Journal of Network and Systems Management (2021) 29:20
https://doi.org/10.1007/s10922-021-09591-y

An Intelligent Tree-Based Intrusion Detection Model for Cyber Security

Mohammad Al-Omari[1] · Majdi Rawashdeh[1] · Fadi Qutaishat[1] · Mohammad Alshira'H[2] · Nedal Ababneh[3]

Received: 19 August 2020 / Revised: 26 January 2021 / Accepted: 10 February 2021 /
Published online: 21 February 2021

Abstract
The widespread use of the Internet of Things and distributed heterogeneous devices has shed light on the implementation of efficient and reliable intrusion detection systems. These systems should be able to efficiently protect data and physical devices from cyber-attacks. However, the huge amount of data with different dimensions and security features can affect the detection accuracy and increase the computation complexity of these systems. Lately, Artificial Intelligence has received significant interest and is now being integrated into these systems to intelligently detect and protect against cyber-attacks. This paper aims to propose an intelligent intrusion detection model to predict and detect attacks in cyberspace. The model is designed based on the concept of Decision Trees, taking into consideration the ranking of the security features. The model is applied to a real dataset for network intrusion detection systems. Moreover, it is validated based on predefined performance evaluation metrics, namely accuracy, precision, recall and Fscore. Meanwhile, the experimental results reveal that our tree-based intrusion detection model can detect and predict cyber-attacks efficiently and reduce the complexity of computation process compared to other traditional machine learning techniques.

Keywords Security features · Decision Trees · Machine learning · Cyber security

1 Introduction

The revolution in communication technologies and the Internet has dramatically changed our daily lives. Also, the advancements in Artificial Intelligence and computing have led to an increasing number of distributed intelligent systems. The scale of networks becomes larger and the network environment becomes more complex day by day [1]. As a result, the amount and categories of data

✉ Mohammad Al-Omari
m.alomari@psut.edu.jo

Extended author information available on the last page of the article

flow in networks are constantly expanding. Users now deal with a huge amount of data that is transferred in cyberspace because of the spread of the Internet of Things (IoT) and cloud services with distributed heterogeneous devices [2]. However, protecting these devices from attacks is extremely important in order to protect users' data and their physical devices, and obtain the most benefit from these cloud services. In this context, cyber security is crucial in order to make cloud services efficient and successful. Firewalls and traditional techniques such as encryption and user authentication are unable to protect devices in cyberspace due to the rapid development of new intrusion techniques [3, 4]. Intrusion Detection Systems (IDS) are security systems that are able to detect and prevent attacks in a particular network environment, such as Denial of Service (DoS) attacks, phishing, malware etc. Moreover, these systems should be able to intelligently identify and classify any abnormal behaviors within a network.

Currently, the demand for intelligent intrusion detection approaches using Machine Learning (ML) techniques is significantly increasing. ML can play a vital role in building IDS that are able to classify and predict attacks in cyberspace. Traditional ML-based methods such as k-nearest neighbor (k-NN) algorithms, Support Vector Machines (SVM), Logistic Regression (LR), NaiveBayes (NB) Models and Decision Trees have a significant role in detecting anomalies and attacks in cyber security [5–7]. Among the machine learning techniques, Decision Trees are one of the most popular predictive models that can be used in building intrusion detection systems based on classification algorithms that fall under supervised learning. These predictive Models are commonly used supervised ML algorithms that can be used for classification [8]. In a tree-based classification model, a model or classifier is constructed to predict the categorical class. For instance, a tree-based classification model can predict whether a particular network activity is "normal" or an "attack". Decisions are made at each node of the tree until the leaf node is reached. The class of the data point (i.e. normal or attack) is determined in the leaf node. In other words, the tree node represents a feature, each edge or branch represents a decision made depending on the information gained for each feature, and each leaf represents a class [8]. Nevertheless, the massive volume of network traffic data and the large number of dimensions or features (i.e. security features) can affect the accuracy of prediction. In addition, they can increase the complexity of computation of the tree-based predictive model (i.e. overfitting and processing time). The need for reliable and efficient intrusion detection systems has become a significant requirement to make cloud services successful and beneficial. The design of an IDS that performs with maximum accuracy with minimum false predictions is a challenging task [9]. In addition, since most AI techniques require learning from big data sets and reasoning using a multitude of classification patterns, it is necessary to create new simplified and collaborative solutions [10]. In this paper, an intrusion detection approach based on the concept of decision trees is proposed. Our approach involves considering the ranking of security features before building the predictive model. The model aims to increase the prediction accuracy and reduce the complexity of computation compared with other traditional ML techniques. The main contributions of this research are summarized as follows:

1. An intrusion detection model is developed based on the concept of Decision Trees to efficiently predict and detect attacks in cyberspace.
2. An approach to security features selection and ranking is developed in order to select the security features with the most importance, which should be processed by the proposed tree-based predictive model.
3. The proposed model is applied to a real dataset with 175,341 records for network intrusion detection systems to evaluate it based on predefined performance evaluation metrics, namely accuracy, precision, recall and Fscore, compared with other traditional ML techniques.

The remainder of this paper is structured as follows. Section 2 investigates related work on intrusion detection models. Section 3 presents and discusses our tree-based intrusion detection model taking into consideration the ranking of security features. Section 4 presents our experiments and evaluation of the proposed model. The last section concludes the paper and highlights our future work.

