Article

IntruDTree: A Machine Learning-Based Cyber Security Intrusion Detection Model

Iqbal H. Sarker [1,2,*], Yoosef B. Abushark [3], Fawaz Alsolami [3] and Asif Irshad Khan [3]

[1] Department of Computer Science and Software Engineering, Swinburne University of Technology, Melbourne, VIC-3122, Australia.

[2] Department of Computer Science and Engineering, Chittagong University of Engineering and Technology, Chittagong-4349, Bangladesh.

[3] Computer Science Department, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah-21589, Saudi Arabia; yabusharkh@kau.edu.sa (Y.B.A.); falsolami1@kau.edu.sa (F.A.); aikhan@kau.edu.sa (A.I.K.)

* Correspondence: msarker@swin.edu.au (I.H.S.)

Abstract: Cyber security has recently received enormous attention in today's security concerns, due to the popularity of the Internet-of-Things (IoT), the tremendous growth of computer networks, and the huge number of relevant applications. Thus, detecting various cyber-attacks or anomalies in a network and building an effective *intrusion detection* system that performs an essential role in today's security is becoming more important. Artificial intelligence, particularly *machine learning* techniques, can be used for building such a data-driven intelligent intrusion detection system. In order to achieve this goal, in this paper, we present an *Intrusion Detection Tree ("IntruDTree")* machine-learning-based security model that first takes into account the *ranking of security features* according to their importance and then build a tree-based generalized intrusion detection model based on the selected important features. This model is not only effective in terms of prediction accuracy for unseen test cases but also minimizes the *computational complexity* of the model by reducing the feature dimensions. Finally, the effectiveness of our IntruDTree model was examined by conducting experiments on cybersecurity datasets and computing the precision, recall, fscore, accuracy, and ROC values to evaluate. We also compare the outcome results of IntruDTree model with several traditional popular machine learning methods such as the naive Bayes classifier, logistic regression, support vector machines, and k-nearest neighbor, to analyze the effectiveness of the resulting security model.

Keywords: cybersecurity; cyber-attacks; anomaly detection; intrusion detection system; machine learning; network behavior analysis; cyber decision making; cybersecurity analytics; cyber threat intelligence.

1. Introduction

In recent days, the demand for cyber security and protection against various types of *cyber-attacks* has been ever increasing. The main reason is the popularity of Internet-of-Things (IoT), the tremendous growth of computer networks, and the huge number of relevant applications that are used by individuals or groups for the purpose of either personal or commercial use. Cyber attacks such as the denial-of-service (DoS) attack [1], computer malware [2], or unauthorized access [1] led to irreparable damage and financial losses in large-scale networks. For instance, according to [3], in May 2017, one ransomware virus brought huge losses to many organizations and industries, including finance, medical care, energy, and universities as well, causing a loss of 8 billion dollars. Other statistics have estimated that on average, a data breach costs an affected organization 3.9 million USD, and 8.19

million USD to the United States [4]. Thus, the demand for cyber security and protection against various types of cyber-attacks is increasing day-by-day according to today's needs in the cyber world.

A cyber security system typically consists of a network security system and a computer security system. Although various systems, such as firewall and encryption, are designed to handle Internet-based cyber-attacks, an intrusion detection system (IDS) is more capable of resisting the computer network from external attacks [5]. Thus, the main purpose of an IDS is to detect various types of malicious network communications and computer systems usage for prevention. The conventional solutions, e.g., firewalls, are unable to perform the tasks well [6–8]. An IDS conducts the process of identifying malicious cyber-attack behavior on a network while monitoring and evaluating the daily activities in a network or computer system to detect security risks or threats such as denial-of-service (DoS). An intrusion detection system also helps to discover, determine and identify unauthorized system behavior such as unauthorized access, or modification and destruction [9–11]. Therefore, detecting various types of cyber-attacks and anomalies in a network and to build an effective IDS that performs an essential role in today's network security is needed to facilitate a system's security.

