IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 7 March 2024, accepted 22 March 2024, date of publication 2 April 2024, date of current version 19 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3384380

APPLIED RESEARCH

Using Triple Modular Redundancy for Threshold Determination in DDOS Intrusion Detection Systems

ALEKSA N. MAKSIMOVIĆ[1], VOJKAN R. NIKOLIĆ[2], DEJAN V. VIDOJEVIĆ[2], MILAN D. RANDJELOVIĆ[3], SLAVIŠA M. DJUKANOVIĆ[1], AND DRAGAN M. RANDJELOVIC[4]

[1]Ministry of Interior Affairs of the Republic of Serbia, 11000 Belgrade, Serbia
[2]Department for Informatics and Computing, University of Criminal Investigation and Police Studies, 11000 Belgrade, Serbia
[3]Science Technology Park Niš, 18000 Niš, Serbia
[4]Faculty of Diplomacy and Security, Univerzitet Union Nikola Tesla, 11000 Belgrade, Serbia

Corresponding author: Vojkan R. Nikolić (vojkan.nikolic@kpu.edu.rs)

ABSTRACT This paper describes an Intrusion Detection System (IDS) which uses several existing known IDS algorithms and Triple Modular Redundancy (TMR) algorithm to make decision about eventual existing attack by majority voting in one constructed ensemble model which solves practically the problem of binary classification. Proposed novel model belongs to so called stacking ensemble methods of machine learning algorithms which uses exactly four algorithms from the group of best binary classification algorithms: Decision trees, Naive Bayes, Support Vector Machine, k-nearest neighbors, logistic regression and AdaBoost and is applicable for any similar problem. Using proposed method, we get a more precisely determined threshold than it is case using whatever of in method individual applied algorithm as well as those algorithms that are the state of the art in the field of binary classification. Besides that, one of the main disadvantages of classic TMR used for classification network traffic, which is the problem of bad over-voting was successfully avoided by improving classic TMR with a new algorithm proposed by the authors. Today a denial of service attacks (DoS) and distributed denial of service (DDoS) are one of most present type on Internet and that is why the authors in this paper paid special attention to them and because of that the authors of proposed method chose to use known KAGGLE dataset which contains data of these type of attacks for the examination of quality of proposed IDS method implemented in suitable software. The dataset itself consists of a wide range of simulated intrusions in a military network environment in United States. Obtained results showed that IDS software with implemented proposed method worked precise and timely, which means alarms was trigger properly and efficiently with better results of quality of classification in most important measures than the individual included algorithms who are the state of the art in binary classification.

INDEX TERMS IDS, DoS, DDoS, triple modular redundancy, bad over-voting problem, Kaggle, ensemble machine learning methods.

I. INTRODUCTION

The rapid development and application of information and communication technologies (ICT) enabled the widespread use of these technologies, where there was a sudden increase in the number of users of ICT systems. These systems have become more complex, and users' devices have high hardware performance and can use very sophisticated applications with great capabilities, while some ordinary users have reached the level of serious ICT system experts. In such

The associate editor coordinating the review of this manuscript and approving it for publication was Barbara Masini.

© 2024 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

an environment, the possibility of abuse of the ICT system has increased, in order to cause damage to the target user and/or achieve material profit.

In order to protect themselves from malicious users of ICT systems, State Governments are establishing computer emergency response units (CERTs) in order to increase ICT security. Their main task is to monitor the happenings on the Internet and to react in case of attacks and incidents on the network, in order to protect their ICT system. In addition to government organizations, large corporations also form their own CERTs, and small and medium-sized companies often hire professional teams to ensure their ICT security. However, despite the measures taken in this way, great damage is caused to organizations and individuals.

This damage is largely caused by Denial of Service (DoS) attacks when protection systems are not at a satisfactory level. In this case, a malicious user of the ICT system tries to simply disable the normal operation of that host, i.e. server, by means of a large number of requests to a specific host. In a more extreme case, attacks can be distributed using infected ICT devices, i.e. bots, where they all attack a specific host. In this case, it is about Distributed Denial of Service (DDoS), which can be organized in such a way that they can paralyze the ICT systems of the entire country if they are strong enough.

Choosing the right strategy for recognizing real attacks is of great importance for the ICT security of an organization [1], [2]. The most common types of these attacks are: flood attacks, protocol attacks, and application-layer attacks [3], [4]. The focus of our research is precisely on flood attacks, like Hypertext Transfer Protocol (HTTP) flood and User Datagram Protocol (UDP) flood attacks. As a common form of protection against such attacks is Intrusion Detection Systems (IDS), which represents a system that monitors network traffic for suspicious activity and alerts when such activity is discovered.

When implementing IDS, attack detection is viewed from two aspects: Host-Based IDS (HIDS) and Network-Based IDS (NIDS). Certain weaknesses of both these systems are overcome by the realization of a hybrid system, which includes the characteristics and functionalities of HIDS and NIDS. This increases the possibility of detecting potential attacks with a lower degree of errors.

Depending on the methods used for attack detection, there are: Signature Detection (SIDS), Anomaly Detection (AIDS) and Hybrid Detection. AIDS approach which is used in proposed ensemble method uses a method based on anomalies and according to one of the taxonomies [5] there are three basic ones: Statistics, knowledge, and machine learning.

For example, the paper [6] presented the Triple Modular Redundancy (TMR) method, as one of the machine learning (ML) ensemble methods, which optimizes the efficiency in determining the threshold when detecting attacks on the network. In this ensemble, there are the attack suggestion algorithm, the k-nearest neighbors algorithm, the cumulative sum algorithm, and the exponentially weighted moving average algorithm in combination with TMR [7], [8], [9], while the role of voting is played by informational TMR in the optimization process. Asymmetric optimization in this work meant an odd number of three IDS algorithms used as one type of asymmetry that allows the decision to start an attack to be made at any time by the majority of votes of the algorithms participating in the proposed method. In doing so, these algorithms look for changes in data traffic and then individually decide whether that traffic is regular or not, but TMR only makes decisions using majority votes.

The two coauthors of this paper were also coauthors in mentioned paper which in its conclusion states ''the subject of the author's interest in future work will be the inclusion and application of different types of algorithms i.e. approaches in the mentioned N-redundant optimization''. These coauthors also observed that with the classic TMR method proposed in mentioned paper, there can be a problem in which two algorithms with worse accuracy over-vote the third one with better characteristics and thus affect the quality of the malicious connection assessment, and thus the decision to trigger the alarm. This was the main driving motive that in this paper, these coauthors with new research team tray to propose a novel TMR based method which uses more than three different algorithms that are state of the art for binary classification and in such way to improve accuracy of the proposed method and overcome the possibly problem of bad over-voting in it. They chosen to tray make this using exactly four algorithms which are the best and most used in solving a problems of binary classification that they aggregated in one special designed ensemble model of machine learning.

It is necessary to have in mind that for the qualified evaluations of the value of the proposed methodology, in addition to comparison with in the literature known state of the art methodologies, it is also important that this is done on a verified dataset. Because of that this paper uses the data set from known KAGGLE platform, where the class variable has two classification categories: normal and with anomaly, at which the complete data set consists two so called .csv files, one for training and one for testing.

Having in mind all above mentioned facts, the authors practically set out to prove in this paper the hypothesis that it is possible to construct a model in which in the ensemble of decision-making by voting using TMR with four algorithms, is avoided the case of wrong generation of attacks due to over-voting of the correct algorithm by two incorrect algorithms, when TMR uses only three algorithms and which is with better characteristics than each individual included as well as any other that are state of the art, thereby used algorithms must be state of the art in the field of binary classification where belongs considered problem.

The main scientific contribution of this paper is proposed method that is implemented in suitable software tool which works precise and timely, what means alarms was trigger properly and efficiently with better results of quality of classification in most important measures than the individual

included algorithms who are the state of the art in binary classification.

The rest of the paper is organized as follows. Section II contains the literature overview for the field that is the subject of consideration in this paper.

There, the authors deal with papers on anomaly detection caused by DOS/DDOS attacks in general in the first subsection. In the second subsection, the authors present the use of classification algorithms in IDS for the detection of DOS/DDOS attacks and especially six algorithms that were used to propose a new ensemble model in this paper: decision trees, naive Bayes, support vector machine, K-nearest neighbors (k-NN), Ada Boost, logistic regression algorithm but also consider other different proposed ensemble methods. Section III contains two subheadings, with Section III-A outlining the material used and Section III-B briefly describing and appropriately citing six algorithms which will be used to propose novel model: Decision trees, Naive Bayes, Support Vector Machine, K-nearest neighbors (k-NN), Ada Boost, Logistic Regression algorithm, and different ensemble methods similar as one that they are going to use. In Section IV, the proposed model is presented, where the following are given: diagram of the proposed model, as well as all included modules and algorithms. Section V Results, discussion and findings consists three parts. In part 5.1 the implementation of the proposed solution is presented, in 5.2 the findings, validation and discussion are given, and in 5.3 the limitation and future work are presented. In section VI the description of the technical solution of the proposed model is given. Finally, Section VII contains conclusions.

II. RELATED WORK

Nowadays security software is a very important part of every serious security organization, as facing security incidents has never been harder than it is today. Vast numbers of people globally now have the baseline technical knowledge needed to abuse systems and cause serious damage. As noted in the introduction, DDoS attacks are one of the biggest threats to IT security of any organization, so naturally many experts are trying to resolve the issue efficiently in their own domains. Because of that the literature review of the papers that deals with DDOS anomaly detection is given in first subchapter of this chapter. For the sake of ease and better overview of the literature, it is specially singled out in second subchapter of this chapter an overview of the use of each of the classification algorithms of the participants in the proposed ensemble model for DOS / DDOS.

A. ANOMALY DETECTION CAUSED WITH DOS/DDOS ATTACKS

In the articles [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], and [31], methods of Anomaly Detection caused with DOS/DDOS attacks are presented and we can find different approach and solution.

In [10], the authors presented a description of an Intrusion Detection System and proposed a prevention mechanism using a puzzle controller based on a stochastic model, in order to eliminate the possibility of DDoS attacks. User behavior is recorded in the behavior matrix, on the basis of which the covariance matrix is calculated. The entropy calculated from the covariance matrix is compared with the threshold for detecting a DDoS attack, and network resources can be accessed only by those clients who have solved the puzzle with a difficulty level determined based on the entropy value.

In [11], the authors focused on the design and evaluation of statistical automated attack detection, where a DDOS attack is considered as an anomaly in network behavior. The basic idea is that unlike a DDOS attack, a flash crowd is characterized by a large increase not only in the number of packets but also in the number of IP connections. A comparison is made between the active connections and the packets sent by those connected users, where the results obtained should be proportional, and if they are not, it refers to the evidence of a DDoS attack.

The classification of DDoS attacks is presented by type in [12], where the classification is done by the degree of automation, by the exploited vulnerability, by the dynamics of the attack rate and by the impact. Each of these metrics can be helpful when it comes to detecting and preventing these attacks and [13] shows which businesses are the most common targets of DDoS attacks, clearly showing that they are gaming sites, online shopping, banks, and stock exchange services.

In [14], the solution that is presented is an approach in which both users and attackers are taken into account. Here it is shown how with the usage of Fast entropy and Flow based analysis, they decreased the number of false positives by looking at both sides and in [3] the proposed solution is to exploit the architecture of the DDoS attack. Certain features of the DDoS attack are looked into and then variables are selected that are needed for detection. With this approach, different phases of the attack can be partitioned and detected successfully.

In [15], a hybrid IDS is considered, where there are several types of attacks that can be detected with an IDS because their behavior is different from the normal one. The type of attack that is our object is DDoS, which can be detected in a few ways [16], for example services are slowed down or interrupted by the attack, and that is achieved by allocating system resources. Due to the attack's many sources, the so-called bot-network can use different approaches for implementing IDS. The hybrid approach is based on the idea of selecting algorithms that already have good results regarding attack detection, which means the algorithm needs to have a high precision rate and low false positive rate, so afterwards they can be combined to minimize their weak sides and to improve their good sides to get better results in the end.

In [17], the authors present a new approach to AD, to be called a posteriori AD for unlabeled anomaly classification where a posteriori indicates that information obtained directly

from processing data is used as new information for subsequent data processing.

In [18], the consistency subset evaluation has been combined with DDoS characteristic features. They are combined with a feature selection algorithm in order to select the best features of the attack to see whether the attack is malicious or normal. A better accuracy and precision rate has been achieved which confirms that our approach is good with a combination of three algorithms.

