F-measure values. Deep Learning stands out with an impressive accuracy of 99.5% and balanced precision, recall, and F-measure scores at 99.5%.

Table 11. Results for the Kyoto 2006+ original and preprocessed dataset

Method	Kyoto 2006+ Dataset				Kyoto 2006+ Preprocessed Dataset			
	Accuracy	Precision	Recall	F-measure	Accuracy	Precision	Recall	F-measure
ANN	0.977	0.975	0.977	0.975	0.975	-	0.975	-
K-NN	0.986	0.985	0.986	0.986	0.986	0.985	0.986	0.986
DTree	**0.995**	**0.996**	**0.996**	**0.996**	0.990	0.990	**0.991**	0.990
NB	0.693	0.951	0.694	0.786	0.695	0.947	0.695	0.787
DTable	0.989	0.989	0.989	0.989	0.979	0.978	0.980	0.978
DL	**0.995**	0.995	0.995	0.995	**0.991**	**0.991**	**0.991**	**0.991**

According to the preprocessed Kyoto2006+ results ANN achieves an accuracy of 97.5% with recall at 97.5%, suggesting that it captures a high proportion of actual positive instances. However, precision and F-measure values are not provided. K-NN demonstrates solid performance with accuracy, precision, recall, and F-measure all at 98.6%. Decision Tree also performs well with an accuracy of 99.0% and balanced precision, recall, and F-measure values at 99.0-99.1%. Naive Bayes shows a lower accuracy of 69.5% but maintains a high precision of 94.7%, suggesting it correctly identifies a substantial portion of positive instances. Decision Table achieves an accuracy of 97.9% with balanced precision, recall, and F-measure values around 97.8-98.0%. Deep Learning stands out with an accuracy of 99.1% and balanced precision, recall, and F-measure scores at 99.1%

Table 12. Results for the Kyoto 2006+ SBS datasets

Method	Kyoto 2006+ SBS			
	Accuracy	Precision	Recall	F-measure
ANN	0.884	0.885	0.884	0.884
K-NN	0.879	0.883	0.880	0.880
DTree	0.912	0.913	0.912	0.912
NB	0.824	0.823	0.824	0.823
DTable	0.833	0.836	0.833	0.830
DL	**0.990**	**0.990**	**0.990**	**0.990**

After the SBS feature selection is made on Kyoto 2006+ dataset; ANN achieves an accuracy of 88.4%, precision of 88.5%, recall of 88.4%, and an F-measure of 88.4%. K-NN closely follows with an accuracy of 87.9%, precision of 88.3%, recall of 88.0%, and an F-measure of 88.0%. Decision Tree performs slightly better with an accuracy of 91.2%, precision of 91.3%, recall of 91.2%, and an F-measure of 91.2%. Naive Bayes lags with an accuracy of 82.4%, precision of 82.3%, recall of 82.4%, and an F-measure of 82.3%. Decision Table exhibits an accuracy of 83.3%, precision of 83.6%, recall of 83.3%, and an F-measure of 83.0%. Finally, Deep Learning stands out with exceptional performance, achieving an accuracy, precision, recall, and F-measure of 99.0%.

4.3. Discussion of Experimental Results

When all experimental studies are taken into account, it can be seen that classification accuracy generally decreases, especially after data preprocessing and feature selection processes. Moreover, it seems that the success of all machine learning algorithms except the K-NN algorithm decreases after the normalization process because, in the K-NN algorithm, each feature is expected to have the same impact on the classification process. In addition, the high number of features provides flexibility to machine learning models. It has been observed that selecting some features for these datasets deprives the model of the unique information contained in that feature, information that could be a critical determinant of the outcome. As a result of the classification model being deprived of some features required for prediction, it was analyzed that the data was insufficient for high success and thus the classification accuracy was determined to decrease. According to the results, it has been observed that the DL model gives better results on the original dataset for both NSL-KDD and Kyoto 2006+ datasets. Thus, it was learned that the DL model is more successful with more data.

Moreover, as an intrusion detection system, the best results of the proposed intrusion detection system on NSL-KDD and Kyoto datasets are compared with the previous studies. When compared with the proposed DL model and the study conducted by Kasongo in 2023, it is seen that they achieve 99.5% and 99.49% success, respectively, for the NSL-KDD dataset. In addition, when the proposed DL model is compared with the study conducted by Bakro in 2024, it is seen that they achieve 99.5% and 99.25% success, respectively, for the Kyoto

2006+ dataset. As a result, the proposed DL model is successful for both datasets.