2 Related Work

Cyber security has attracted the interest of many researchers due to the increasing demand for reliable and efficient intrusion detection methods. Intrusion Detection Systems (IDS) are developed to detect abnormal activities and attacks in cyberspace. Machine Learning and its applications can play a significant role in building intelligent and efficient IDS. Much research work has focused on implementing a variety of ML techniques in building IDS while seeking efficiency and effectiveness [11–14]. The work presented in [15–23] introduced different intrusion detection methods using Deep Learning, Decision Trees and other techniques. Aloqaily et al. [24] introduced an intrusion detection system against security attacks for connected vehicles in smart cities. The system is based on deep learning and decision trees mechanisms. Currently, the Tree-based technique is one of the common Machine Learning techniques and predictive models that is used by researchers for building IDS to predict and detect attacks in communication networks [12]. In the literature, there are a considerable number of studies that propose tree-based intrusion detection models taking into consideration the ranking and selection of security features. This process can enhance the prediction accuracy and minimize the complexity of computation [25–27]. Ingre et al. [28] proposed a decision tree based intrusion system using the Feature Correlation Selection (FCR) method in order to increase the prediction accuracy of the model. Moon et al. [29] presented an IDS based on the concept of decision trees using behavior analysis to prevent Advanced Persistent Threats (APT) attacks especially in social media networks. Sarker et al. [30] proposed a behavioral decision tree model, which predicts users' diverse behaviors considering multi-dimensional contexts. A number of research studies have also proposed enhanced prediction algorithms to detect attacks efficiently in a particular network. For example, Puthran and Shah [31] highlighted the poor performance of the ID3 algorithm for Probe, R2L and U2R attacks. Moreover, an Improved

Decision Tree algorithm using Binary Split (IDTBS) and an improved decision tree algorithm is proposed using quad split (IDTQS) to improve the detection rate of Probe, U2R and R2L attacks. Rai et al. [32] developed a decision tree algorithm based on the C4.5 decision tree approach taking into consideration feature selection and split value. A machine-learning-based security model called IntruDtree was proposed, taking into consideration the ranking of the security data featured [33]. This model was developed to increase the prediction accuracy and reduce the complexity of computation (i.e. overfitting, time). Decision Trees can play a significant role in building intrusion detection systems. However, it is vital that such systems have the ability to deal with the huge volume of network traffic data, with many dimensions and security features, so that the detection process is reliable and efficient with high accuracy and reduced complexity of computation. Nevertheless, high variance with regard to over-fitting, high complexity and low prediction accuracy are common limitations of tree-based models when building intrusion detection systems, especially when the predictive model processes many security features with high dimensions.

To this end, the process of evaluating such methods depends on many factors such as the volume of a given dataset, data consistency, the number of security features and the parameters used in the experiments. As a result, it is difficult to conclude that a particular ML technique is better than other techniques unless these factors are considered. However, unlike the proposed models mentioned above, a tree-based intrusion detection model is proposed; it considers the ranking of security features before building the prediction decision tree to overcome the shortcomings of tree-based models mentioned above. Besides, the model is applied to a real dataset with 175,341 records and follows the main steps required in ML, especially at the early stages of building such models. In the following section, our tree-based intrusion detection model is proposed and discussed in detail.

3 Tree-Based Intrusion Detection Model

In this section, the Tree-based Intrusion Detection Model is introduced and discussed in details.

3.1 Model

The proposed intrusion detection model is composed of three main modules. The first module consists of three processes, namely data exploration, data preprocessing and standardization, and features ranking and selection. These processes are crucial in order to build our tree-based intrusion detection approach based on feature ranking and selection. The last two modules are concerned with model training and testing in order to build a classification model that is able to detect attacks in cyberspace. Figure 1 illustrates our proposed model, and each step in the model is discussed in detail in the following sections.

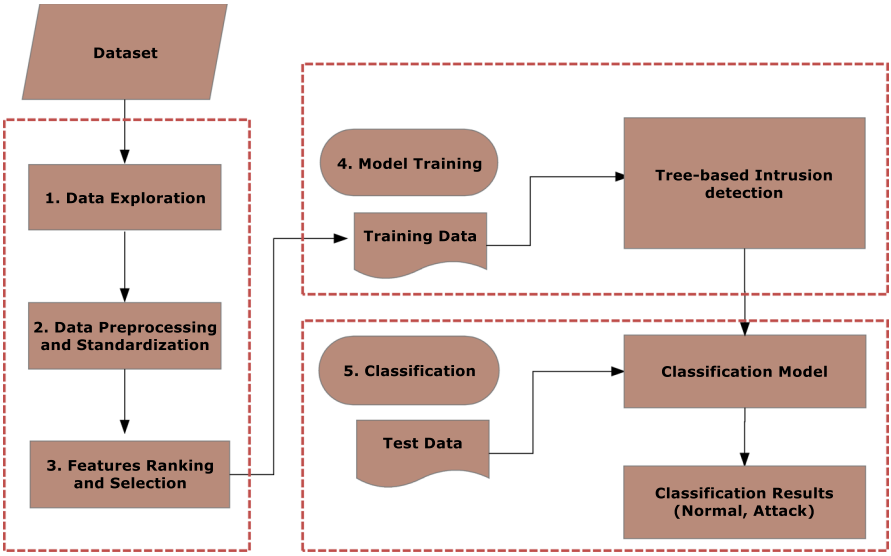

Fig. 1 Tree-based intrusion detection model

3.2 Data Exploration

In Data Mining (DM) and ML techniques, the quality of the data is considered one of the most crucial assets that can radically affect the prediction accuracy of any proposed prediction model. Therefore, the data exploration process in our model examines the data in order to understand its features, identify any integrity issues and apply the data cleansing process. In addition, data types (i.e. feature types) are reviewed in order to determine whether a particular feature is numerical or categorical. This process is important to correctly apply any statistical or prediction measurements and accordingly draw conclusions regarding certain assumptions about the data. In this research, a dataset with 175,341 records for network intrusion detection systems taken from the comprehensive dataset "UNSW-NB 15" is used; this dataset is available on the Kaggle website [34]. The dataset was created in the Cyber Range Lab of the Australian Centre for Cyber Security and consists of 42 features excluding the class label (i.e. 0 for normal records, 1 for attack records). The class feature in the dataset is used to determine whether a particular activity is Normal or an Attack. Moreover, the type of attack, which is one of the dataset features, is excluded from our work, as it is outside the scope of this research. After completing the Data Exploration process, the 42 features are selected for further processing as shown in Table 1.