In [19], the deep learning solution is used with a convolutional neural network to develop a deep learning model. It investigates some of the advanced DDoS attacks. Higher precision and accuracy have been achieved by using not only a binary classification system but a multiclass system.

In [20], an analysis of existing machine learning algorithms has been presented together with a new GTCS dataset. In addition, the new adaptive classifier model has been presented that is assembled from different learning models in order to improve the accuracy and false positive rates of DDoS attacks.

In [21], authors proposed an ensemble and adaptive classifier model composed of multiple classifiers with different learning paradigms to address the issue of the accuracy and false alarm rate in IDSs.

In [22], it is explained that every detection algorithm must have some threshold value that is calculated. Threshold values will be discussed further in this paper, but first we need to understand why this is important for IDS. Threshold can be calculated using static or dynamic approach. Dynamic approach requires having an algorithm trained with previously recorded data, and also needing a human to analyze that data, which is not always an option, and it gives better results than static value but requires a certain amount of information [23]. The IDS solution that is proposed by using TMR relies on those dynamic threshold values. Threshold is a standard when it comes to the detection of illegitimate traffic, and there are few parameters that can be traced and recorded during an attack. The number of hosts is one parameter, as is the number of different IP addresses, when it comes to the DDoS port scan feature it can be a valuable indicator that a DDoS attack is coming. Looking at those parameters, the threshold value can be calculated more effectively and precisely because the more data and more parameters to consider, the better the results.

In the last few years, the interest in the study and development of new, more efficient methodologies in attack detection systems has intensified, and in addition to many other papers on the subject of this paper, the papers listed here are particularly interesting.

In [24], Internet of Things (IoT) devices are observed because of their lack of security, which are common targets of DDoS attacks. It uses an entropy and hybrid approach to prevent attackers into training models and tricking them that attack traffic is a regular one. Multiple fields have been observed from network traffic to achieve the best possible result but in [25] we find an explanation of the attempt to use the statistical process control applied in the IDS where principal component analysis is used to overcome the problems caused by a big number of quality characteristics.

In [26], the authors proposed an end-to-end abnormal behavior detection method based on sequential information preserving log embedding algorithms and machine learning-based anomaly detection algorithms.

In [27], a notification management system is presented, which is very important when it comes to IDS, and right time alert is a key part in such kinds of systems as well as privacy protection, which is especially highlighted here.

In [28], IoT networks are discussed where anomaly detection is the most important part when it comes to stability and security of these networks. IoT networks are highly exposed to attacks because of their lack of security.

In [29], a train communication network intrusion detection system based on anomaly detection and attack classification is proposed. Firstly, the built an anomaly detection model based on support vector machines (SVM). The particle swarm optimization-support vector machines (PSO-SVM), and genetic algorithm-support vector machines (GA-SVM) optimization algorithms are used to optimize the kernel function parameters of SVM. Secondly, the built two attack classification models based on random forest. They are iterative dichotomiser 3 (ID3) and classification and regression tree (CART).

In [30], a deep convolutional neural network framework was used in software defined networks and has been evaluated on the latest datasets for efficiency, and similar datasets are used for evaluation of our solution.

In [31], DDoS attack types are described and classified as well as defensive counter measures, which was helpful to this research.

B. USING CLASSIFICATION ALGORITHMS IN IDS FOR DOS/DDOS ATTACKS DETECTION

1) DECISION TREES ALGORITHM

In [32], authors combine the spatial feature and temporal feature, and fuse the GBDT model and the GRU model to make a quadratic ensemble model as the final intrusion detection system.

In [33], five attack scenarios were designed by performing various DDoS attacks on SCADA systems. The test results of various DDoS attacks demonstrated that the hybrid model and the decision tree model are the most suitable for such environments.

In [34], the proposed ensemble multi binary attack model (EMBAM) is an intrusion detection system (IDS) that offers a unique anomaly based IDS to detect normal behavior and abnormal attacks, for example, threats in a network.

2) NAIVE BAYES METHOD

In [35], authors proposed a Double-Layered Hybrid Approach (DLHA) designed specifically to detect anomalies and unseen attacks. DLHA deploys Naive Bayes classifier as

Layer 1 to detect DoS and Probe, and adopts SVM as Layer 2 to distinguish R2L and U2R from normal instances.

In [36], The Likelihood Naive Bayes (LNB) classification approach is implemented to accurately predict the classified label as to whether normal or attack.

In [37], the proposed method MANN-AM (Modified Artificial Neural Network with Attention Mechanism) is compared internally by using Random Forest, Naive Bayes and K-Nearest Neighbor to show the effectiveness of the proposed model.

3) SUPPORT VECTOR MACHINE

In [38], the authors tested and described different supervised Machine Learning classifiers (Logistic Regression, Random Forest and Support Vector Machine) combined with two topic-modelling techniques of NLP, (Latent Semantic Analysis and Latent Dirichlet Allocation).

In [39], authors use five machine learning models, including K-means clustering, decision tree, random forest, and support vector machine to classify bearing faults.

In [40], authors use deep learning to extract essential feature representations automatically and realize high detection performance efficiently. An effective stacked contractive auto-encoder (SCAE) method is presented for unsupervised feature extraction. A novel cloud intrusion detection system is designed on the basis of the SCAE and support vector machine (SVM) classification algorithm. The SCAE+SVM approach combines both deep and shallow learning techniques, and it fully exploits their advantages to significantly reduce the analytical overhead.

4) k-NN ALGORITHM

In [41], the k-Nearest Neighbor algorithm is used for DDoS attack classification, this algorithm has been chosen for our research because of its effectiveness and training ability. With k-NN optimization, the detection process is simpler and more efficient.

In [42], authors propose a novel data filtering strategy for k-NN search algorithms on multicore platforms.

In [43], the authors presented a fast k-nearest neighbor algorithm which combines k-NN with a cluster-space data representation.

5) LOGISTIC REGRESSION

In [44] the authors deal with the detection of DDoS attacks from all service requests and classify them according to DDoS classes.Two different machine learning approaches, SVM and Logistic Regression, are implemented in the dataset for detecting and classifying DDoS attacks and a comparative study is accomplished among them in terms of accuracy, precision and recall rates.

In [45] the authors conducted an extensive examination of diverse classifiers aimed at identifying ingress DDoS attack traffic within an SDN environment. To evaluate the efficacy of their machine learning classifiers, including decision tree (DT), multilayer perceptron (MLP), AdaBoost (AB), RF, and logistic regression (LR), the researchers employed TCP-SYN flooding attacks. The assessment of these classifiers involved the application of a cross-validation technique to validate the proposed classification models. The outcomes derived from the experiments demonstrated that the approach introduced by the authors exhibited noteworthy performance, as illustrated by the high-quality results obtained.

6) ADA BOOST

In [46] the focus is on boosting the classification accuracy by improving feature selection and weighing the ensemble model with the crow search algorithm (CSA). The feature selection is handled by combining both filters and automated models to obtain improved feature sets. The ensemble classifier is made up of machine and deep learning models such as long short-term memory (LSTM), support vector machine (SVM), XGBoost, and a fast learning network (FLN).

In [47], the authors of this article used AdaBoost and RUSBoost for classification and prediction purposes and a realistic IoT dataset named ToN-IoT, which contains various network data, to detect DDOS attacks. Through this ensemble classifier learning, the confusion matrix has been generated, and the performance of both models has been compared. AdaBoost achieved better precision, while RUSBoost achieved better accuracy.

In [48], the authors of this article propose a solution an Intrusion Detection System based on Artificial Intelligence (AI, AdaBoost) for IoT system. The method used in this study is supervised learning which measures the accuracy of predictions in detecting DoS on IoT network data.

7) ENSEMBLE ALGORITHMS

In reviewing the literature related to ensemble models, the authors did not find a model similar to the one they propose.

In [49] the ensemble combines well-known grouping methods such as Naïve Bayes, Multilayer Perceptron (MLP), and SVM, Decision trees.

In [50] authors use a voting method and average of probabilities, we present an ensemble classifier that used K-means, One-Class SVM, DBSCAN, and Expectation-Maximization, abbreviated (KODE) as an enhanced classifier that consistently classifies the asymmetric probability distributions between malicious and normal instances.

In [51] authors have an approach for generating optimized ensemble IDS is developed. Six feature selection methods are used and compared, ie: Information Gain (IG), Gain Ratio (GR), Symmetrical Uncertainty (SU), Relief-F (RF), One-R (OR) and Chi-Square (CS). The feature selection techniques produce sets of selected features. Each best selected number of features that are obtained from feature ranking step for respective feature selection technique will be used to classify attacks via four classification methods, ie: Bayesian Network (BN), Naive Bayesian (NB), Decision Tree: J48 and SOM. Then, each feature selection technique with its respective best

features is combined with each classifier method to generate ensemble IDSs.

In the paper [52], we can find that the authors use so called RFE-XGBoost (Recursive Feature Elimination-eXtreme Gradient Boosting) scheme which is based on the feature selection with a majority vote ensemble method.

III. MATERIALS AND METHODS

To insert an introductory paragraph in chapter 3 Materials and methods in the form of e.g. for the purposes of realizing the set goal of this work, it was necessary to first select a qualified dataset as the material on which the proposed method will be treated in one subchapter.

A. MATERIALS

The basic data model used to create the paper was downloaded from the online platform KAGGLE. KAGGLE is an open platform dedicated to data science researchers and machine learning enthusiasts. KAGGLE allows users to collaborate with each other, find and publish datasets, use shared workbooks, and compete with other data scientists to solve data science challenges. Data were taken from such a dataset. The dataset itself consists of a wide range of simulated intrusions in a military network environment. An environment was created to collect raw TCP/IP data to simulate a typical US Air Force LAN. The network is configured like a solid environment and attacks with multiple attacks. Each connection is determined by a sequence of TCP packets that start and end at a certain time, between which data from the source to a certain IP address according to some defined protocol. Also, each connection is marked as normal or as an attack. For each TCP/IP connection, the data are 41 quantitative and qualitative characteristics. The class variable has two categories: normal and anomaly. The complete dataset consists of two .csv files, one for training and one for testing, which can be found at the following link:

- https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection?select=Train_data.csv
- https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection?select=Test_data.csv

Below is a table that describes each of the dataset attributes in more detail.

B. METHODS

Today, security software is essential for every organization, because the fight against security risks and threats has never been more demanding. One of the reasons for this is the fact that today many people have some kind of IT knowledge, which some individuals can unfortunately misuse. DDoS attacks constitute one of the biggest security threats, and organizations are trying to solve this problem. As we know, it is possible for a long time to pass before some malicious program or malicious activity is detected by anti-virus solutions, so they are practically an everyday occurrence. In this paper, we analyze the network traffic in order to determine the characteristics that characterize it as abnormal, and after recognizing it, react appropriately. As we already mentioned in introduction one good approach uses TMR and using three state of the art algorithms constructs one with better characterictics than each individually used. However, the authors of this paper observed that with this classic TMR method, there is a problem in which two algorithms with poorer accuracy overvote the third one and thus affect the quality of the assessment of malicious connections, and thus the decision to trigger the alarm. Therefore, in this paper we present a method that uses an upgraded TMR with four state of the art binary classificacion algorithms to detect malicious network traffic, which will help to improve accuracy by overcoming the problem of bad overvotes.

1) DECISION TREES METHOD

Decision trees are a tool that provides decision support. It is a way to represent an algorithm that contains conditional statements. It is most often used in operations research, especially in decision analysis, to help determine the strategy most likely to achieve a particular goal, but they are also a popular tool in machine learning. Decision trees are among the most popular machine learning algorithms because of their comprehensibility and simplicity. The tree is built by splitting the original set, which forms the root node of the tree, into subsets that form the descendants. The segmentation is based on a set of segmentation rules based on classification features. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The ID3 algorithm starts with the original set as the base node. At each iteration, the algorithm iterates through each unused attribute of the set and calculates the entropy or information gain of that attribute. It then selects the attribute that has the lowest entropy (or highest information gain) value.

Briefly enumerated steps of the algorithm:

1. Calculate the entropy of each attribute of the data set.
2. Divide the set into subsets using the attribute for which the resulting entropy after splitting is minimized, or, equivalently, the information gain is maximized.
3. Form a tree node containing that attribute.
4. Return to subsets using the remaining attributes.