Intrusion detection systems could be different categories according to the usage scope. For instance, the most common types of intrusion detection systems are host-based and network-based, which are in the scope of single computers to large networks [12]. A host-based intrusion detection system (HIDS) relies on an individual system and monitors important operating system files for suspicious or malicious activities, which is very limited to detect unknown malicious code [12]. On the other hand, in a network intrusion detection system (NIDS), the system analyzes and monitors network connections for suspicious traffic. Similarly, based on detection approach, the most well known variants are misuse or signature-based, and anomaly-based detection that have been studied worldwide by the security research community for many years [13]. In a signature-based IDS, a specific pattern is identified as the detection of corresponding attacks. For instance, a byte sequence in a network traffic, or known patterns or sequences used by malware could be considered as a signature. Anti-virus software uses such types of patterns as a signature in order to the detect the attacks by matching. This signature-based intrusion detection system is capable of easily detecting known attacks; however, it is challenging to detect new unseen attacks using such known signature for which no pattern is available [14]. On the other hand, an anomaly-based intrusion detection system examines the behavior of the network and finds patterns, automatically creates a data-driven model for profiling the normal behavior, and thus detects deviations in the case of any anomalies. The main advantage of this anomaly-based IDS over a signature-based IDS is that it can detect attempts to exploit new and unseen vulnerabilities or cyber-threats. However, it may produce high false alarm rates considering previously unseen system behaviors as anomalies [10]. Thus, a *machine learning* based effective detection approach is needed to minimize these issues, in which we are interested in this paper.

Developing computational security models to analyze different cyber incident patterns and eventually predict the threats utilizing cybersecurity data can be used for building a data-driven intelligent IDS. Thus, the knowledge of artificial intelligence, particularly *machine learning* techniques that can learn from a security dataset well, can play a key role in this [14,15]. Among the machine learning techniques, a tree-based method performs well in the area of predictive analytics [16,17]. However, effectively modeling cyber-attacks or anomalies is *challenging* as today's security datasets may contain high dimensions of security features that may include features with less importance or not important at all. Thus, a security model with these features may cause several issues such as *high variance* that usually causes over-fitting in a tree-based model as it learns from only one pathway of decisions, high *computational cost and time* for model training, lack of *generalization* in the model, and consequently, may decrease the *prediction accuracy* for unseen test cases. Therefore, the research question addressing in this paper is - *"RQ: How can we minimize these security issues and build an effective data-driven intrusion detection model in the domain of cybersecurity?"*

In order to answer this research question, in this paper, we present an *Intrusion Detection Tree ("IntruDTree")* machine-learning-based security model that minimizes the above mentioned issues.

In our approach, we first took into account the *ranking of security features* according to their importance in modeling. We then build a tree-based generalized intrusion detection model based on the selected important features. Once the complete tree has been built by utilizing the training security data, the test data is used to validate the model. This approach is not only effective in terms of *prediction accuracy* for unseen test cases by minimizing the over-fitting in modeling but also minimizes the *computational complexity* of the model by reducing the feature dimensions while building the resultant model.

The contributions of this work can be summarized as follows.

- We first highlight the importance of *security features* for high dimensions in a machine learning-based intrusion detection model.
- We then present an intrusion detection tree *"IntruDTree"* machine-learning-based security model that first takes into account the *ranking of security features* according to their importance and then build a tree-based generalized model based on the selected important features.
- Finally, we conduct *experiments* to evaluate the effectiveness of our intrusion detection model IntruDTree. The experimental results show that our IntruDTree model significantly outperforms previous ones for detecting cyber intrusions in various unseen test cases.

The rest of the paper is organized as follows. Section 2 provides the background and related work of intrusion detection techniques. In Section 3, we present our intrusion detection tree IntruDTree machine-learning-based security model by taking into account feature importance. We evaluate the resultant security model and report the experimental results by conducting experiments on cybersecurity dataset in Section 4. Finally, Section 5 concludes this paper and highlights the future work.

2. Background and Related Work

An intrusion detection system is typically used to identify malicious cyber-attack behavior on a network while monitoring and evaluating the daily activities in a network or computer system to detect security risks or threats [9,11]. A number of research has been conducted in the area of cybersecurity with the capability of detecting and preventing cyber attacks or intrusions. Signature-based network intrusion detection is one of the well known systems used in the cyber industry [14]. This system takes into account a known signature and has seen widespread adoption including commercial success in recent time. On the other hand, the anomaly-based approach has advantage over the signature-based approach for detecting unseen attacks, including the ability to identify unknown or zero-day attacks [10,18]. This approach monitors network traffic and finds attack behavioral patterns by analyzing the relevant security data. Various data mining and machine learning techniques are used to analyze such security incident patterns for making useful decisions [5,16,19]. The main drawback of the anomaly-based approach is that it may produce high false alarm rates as it may categorize the previously unseen system behaviors as anomalies [10]. Thus, limiting the false positive rates of an intrusion detection system must be a top priority [13]. Therefore, a machine learning-based effective detection approach is needed to minimize these issues.