Table 1 clearly shows that all of the features are quantitative except the *proto*, *service* and *state* features, which are nominal. As a result, these features (i.e. independent variables) must be subjected to Feature Encoding (i.e. Feature Engineering) in order to fit our ML-based intrusion detection model. Feature Encoding transforms nominal values into numerical values. Another aspect that should be taken

Table 1 Security features of the selected dataset

Feature name	Feature type	Feature name	Feature type
Dur	Float	*dtcpb*	Integer
proto	Nominal	*dwin*	Integer
service	Nominal	*tcprtt*	Float
state	Nominal	*synack*	Float
spkts	Integer	*ackdat*	Float
dpkts	Integer	*smean*	Integer
sbytes	Integer	*dmean*	Integer
dbytes	Integer	*trans_depth*	Integer
rate	Float	*response_body_len*	Integer
sttl	Integer	*ct_srv_src*	Integer
dttl	Integer	*ct_state_ttl*	Integer
sload	Float	*ct_dst_ltm*	Integer
dload	Float	*ct_src_dport_ltm*	Integer
sloss	Integer	*ct_dst_sport_ltm*	Integer
dloss	Integer	*ct_dst_src_ltm*	Integer
sinpkt	Float	*is_ftp_login*	Binary
dinpkt	Float	*ct_ftp_cmd*	Integer
sjit	Float	*ct_flw_http_mthd*	Integer
djit	Float	*ct_src_ltm*	Integer
swin	Integer	*ct_srv_dst*	Integer
stcpb	Integer	*is_sm_ips_ports*	Binary

into consideration is Data Standardization. It involves rescaling the distribution of feature values so that the mean of the values is 0 and the standard deviation is 1. This process is important when the features values are in different ranges. In the following section, the feature encoding and standardization are discussed in detail.

3.3 Data Preprocessing and Standardization

This process is considered one of the most vital steps in machine learning. In this process, the Security Feature Encoding and Security Feature Standardization take place as discussed in the following points.

3.3.1 Security Feature Encoding

In the previous section, the nominal security features that must be encoded were identified, namely *proto*, *service*, and *state,* as shown previously in Table 1. Two methods can be used in this context namely, One Hot Encoding and Label Encoding. In this study, Label Encoding is used to encode all of the nominal security features as the One Hot Encoding method can significantly increase the feature dimensions by creating additional features based on the number of unique values in each nominal feature [35]. The Label Encoding method makes all of the feature values

numeric. For example, if the security feature *state* has the values [ACC, CLO, CON, CLO, INT, INT], then these values can be converted to the vector $V = [0,1,2,1,3,3]$. This process is implemented in Python using the *LabelEncoder* method in the *sklearn* class for all of the security features mentioned above.

3.3.2 Security Feature Standardization

The next step is concerned with features that have different value distributions or different scales. This process is considered vital in data preprocessing and it must be completed before the data is processed by our tree-based intrusion model. In the dataset, all features of the data that have a significant difference in data scales are rescaled so that the values for each feature have a zero-mean and unit-variance. The calculation method is shown in formula (1).

$$X_{Scaled} = \frac{X_{original} - \overline{X}}{\sigma}$$

where X_{scaled} denotes the new-scaled value of the feature, $X_{original}$ denotes the original value of the feature, $\overline{X}$ denotes the mean of the feature values and σ is the standard deviation.

The *sklearn* class in Python is used to rescale the values of all of the features that have different value distributions. For instance, the security features *dur, sload, sinpkt and rate* have different value distributions and must be scaled in order to fit in our tree-based intrusion detection model. The density plot is used in order to understand the spread of values for each feature. Figure 2 shows the different density plots for each of the features mentioned above.

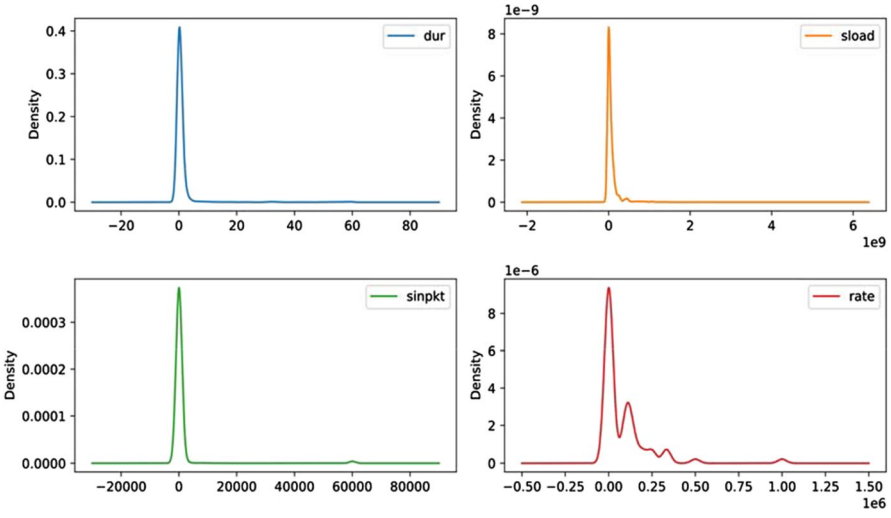

Fig. 2 Different density plots for the *dur*, *sload*, *sinpkt* and *rate* features

As can be seen from the figure above, the density plots for each of these features indicate that they have different distributions. To this end, all of the features in the dataset are scaled (i.e. normalized) and encoded so that the data is ready for the feature ranking and selection process, as discussed in the following section.