5. Conclusions

In this study, a machine learning-based approach has been studied in the development of intelligent intrusion detection systems. Within the scope of the study, NSLKDD and Kyoto 2006+ datasets, which are frequently used in the literature, were used. The datasets were given to the ANN, K-NN, DTree, NB, DTable, and DL classification algorithms used in the study, first in their original form and then with discretization, feature selection, data reduction, and normalization preprocessing techniques, and experiments were carried out by cross-correcting. As a result of the experimental studies, it was observed that the DL model developed within the scope of the study and the DTree algorithm were both successful. It was observed that the ANN, KNN, and DTable algorithms used in the study obtained similar results and the NB algorithm showed the worst performance.

Another important analysis obtained from the study is that, for intrusion detection datasets, the DL model achieves more successful results with the original datasets, that is, with more data, but its success decreases especially as a result of the data preprocessing operations.

To improve the performance of the developed model in future studies, it is recommended to use it as a hybrid and test the resulting model on the real system.

Declaration of Ethical Standards
The authors declare that they comply with all ethical standards.

Credit Authorship Contribution Statement
Author 1: Investigation, Methodology, Experimental study, Writing
Author 2: Investigation, Methodology, Experimental study, Writing
Author 3: Investigation, Methodology, Experimental study, Writing

Declaration of Competing Interest
The authors have no conflicts of interest to declare regarding the content of this article.

Data Availability Statement
All data generated or analyzed during this study are included in this published paper.

Acknowledgement
This research was supported by Necmettin Erbakan University Scientific Research Projects Coordination (Project No: 201219009)

6. References

Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J. and Ahmad, F., 2021. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, **32(1),** e4150 https://doi.org/10.1002/ett.4150

Al Shalabi, L., and Shaaban, Z., 2006. Normalization as a preprocessing engine for data mining and the approach of preference matrix. *In 2006 International conference on dependability of computer systems,* 207-214. https://doi.org/10.1109/DEPCOS-RELCOMEX.2006.38

Anuse, A. and Vyas, V., 2016. A novel training algorithm for convolutional neural network. *Complex & Intelligent Systems*, **2(3)**, 221-234. https://doi.org/10.1007/s40747-016-0024-6

Bakro, M., Kumar, R. R., Husain, M., Ashraf, Z., Ali, A., Yaqoob, S. I., ... and Parveen, N., 2024. Building a Cloud-IDS by Hybrid Bio-Inspired Feature Selection Algorithms Along With Random Forest Model. *IEEE Access*, **12**, 8846 - 8874. https://doi.org/10.1109/ACCESS.2024.3353055

Budak, H., 2018. Özellik seçim yöntemleri ve yeni bir yaklaşım. *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, **22**, 21-31.

Chary, S. N. and Rama, B., 2017. A survey on comparative analysis of decision tree algorithms in data mining. *International Journal of Advanced Scientific Technologies, Engineering and Management Sciences*, **3(1)**, 91-95.

Chitrakar, R. and Huang, C., 2014. Selection of candidate support vectors in incremental SVM for network intrusion detection. *Computers & Security*, **45**, 231-241. https://doi.org/10.1016/j.cose.2014.06.006

Datti, R. and Verma, B., 2010. Feature reduction for intrusion detection using linear discriminant analysis. *International Journal on Engineering Science and Technology*, **2(4)**, 1072-1078.

Dhanabal, L. and Shantharajah, S. P. (2015). A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. *International Journal of Advanced Research in Computer and Communication Engineering*, **4(6)**, 446-452.

Diro, A. A. and Chilamkurti, N., 2018. Distributed attack detection scheme using deep learning approach for Internet of Things. *Future Generation Computer Systems*, **82**, 761-768. https://doi.org/10.1016/j.future.2017.08.043

Dong, B. and Wang, X., 2016. Comparison deep learning method to traditional methods using for network intrusion detection. *In 2016 8th IEEE international conference on communication software and networks (ICCSN)*, 581-585.

Dong, Y., 2018. An application of deep neural networks to the in-flight parameter identification for detection

and characterization of aircraft icing. *Aerospace Science and Technology*, **77**, 34-49. https://doi.org/10.1016/j.ast.2018.02.026

Du, J., Yang, K., Hu, Y. and Jiang, L., 2023. NIDS-CNNLSTM: Network intrusion detection classification model based on deep learning. *IEEE Access*, **11**, 24808-24821. https://doi.org/10.1109/ACCESS.2023.3254915

Duan, L., Han, D. and Tian, Q., 2019. Design of intrusion detection system based on improved ABC_elite and BP neural networks. *Computer Science and Information Systems*, **16(3)**, 773-795. https://doi.org/10.2298/CSIS181001026D

El Aboudi, N. and Benhlima, L., 2016. Review on wrapper feature selection approaches. *In 2016 international conference on engineering & MIS (ICEMIS)*, 1-5.

Gorunescu, F., 2011. Data Mining: Concepts, models and techniques, 12, Springer Science & Business Media.