Entropy is a measure of the amount of uncertainty in a data set. The formula for calculating entropy is

$$H(S) = \sum_{x \in X} -p(x) \log_2 p(x) \tag{1}$$

wherein,

S - Current data set for which entropy is calculated. This is changed at each step of the ID3 algorithm, either to a subset of the previous set in the case of a split on an attribute, or to a twin node in the case that the recursion was previously terminated.

X - Set of classes in S

TABLE 1. Dataset attributes description.

#	Feature Name	Description	Type	Value Type
1	Duration	Length of time duration of the connection	Continuous	Integers
2	Protocol Type	Protocol used in the connection	Categorical	Strings
3	Service	Destination network service used	Categorical	Strings
4	Flag	Status of the connection – Normal or Error	Categorical	Strings
5	Heart Bytes	Number of data bytes transferred from source to destination in a single connection	Continuous	Integers
6	Dst Bytes	Number of data bytes transferred from destination to source in a single connection	Continuous	Integers
7	Land	If source and destination IP addresses and port numbers are equal then, this variable takes value 1 else 0	Binary	Integers
8	Wrong Fragment	Total number of wrong fragments in this connection	Discreet	Integers
9	Urgent	Number of urgent packets in this connection. Urgent packets are packets with the urgent bit activated	Discreet	Integers
10	Hot	Number of "hot" indicators in the content such as: entering a system directory, creating programs and executing programs	Continuous	Integers
11	Num Failed Logins	Count of failed login attempts	Continuous	Integers
12	Logged In	Login Status: 1 if successfully logged in; 0 otherwise	Binary	Integers
13	Num Compromised	Number of "compromised" conditions	Continuous	Integers
14	Root Shell	1 if root shell is obtained; 0 otherwise	Binary	Integers
15	Su Attempted	1 if "su root" command attempted or used; 0 otherwise	Discrete (Dataset contains '2' value)	Integers
16	Num Root	Number of "root" accesses or number of operations performed as a root in the connection	Continuous	Integers
17	Num File Creations	Number of file creation operations in the connection	Continuous	Integers
18	Num Shells	Number of shell prompts	Continuous	Integers
19	Num Access Files	Number of operations on access control files	Continuous	Integers
20	Num Outbound Cmds	Number of outbound commands in an ftp session	Continuous	Integers
21	Is Hot Logins	1 if the login belongs to the "hot" list ie, root or admin; else 0	Binary	Integers
22	Is Guest Login	1 if the login is a "guest" login; 0 otherwise	Binary	Integers
23	Count	Number of connections to the same destination host as the current connection in the past two seconds	Discreet	Integers
24	Srv Count	Number of connections to the same service (port number) as the current connection in the past two seconds	Discreet	Integers
25	Error Rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count (23)	Discreet	Floats (hundredths of a decimal)
26	Srv Serror Rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in srv_count (24)	Discreet	Floats (hundredths of a decimal)
27	Error Rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in count (23)	Discreet	Floats (hundredths of a decimal)
28	Srv Error Rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in srv_count (24)	Discreet	Floats (hundredths of a decimal)
29	The Srv Rates themselves	The percentage of connections that were to the same service, among the connections aggregated in count (23)	Discreet	Floats (hundredths of a decimal)
30	Diff Srv Rate	The percentage of connections that were to different services, among the connections aggregated in count (23)	Discreet	Floats (hundredths of a decimal)
31	Srv Diff Host Rate	The percentage of connections that were to different destination machines among the connections aggregated in srv_count (24)	Discreet	Floats (hundredths of a decimal)
32	Dst Host Count	Number of connections having the same destination host IP address	Discreet	Integers
33	Dst Host Srv Count	Number of connections having the same port number	Discreet	Integers
34	Dst Host Same Srv Rate	The percentage of connections that were to different services, among the connections aggregated in dst_host_count (32)	Discreet	Floats (hundredths of a decimal)

TABLE 1. *(Continued.)* **Dataset attributes description.**

35	Dst Host Diff Srv Rate	The percentage of connections that were to different services, among the connections aggregated in dst_host_count (32)	Discreet	Floats (hundredths of a decimal)
36	Dst Host Same Src Port Rate	The percentage of connections that were to the same source port, among the connections aggregated in dst_host_srv_count (33)	Discreet	Floats (hundredths of a decimal)
37	Dst Host Srv Diff Host Rate	The percentage of connections that were to different destination machines, among the connections aggregated in dst_host_srv_count (33)	Discreet	Floats (hundredths of a decimal)
38	Dst Host Serror Rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst_host_count (32)	Discreet	Floats (hundredths of a decimal)
39	Dst Host Srv Serror Rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst_host_srv_count (33)	Discreet	Floats (hundredths of a decimal)
40	Dst Host Error Rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst_host_count (32)	Discreet	Floats (hundredths of a decimal)
41	Dst Host Srv Rerror Rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst_host_srv_count (33)	Discreet	Floats (hundredths of a decimal)
42	Class	Classification of the traffic input	Categorical	Strings

p(x) – the ratio of the number of elements in class x to the number of elements in set S

When H(S) = 0, then the set S is ideally classified

The information gain IG(A) is a measure of the difference in entropy between the state before and after the set S is partitioned by attribute A. In other words, how much the uncertainty in the set S is reduced after splitting the set S based on the attribute A. The information gain is calculated by the following formula:

$$IG(S, A) = H(S) - \sum_{t \in T} p(t) H(t) = H(S) - H(S|A) \tag{2}$$

wherein,

H(S) - Entropy of the set
T - A subset created by dividing the set based on the attribute such that it holds
p(t) – the ratio of the number of elements in utu to the number of elements in the set S
H(t) - Entropy of subset t

2) NAIVE BAYES

The Naive Bayes classifier is a machine learning model based on conditional probability used for classification problems. The essence of the classifier is based on Bayes theorem.

$$P(A|B) = \frac{P(B|A)\, P(A)}{P(B)} \tag{3}$$

In statistics, naive Bayes classifiers are a family of simple ''probabilistic classifiers'' based on the application of Bayes' theorem with strong (naive) assumptions of independence between features. Using Bayes' theorem, it is possible to find the probability of A happening, given that B has happened. Here B is evidence and A is hypothesis. The assumption here is that the features are independent. This means that the presence of one feature does not affect the other. Naive Bayesian classifiers are highly scalable and require a set of parameters that are linear in the number of variables (features/predictors) in the learning problem. From the perspective data set, Bayes theorem can be written in the following form:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \tag{4}$$

The variable y is the class variable, which represents the dependent variable, that is, the outcome in the case of classification under the conditions of the independent variable X, which represents the parameters/characteristics. X is given in the form:

$$X = (x_1, x_2, x_3, \ldots, x_n) \tag{5}$$

Here $x_1, x_2, x_3, \ldots, x_n$ they represent the characteristics, ie. can be mapped according to e.g. appearance, temperature, humidity and wind. Substituting for X and expanding using the chain rule gives:

$$P(y|x_1, \ldots, x_n) = \frac{P(x_1|y)P(x_2|y)\ldots P(x_n|y)P(y)}{P(x_1)P(x_2)\ldots P(x_n)} \tag{6}$$

3) SUPPORT VECTOR MACHINE (SVM)

Support Vector Machine is another simple algorithm that every machine learning expert should have in their arsenal. Many prefer a support vector machine because it produces significant accuracy with less computing power. Support Vector Machine, SVM for short, can be used for both regression and classification tasks. But it is widely used for classification purposes. The goal of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N — number of features) that clearly classifies the data points. To separate the two classes of data points, there are many possible hyperwaves that can be chosen. Our goal is to find

the plane that has the maximum margin, that is, the maximum distance between the data points of both classes. Maximizing the margin distance provides some boost so that future data points can be classified with more confidence. Hyperplanes are decision boundaries that help classify data points. Data points falling on both sides of the hyperplane can be assigned to different classes. Also, the dimension of the hyperplane depends on the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to visualize when the number of features exceeds 3. Support vectors are data points that are closer to the hyperplane and affect the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vector will change the position of the hyperplane. These are the points that help us build our SVM.

4) KNN ALGORITHM

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve classification and regression problems. A classification problem has a discrete value as an output. For example, ''likes pineapple on pizza'' and ''dislikes pineapple on pizza'' are discrete. There is no middle ground. The data in a classification problem would look like there is a predictor (or set of predictors) and a label. Standard practice is to represent the output (label) of a classification algorithm as an integer such as 1, −1, or 0. Mathematical operations should not be performed on them as this would be meaningless. A regression problem has a real number (a number with a decimal point) as the output. The data used in regression analysis will look similar. There is an independent variable (or set of independent variables) and a dependent variable (the thing we are trying to predict given our independent variables). Also, each row is usually called a sample, observation, or data point, while each column (not including the label/dependent variable) is often called a predictor, dimension, independent variable, or characteristic. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are close to each other. The KNN algorithm depends on this assumption being correct enough for the algorithm to be useful. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some math we may have learned as children—calculating the distance between points on a graph. There are other ways of calculating distance, and one way might be preferable depending on the problem we are solving. However, straight-line distance (also called Euclidean distance) is a popular and well-known choice. The advantage of the algorithm is that it is simple and easy to implement, there is no need to create a model, adjust several parameters or make additional assumptions. Also, the algorithm is versatile, it can be used for classification, regression and search. On the other hand, the disadvantages are that the algorithm becomes significantly slower as the number of examples and/or predictors/independent variables increases.

KNN algorithm by steps:

1. Load data
2. Initialize K to the chosen number of neighbors
3. For each example in the data
   a. Calculate the distance between the query example and the current example from the data.
   b. Add distance and sample index to ordered collection
4. Sort an ordered collection of distances and indices from smallest to largest (in ascending order) by distances
5. Select the first K entries from the sorted collection
6. Get the labels of the selected K entries
7. If regression, return the mean of the K labels
8. If classification, return K tag mode

5) LOGISTIC REGRESSION (LR)

Logistic regression (LR) is a technique in the field of statistics used for binary classification tasks. It is primarily used to predict binary outcomes of the form Yes/No, as well as True/False. Here, the logistic function or the so-called sigmoid function used to model the probability, where the input belongs to a certain class. The sigmoid function is responsible for ensuring that there are predicted probabilities in the range from 0 to 1. Logistic regression is given by the equation:

$$P(Y=1)\frac{1}{1+e^{-(b_0+b_1X_1+b_2X_2+\cdots+b_nX_n)}} \tag{7}$$

In this equation:

- $P(Y=1)$ represents the probability of the dependent variable Y being 1.
- E is the base of the natural logarithm.
- $b_0, b_1, \ldots, b_n$ are the coefficients of the model.
- $X_1, X_2, \ldots, X_n$ denote the input features.

The basic task of logistic regression is to estimate coefficients in order to minimize the difference between predicted probabilities and actual outcomes. the wide application of logistic regression is due to its simplicity and efficiency when solving problems within the framework of binary classification.

6) ADA BOOST (AB)

Adaptive Boosting (AdaBoost) is an ensemble learning technique applied to solving problems in the field of classification and regression. It is one of the first algorithms in its class, and it refers to the idea of Boosting, which means: combining multiple ''weak classifiers'' into a single ''strong classifier''.

The AdaBoost algorithm iteratively trains a series of weak classifiers on different subsets of the training data, where at each iteration the algorithm assigns higher weights to misclassified samples from the previous iteration, focusing on more demanding examples. This approach allows subsequent weak classifiers to pay more attention to previously misclassified samples and improve their performance.

The typical AdaBoost iteration involves the following steps:

- Training a weak learner on the dataset and assessing its performance.
- Adjusting the weights of misclassified examples, subsequently retraining the weak learner on the updated dataset.
- Repeating the process for a predefined number of iterations or until achieving a near-perfect predictor.

The main advantages of the AdaBoost algorithm are its adaptability and efficiency in improving model performance, where special focus is placed on instances that are of great importance for classification.

7) ENSEMBLE METHOD

The ensemble method involves combining a number of models, which may not be good enough on their own, in order to make a better joint decision. By averaging, independent errors cancel out, which significantly improves prediction accuracy. There are different approaches to ensemble construction, but some of them are consistently among the best methods for solving different problems in terms of the accuracy they offer. Typically when data are represented in vector form, ensembles represent the most reliable approach to achieving high prediction accuracy. Of course, their quality and efficiency also depend on the type of models that make up the ensemble.