The area of machine learning is typically considered as a branch of artificial intelligence, which is closely related to computational statistics, data mining and data science, and mainly focuses on making computers to learn from data [19,20]. It has strong ties to mathematical theories, methods, statistical analysis, optimization, and various real-world application domains in the field. Thus, in the area of cybersecurity, machine learning is a type of data-driven method in which understanding the raw security data is the first step to build an intelligent security model for making predictions about future incidents. Although, the association analysis is popular in machine learning techniques to build rule-based intelligent systems [21–23], in this work, we primarily focus on classification learning techniques [16,24] that are typically used to build a predictive model by utilizing a given training dataset. For instance, to build a data-driven predictive model, several popular techniques such as the probability-based naive Bayes classifier, hyperplane-based support vector machines, instance-learning-based k-nearest neighbors, the sigmoid function based logistic regression technique, as well as rule-based classification like decision trees have been used [16,19].

In the domain of cybersecurity, particularly for detecting intrusions or cyber-attacks, a number of researchers used the machine-learning classification techniques mentioned above. For instance, Li et al. [25] presented an approach to classify the predefined attack categories such as DoS, Probe or Scan, U2R, R2L, as well as normal traffic utilizing the most popular KDD'99 cup dataset by using the hyperplane-based support vector machine classifier with a RBF kernel. In [26], Amiri et al. used a least-squared support vector machine classifier to train the model utilizing large data sets to create a faster system. In [27], Hu et al. used a variation of the support vector machine classifier in their study in order to classify the anomalies. Wagner et al. [28] employed a one-class support vector machine classifier in their study for the purpose of anomaly detection and different types of attacks such as NetBIOS scans, DoS attacks, POP spams, and Secure Shell (SSH) scans. The authors in [29] used this support vector machine classifier for the purpose of detecting unknown computer worms' activity. Similarly, Kotpalliwar et al. [30], Saxena et al. [31], Pervez et al. [32], Li et al. [25], Shon et al. [33], and Kokila et al. [34] used a support vector machine classifier in their studies for the purpose of building intrusion detection systems.

In addition to the support vector machine classifier discussed above, several other classifiers are used to detect intrusions. For instance, Kruegel et al. [35] used a probability-based Bayesian network to classify the events that process TCP/IP packets. A denial-of-service (DoS) intrusion detector using the same Bayesian network has been described by Benferhat et al. in their study [36]. Panda et al. [37] have employed the probability-based naïve Bayes classifier for the purpose of analyzing the KDD'99 cup data set, in which the data consist of four categories of attacks—Probe or Scan, DoS, U2R, and R2L. Koc et al. [38] used the same naive Bayes classifier in order to build a multi-class intrusion detection system. An instance-based learning algorithm, the KNN, is another popular machine learning method in which the classification of a point is determined by the k-nearest neighbors of that data point. Shapoorifard et al. [39], Vishwakarma et al. [40], and Sharifi et al. [41] used KNN classification technique in their studies for the purpose of intrusion detection systems. Several research studies [42,43] have been done using the logistic regression model as well for identifying malicious traffic and intrusions. In addition to these approaches, authors in [44] considered neural classifier, and those in [45] considered wavelet transform for anomaly detection, particularly DoS attacks.

Among the machine learning techniques, a tree-based method known as decision tree is considered as one of the most popular machine learning classification methods for the purpose of building predictive models. The best known methods for automatically building decision trees are the ID3 [46] and C4.5 [47] algorithms. Recently, a behavioral decision tree algorithm called BehavDT for analyzing behavioral patterns was proposed by Sarker et al. [48]. A significant amount of research in the domain of cybersecurity, such as Ingre et al. [49], Malik et al. [50], Relan et al. [51], Rai et al. [52], Puthran et al. [53], Moon et al. [54], Balogun et al. [55], and Sangkatsanee et al. [56] used the decision tree classification approach in their studies for the purpose of building intrusion detection systems. However, with the high dimensions of security features, a decision tree model may cause several issues such as *high variance* with over-fitting, high *computational cost and time*, and low *prediction accuracy*.

Unlike the above approaches, in this work, we present a machine learning-based security model *"IntruDTree"* that first takes into account the *ranking of security features* according to their importance and then build a generalized tree for detecting intrusions based on the selected important features in order to resolve the issues highlighted above.

3. Materials and Methods

In this section, we present our machine learning-based cyber security intrusion detection model IntruDTree. This comprised of several processing steps: exploring the security dataset, preparing raw data, determining feature importance and ranking, and building the resultant tree-like model. In the following section, we briefly discuss these steps one by one in order to achieve our goal.