Guan, S. U., Liu, J. and Qi, Y., 2004. An incremental approach to contribution-based feature selection. *Journal of Intelligent Systems,* **13(1)**, 15-42. https://doi.org/10.1515/JISYS.2004.13.1.15

Gurung, S., Ghose, M. K. and Subedi, A., 2019. Deep learning approach on network intrusion detection system using NSL-KDD dataset. *International Journal of Computer Network and Information Security*, **3**, 8-14. htttps://doi.org/10.5815/ijcnis.2019.03.0

Guyon, I. and Elisseeff, A., 2003. An introduction to variable and feature selection. *Journal of machine learning research*, **3**, 1157-1182.

Hodge, V. J., O'Keefe, S. and Austin, J., 2006. A binary neural decision table classifier. *Neurocomputing*, **69(16)**, 1850-1859. https://doi.org/10.1016/j.neucom.2005.11.012

Kabir, M. M., Islam, M. M. and Murase, K., 2010. A new wrapper feature selection approach using neural network. *Neurocomputing*, **73(16-18)**, 3273-3283. https://doi.org/10.1016/j.neucom.2010.04.003

Kasongo, S. M., 2023. A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework. *Computer Communications*, **199**, 113-125. https://doi.org/10.1016/j.comcom.2022.12.010

Khan, M., Ding, Q. and Perrizo, W., 2002. K-nearest neighbor classification on spatial data streams using P-trees. *In Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 517-528. https://doi.org/10.1007/3-540-47887-6_51

Khraisat, A., Gondal, I. and Vamplew, P., 2018. An anomaly intrusion detection system using C5 decision tree classifier. *In Trends and Applications in Knowledge Discovery and Data Mining: PAKDD 2018*. https://doi.org/10.1007/978-3-030-04503-6_14

Kim, G., Lee, S. and Kim, S., 2014. A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Systems with Applications*, **41(4)**, 1690-1700. https://doi.org/10.1016/j.eswa.2013.08.066

Krose, B. and Smagt, P. V. D., 1996. An introduction to neural networks. *Journal of Computer Science*, **(48)**.

Ladha, L. and Deepa, T., 2011. Feature selection methods and algorithms. *International Journal on Computer Science and Engineering*, **3(5)**, 1787-1797.

Marill, T. and Green, D., 1963. On the effectiveness of receptors in recognition systems. *IEEE transactions on Information Theory*, **9(1)**, 11-17. https://doi.org/10.1109/TIT.1963.1057810

Meena, G. and Choudhary, R. R., 2017. A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA. *In 2017 International Conference on Computer, Communications and Electronics*, 553-558. https://doi.org/10.1109/COMPTELIX.2017.8004032

Mohsen, H., El-Dahshan, E. S. A., El-Horbaty, E. S. M. and Salem, A. B. M., 2018. Classification using deep learning neural networks for brain tumors. *Future Computing and Informatics Journal*, **3(1)**, 68-71. https://doi.org/10.1016/j.fcij.2017.12.001

Oğuzlar, A., 2003. Veri ön işleme. *Erciyes Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi*, (**21**).

Park, K., Song, Y. and Cheong, Y. G., 2018. Classification of attack types for intrusion detection systems using a machine learning algorithm. *In 2018 IEEE fourth international conference on big data computing service and applications*, 282-286. https://doi.org/10.1109/BigDataService.2018.00050

Patro, S. and Sahu, K. K., 2015. Normalization: A preprocessing stage. *arXiv preprint*. https://doi.org/10.48550/arXiv.1503.06462

Prasad, R. and Rohokale, V., 2020. Artificial intelligence and machine learning in cyber security. *Cyber security: the lifeline of information and communication technology*, 231-247. https://doi.org/10.1007/978-3-030-31703-4_16

Pudil, P., Novovičová, J. and Kittler, J., 1994. Floating search methods in feature selection. *Pattern recognition letters*, **15(11)**, 1119-1125. https://doi.org/10.1016/0167-8655(94)90127-9

Puzis, R., Klippel, M. D., Elovici, Y. and Dolev, S., 2008. Optimization of NIDS placement for protection of intercommunicating critical infrastructures. *In European Conference on Intelligence and Security Informatics, 191-203*. https://doi.org/10.1007/978-3-540-89900-6_20

Qassim, Q., Zin, A. M. and Ab Aziz, M. J., 2016. Anomalies Classification Approach for Network-based Intrusion Detection System. *International Journal of Network Security*, **18(6)**, 1159-1172.

Revathi, S. and Malathi, A., 2013. A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection. *International Journal of Engineering Research & Technology (IJERT)*, **2(12)**, 1848-1853.

Rish, I., 2001. An empirical study of the naive Bayes classifier. *In IJCAI 2001 workshop on empirical methods in artificial intelligence*, **3(22)**, 41-46.