The most famous ensemble methods are:

- Bagging is the type of ensemble technique in which a single training algorithm is used on different subsets of the training data where the subset sampling is done with replacement (bootstrap). Once the algorithm is trained on all the subsets, then bagging predicts by aggregating all the predictions made by the algorithm on different subsets.
- Boosting refers to a family of algorithms which converts weak learners to strong learners. Boosting is an ensemble method for improving the model predictions of any given learning algorithm. The idea of boosting is to train weak learners sequentially, each trying to correct its predecessor. The weak learners are sequentially corrected by their predecessors and, in the process, they are converted into strong learners.
- Stacking is an ensemble learning method that combines multiple machine learning algorithms via meta-learning, In which base level algorithms are trained based on a complete training data-set, the meta-model is trained on the final outcomes of the all base-level model as a feature. We have a deal with bagging and boosting methods for handling bias and variance. Now we can learn stacking which improves your model prediction accuracy.
- Voting works because the opinion of the majority holds more weight than the vote on an individual. Max Voting is used when we have discrete options on which the models can take a vote. The option that has the most number of votes is considered the chosen one. It is used for classification problems. Each machine learning model makes a vote, and the option with the maximum vote is the selected option.
- Averaging - For a data point where trying to predict, multiple predictions are made by various models. The average of the model predictions is the final prediction that we consider.

The voting method was used in this paper through application of TMR but from the standpoint that it is applied one ensemble machine learning method simultaneously it could be said that proposed algorithm belongs to stacking ensemble methods because it uses more, exactly four different classification algorithms and TMR voting as so called combiner algorithm.

8) CLASSIFICATION QUALITY MEASUREMENT PARAMETERS

To evaluate the quality of the classification, the following parameters were calculated: accuracy (eng. Accuracy), precision (eng. Precision), sensitivity (eng. Sensitivity), F1-score rating of the accuracy of the model on the data set, calculated on the basis of sensitivity and precision, as well as the true positive ratio (TPR) and false positive ratio (FPR) on the end. In order for these parameters to be calculated, the first step was to calculate the value of TP, True Positive, which represents the number of correct positive predictions that in our case make up successfully recognized attacks, then TN, True Negative, which represents the number of actually predicted negatives, which in our case represent connections where the normal state is recognized. FP, false positive (Eng. False Positive) which represents the number of wrong positive predictions and finally false negative, FN (Eng. False negative), which represents the number of wrongly predicted negatives. After that, other parameters were calculated according to the following formulas

$$Accuracy = \frac{(TP + TN)}{(TP + FP + FN + TN)} \tag{8}$$

$$Precision = \frac{TP}{(TP + FP)} \tag{9}$$

$$Sensitivity = \frac{TP}{(TP + FN)} \tag{10}$$

$$F1\_score = \frac{2^{*}\ (Sensitivity^{*}Precision)}{(Sensitivity + Precision)} \tag{11}$$

Also, in machine learning, performance measurement is an essential task. So, when it comes to the problem of classification, we can count on the AUC - ROC curve. When we need to check or visualize the performance of a multi-class classification problem, we use the AUC (Area Under the Curve) ROC (Receiver Operating Characteristics) curve. It is one of the most important evaluation metrics to check the performance of any classification model. Also written as AUCROC (Area Under Curve Receiver Operating Characteristics). It tells how well the model is able to distinguish classes. The higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1. The ROC curve is plotted with TPR versus FPR where TPR is on the y-axis and FPR is on the x-axis.

It is very important to notice that exist many other performance measures and for more comparsion and

sensitivity/stability analysis of obtained results with prposed model in this paper we will use also two more evaluation measures indicated in literature [51] that they are favourable over alredy mentioned in binary classification evaluation on imbalanced datasets. This two performance measures are:

The Matthew score relation coefficient (MCC) is one of the best measures for imbalanced datasets, which takes into account both true and false positives and negatives. The MCC can be calculated with next formula:

$$MCC = \frac{TP^*TN - FP^*FN}{\sqrt{(TN + FN)(FP + TP) + (TN + FP)(FN + TP)}} \tag{12}$$

MCC varies in interval $(-1; 1)$, with value 1 as perfect prediction, value 0 as random prediction and value -1 as total disagreement between the actual and prediction.

We will as most appropriate measure for goodness of binary classification in the case of imbalanced datasets use Precision-Recall Curve (PRC) and its AUCPRC value which is more informative than ROC curve and its AUCROC value in the case of binary classification of imbalanced datasets. Recall curve shows a trade-off between precision and recall.The area under the curve(PRC) is called AUC PRC. There are several aspects in which PRC is different from ROC curve. Firstly,the base line for ROC curve is fixed, diagonal connecting (0,0) and (1,1) in coordinate system, while the base line for PRC is correlated with the class distribution P/(N+P).

Secondly, interpolation methods for PRC analysis uses non-linear interpolation andROCanalysis uses linear.

IV. PROPOSED MODEL

The diagram of the proposed solution with its functions is presented below using unified modeling language (UML). As it is shown, IDS is working by first check if the considered training dataset is imbalanced (in the case that one of the two possible output values is present in less than 20% of total number) and in the case it is proposed model uses the AUCPRC measure for selection best classification algorithm, otherwise AUCROC. It will be showed that in our case we have balanced dataset and because of that AUCROC measure is used for this purpose. Then IDS select the four algorithms that will participate in the model. These algorithms are determined as the best four from a group of six most used and state of the art binary selection algorithms [52] on concrete dataset with AUC ROC (AUC PRC) measure as one which is most approprite [53] for also already presented classification quality measures. In our case, these are already described algorithms in previous section: DT, KNN, NB, SVM, LR and AB and it will be showed that the best four are KNN, AB, DT, SVM. In addition, all combinations without repetitions of these best four algorithms which will be also used in proposed model are determined in the corresponding TMR, and in our case they are KAD (KNN-AB-DT), KSD (KNN-SVM-DT), KAS (KNN-AB-SVM), ADS (AB–DT-SVM).

This is followed by the training phase, which involves training all these algorithms and each TMR combination on the test data set and determining their ROC(AUC) values. In addition, ROC(AUC) (AUC PRC) values are also calculated for each TMR combination. This is done in order to determine the best TMR, i.e. TMR with best value of ROC(AUC) (AUC PRC), which completes the training phase.

After the training phase and determination of all ROC(AUC) (AUC PRC) values, the best TMR is selected and used in the test data set. As shown in the figure here, the algorithm enters the first loop, where, first, the counter is set to the initial value i =0. Also, the stop variable is defined, in which the length of the data set itself is placed, so that the algorithm knows when to stop execution.

Further, each connection from the data set is processed and for each checks whether the mentioned bad overvoting problem has occurred. If it is not, the output of the entire algorithm will be equal to the output of the selected or the best TMR, while if it is, for those cases, the TMR that has next best value of ROC(AUC) (AUC PRC) is used when deciding. After that, it is checked again if there is a problem of bad over-voting in the replacement TMR, and if not, the output of the whole algorithm will be equal to the output of the replacement TMR, while if it is, the algorithm with the highest ROC(AUC) (AUC PRC) value from first TRM is used for decision making. Finally, depending on whether the connection being analyzed is recognized as valid network traffic, it is passed, the counter is incremented, the training data set is updated for future training and the next connection is examined, while if the connection is recognized as malicious, an alarm is activated, the training data set is also updated and the network traffic is interrupted.

Each of the moduls of the algorithm is described below.

As shown in the picture, in the module for determining the output values of each TMR, a comparison of the outputs of the individual algorithms that make up a specific TMR is performed, on the basis of which this value is determined. Thus, in the case of TMR, which implies a combination of naive bayes, knn and a decision tree (TMR_), if the output of algorithms NB and KNN are equal, the output of the entire TMR will take the value of the output of NB. In the case that the output of the NB and DT algorithms is equal, the output will be equal to the value of the NB output, while in the case that the NB output differs from the output of KNN and DT, the output of TMR will be equal to the output of the KNN algorithm. Furthermore, in the case of TMR, which implies a combination of support vector machines, knn and a decision tree (TMR_SKD), if the output of the SVM and KNN algorithms is equal, the output of the entire TMR will take the value of the output of the SVM. In the event that the output of the SVM and DT algorithms is equal, the TMR output will be equal to the SVM output, while in the event that the SVM output differs from both the KNN and DT outputs, the TMR output will be equal to the KNN algorithm output.

Similarly, in the case of TMR, which implies a combination of naive bayes, support vector machines and knn

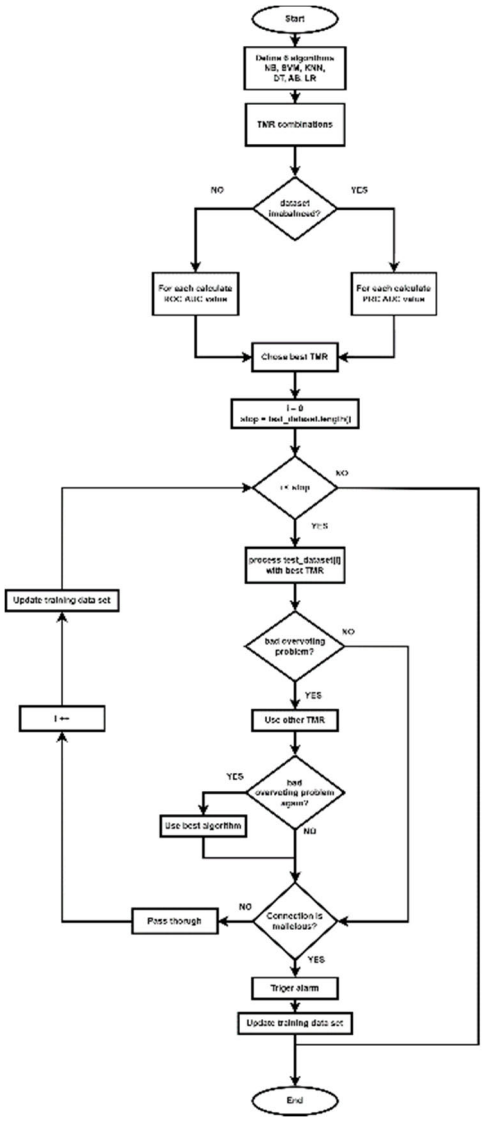


FIGURE 1. Diagram of the proposed model.

(TMR_NSK) algorithms, if the output of the NB and SVM algorithms is equal, the output of the entire TMR will take the value of the NB output. In the case that the output of the NB and KNN algorithms is equal, the output of TMR will be equal to the output of NB, while in the case that the output of NB differs from both the output of SVM and KNN, the output of TMR will be equal to the output of the SVM algorithm. Finally, in the case of TMR, which implies a combination of naive Bayes, support vector machine and decision tree (TMR_NSD) algorithms, if the output of the NB and SVM algorithms is equal, the output of the entire TMR will take the value of the NB output. In the case that the output of the NB and DT algorithms are equal, the output will be equal to the value of the NB output, while in the case that the NB output is different from the SVM and DT outputs, the TMR output will be equal to the SVM algorithm output.

In the algorithm module that serves to determine the best algorithm, their auc values are compared. First, the

```
//module for determining TMR outputs
TMR_KDA:
if OUTPUT_KNN = OUTPUT_DT or OUTPUT_KNN = OUTPUT_AB then
    OUTPUT_KDA → OUTPUT_KNN;
else
    OUTPUT_KDA → OUTPUT_DT;

TMR_SDA:
if OUTPUT_SVM = OUTPUT_DT or OUTPUT_SVM = OUTPUT_AB then
    OUTPUT_SDA → OUTPUT_SVM;
else
    OUTPUT_SDA → OUTPUT_DT;

TMR_SKA:
if OUTPUT_SVM = OUTPUT_KNN or OUTPUT_SVM = OUTPUT_AB then
    OUTPUT_SKA → OUTPUT_SVM;
else
    OUTPUT_SKA → OUTPUT_KNN;

TMR_SKD:
if OUTPUT_SVM = OUTPUT_KNN or OUTPUT_SVM = OUTPUT_DT then
    OUTPUT_SKD → OUTPUT_SVM;
else
    OUTPUT_SKD → OUTPUT_KNN;
```

FIGURE 2. Module for determining TMR output values.

```
//module for chosing the best algorithm
best_alg→ALG_DT
if best_alg.auc < ALG_KNN.auc then
    best_alg→ALG_KNN
if best_alg.auc < ALG_AB.auc then
    best_alg→ALG_AB
if best_alg.auc < ALG_SVM.auc then
    best_alg→ALG_SVM
```

FIGURE 3. Module for determining the best algorithm.

variable best_alg is initialized to the value of the decision tree algorithm. Then, if the auc value of that variable is smaller than the next algorithm, it will take its value and so on for all four algorithms.