3.4 Features Ranking and Selection

In supervised machine learning methods such as decision trees, it is important to choose a suitable method to identify the features that significantly influence the decision making process. There are two common methods in this context, namely Information Gain and the Gini Index. The former implies that the feature with the highest information gain is used as the root to start building a particular decision tree. The latter implies that the feature with a lower Gini index should be chosen for a binary split (i.e. two decisions for each node) [8]. The Gini index (i.e. Gini impurity) is used by Classification and Regression Trees (CART) algorithms and is easy to implement especially for bigger distributions. Therefore, to achieve our goal, a feature ranking approach is proposed using the Gini index method in order to identify the impurity of the features and then rank them based on the Gini impurity (i.e. entropy) before building our decision tree. By achieving this goal, we can then build our tree-based intrusion detection approach with the features that have the lowest Gini index. The Gini index is calculated by deducting the sum of squared of probabilities of each class from one. The more a feature decreases the impurity, the more important the feature is. According to [8, 36], the Gini index for a node n is calculated as shown in formula (2).

$$\boldsymbol{G_I(n)} = 1 - \sum_{i=1}^{c} (\boldsymbol{P_i})^2 \tag{2}$$

where P_i denotes the probability of a tuple in n belonging to a distinct security class. The Gini index is calculated for all features in the dataset in Python and the feature importance scores are ordered for the features as shown in Fig. 3.

In this research, a threshold value of 0.02 (i.e. $t = 0.02$) is chosen to select the most important features that should be processed in the proposed tree-based model. It is worth mentioning here that this value can be changed depending on the dataset used. Therefore, the number of features is reduced to 19 based on the score for each feature. Figure 4 illustrates these features that will be used to build our tree-based intrusion detection.

To this end, the data is ready to be processed by our proposed tree-based model, taking into consideration 19 features instead of 42 features. This study aims to decrease the computation complexity in building our tree-based intrusion detection model and improve its accuracy in regard to attack prediction, as the selected feature has a significant influence on the decision-making process. In the following section, the building of the tree-based intrusion detection model is outlined.

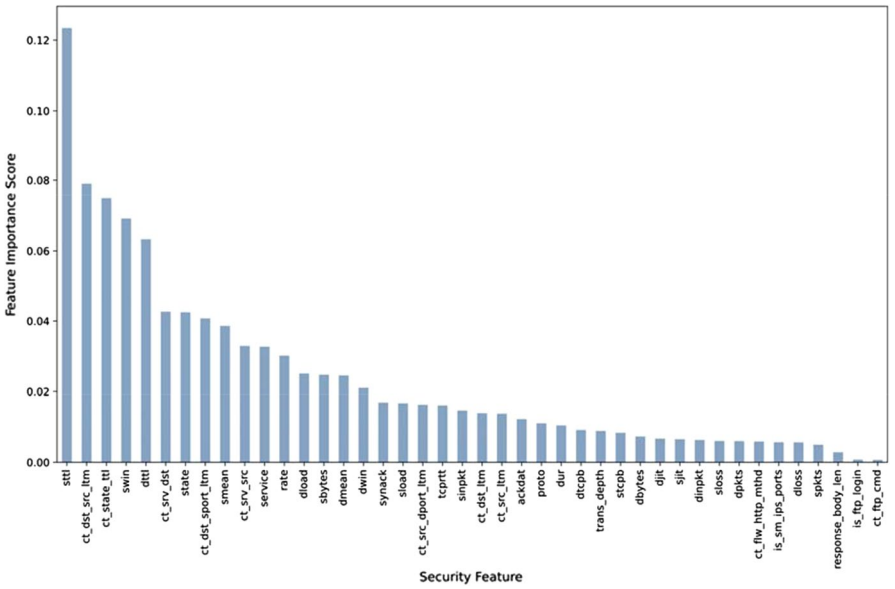

Fig. 3 Security feature importance score

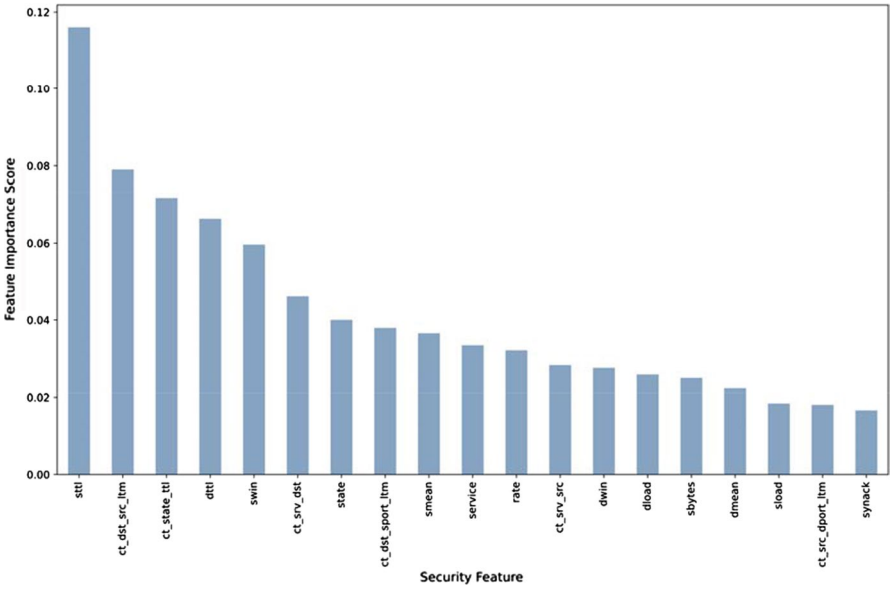

Fig. 4 Selected features based on threshold value of 0.02

3.5 Tree-Based Intrusion Detection

At this level, our tree-based intrusion detection model can be built after all of the previous steps have been completed. Our model is constructed based on reduced dimensions of security features, which can reduce the complexity of the model computation. Besides, it is built using the highest ranked security features that can significantly improve its prediction accuracy. To start our tree-based model, we should identify the root node that will break down the dataset into smaller subsets and then create the branches of the tree. This process is achieved using the Gini Index discussed earlier in this paper. The leaf node is labelled with our target class, which determines whether a particular activity is classified as normal or an attack. The tree-based model is implemented in Python and a sample of our intrusion decision tree is illustrated in Fig. 5.