3.1. Exploring Security Dataset

Security datasets typically represent a collection of information records consisting of several security features and related facts that can be used for the purpose of building a data-driven cybersecurity intrusion detection model [15]. Thus, it is important to understand the nature of raw cybersecurity data and the patterns of security incidents in order to detect malicious behavior or anomalies. In this work, we used an intrusion dataset consisting of two categories of class variables—normal and anomaly—publicly available in Kaggle [57], the world's largest machine learning and data science community. The dataset consists of a total of 41 features, among them, 3 features are qualitative, the *protocol_type*, *flag*, and the other 38 features are quantitative, including *duration*, *logged_in*, *srv_serror_rate*, *dst_host_count*, *dst_bytes*. Table 1 shows all the security features, including their value type. The dataset contains more than twenty five thousand instances that were collected from a wide variety of intrusions simulated in a military network environment. To collect these raw data, including TCP/IP dump data for a network, an environment was created by simulating a typical US Air Force local area network. The network was created like a real cyber environment and blasted with multiple cyber-attacks that are known as anomalies [57].

Table 1. Dataset features with value type.

Feature Name	Value Type	Feature Name	Value Type
dst_host_srv_count	Integer	*same_srv_rate*	Float
flag	Nominal	*dst_host_same_srv_rate*	Float
srv_serror_rate	Float	*dst_host_srv_serror_rate*	Float
dst_host_serror_rate	Float	*count*	Integer
protocol_type	Nominal	*logged_in*	Integer
dst_host_same_src_port_rate	Float	*dst_host_srv_diff_host_rate*	Float
rerror_rate	Float	*src_bytes*	Integer
dst_host_srv_rerror_rate	Float	*service*	Nominal
srv_rerror_rate	Float	*dst_host_rerror_rate*	Float
dst_host_count	Integer	*dst_host_diff_srv_rate*	Float
srv_count	Integer	*wrong_fragment*	Integer
serror_rate	Float	*num_compromised*	Integer
srv_diff_host_rate	Float	*dst_bytes*	Integer
hot	Integer	*diff_srv_rate*	Float
duration	Integer	*is_guest_login*	Integer
root_shell	Integer	*land*	Integer
urgent	Integer	*num_failed_logins*	Integer
su_attempted	Integer	*num_root*	Integer
num_file_creations	Integer	*num_shells*	Integer
num_access_files	Integer	*num_outbound_cmds*	Integer
is_host_login	Integer	-	-

All the security features mentioned in Table 1 are not identical in terms of data distribution, and vary from feature to feature. For instance, Figure 1 and Figure 2 show the data distribution for two different features, *duration* and *dst_bytes*, respectively. In order to build our machine learning-based intrusion detection model, we first prepared the raw dataset with the feature values mentioned above. Effectively processing and ranking these security features according to the requirements, building a target machine-learning based detection model, and eventually, data-driven pattern-based decision making, could play a role in providing intelligent cybersecurity services, particularly detecting anomalies or intrusions.

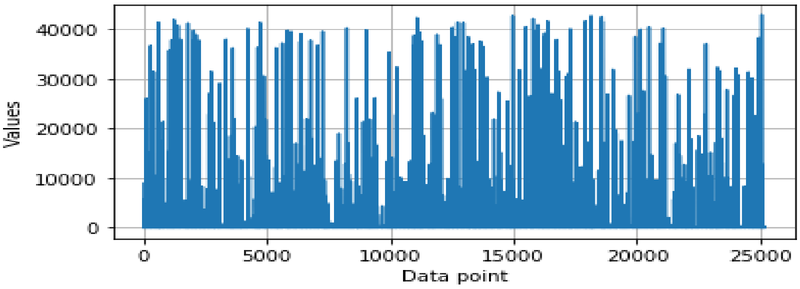

Figure 1. Data distribution of the secuity feature '*duration*'.

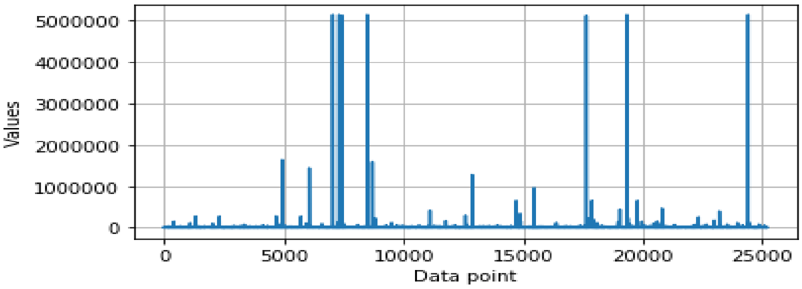

Figure 2. Data distribution of the secuity feature '*dst_bytes*'.