Accordingly, the module for determining the best TMR compares the auc values of all TMRs. As in the module for determining the best algorithm, the variable named best_tmr is initialized to the value TMR_NKD. After that, if its auc value is less than the auc value of the next tmr, it is set to the value of that algorithm, and so for all four tmr.

```
//module for chosing the best TMR
best_tmr→TMR_KDA
if best_tmr.auc < TMR_SDA.auc then
    best_tmr→TMR_SDA
if best_tmr.auc < TMR_SKA.auc then
    best_tmr→TMR_SKA
if best_tmr.auc < TMR_SKD.auc then
    best_tmr→TMR_SKD
```

FIGURE 4. Module for determining the best TMR.

```
//module for detecting bad overvoting problem
bad_overvoting_problem→FALSE
if(best_tmr.alg1.auc > best_tmr.alg2.auc
    and
    best_tmr.alg1.auc > best_tmr.alg3.auc
    and
    best_tmr.alg1.predicted != best_tmr.predicted)
then bad_overvoting_problem→TRUE
if(best_tmr.alg2.auc > best_tmr.alg1.auc
    and
    best_tmr.alg2.auc > best_tmr.alg3.auc
    and
    best_tmr.alg2.predicted != best_tmr.predicted)
then bad_overvoting_problem→TRUE
if(best_tmr.alg3.auc > best_tmr.alg1.auc
    and
    best_tmr.alg3.auc > best_tmr.alg2.auc
    and
    best_tmr.alg3.predicted != best_tmr.predicted)
then bad_overvoting_problem→TRUE
```

FIGURE 5. Bad over-voting problem detection module.

```
//module for determining the worst algorithm in TMR
x→best_tmr.alg1.auc
y→best_tmr.alg2.auc
z→best_tmr.alg3.auc
if((x<y) and (x<z)) then
    worst_algoritm → best_tmr.alg1
if((y<x) and (y<z)) then
    worst_algoritm → best_tmr.alg2
if((z<x) and (z<y)) then
    worst_algoritm → best_tmr.alg3
```

FIGURE 6. Module for determining the worst algorithm in TMR.

As for the module for checking the mentioned overvoting problem, it consists in first setting the value of the variable with the name bad_overvoting_problem to the value FALSE. Then, if the output value of the algorithm with the highest auc value in tmr is different from the output value of the entire tmr, this variable becomes TRUE.

In order for the optimized TMR algorithm to be able to perform a potential replacement of one tmr with another, it is necessary to know what caused the problem of bad overvoting, that is, which algorithm is responsible for it. For this purpose, a module was developed for determining the worst algorithm within tmr. The logic behind this module is that first the variables x, y and z are initialized to the value of the auc value of the respective algorithms, so that the variable x represents the auc value of the first algorithm in tmr. The variable y represents the auc value of the second algorithm in tmr, az the auc value of the third one. After that, depending on which variable has the smallest value, it is determined which algorithm is the worst, as in the picture below.

Further in the algorithm, if there is an overvoting problem and after a deviant algorithm has been determined, it is possible to replace the TMR. For this purpose, an algorithm module was developed that applies the following logic as shown in the figure. This module consists of nested case structures. The first case checks which tmr is in use, so there are more cases:

1. If the best tmr is TMR_KDA, a new case is then entered to check which is the worst algorithm out of the three in that tmr. If it is the algorithm ALG_KNN, the tmr with which the original tmr is changed is TMR_SDA, if it is the algorithm ALG_DT, the replacement tmr is TMR_SKA, while if it is the algorithm ALG_AB, the tmr with which the original tmr will be replaced is TMR_SKD.
2. The case when the best tmr is TMR_SDA, in the other case it is checked which is the worst algorithm out of three in that tmr. If it is the algorithm ALG_SVM, the tmr with which the original tmr is changed is TMR_KDA, if it is the algorithm ALG_DT, the replacement tmr is TMR_SKA, while if it is the algorithm ALG_AB, the tmr with which the original tmr will be replaced is TMR_SKD.
3. The best tmr is TMR_SKA. If the worst algorithm is ALG_SVM, the tmr with which the original tmr is changed is TMR_KDA, if it is the ALG_KNN algorithm, the replacement tmr is TMR_SDA, while if it is the ALG_AB algorithm, the tmr with which the original tmr will be replaced is TMR_SKD.
4. TMR_SKD is the best. If the worst algorithm is ALG_SVM, the tmr that replaces the original tmr is TMR_KDA, if it is the algorithm ALG_KNN, the replacement tmr is TMR_SDA, while if it is ALG_DT, the replacement tmr is TMR_SKA.

```
//module for changing TMR
    case (best_tmr) of
        TMR_KDA:case (worst_algoritm) of
            ALG_KNN : best_tmr→TMR_SDA
            ALG_DT : best_tmr→TMR_SKA
            ALG_AB : best_tmr→TMR_SKD

        TMR_SDA:case (worst_algoritm) of
            ALG_SVM : best_tmr→TMR_KDA
            ALG_DT : best_tmr→TMR_SKA
            ALG_AB : best_tmr→TMR_SKD

        TMR_SKA:case (worst_algoritm) of
            ALG_SVM : best_tmr→TMR_KDA
            ALG_KNN : best_tmr→TMR_SDA
            ALG_AB : best_tmr→TMR_SKD

        TMR_SKD:case(worst_algoritm) of
            ALG_SVM : best_tmr→TMR_KDA
            ALG_KNN : best_tmr→TMR_SDA
            ALG_DT : best_tmr→TMR_SKA
```

FIGURE 7. TMR changeover module.

Finally, the module for triggering the alarm consists in initializing the variable alarm to the value FALSE, so if the

output value optimizedTMR is equal to 1, this variable gets the value TRUE and the alarm is triggered.

```
//module for triggering alarm
alarm→FALSE
if(best_tmr.predicted = 1) then
    alarm→TRUE
```

FIGURE 8. Alarm triggering module.

V. RESULTS, DISCUSSION AND FINDINGS

In this chapter, the authors of the paper dealt with the implementation that can be used as a useful solution in the form of an application for stationary and mobile devices in Python.

A. IMPLEMENTATION

The proposed solution was developed and implemented by the authors on the system that has a processor Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz and 16GB of RAM. As for the software, the PyCharm Community Edition, IDE for Python developed by JetBrains, version 2023.3 was used. Operating system was Linux Ubuntu 22.04.3 LTS. The proposed solution was developed and imlpementated in Python programming language due to its relatively simple use and the existence of a large number of libraries that enable working with machine learning algorithms and metrics, such as pandas and sklearn, which will be discussed later in next separate chapter Technical solution of proposed model. However, first of all, the original data of the training data set had to be transformed, because they were not edited and therefore inadequate for processing, so it was necessary to bring them into a more adequate form. Here we are specifically talking about an excessive number of modalities, i.e. the appearance of some of the characteristics, so for this purpose the Struges rule for data grouping was applied. When the training data set is finally formed, all the algorithms trained on it are formed. In order to group the data we use the Struges rule, to determine the number of modalities we use:

$$K = 1 + 3.3^* \log N \tag{13}$$

And to determine the width of the interval we use:

$$i = \frac{x_{\max} - x_{\min}}{K} \tag{14}$$

For this purpose, for each column that needs a new grouping, a maximum and a minimum value are determined. After that, the original values are mapped to the new ones according to the corresponding membership. This is the training set completed and used during implementation.

B. FINDING AND DISCUSSION

First of all now corrected algorithm check in the first added block of algorithm, is the used training dataset imbalanced. The resultats showed in Table 2 and based on those results concluded that in our case of used Kaggle dataset we have balanced dataset and based on those results the model should

TABLE 2. Measuring of balancing of used dataset.

CONNECTION TYPE	NUMBER	%
ANOMALY	11743	46.61%
REGULAR	13449	53.39%
TOTAL	25192	100%

TABLE 3. Achieved results in mutual comparison in all measured values.

algoritm	TP	TN	FP	FN	ACC	F1	PRE	SEN	PRC-AUC	MCC	ROC-AUC
OPT	2221	2640	48	91	0,9723	0,9698	0,9790	0,9610	0,9972	0,9445	0,9968
SDA	2209	2673	15	103	0,9763	0,9738	0,9931	0,9552	0,9945	0,9527	0,9963
SKA	2206	2670	18	106	0,9754	0,9728	0,9921	0,9543	0,9945	0,9509	0,9962
KDA	2247	2656	32	65	0,9805	0,9787	0,9859	0,9717	0,9902	0,9609	0,9961
SKD	2252	2662	26	60	0,9826	0,9810	0,9884	0,9738	0,9964	0,9651	0,9961
KNN	2257	2643	45	55	0,9801	0,9784	0,9806	0,9762	0,9888	0,9599	0,9940
AB	2190	2628	60	122	0,9635	0,9599	0,9731	0,9471	0,9937	0,9267	0,9937
DT	2265	2648	40	47	0,9825	0,9810	0,9826	0,9795	0,9774	0,9648	0,9890
SVM	2101	2672	16	211	0,9545	0,9486	0,9923	0,9086	0,9919	0,9108	0,9888

be applied in the continuing the AUC ROC as a evaluation measure of goodnes of classification of each used algorithm in proposed model.

All tmr are tested to see all the advantages of the presented solution. After marking the malicious connections, we started measuring for all tmr combinations to see how they perform and compare. Finally, after the proposed cutting was tested, all the results were combined and compared with each other. Thus, as you can see in the table below, the proposed solution has the best results looking ROC-AUC and PRC-AUC categories which was the main goal of research and it recorded good results in other measured categories as well-Table 3.

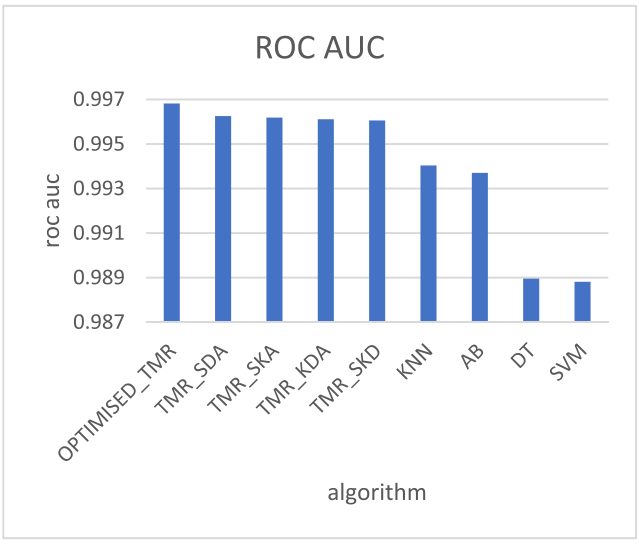
In addition, below are graphs 1 - graphs 6 with the results of all measured categories. The x-axis shows the compared ensembles as well as the proposed solution, while the y-axis shows the values recorded by them in each of the measured categories. As can be seen in the graphic, all the tested ansabml methods have a high ROC AUC value, over 0.996, but the proposed solution surpassed that and has the highest ROC AUC value of all with a value 0.9968.

As can be seen in the graphic, all the tested ansabml methods have a high PRC AUC value, over 0.993, except the TMR_KDA with 0.9902, but the proposed solution has the highest ROC AUC value of all with a value of 0.9972.

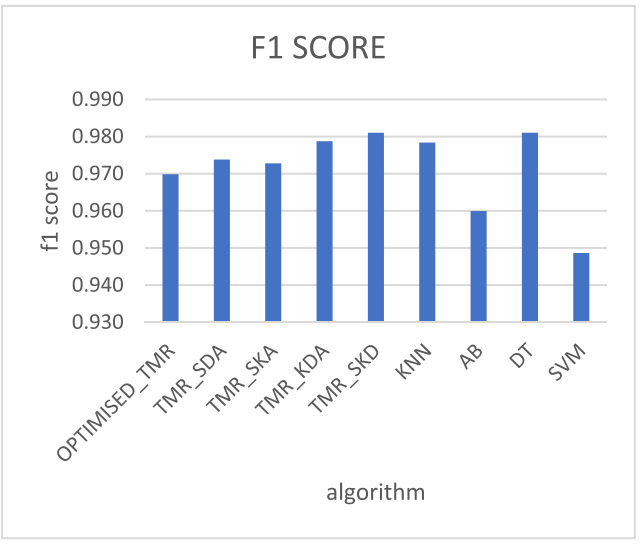
When it comes to the f1-score, as can be seen from the graph, the proposed solution has the lowest achieved result. But, it should be kept in mind that the differences between all of them are very small, bearing in mind that not a single ensemble method has a result worse than approximately 0.97.