Rojas, R., 2013. Neural networks: a systematic introduction. Springer Science & Business Media.

Sahani, R., Shatabdinalini, Rout, C., Chandrakanta Badajena, J., Jena, A. K. and Das, H., 2018. Classification of intrusion detection using data mining techniques. *In Progress in Computing, Analytics and Networking: Proceedings of ICCAN 2017*, 753-764. https://doi.org/10.1007/978-981-10-7871-2_72

Sahu, S. and Mehtre, B. M., 2015. Network intrusion detection system using J48 Decision Tree. In 2015 *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2023-2026. https://doi.org/10.1109/ICACCI.2015.7275914

Sarker, I. H. (2021). Deep cybersecurity: a comprehensive overview from neural network and deep learning perspective. *SN Computer Science*, **2(3)**, 154. https://doi.org/10.1007/s42979-021-00535-6

Shone, N., Ngoc, T. N., Phai, V. D. and Shi, Q., 2018. A deep learning approach to network intrusion detection. *IEEE transactions on emerging topics in computational intelligence*, **2(1)**, 41-50. https://doi.org/10.1109/TETCI.2017.2772792

Song, J., Takakura, H., Okabe, Y., Eto, M., Inoue, D. and 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*, 29-36. https://doi.org/10.1145/1978672.1978676

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, **15**, 1929-1958.

Stolfo, S. J., Fan, W., Lee, W., Prodromidis, A. and Chan, P. K., 2000. Cost-based modeling for fraud and intrusion detection: Results from the JAM project. *In Proceedings DARPA Information Survivability Conference and Exposition*, 130-144. https://doi.org/10.1109/DISCEX.2000.821515

Swathi, K. and Rao, B. B., 2019. Impact of PDS based kNN classifiers on Kyoto dataset. *International Journal of Rough Sets and Data Analysis (IJRSDA)*, 6(2), 61-72. http://dx.doi.org/10.4018/IJRSDA.2019040105

Tavallaee, M., Bagheri, E., Lu, W. and Ghorbani, A. A., 2009. A detailed analysis of the KDD CUP 99 data set. *In 2009 IEEE symposium on computational intelligence for security and defense applications*, 1-6. https://doi.org/10.1109/CISDA.2009.5356528

Vasilomanolakis, E., Karuppayah, S., Mühlhäuser, M. and Fischer, M., 2015. Taxonomy and survey of collaborative intrusion detection. *ACM computing surveys*, **47(4)**, 1-33. https://doi.org/10.1145/2716260

Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A. and Venkatraman, S., 2019. Deep learning approach for intelligent intrusion detection system. *IEEE Access*, **7**, 41525-41550. https://doi.org/10.1109/ACCESS.2019.2895334

Wei, L., Ding, Y., Su, R., Tang, J. and Zou, Q., 2018. Prediction of human protein subcellular localization using deep learning. *Journal of Parallel and Distributed Computing*, **117**, 212-217. https://doi.org/10.1016/j.jpdc.2017.08.009

Whitney, A. W., 1971. A direct method of nonparametric measurement selection. *IEEE transactions on computers*, **100(9)**, 1100-1103. https://doi.org/10.1109/T-C.1971.223410

Witlox, F., Antrop, M., Bogaert, P., De Maeyer, P., Derudder, B., Neutens, T., ... and Van de Weghe, N. , 2009. Introducing functional classification theory to land use planning by means of decision tables. *Decision Support Systems*, **46(4)**, 875-881. https://doi.org/10.1016/j.dss.2008.12.001

Yan, K., Ma, L., Dai, Y., Shen, W., Ji, Z. and Xie, D., 2018. Cost-sensitive and sequential feature selection for chiller fault detection and diagnosis. *International Journal of Refrigeration*, **86**, 401-409. https://doi.org/10.1016/j.ijrefrig.2017.11.003

Zakariah, M., AlQahtani, S. A., Alawwad, A. M. and Alotaibi, A. A., 2023. Intrusion Detection System with Customized Machine Learning Techniques for NSL-KDD Dataset. *Computers, Materials & Continua*, **77(3)**. 4025-4054 https://doi.org/10.32604/cmc.2023.043752

Zhang, X. and Liu, C. A. (2023). Model averaging prediction by K-fold cross-validation. *Journal of Econometrics*, **235(1)**, 280-301. https://doi.org/10.1016/j.jeconom.2022.04.007

Zhang, Y., Cao, G., Wang, B. and Li, X., 2019. A novel ensemble method for k-nearest neighbor. Pattern Recognition, 85, 13-25.

https://doi.org/10.1016/j.patcog.2018.08.003

Zhu, Z., Ong, Y. S. and Dash, M., 2007. Wrapper–filter feature selection algorithm using a memetic framework. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, **37(1)**, 70-76. https://doi.org/10.1109/TSMCB.2006.883267