3.2. Preparing Raw Security Data

Data preparation comprises both feature encoding and scaling according to the characteristics of the given intrusion dataset.

- *Feature encoding:* As mentioned earlier, the dataset contains both the numeric and nominal values of the given security features. Although most of the features are numerically valued, several are nominally valued, such as *protocol_type*, *service*, *flag*, shown in Table 1, and the class value [anomaly, normal] as well. Thus, it is needed to convert all the nominal valued features into vectors in order to fit these data to the target machine learning-based intrusion detection model. Although, "One Hot Encoding" is a popular approach, we used "Label Encoding" in this work. The reason is that, in one hot encoding technique, a significant number of feature dimensions increase. On the other hand, the label-encoding approach directly converts the feature values into particular numeric values. Let us consider an example in terms of the feature *protocol_type*. Label encoding can turn the values [tcp, udp, icmp, udp, icmp] into vectors [0, 1, 2, 1, 2].
- *Feature scaling:* In data pre-processing, feature scaling is also known as data normalization. The values of the security features are in different ranges that vary from feature to feature. For instance, Figure 1 and Figure 2 show the data distributions of two different features, *duration* and *dst_bytes*, respectively. For some data points, the value is very low while for some data points, it is much higher, as shown in Figures 1 and 2. Thus, data scaling method is used to normalize the range of the feature values, known as the independent variables as well. In order to do this, we used a Standard Scaler that normalizes the security features with the mean value = 0 and standard deviation = 1. The normalized values are then ready for further analysis in order to build the security model.

3.3. Determining Feature Importance and Ranking

In our approach, once the data exploration and preparation have been done, we calculate the importance score of each security feature discussed above and rank them according to their importance score, in order to select a set of significant features for further processing. Generally, feature importance provides a score that indicates how useful or valuable each feature in a cybersecurity dataset, to build a tree-based intrusion detection model. Feature importance is estimated as the decrease in node impurity weighted by the probability of reaching that node. The node probability is simply calculated by the number of instances that reach a particular node divided by the total number of instances. The higher the value, the more important the feature. The value varies from 0 to 1. A value of 0 means that the output of the model does not depend on the feature at all and 1 means that the output of the model is directly associated with the feature. Thus, the importance of each feature is derived from how "pure" it is.

In statistics and data mining, "Gini Index" is a well known measure of the impurity of a node that typically measures how often a random element would be wrongly detected [19]. Thus, it is the probability of wrongly classifying a randomly chosen element in a security dataset according to the class distribution in the dataset. For a binary split, the Gini Index [19] of a node n can be expressed as -

$$I_G(n) = 1 - \sum_{i=1}^{c} {p_i}^2 \tag{1}$$

$$\Delta I_G(n_p) = I_G(n_p) - p_l I_G(n_l) - p_r I_G(n_r) \tag{2}$$

where p_i represents the probability of an element being classified to a particular security class and p_l and p_r are the proportions of examples in node n_p that are assigned to child nodes n_l and n_r, respectively. Therefore, how much each feature decreases the impurity is then computed using the Gini impurity formulas defined above. The more a feature decreases the impurity, the more important the feature is. We then rank the security features according to their calculated importance score. Thus, after ranking the features, our approach is capable of selecting the top n security features according to their importance score values for the purpose of building an effective tree-based security model by taking into account the selected n features. This thus enables us to identify correlations and patterns in a security data set to transform them into significantly lower dimension datasets without loss of any significant information for modeling.

3.4. Designing Intrusion Detection Tree

Once the security features are ready to process, we design a tree-like model in order to make decisions in a data-driven intelligent intrusion detection system. In our model, we take into account the selected security features that are determined according to their importance score and ranking, rather than using all the features available in the given dataset. To design a tree-like model, we start from a root node. It breaks down a given training dataset into smaller subsets and an associated branch of the tree is incrementally created. To identify the attribute for the root node in each level, the "Gini Index" [19] defined in the earlier section is used. An attribute with a lower Gini index is selected to incrementally develop the tree. Thus, we expand the tree by creating the required number of branches consisting of internal and leaf nodes and their connecting edges or arcs. Internal nodes are labelled with the security features selected earlier and each leaf node of the tree is labelled with a target class, either anomaly or normal in this work. The arcs of the tree coming from a node are labeled with each of the possible values of the relevant feature. The final result is a multi-level tree with a number of terminal or leaf nodes that includes anomaly or normal, as shown in Figure 3. Overall, our intrusion detection model IntruDTree takes into account mainly two properties, (i) reducing the feature dimensions by determining the feature importance and ranking, and (ii) to construct a multi-level tree by taking into account the selected important features. Figure 3 shows an example of an IntruDtree considering several features like $flag$, $service$, $duration$, $logged\_in$ and their sample values in a given intrusion dataset.