As shown in the graphic, when it comes to sensitivity, all methods achieved results above 0.95 and proposed method has good result over 0,96.

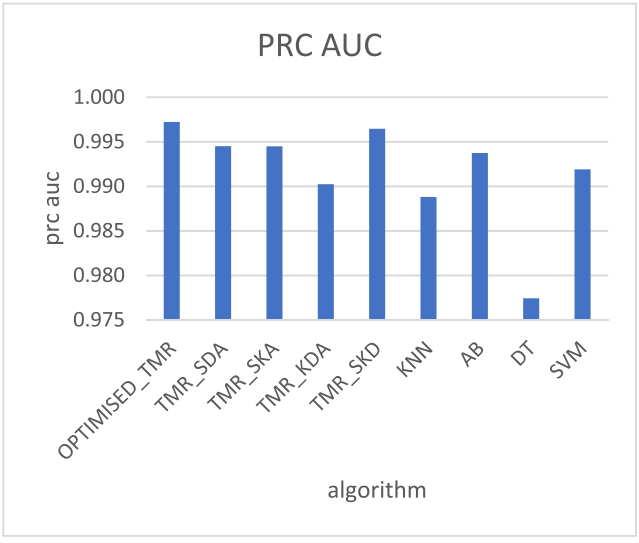
As shown in the graphic, proposed solution has lowes value for accuracy but all the tested ensemble methods have a high



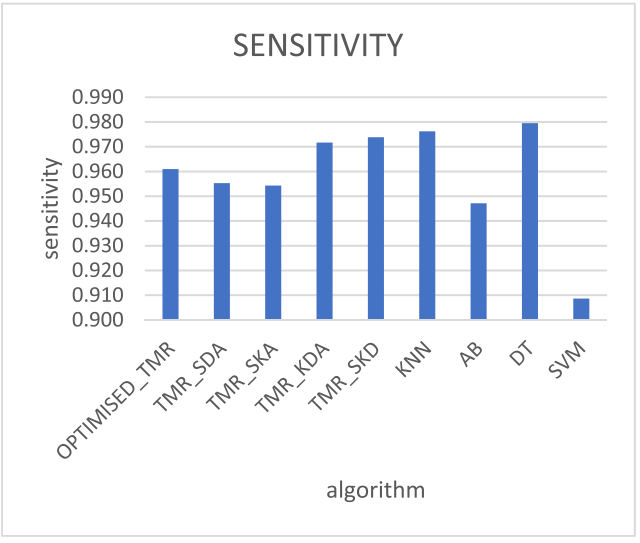
GRAPH 1. ROC AUC values of the proposed solution and other TMRs.



GRAPH 3. F1-score of the proposed solution and other TMRs.



GRAPH 2. PRC AUC values of the proposed solution and other TMRs.



GRAPH 4. Sensitivity values of the proposed solution and other TMR.

score, approximately 0.97 and the best is TMR_SKD with score of approximately 0,98.

Regarding the precision, as shown in the graphic, proposed solution has lowest value but all the tested ensemble methods have a high score, approximately 0.98 and the best is TMR_SKD with score of approximately 0,99.

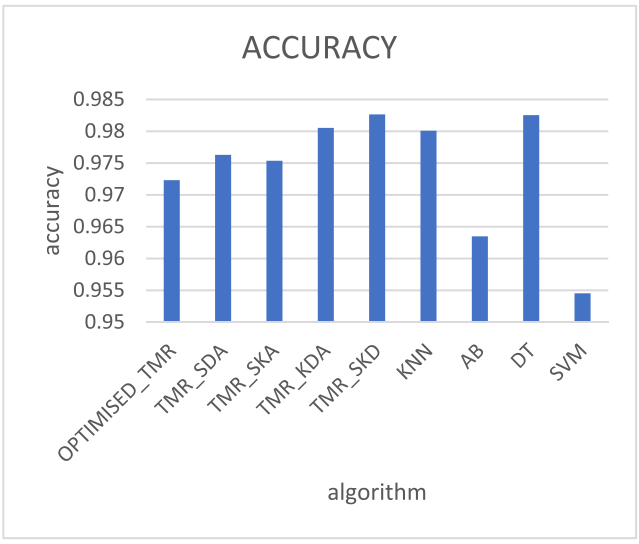
C. VALIDATION

For the validation of the results, a five-fold cross-validation was performed, which meant that the training data set was divided into a part for training and for verification using the method of random selection, in a 60/40 ratio. Using the first part of the data set, the prediction of the output value of each tmr according to the module of the algorithm described above was determined, then the auc values for all algorithms and all tmr were calculated in order to determine the best one. Finally, by modules for checking bad overvoting problems and replacing tmr, a part of optimizedTMR is implemented. Finally, according to the alarm triggering module, the classification of new connections to normal network traffic, i.e. attack, is determined.

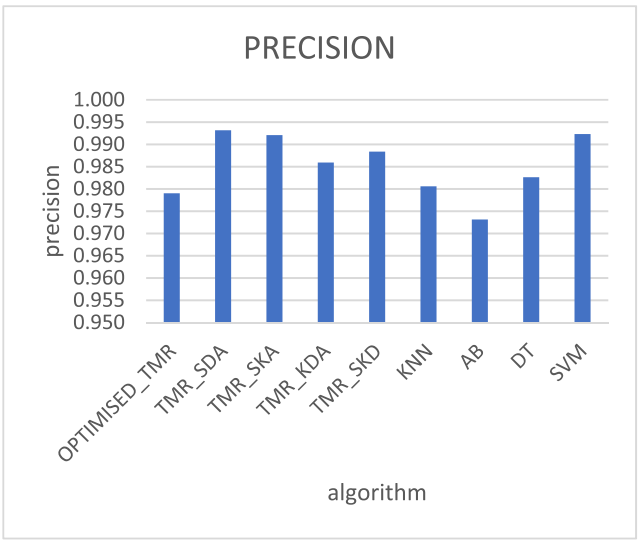
D. LIMITATION AND FUTURE WORK

The execution time of the proposed model and its working with imbalanced datasets authors considered as a potential problems and limitations of its application.

What can be a potential limitation of the proposed solution is only the execution time, because the solution involves more i.e. N algorithms, that is, in this concrete case, 4 algorithms. All those algorithms, as already described, should be trained first, then all combinations of TMR should be determined and in them the corresponding algorithms should make a joint decision, after which it should be checked whether the problem of bad overvoting occurs and, depending on that, if it



GRAPH 5. Accuracy values of the proposed solution and other TMR.



GRAPH 6. Precision values of the proposed solution and other TMR.

is necessary to replace the TMR. And as is known, these types of solutions must not be slow and must react immediately. However, taking into account that the training and determination of TMR combinations, as well as the determination of the best and worst algorithms constitute preprocessing, as well as the fact that with the development of multi-threaded systems it is possible to execute certain system functionalities asynchronously and it does not represent such a big obstacle in terms of performance.

As a particulary important fact and potential limitation in application of proposed model, the authors state that the problem of working with an imbalanced dataset for the application of the proposed model, which is a frequent case when it comes toDOS/DDOS attacks, they had solve it by implementig recommandations from the literature that are state of the art in solving this problem [54], [55], [56], [57], [58], [59], [60]. Namely, the authors:

1. Used the right evaluation metrics
2. Used 5-fold cross-validation in the right way in training
3. Used random selection in 60/40 ratio in validation
4. Used different ensemble methods
5. Used the recommanded Ada Boost.

Also, research gap which is directly connected with the problem of using the right evaluation metrics in the model, caused with type of balancing of used dataset, is solved in the way that is added one block on the beginning of diagram of proposed model. This block first check if the considered training dataset is imbalanced and, in the case it is, proposed model uses the AUCPRC measure for selection best classification algorithm, otherwise AUCROC.

In the future work, the authors will investigate further possible optimization of proposed solution in the sense of using other machine learning algorithms that would make up the future solution, but also, apart from other algorithms, it is certainly important to conduct research in terms of increasing the number of algorithms included in the solution. In addition, bearing in mind that these things are not mutually exclusive, the future work of the author would be based on increasing the number of modalities of the target variable, because the fact is that not all types of attacks on the network are equally dangerous for the protected system itself and it is not necessary, and sometimes it is even desirable to react to each type of attack in the same way.

Finally, there is certainly work with newer data sets with recorded new attacks based on newer technologies and their potential vulnerabilities.

VI. TECHNICAL SOLUTION OF THE PROPOSED MODEL

In this part, as it was said at the beginning of the first subchapter of previous chapter, technical matters related to the implementation of the proposed model in the Python programming language will be described-Figure 9. So, as it was said, the Python programming language was chosen because it offers several libraries that enable a relatively simple implementation of all artificial intelligence algorithms, as well as the metrics needed to compare and validate the obtained results.

At the beginning of the program, it is necessary to enter data. As explained earlier, the data is stored in a csv file, so the pandas library is used to load it into the data frame. The Pandas library must first be imported into the program with the command:

import pandas as pd

Then, to load the data from the csv file into the dataframe, it is necessary to call the *read_csv function*, which receives the path to the file as an argument:

data_train_df = pd.read_csv(“data_files/train.csv”)

Once that is done, it is necessary to determine the inputs and the target variable. The inputs behind the data frame are obtained by extracting the class column containing the dependent variable. This is done by calling the drop function:

inputs_train=data_train_df.drop(“class”,axis=’columns’

inputs_train=data_train_df.drop(“class”, axis=’columns’)

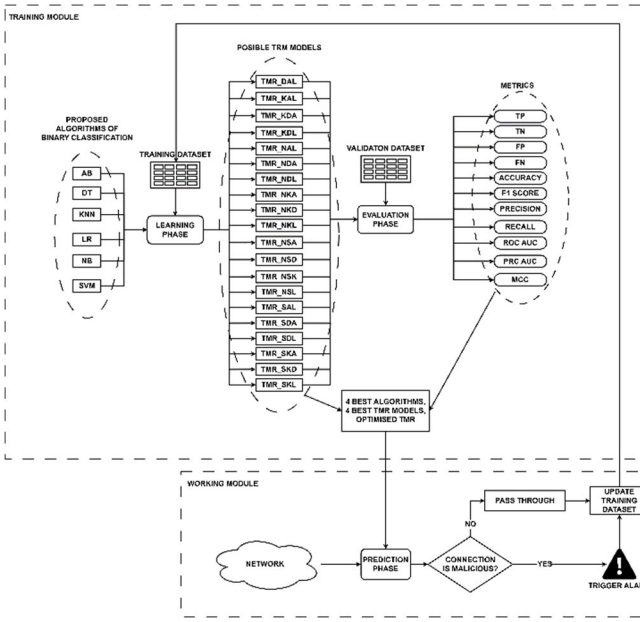


FIGURE 9. Architecture of proposed technical solution.