In Fig. 5 above, a depth value of 3 (i.e. $d=3$) is selected to illustrate part of the intrusion detection tree based on the selected features mentioned earlier. For example, the *sttl* feature was chosen based on the Gini index as the root node and then the branches of the tree were expanded. Each decision node shows the feature name, Gini index, samples, values captured and class name. The class name indicates that a particular activity is Normal or an Attack. To this end, our tree-based intrusion detection model is built and implemented using the selected features. The following

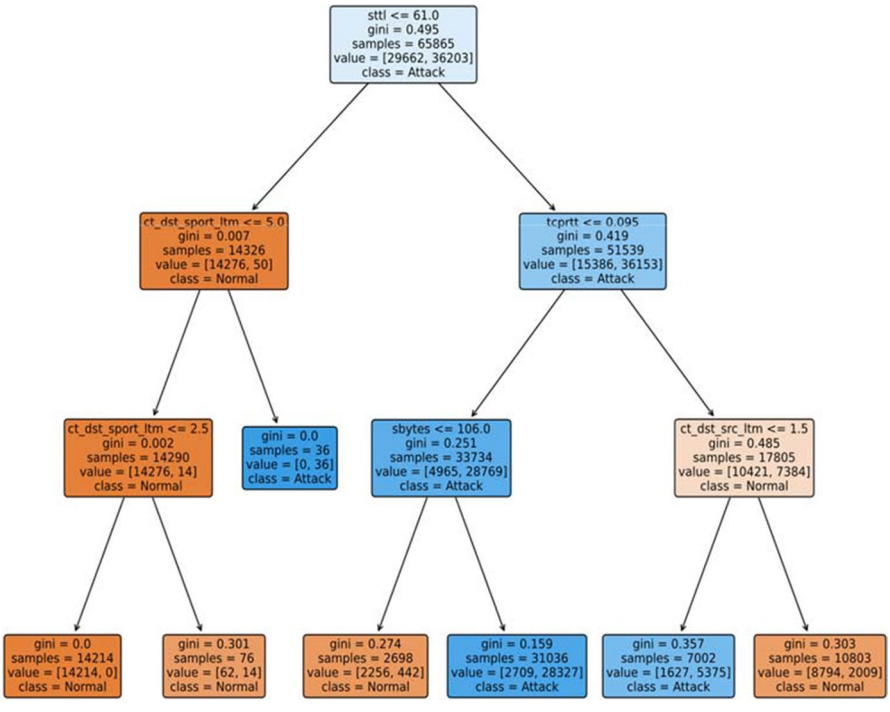

Fig. 5 A snapshot of our Intrusion detection Tree

section summarizes our experiments to evaluate the proposed model and compare the results with other models.

4 Experiments

In this section, the experiments are summarized using the cyber security dataset mentioned earlier in this paper. In addition, the proposed model is evaluated based on Accuracy, Precision, Recall and Fscore and the results are compared with other traditional ML models.

4.1 Evaluation Metrics

The values of Accuracy, Precision, Recall and Fscore are significant metrics for evaluating the efficiency of IDS. These values are calculated based on the following terms [8]:

- *True Positives (TP)* The number of tuples that are truly detected as an intrusion at the end of the detection process.
- *True Negatives (TN)* The number of tuples that are truly detected normally at the end of the detection process.
- *False Positives (FP)* The number of tuples that are safe but are detected as an intrusion at the end of the detection process.
- *False Negatives (FN)* The number of tuples that are harmful but are detected normally at the end of the detection process.

The Accuracy metric is the total number of correct predictions divided by the total number of predictions made for a dataset. It is calculated as shown in formula (3).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

The Precision metric quantifies the number of positive class predictions that actually belong to the positive class. It is calculated as shown in formula (4).

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

The Recall metric quantifies the number of positive class predictions made out of all positive examples in the dataset. It is calculated as shown in formula (5).

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

The Fscore metric provides a single score that balances both the concerns of precision and recall in one number. It is calculated as shown in Formula (6).

$$Fscore = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{6}$$

4.2 Dataset

In this study, a real dataset with 175,341 records taken from the comprehensive dataset "UNSW-NB 15" is used. It was created by the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS). The aim was to generate a hybrid of real modern normal activities and synthetic contemporary attack behaviors [34]. The dataset is publicly available on Kaggle website [34]. It is stored in a ".csv" file that can be processed in Jupyter Notebook using Python.

As stated earlier in this research, the dataset consists of 42 features excluding the class label (see Table 1). The class label is used to determine whether a particular activity is Normal or an Attack. Besides, the type of attack, which is one of the dataset features, is excluded from this study, as it is outside the scope of this research.

4.3 Experiment Design

The first step in our experiments was to complete the processes discussed in Sect. 3. Then, we split the dataset into two sets, namely the training and test sets. The training set comprised 80% of the total records (i.e. randomly selected) in the dataset. It was used to train our proposed model. On the other hand, the test set comprised 20% of the total number of records. It was used to test and validate the proposed model. Two experiments were conducted. The first experiment involved applying our proposed model to the selected dataset taking into consideration the ranking of security features discussed in Sect. 3. The second experiment involved applying traditional ML models such as the k-nearest neighbor (k-NN) algorithm, Support Vector Machines (SVM), Logistic Regression (LR) and NaiveBayes (NB) as common baseline methods. All experiments are implemented in Python using a personal computer with 1.8 GHz processor speed and 4 GB RAM. Table 2 outlines the implementation environment of experiments.

Table 2 Implementation environment

Environment	Description
Personal Computer	Windows 10, 1.8 GHz processor speed and 4 GB RAM
Implementation Platform	Jupyter Notebook is used for developing and implementing the proposed model. Jupyter is a powerful interactive computational environment used for AI applications and development
Programming Language	Python version 3.8 using ML libraries
Database Management System	MS Excel 2016 is used to store the experiments' dataset

Table 3 Results of experiment 1

Class	Accuracy	Precision	Recall	Fscore
Normal	96.7%	96%	96%	96%
Attack	96.7%	97%	97%	97%

Table 4 Full classification report of experiment 1

Classification report				
	Precision	Recall	FScore	Support
Normal	0.96	0.96	0.96	7338
Attack	0.97	0.97	0.97	9129
Macro avg.	0.97	0.97	0.97	16,467
Weighted avg.	0.97	0.97	0.97	16,467
Accuracy: 0.9672				

5 Results and Evaluation

5.1 Experiment 1

In the first experiment, the proposed model is applied to the selected dataset and the intrusion decision tree is built based on the ranking of the selected security features. As mentioned earlier in this research, the performance evaluation metrics were used, namely accuracy, precision, recall and Fscore to validate our proposed model. The accuracy metric is one of the most popular performance metrics that can be used in classification algorithms; it can be simply defined as the percentage of correct predictions. Table 3 shows the results with respect to each of these metrics.