The overall process for building an IntruDTree is set out in Algorithm 1. Given a training intrusion dataset, $DS = \{X_1, X_2, ..., X_m\}$, where m represents the data size. Each instance is represented by n-dimensional features. The training data also belong to diverse cyber-attack classes $CA = \{normal, anomaly\}$. The outcome is an IntruDTree, which is a rule-based classification tree associated with DS. For instance, according to Figure 3, an example rule with single feature could be "if the flag value is $RSTR$, then the outcome is *anomaly*". Similarly, another rule with multiple features could be "if flag value is SF, service is ftb, and duration $<= 4$, then the outcome is *anomaly*". Thus, a number of security rules can be extracted by traversing the generated IntruDTree, that can be used to detect whether the test case is normal or an anomaly.

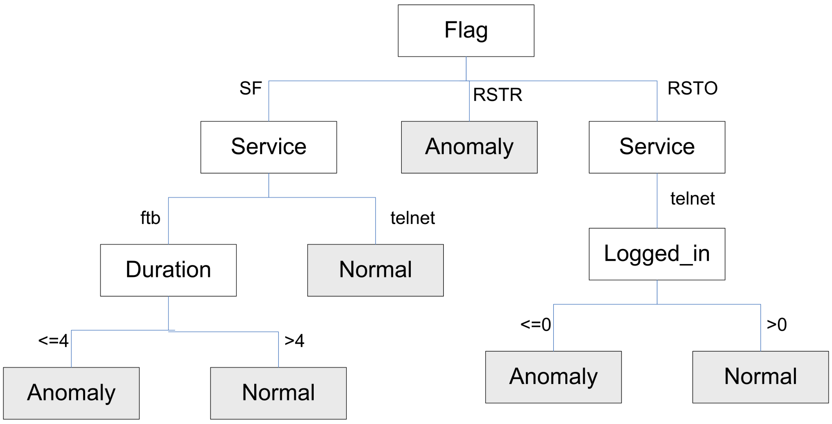

Figure 3. An example of an IntruDTree considering several features.

```
Algorithm 1: IntruDTree Induction
  Data: Dataset: $DS = X_1, X_2, ..., X_m$ // each instance $X_i$ contains a number of features and
        corresponding cyber-attack class $CA$
  Result: An IntruDTree
Procedure IntruDTree ($DS, feature\_list, CAs$);
//calculate feature importance score
$imp_{score} \leftarrow calculateScore(feature\_list)$
//select important features
$imp\_feature\_list \leftarrow selectFeatrues(feature\_list, imp_{score}, n)$
$TreeGen(DS, imp\_feature\_list, CAs)$
$N \leftarrow createNode()$ //create a root node for the tree
if all instances in $DS$ belong to the same class $CA$ then
    return $N$ as a leaf node labeled with the class $CA$.
end
if imp_feature_list is empty then
    return $N$ as a leaf node labeled with the majority class in $DS$; // majority voting
end
identify the highest precedence feature $F_{split}$ for splitting and assign $F_{split}$ to the node $N$.
foreach feature value $val \in F_{split}$ do
    create subset $DS_{sub}$ of $DS$ containing $val$.
    if $DS_{sub} \neq \phi$ then
        attach the node returned by $TreeGen(DS_{sub}, \{imp\_feature\_list - F_{split}\}, CAs))$ to node
          N;
    end
    attach a leaf labeled with the majority class in $DS$ to node $N$;
end
return $N$
```

4. Experimental Results and Discussion

In this section, we briefly analyze and report the experimental results of this work. For this, we first setup our experiments to evaluate our IntruDTree cyber security model, and then discuss the experimental results in various dimensions related to our analysis.

4.1. Experimental Setup

In order to evaluate our IntruDTree cyber security model, we aim to answer the following three questions:

- Question 1: Does the feature importance score and corresponding ranking strategy in IntruDTree model simplify the security dataset by reducing the negligible features, and help to build a generalized data-driven security model?
- Question 2: Is the IntruDTree machine-learning-based security model able to effectively detect cyber intrusions and to provide significant outcome results for unseen test cases?
- Question 3: How effective is our IntruDTree model compared to traditional machine learning classification-based methods?