In contrast, the target variable is obtained by taking only the class variable from the data frame:

target_train = data_train_df["class"]

Next in the implementation is the definition of the model, that is, the machine learning classification algorithms used in the proposed model. As mentioned earlier, these are decision trees, support vector machine, k nearest neighbors and naive bayes. In order for them to be used, they must first be imported from the sklearn python library. Shown here are the commands to import decision trees, support vector machine, k nearest neighbors and naive bayes respectively:

from sklearn import tree

from sklearn.Svm import SV

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive_bayes import GaussianN

When classification models are imported into the main program, it is possible to define variables with them, in our case these are variables:

modelNb = GaussianNB()

modelSVM = SVC()

modelKnn = KNeighborsClassifier(

modelDt = three.DecisionTreeClassifier()

modelAb = AdaBoostClassifier()

modelLr = LogisticRegression()

When all the classification models are loaded and defined, it is possible to train them on the training data. This is done by the function fit, which receives the input variables and the output or target variable as parameters:

model.fit(inputs_train, target_train)

After the training, it is possible to perform the classification of the test data, with the fact that they are loaded from the csv file as well as the training data. This is done by calling the predict function, which receives as a parameter the input variables of the test data frame, which are extracted in the same way as the input variables of the training data frame.

y_predicted=model.predict(inputs_test)

After predicting or classifying each of the connections in the test data set, it is necessary to enter them into the data frame. This is done by simply defining a new column in the data frame and assigning a value to it:

data_test_df["model-predicted"]=y_predicted_nb

As for the ROC(AUC) values, they were calculated using the roc_auc_score function. In order for it to be used, it must first be introduced into the main program using the command:

from sklearn.metrics import roc_auc_scor

In order for this function to be able to calculate all the necessary ROC(AUC), the probabilities for belonging to one of the target classes had to be calculated first, whether it was normal network traffic or an attack. These probabilities are calculated using the predict_prob function which receives the input variables of the test data set as an input parameter

data_test_df["probs"]=model.predict_test(inputs_test

After the calculated probabilities, it is possible to calculate the ROC(AUC) value by calling the mentioned function roc_auc_score, which receives as a parameter the target variable of the test data set and the calculated probabilities:

roc_auc=roc_auc_score(target_test,data_test_df["probs"]

Finally, all the results are written into a new csv file using the to_csv function, which receives the path and name of the new file as a parameter, in our case it is:

data_test_df.to_csv(‘‘result/results.csv’’)

VII. CONCLUSION

In this paper, a new approach of applying the TMR triple modular redundancy in one novel ensemble method is presented. What prompted the authors to do this research is that the problem of bad over-voting was observed in using the classic TMR method, which, as described in the paper, represents a problem in which two of the three algorithms that make up TMR give the wrong prediction, i.e. perform the classification incorrectly and so overvote the algorithm that performed the classification correctly. Based on the results obtained from the research and comparison with the classic TMR and other state of the art binary classification algorithms, it is concluded that the proposed solution contributes to increasing the quality and reliability of network traffic classification, which confirms the basic hypothesis of the research goal, which is that it is possible to combine classification models into one aggregated system, but so that the aforementioned bad over-voting problem is avoided.

First of all, the motivation of the authors was to create this work because they did not find stacking ensemble methods that using proposed TMR voting i.e. decision as combiner algorithm in the literature.

Second, the reason stacking technique was chosen is because this type is the only one that can include TMR.

The proposed solution consists in the application of the so-called optimized TMR, solutions that imply the use of 4 algorithms and the implementation of all combinations

without repetition of their TMRs, in our case 4. The selected classification algorithms are Knn, Ada Boost, Decision trees and Support vector machine, and combinations of these algorithms combined into ensemble methods for Triple Modular Redundancy are TMR_KAD, TMR_KSD, TMR_KAS and TMR_ADS. As described in the part with the architecture of the proposed platform, the proposed solution consists in using the best TMR during classification, but checking for the existence of the problem of bad over-voting, and if it occurs, the TMR is replaced in suitable way.

The significance is twofold:

- first of all, this paper proposed a new, optimization method that eliminates the shortcomings of known TMR models and at the same time it has better characteristics in terms of the most important measures of binary classification than those but all other algorithms that are state of the art in that field.
- while on the other hand, this research implied the creation of a practical tool in which the method was implemented, as shown in the earlier part of the paper.

So, a big advantage of this solution is that it is possible to combine it as a hybrid solution with other existing ones and that by using Python it is relatively easy to implement, so it is also suitable for in-house development models in the case of data confidentiality and implementation in security protected structures and systems.

REFERENCES

[1] S. Pu, "Choosing parameters for detecting DDoS attack," in *Proc. Int. Conf. Wavelet Act. Media Technol. Inf. Process. (ICWAMTIP)*, Chengdu, China, Dec. 2012, pp. 239–242.

[2] K. Lee, J. Kim, K. H. Kwon, Y. Han, and S. Kim, "DDoS attack detection method using cluster analysis," *Expert Syst. Appl.*, vol. 34, no. 3, pp. 1659–1665, Apr. 2008, doi: 10.1016/j.eswa.2007.01.040.

[3] *DDoS Attack Types and Mitigation Methods*. [Online]. Available: https://www.imperva.com/learn/ddos/ddos-attacks

[4] A. Sanmorino and S. Yazid, "DDoS attack detection method and mitigation using pattern of the flow," in *Proc. Int. Conf. Inf. Commun. Technol. (ICoICT)*, Bandung, Indonesia, Mar. 2013, pp. 12–16.

[5] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, Dec. 2019.

[6] I. Babić, A. Miljković, M. Čabarkapa, V. Nikolić, A. Dordević, M. Randelović, and D. Randelović, "Triple modular redundancy optimization for threshold determination in intrusion detection systems," *Symmetry*, vol. 13, no. 4, p. 557, Mar. 2021, doi: 10.3390/sym13040557.

[7] Z. Zhang, D. Liu, Z. Wei, and C. Sun, "Research on triple modular redundancy dynamic fault-tolerant system model," in *IEEE MTT-S Int. Microw. Symp. Dig.*, vol. 1, Hanzhou, China, Jun. 2006, pp. 572–576, doi: 10.1109/IMSCCS.2006.119.

[8] R. E. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," *IBM J. Res. Develop.*, vol. 6, no. 2, pp. 200–209, Apr. 1962, doi: 10.1147/rd.62.0200.

[9] J. A. Abraham and D. P. Siewiorek, "An algorithm for the accurate reliability evaluation of triple modular redundancy networks," *IEEE Trans. Comput.*, vol. C-23, no. 7, pp. 682–692, Jul. 1974, doi: 10.1109/T-C.1974.224016.

[10] K. M. Santosh and E. Isaac, "Defending DDoS attack using stochastic model based puzzle controller," *Int. J. Comput. Sci. Netw. Secur.*, vol. 13, pp. 100–105, Jan. 2013.

[11] H. Rahmani, N. Sahli, and F. Kamoun, "A traffic coherence analysis model for DDoS attack detection," in *Proc. Int. Conf. Secur. Cryptogr.*, Milan, Italy, 2009, pp. 148–154.

[12] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: A classification," in *Proc. 3rd IEEE Int. Symp. Signal Process. Inf. Technol.*, Darmstadt, Germany, Dec. 2003, pp. 190–193.

[13] T. Mahjabin, Y. Xiao, G. Sun, and W. Jiang, "A survey of distributed denial-of-service attack, prevention, and mitigation techniques," *Int. J. Distrib. Sensor Netw.*, vol. 13, no. 12, Dec. 2017, Art. no. 155014771774146, doi: 10.1177/1550147717741463.

[14] J. David and C. D. Thomas, "DoS attack detection using fast entropy approach on flow-based network traffic," in *Proc. 2nd Int. Symp. Big Data Cloud Comput. Challenges*. Chennai, India: VIT Univ., 2015, pp. 30–36.

[15] Y. Wang, "A hybrid intrusion detection system," Ph.D. thesis, Iowa State Univ., Ames, IA, USA, 2004.

[16] S. E. Smaha, "Haystack: An intrusion detection system," in *Proc. 4th Aerosp. Comput. Secur. Appl.*, Orlando, FL, USA, Sep. 1988, pp. 37–44, doi: 10.1109/ACSAC.1988.113412.

[17] Y. Wang, L.-C. Lee, B. Xue, L. Wang, M. Song, C. Yu, S. Li, and C.-I. Chang, "A posteriori hyperspectral anomaly detection for unlabeled classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 6, pp. 3091–3106, Jun. 2018, doi: 10.1109/TGRS.2018.2790583.

[18] A. R. Yusof, N. I. Udzir, A. Selamat, H. Hamdan, and M. T. Abdullah, "Adaptive feature selection for denial of services (DoS) attack," in *Proc. IEEE Conf. Appl., Inf. Netw. Secur. (AINS)*, Miri, Malaysia, Nov. 2017, pp. 81–84, doi: 10.1109/AINS.2017.8270429.

[19] J. Kim, J. Kim, H. Kim, M. Shim, and E. Choi, "CNN-based network intrusion detection against denial-of-service attacks," *Electronics*, vol. 9, no. 6, p. 916, Jun. 2020, doi: 10.3390/electronics9060916.

[20] A. Mahfouz, A. Abuhussein, D. Venugopal, and S. Shiva, "Ensemble classifiers for network intrusion detection using a novel network attack dataset," *Future Internet*, vol. 12, no. 11, p. 180, Oct. 2020, doi: 10.3390/fi12110180.

[21] Y. Liu, X. Yu, J. X. Huang, and A. An, "Combining integrated sampling with SVM ensembles for learning from imbalanced datasets," *Inf. Process. Manage.*, vol. 47, no. 4, pp. 617–631, Jul. 2011, doi: 10.1016/j.ipm.2010.11.007.

[22] M. A. Faizal, M. M. Zaki, S. Shahrin, Y. Robiah, S. S. Rahayu, and B. Nazrulazhar, "Threshold verification technique for network intrusion detection system," 2009, *arXiv:0906.3843*.

[23] N. Idika and A. Mathur, *Survey of Malware Detection Techniques*. West Lafayette, IN, USA: Purdue Univ., 2007.

[24] D. Patel, K. Srinivasan, C.-Y. Chang, T. Gupta, and A. Kataria, "Network anomaly detection inside consumer networks—A hybrid approach," *Electronics*, vol. 9, no. 6, p. 923, Jun. 2020, doi: 10.3390/electronics9060923.

[25] M. Ahsan, M. Mashuri, H. Kuswanto, and D. D. Prastyo, "Intrusion detection system using multivariate control chart hotelling's T2 based on PCA," *Int. J. Adv. Sci., Eng. Inf. Technol.*, vol. 8, no. 5, pp. 1905–1911, Oct. 2018, doi: 10.18517/ijaseit.8.5.3421.

[26] C. Kim, M. Jang, S. Seo, K. Park, and P. Kang, "Intrusion detection based on sequential information preserving log embedding methods and anomaly detection algorithms," *IEEE Access*, vol. 9, pp. 58088–58101, 2021, doi: 10.1109/ACCESS.2021.3071763.

[27] L. A. Silva, V. R. Q. Leithardt, C. O. Rolim, G. V. González, C. F. R. Geyer, and J. S. Silva, "PRISER: Managing notification in multiples devices with data privacy support," *Sensors*, vol. 19, no. 14, p. 3098, Jul. 2019.

[28] A. S. Mendes, D. M. Jiménez-Bravo, M. Navarro-Cáceres, V. R. Q. Leithardt, and G. V. González, "Multi-agent approach using LoRaWAN devices: An airport case study," *Electronics*, vol. 9, no. 9, p. 1430, Sep. 2020, doi: 10.3390/electronics9091430.

[29] R. Duo, X. Nie, N. Yang, C. Yue, and Y. Wang, "Anomaly detection and attack classification for train real-time Ethernet," *IEEE Access*, vol. 9, pp. 22528–22541, 2021, doi: 10.1109/ACCESS.2021.3055209.

[30] S. Haider, A. Akhunzada, I. Mustafa, T. B. Patel, A. Fernandez, K. R. Choo, and J. Iqbal, "A deep CNN ensemble framework for efficient DDoS attack detection in software defined networks," *IEEE Access*, vol. 8, pp. 53972–53983, 2020, doi: 10.1109/ACCESS.2020.2976908.

[31] B. B. Gupta and A. Dahiya, *Distributed Denial of Service (DDoS) Attacks: Classification, Attacks, Challenges and Countermeasures*, 1st ed. Boca Raton, FL, USA: CRC Press, 2021.

[32] J. Yang, Y. Sheng, and J. Wang, "A GBDT-paralleled quadratic ensemble learning for intrusion detection system," *IEEE Access*, vol. 8, pp. 175467–175482, 2020, doi: 10.1109/ACCESS.2020.3026044.

[33] E. Söğüt and O. A. Erdem, "A multi-model proposal for classification and detection of DDoS attacks on SCADA systems," *Appl. Sci.*, vol. 13, no. 10, p. 5993, May 2023, doi: 10.3390/app13105993.

[34] A. A. Alhabshy, B. I. Hameed, and K. A. Eldahshan, "An ameliorated multiattack network anomaly detection in distributed big data system-based enhanced stacking multiple binary classifiers," *IEEE Access*, vol. 10, pp. 52724–52743, 2022, doi: 10.1109/ACCESS.2022.3174482.

[35] T. Wisanwanichthan and M. Thammawichai, "A double-layered hybrid approach for network intrusion detection system using combined naive Bayes and SVM," *IEEE Access*, vol. 9, pp. 138432–138450, 2021, doi: 10.1109/ACCESS.2021.3118573.