In Table 3 above, the metrics for each class are presented. As stated earlier in Sect. 4.1, the Accuracy metric is the percentage of test samples that are correctly classified by the model. The Precision metric is the ratio of true positives to the total of the true positives and false positives. The Recall metric quantifies the number of positive class predictions made out of all positive examples in the dataset. The Fscore metric provides a single score that balances both precision and recall values. The number of samples of the true response that lie in each class (i.e. Normal or Attack) can be presented in the full classification report, as shown in Table 4.

In this context, another performance metric that can be used to evaluate our proposed model, is the Receiver Operating Curve (ROC). It provides an indication of the capability of our predictive model in regard to distinguishing the security classes. It is created by plotting the True Positive Rate (TPR) (i.e. same value of Recall) versus the False Positive Rate (FPR). The FPR is the total number of false positives divided by the number of false positives and the number of true negatives. The higher Area under the curve (AUC), the better the predictive model. Figure 6 shows the ROC curve of our proposed model with AUC = 0.97.

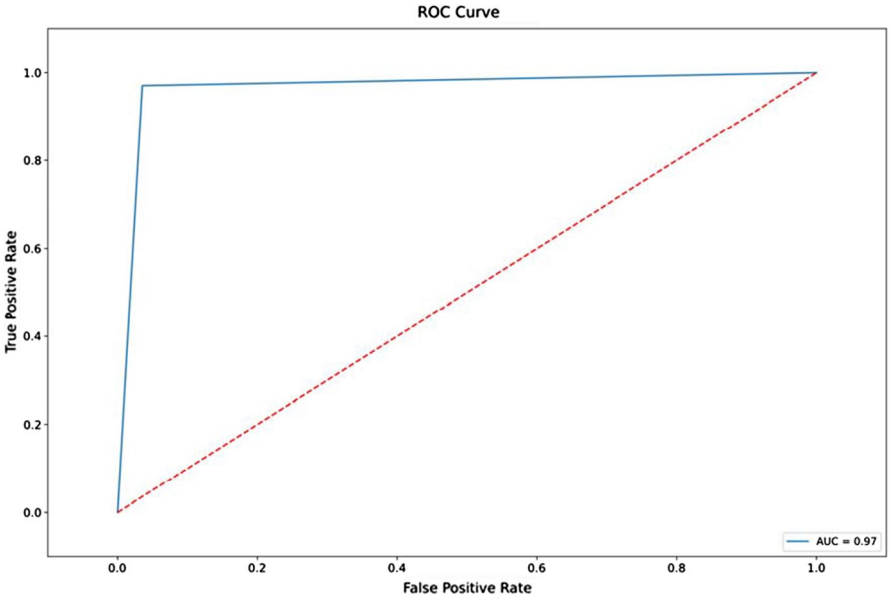

Fig. 6 The ROC curve of our proposed model

5.2 Experiment 2

In this experiment, the traditional ML models mentioned earlier in this research are used and applied to the same dataset. To achieve our goal in this research, however, our approach to ranking each security feature was excluded from this experiment in order to evaluate our proposed model. Other steps such as data encoding and scaling remained, as in the first experiment. First, the k-nearest neighbor (k-NN) algorithm, Support Vector Machines (SVM), Logistic Regression (LR) and NaiveBayes (NB) models were used. Figure 7 shows a summary of the results of experiment 2 for each of these methods compared with the proposed model.

The results in Fig. 7 above show the performance metrics for each of the baseline methods compared with our proposed model. The models were equally applied to the selected dataset in the same environment. The proposed model provides better performance than the other models. In addition, our approach of selecting the highly ranked security features reduced the complexity of computation in terms of time processing and over-fitting (i.e. reduced number of security features).

6 Conclusion and Future Work

In this paper, we present an intelligent tree-based model that is capable of efficiently and effectively predicting and detecting attacks in cyberspace. Within the model, the main steps in machine learning are followed such as data rescaling and encoding.

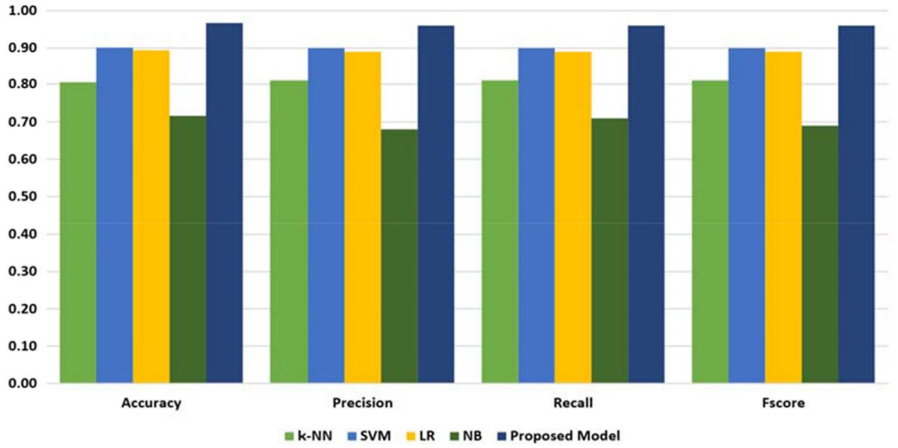

Fig. 7 Results of experiment 2 for the selected baseline traditional methods

Moreover, an approach was developed to select the security features that should be processed based on the ranking of each security feature before building our tree-based intrusion model. The Gini Index was used to measure the impurity of the security features. Specifically, for efficient and accurate results, the highly ranked features were used to train and test the proposed model instead of using all of the security features. In our experiments, we presented the efficiency and effectiveness of our model compared with other popular ML methods.