In answering these questions, we have conducted experiments on a security dataset consisting of the anomalies and normal classes discussed in an earlier section. We implemented all the methods in Python programming language, in which we used Scikit-learn, the most popular machine learning library for predictive data analysis, and executed them on a Windows PC. In the following subsections, we first define the evaluation metrics that are taken into account to evaluate our security model and then discuss the experimental results that answer the above questions identified for this experimental analysis.

4.2. Evaluation Metric

In order to measure the effectiveness of our IntruDTree cyber security model, we compute the outcome results in terms of precision, recall, fscore, ROC value, and overall accuracy, defined as below [19] -

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$Fscore = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

where, TP denotes true positives, FP denotes false positives, TN denotes true negative, and FN denotes false negatives in these above formal definitions of precision, recall, fscore, and overall accuracy. In addition to these evaluation metrics, we also take into account the ROC (Receiver Operating Characteristic) curve that is created by plotting the true positive rate (TPR) against the false positive rate (FPR) of the resultant security model.

4.3. Effect of Feature Importance Score and Ranking

In order to answer the first question highlighted above, we show the importance score of each feature in the dataset and the significance of this score in our IntruDTree model in this experiment. For this, Figure 4 shows the calculated importance score for various features utilizing the given security dataset. If we observe this figure, we see that the calculated score of all features is not identical in a given dataset and may vary from feature-to-feature according to their effect on the target class. According to Figure 4, the feature *src_bytes* has the highest score of 0.258 (around 25%), whereas another feature *is_host_login* has a score closer to the value 0 for this dataset. In order to show the comparative significance of each feature, Figure 4 shows their importance score values in a descending order, where the values are arranged from the largest to the smallest number. The higher the value, the more important the feature in our IntruDTree model. Thus, based on the score values, we can conclude that all the features in a given security dataset might not be similar significant to build a data-driven security model and a number of features comparatively with very low score values, less than a threshold t, can be reduced to simplify the dataset. In this experiment, we have selected all the features to build the model that satisfy the threshold value $t = 0.02$. As a result, we finally selected the top 14 features according to their importance score values shown in Table 2. Therefore, according to the results shown in Figure 4 and Table 2, we can conclude that the feature importance score and corresponding ranking strategies in our IntruDTree model is able to simplify the security dataset by reducing the negligible features. Consequently, it can help to build a generalized data-driven security model with a smaller number of security features.

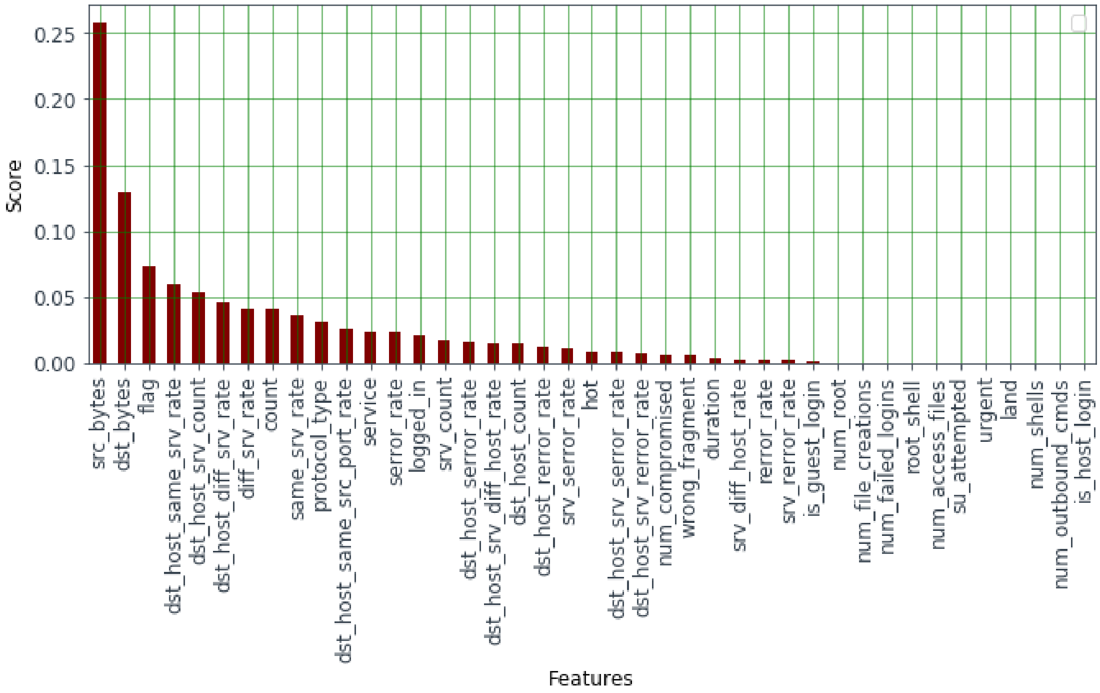

Figure 4. The importance of the features in terms of score values.