[36] S. Shitharth, P. R. Kshirsagar, P. Kumar Balachandran, K. H. Alyoubi, and A. O. Khadidos, "An innovative perceptual pigeon galvanized optimization (PPGO) based likelihood Naïve Bayes (LNB) classification approach for network intrusion detection system," *IEEE Access*, vol. 10, pp. 46424–46441, 2022, doi: 10.1109/ACCESS.2022.3171660.

[37] A. Alqahtani, S. Alsubai, A. Binbusayyis, M. Sha, A. Gumaei, and Y.-D. Zhang, "Prediction of urinary tract infection in IoT-fog environment for smart toilets using modified attention-based ANN and machine learning algorithms," *Appl. Sci.*, vol. 13, no. 10, p. 5860, May 2023, doi: 10.3390/app13105860.

[38] A. Bonetti, M. Martínez-Sober, J. C. Torres, J. M. Vega, S. Pellerin, and J. Vila-Francés, "Comparison between machine learning and deep learning approaches for the detection of toxic comments on social networks," *Appl. Sci.*, vol. 13, no. 10, p. 6038, May 2023, doi: 10.3390/app13106038.

[39] S. Fu, Y. Wu, R. Wang, and M. Mao, "A bearing fault diagnosis method based on wavelet denoising and machine learning," *Appl. Sci.*, vol. 13, no. 10, p. 5936, May 2023, doi: 10.3390/app13105936.

[40] W. Wang, X. Du, D. Shan, R. Qin, and N. Wang, "Cloud intrusion detection method based on stacked contractive auto-encoder and support vector machine," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 1634–1646, Jul. 2022, doi: 10.1109/TCC.2020.3001017.

[41] N. H. Vu, Y.-S. Choi, and M. Choi, "DDoS attack detection using K-nearest Neighbor classifier method," in *Proc. IASTED*, Baltimore, MD, USA, 2008, pp. 248–253.

[42] X. Tang, Z. Huang, D. Eyers, S. Mills, and M. Guo, "Scalable multicore k-NN search via subspace clustering for filtering," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, pp. 3449–3460, Dec. 2015, doi: 10.1109/TPDS.2014.2372755.

[43] X. Jia and J. A. Richards, "Fast k-NN classification using the cluster-space approach," *IEEE Geosci. Remote Sens. Lett.*, vol. 2, no. 2, pp. 225–228, Apr. 2005, doi: 10.1109/LGRS.2005.846437.

[44] M. A. Ullah, A. A. Jamal, R. A. Tuhin, and S. Akhter, "Detecting distributed denial of service attack using logistic regression and SVM methods," in *Proc. Int. Conf. Appl. Inf. Technol. Innov.*, Bali, Indonesia, 2019.

[45] K. Kumari and M. Mrunalini, "Detecting denial of service attacks using machine learning algorithms," *J. Big Data*, vol. 9, no. 1, p. 56, Dec. 2022, doi: 10.1186/s40537-022-00616-0.

[46] M. Arshi, M. D. Nasreen, and K. Madhavi, "A survey of DDOS attacks using machine learning techniques," in *Proc. E3S Web Conf.*, 2020, p. 1052, doi: 10.1051/e3sconf/202018401052.

[47] F. Chishti and G. Rathee, "ToN-IoT set: Classification and prediction for DDoS attacks using AdaBoost and RUSBoost," in *Proc. 3rd Int. Conf. Advance Comput. Innov. Technol. Eng. (ICACITE)*, May 2023, pp. 2842–2847, doi: 10.1109/ICACITE57410.2023.10183100.

[48] S. Rachmadi, S. Mandala, and D. Oktaria, "Detection of DoS attack using AdaBoost algorithm on IoT system," in *Proc. Int. Conf. Data Sci. Its Appl. (ICoDSA)*, Bandung, Indonesia, Oct. 2021, pp. 28–33, doi: 10.1109/ICoDSA53588.2021.9617545.

[49] M. Bakro, R. R. Kumar, A. A. Alabrah, Z. Ashraf, S. K. Bisoy, N. Parveen, S. Khawatmi, and A. Abdelsalam, "Efficient intrusion detection system in the cloud using fusion feature selection approaches and an ensemble classifier," *Electronics*, vol. 12, no. 11, p. 2427, May 2023, doi: 10.3390/electronics12112427.

[50] E. Jaw and X. Wang, "Feature selection and ensemble-based intrusion detection system: An efficient and comprehensive approach," *Symmetry*, vol. 13, no. 10, p. 1764, Sep. 2021, doi: 10.3390/sym13101764.

[51] D. Stiawan, A. Heryanto, A. Bardadi, D. P. Rini, I. M. I. Subroto, M. Y. B. Idris, A. H. Abdullah, B. Kerim, and R. Budiarto, "An approach for optimizing ensemble intrusion detection systems," *IEEE Access*, vol. 9, pp. 6930–6947, 2021, doi: 10.1109/ACCESS.2020.3046246.

[52] N. V. Sharma and N. S. Yadav, "An optimal intrusion detection system using recursive feature elimination and ensemble of classifiers," *Microprocessors Microsystems*, vol. 85, Sep. 2021, Art. no. 104293, doi: 10.1016/j.micpro.2021.104293.

[53] M. Gong, "A novel performance measure for machine learning classification," *Int. J. Manag. Inf. Technol.*, vol. 13, no. 1, pp. 11–19, Feb. 2021, doi: 10.5121/ijmit.2021.13101.

[54] V. Bahel, S. Pillai, and M. Malhotra, "A comparative study on various binary classification algorithms and their improved variant for optimal performance," in *Proc. IEEE Region 10 Symp. (TENSYMP)*, Jun. 2020, pp. 495–498, doi: 10.1109/TENSYMP50017.2020.9230877.

[55] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," *PLoS ONE*, vol. 10, no. 3, Mar. 2015, Art. no. e0118432, doi: 10.1371/journal.pone.0118432.

[56] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.

[57] W. Wang and D. Sun, "The improved AdaBoost algorithms for imbalanced data classification," *Inf. Sci.*, vol. 563, pp. 358–374, Jul. 2021, doi: 10.1016/j.ins.2021.03.042.

[58] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[59] Y. Sun, A. K. C. Wong, and M. S. Kamel, "Classification of imbalanced data: A review," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 23, no. 4, pp. 687–719, Jun. 2009, doi: 10.1142/s0218001409007326.

[60] P. Kumar, R. Bhatnagar, K. Gaur, and A. Bhatnagar, "Classification of imbalanced data: Review of methods and applications," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 1099, no. 1, Mar. 2021, Art. no. 012077, doi: 10.1088/1757-899x/1099/1/012077.

[61] A. Al-Ashoor and S. Abdullah, "Examining techniques to solving imbalanced datasets in educational data mining systems," *Int. J. Comput.*, pp. 205–213, Jun. 2022, doi: 10.47839/ijc.21.2.2589.

[62] L. Wang, M. Han, X. Li, N. Zhang, and H. Cheng, "Review of classification methods on unbalanced data sets," *IEEE Access*, vol. 9, pp. 64606–64628, 2021, doi: 10.1109/ACCESS.2021.3074243.

ALEKSA N. MAKSIMOVIĆ was born in Belgrade, Serbia, in 1995. He received the B.S. and M.Sc. degrees in informatics and computing from the Department of Informatics and Computing, University of Criminal Investigation and Police Studies, Belgrade, in 2018 and 2020, respectively, where he is currently pursuing the Ph.D. degree.

He is an Assistant Researcher with the Department of Applied Software Engineering and Information Technologies, Faculty of Diplomacy and Security, Belgrade. He has participated in national and COST projects. His research interests include information systems, databases, intelligent decision-making, software engineering, information retrieval, machine learning, and network security.

VOJKAN R. NIKOLIĆ was born in Aleksinac, Serbia, in 1971. He received the B.S. degree in computer technics and informatics from the Faculty of Electronic Engineering, University of Niš, Serbia, in 2001, the M.Sc. degree in industrial informatics from the Technical Faculty Bor, University of Belgrade, Serbia, in 2010, and the Ph.D. degree in information technology from the Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia, in 2016.

He is currently an Associate Professor with the Department of Informatics and Computing, University of Criminal Investigation and Police Studies, Belgrade, Serbia. He has published more than 100 journal articles of which more than ten are from the SCI list, as an author. He has participated in many national and IPA projects and one COST project. His research interests include information systems, databases, intelligent decision-making, software engineering, information retrieval, NLP, and document mining.

Dr. Nikolić is a member of the editorial board for computer sciences in the scientific journal, such as *Journal of Computer and Forensic Sciences*.

DEJAN V. VIDOJEVIĆ was born in Gornji Milanovac, Serbia, in 1972. He received the bachelor's, master's, and Ph.D. degrees from the Faculty of Mechanical Engineering, University of Kragujevac, in 1998, 2004, and 2009, respectively.

He is currently an Assistant Professor with the Department of Informatics and Computing, University of Criminal Investigation and Police Studies, Belgrade, Serbia. He has published more than 20 journal articles of which more than five are from the SCI list, as an author. He has participated in many national and IPA projects. His research interests include applied computing, intelligent decision-making, software engineering, information retrieval, and document mining.

Dr. Vidojevićis a member of the editorial board of computer sciences in the scientific journal, such as *Journal of Computer and Forensic Sciences*.

MILAN D. RANDJELOVIĆ was born in Niš, Serbia, in 1980. He received the B.S. and M.Sc. degrees in organization and economics of production from the Faculty of Agriculture, University of Pristina, Serbia, in 2003 and 2006, respectively, and the Ph.D. degree in economics from the Faculty of Economics, University of Niš, Niš, in 2017.

He is currently a Docent with the Department for Applied Software Engineering and IT, Faculty for Diplomacy and Security, Univerzitet Union Nikola Tesla, Belgrade, Serbia. He is also the Managing Director of the Science and Technology Park Niš, Serbia, working on the regional transition to a knowledge-based economy through the development of regional innovation and start-up ecosystem. He is active in the academic field as an expert with numerous publications in scientific journals and textbooks in the field of economic modeling and decision-making. He has authored more than 30 published papers. He is an experienced professional with 20 years in entrepreneurship, business development, economic development, investment management, FDI, and business infrastructure development projects, such as a direct, result-oriented, problem-solving professional. He is a supporter of modern technologies, digitalization, and knowledge-based society. Under his leadership development agency of Niš, has made tremendous progress in supporting and advancing of business and entrepreneurship climate in Niš. In five years increased number of employed people, companies, and entrepreneurs by 30% and attracted almost EUR €1 billion FDI to the region.

SLAVIŠA M. DJUKANOVIĆ was born in Foča, Bosnia and Herzegovina, in 1976. He received the B.S. degree in section-technical services, specially in electronic systems, from the Military Engineering Academy VSCG, Serbia, in 2000, and the M.Sc. degree in digital transmission of information from the School of Electrical Engineering, University of Belgrade, Serbia, in 2002. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the Faculty of Technical Sciences Čačak, University of Kragujevac.

He is the Assistant Head of the Sector for Analytics, Telecommunications and Information Technology, Ministry of Interior Affairs of the Republic of Serbia. He has participated in many national and IPA projects. His research interests include information systems, databases, intelligent decision-making, and information retrieval.

DRAGAN M. RANDJELOVIĆ was born in Niš, Serbia, in 1953. He received the B.S. degree in electrical engineering and the M.Sc. degree in applied mathematics from the Faculty of Electronic Engineering, University of Niš, Niš, in 1977 and 1984, respectively, and the Ph.D. degree in computer science from the Faculty of Sciences, University of Pristina, Serbia, in 1999.

He is currently a Full Professor and the Chief of the Department for Applied Software Engineering and IT, Faculty for Diplomacy and Security, Univerzitet University Union- Nikola Tesla, Belgrade, Serbia. Since 2014, he has been the Chief of the Department for Informatics and Computing, University of Criminal Investigation and Police Studies, Belgrade, in two mandates. He has published more than 200 journal articles of which more than 20 are from the SCI list, author or editor of more than five monographs, and more than 15 books. He has participated in more than ten national scientific and research projects in Serbia and has led two of them; he also participated in some EU projects. His research interests include parallel computing, intelligent decision-making, cyber security and forensics, information retrieval, numerical mathematics, and statistics.

Dr. Randjelovićis a member of the editorial board of *International Journal of Innovations in Engineering and Technology* and *Computer System Networking and Telecommunications*. He is the Editor-in-Chief of *International Journal of Recent Advances in Information Technology and Management* and *Futuristic Information Technology*.