Meanwhile, our future work will involve working out how to predict what types of attacks will occur in cyberspace using our model and assessing its effectiveness with more dimensions of security features. Besides, for the features selection and ranking process, we intend to apply combined methods such as feature filtering and wrapping into our model in order to improve its performance.

Funding Not applicable.

Data Availability The dataset used in this research is publicly available on the Kaggle website.

Code Availability All experiments in this research were implemented in Jupyter Notebook, Python using predefined machine learning packages and libraries, namely *sklearn* and *matplotlib*.

Compliance with Ethical Standards

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

References

1. Otoum, S., Kantarci, B., Mouftah, H.: A Comparative Study of AI-based Intrusion Detection Techniques in Critical Infrastructures. arxiv.org. (2020)

2. Hesselman, C., Grosso, P., Holz, R., Kuipers, F., Xue, J.H., Jonker, M., de Ruiter, J., Sperotto, A., van Rijswijk-Deij, R., Moura, G.C.M., Pras, A., de Laat, C.: A responsible internet to increase trust in the digital world. J. Netw. Syst. Manag. **28**, 882–922 (2020). https://doi.org/10.1007/s10922-020-09564-7
3. Tavallaee, M., Stakhanova, N., Ghorbani, A.A.: Toward credible evaluation of anomaly-based intrusion-detection methods. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. **40**, 516–524 (2010). https://doi.org/10.1109/TSMCC.2010.2048428
4. Tapiador, J.E., Orfila, A., Ribagorda, A., Ramos, B.: Key-recovery attacks on KIDS, a keyed anomaly detection system. IEEE Trans. Dependable Secur. Comput. **12**, 312–325 (2015). https://doi.org/10.1109/TDSC.2013.39
5. Buczak, A.L., Guven, E.: A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun. Surv. Tutorials **18**, 1153–1176 (2016). https://doi.org/10.1109/COMST.2015.2494502
6. Mishra, P., Varadharajan, V., Tupakula, U., Pilli, E.S.: A detailed investigation and analysis of using machine learning techniques for intrusion detection. IEEE Commun. Surv. Tutorials **21**, 686–728 (2019). https://doi.org/10.1109/COMST.2018.2847722
7. Nisioti, A., Mylonas, A., Yoo, P.D., Katos, V.: From intrusion detection to attacker attribution: a comprehensive survey of unsupervised methods. IEEE Commun. Surv. Tutorials **20**, 3369–3388 (2018). https://doi.org/10.1109/COMST.2018.2854724
8. Thomas, T., Vijayaraghavan, A.P., Emmanuel, S.: Machine Learning Approaches in Cyber Security Analytics. Springer, Singapore (2019)
9. Otoum, S., Kantarci, B., Mouftah, H.T.: A novel ensemble method for advanced intrusion detection in wireless sensor networks. In: IEEE International Conference on Communications. Institute of Electrical and Electronics Engineers Inc. (2020)
10. Al Ridhawi, I., Otoum, S., Aloqaily, M., Boukerche, A.: Generalizing AI: challenges and opportunities for plug and play AI solutions. IEEE Netw. (2020). https://doi.org/10.1109/MNET.011.2000371
11. Ferrag, M.A., Maglaras, L., Moschoyiannis, S., Janicke, H.: Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study. J. Inf. Secur. Appl. **50**, 102419 (2020). https://doi.org/10.1016/j.jisa.2019.102419
12. Gumusbas, D., Yldrm, T., Genovese, A., Scotti, F.: A comprehensive survey of databases and deep learning methods for cybersecurity and intrusion detection systems. IEEE Syst. J. (2020). https://doi.org/10.1109/jsyst.2020.2992966
13. Shapoorifard, H., Shamsinejad, P.: Intrusion detection using a novel hybrid method incorporating an improved KNN. Int. J. Comput. Appl. **173**, 5–9 (2017). https://doi.org/10.5120/ijca2017914340
14. Ji, S.Y., Choi, S., Jeong, D.H.: Designing an internet traffic predictive model by applying a signal processing method. J. Netw. Syst. Manag. **23**, 998–1015 (2015). https://doi.org/10.1007/s10922-014-9335-3
15. Ambusaidi, M.A., He, X., Nanda, P., Tan, Z.: Building an intrusion detection system using a filter-based feature selection algorithm. IEEE Trans. Comput. **65**, 2986–2998 (2016). https://doi.org/10.1109/TC.2016.2519914
16. Amiri, F., Rezaei Yousefi, M., Lucas, C., Shakery, A., Yazdani, N.: Mutual information-based feature selection for intrusion detection systems. J. Netw. Comput. Appl. **34**, 1184–1199 (2011). https://doi.org/10.1016/j.jnca.2011.01.002
17. Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., Gao, M., Hou, H., Wang, C.: Machine learning and deep learning methods for cybersecurity. IEEE Access **6**, 35365–35381 (2018). https://doi.org/10.1109/ACCESS.2018.2836950
18. Mahdavifar, S., Ghorbani, A.A.: Application of deep learning to cybersecurity: a survey. Neurocomputing **347**, 149–176 (2019). https://doi.org/10.1016/j.neucom.2019.02.056
19. Sultana, N., Chilamkurti, N., Peng, W., Alhadad, R.: Survey on SDN based network intrusion detection system using machine learning approaches. Peer-to-Peer Netw. Appl. **12**, 493–501 (2019). https://doi.org/10.1007/s12083-017-0630-0
20. Kang, M.-J., Kang, J.-W.: Intrusion detection system using deep neural network for in-vehicle network security. PLoS One **11**, e0155781 (2016). https://doi.org/10.1371/journal.pone.0155781
21. Feng, F., Liu, X., Yong, B., Zhou, R., Zhou, Q.: Anomaly detection in ad-hoc networks based on deep learning model: a plug and play device. Ad Hoc Netw. **84**, 82–89 (2019). https://doi.org/10.1016/j.adhoc.2018.09.014

22. Zhao, G., Zhang, C., Zheng, L.: Intrusion detection using deep belief network and probabilistic neural network. In: Proceedings—2017 IEEE International Conference on Computational Science and Engineering and IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, CSE and EUC 2017, pp. 639–642. Institute of Electrical and Electronics Engineers Inc. (2017)
23. Mohammadi, S., Mirvaziri, H., Ghazizadeh-Ahsaee, M., Karimipour, H.: Cyber intrusion detection by combined feature selection algorithm. J. Inf. Secur. Appl. **44**, 80–88 (2019). https://doi.org/10.1016/j.jisa.2018.11.007
24. Aloqaily, M., Otoum, S., Al Ridhawi, I., Jararweh, Y.: An intrusion detection system for connected vehicles in smart cities. Ad Hoc Netw. **90**, 101842 (2019). https://doi.org/10.1016/j.adhoc.2019.02.001
25. Peng, Y., Wu, Z., Jiang, J.: A novel feature selection approach for biomedical data classification. J. Biomed. Inform. **43**, 15–23 (2010). https://doi.org/10.1016/j.jbi.2009.07.008
26. Kang, S.H., Kim, K.J.: A feature selection approach to find optimal feature subsets for the network intrusion detection system. Clust. Comput. **19**, 325–333 (2016). https://doi.org/10.1007/s10586-015-0527-8
27. Eesa, A.S., Orman, Z., Brifcani, A.M.A.: A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems. Expert Syst. Appl. **42**, 2670–2679 (2015). https://doi.org/10.1016/j.eswa.2014.11.009
28. Ingre, B., Yadav, A., Soni, A.K.: Decision tree based intrusion detection system for NSL-KDD dataset. In: Satapathy S., Joshi A. (eds.) Information and Communication Technology for Intelligent Systems (ICTIS 2017) - Vol. 2, ICTIS 2017. Smart Innovation, Systems and Technologies, pp. 207–218. Springer Science and Business Media Deutschland GmbH (2018)
29. Moon, D., Im, H., Kim, I., Park, J.H.: DTB-IDS: an intrusion detection system based on decision tree using behavior analysis for preventing APT attacks. J. Supercomput. **73**, 2881–2895 (2017). https://doi.org/10.1007/s11227-015-1604-8
30. Sarker, I.H., Colman, A., Han, J., Khan, A.I., Abushark, Y.B., Salah, K.: BehavDT: a behavioral decision tree learning to build user-centric context-aware predictive model. Mob. Netw. Appl. **25**, 1151–1161 (2020). https://doi.org/10.1007/s11036-019-01443-z
31. Puthran, S., Shah, K.: Intrusion detection using improved decision tree algorithm with binary and quad split. In: Mueller P., Thampi S., Alam Bhuiyan M., Ko R., Doss R., Alcaraz Calero J. (eds.) Security in Computing and Communications, pp. 427–438. Springer (2016)
32. Rai, K., Syamala Devi, M., Guleria, A.: Decision tree based algorithm for intrusion detection. Int. J. Adv. Netw. Appl. **7**, 2828–2834 (2016)
33. Sarker, I.H., Abushark, Y.B., Alsolami, F., Khan, A.I.: IntruDTree: a machine learning based cyber security intrusion detection model. Symmetry (Basel) **12**, 754 (2020). https://doi.org/10.3390/SYM12050754
34. Kaggle, https://www.kaggle.com (2020). Accessed 24 July 2020
35. Zheng, A., Casari, A.: Feature Engineering for Machine Learning. O'Reilly Media, Sebastopol (2018)
36. Han, J., Kamber, M., Pei, J.: Data mining: Concepts and Techniques. Elsevier, Amsterdam (2012)

Mohammad Al-Omari is an assistant professor in the Department of Business Information Technology, Princess Sumaya University for Technology, Jordan. He received his PhD in Computer Science from De Montfort University, UK. His research interests include, but are not limited to, Machine Learning, Cyber Security, Software Engineering, Data Mining, Big Data, and Cloud Computing.

Majdi Rawashdeh received his Ph.D. degree in Computer Science from the University of Ottawa, Canada. He is currently an associate professor at Princess Sumaya University for Technology, Jordan. His research interests include social media, recommender systems, smart cities, and big data.

Fadi Qutaishat is an associate professor teaching various subjects in the Department of Business Information Technology, Princess Sumaya University for Technology, Jordan. He received his PhD in Information Systems from Loughborough University, UK. His research interests include, but are not limited to, Enterprise Systems, Knowledge Management, Systems Analysis and Design, Database Management Systems.

Mohammad Alshira'H is an assistant professor in the Department of Information Systems, Al al-Bayt University, Jordan. He obtained his PhD in Software Engineering from the University of Leicester, UK.

Nedal Ababneh is an assistant professor in the Department of Information Security Engineering Technology at Abu Dhabi Polytechnic. He received his PhD from The University of Sydney, Australia. His main research interests include Internet of Things, Wireless Sensor Networks, Blockchain, Network and Information Security, and Steganography.

Authors and Affiliations

Mohammad Al-Omari[1] · Majdi Rawashdeh[1] · Fadi Qutaishat[1] · Mohammad Alshira'H[2] · Nedal Ababneh[3]

Majdi Rawashdeh
m.rawashdeh@psut.edu.jo

Fadi Qutaishat
f.qutaishat@psut.edu.jo

Mohammad Alshira'H
alshirah@aabu.edu.jo

Nedal Ababneh
nedal.ababneh@adpoly.ac.ae

[1] Department of Business Information Technology, Princess Sumaya University for Technology, Amman, Jordan

[2] Department of Information Systems, Al al-Bayt University, Al-Mafraq, Jordan

[3] Department of Information Security Engineering Technology (ISET), Abu Dhabi Polytechnic, Abu Dhabi, UAE