Table 2. Top ranked features with corresponding importance score values for a given security dataset.

Ranking	Security Feature Name	Importance Score
01	*src_bytes*	0.258093
02	*dst_bytes*	0.129825
03	*flag*	0.073396
04	*dst_host_same_srv_rate*	0.059504
05	*dst_host_srv_count*	0.053630
06	*dst_host_diff_srv_rate*	0.046281
07	*diff_srv_rate*	0.041144
08	*count*	0.040548
09	*same_srv_rate*	0.036620
10	*protocol_type*	0.031650
11	*dst_host_same_src_port_rate*	0.025566
12	*service*	0.023904
13	*serror_rate*	0.023188
14	*logged_in*	0.020901

4.4. Outcome Results of IntruDTree Model

In order to answer the second question highlighted above, in this experiment, we show the outcome results of our machine learning-based cyber security intrusion detection model IntruDTree. To calculate the outcome results for unseen test cases, we first built the model using a subset of 80% data of the given dataset and used the remaing 20% data for testing the model. The results are calculated by generating a confusion matrix that reports the number of false positives, false negatives, true positives, and true negatives. Based on these values, Table 3 reports the prediction results in terms of precision, recall, fscore, and accuracy for each individual class, either normal or anomaly, utilizing the given dataset, in order to show the outcome results of our IntruDTree model. If we observe Table 3, we see that for each class, this model gives significant results of the precision, recall, fscore, and accuracy, that ensures the model effectiveness while predicting anomalies or intrusions. In addition, Figure 5 shows the outcome results as an ROC curve that represents the TPR (true positive rate) against the FPR (false positive rate) of our IntruDTree model. From Figure 5, we can see that the true positive rate, i.e., correctly classified, is high close to the maximum value 1, while the false positive rate, i.e., falsely classified, is low. Thus, from the overall experimental results shown in Table 3 and Figure 5, we can conclude that our machine learning-based IntruDTree cyber security intrusion detection model is able to effectively detect either the anomalies or normal class according to their occurring patterns in the security dataset and consequently gives a significant outcome for unseen test cases.

Table 3. The outcome results of our IntruDTree model for each individual class.

Class	Precision	Recall	FScore	Accuracy
Normal	0.98	0.98	0.98	0.98
Anomaly	0.98	0.98	0.98	0.98

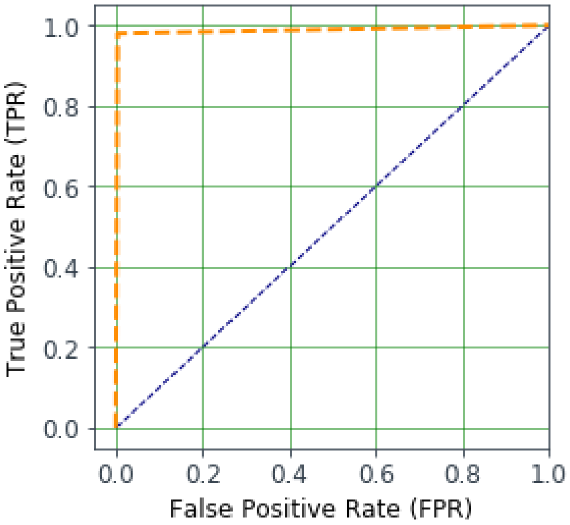

Figure 5. The ROC curve representing TPR (true positive rate) against FPR (false positive rate) of our IntruDTree model.

4.5. Effectiveness Comparison

In order to answer the third question, in this experiment, we compute and compare the effectiveness of our cyber security intrusion detection model IntruDTree, with the traditional popular machine learning-based methods. To show the effectiveness of various machine learning-based security models, we first select several popular baseline methods, such as NaiveBayes (NB), Logistic Regression (LR), K-Nearest Neighbor (KNN), and Support Vector Machines (SVM) for the purpose of comparison. For each model, we compute the outcome results utilizing the same security datasets, in order to compare the models fairly.

To compare the effectiveness mentioned above, Figure 6 shows the relative comparison of precision, recall, fscore and accuracy, utilizing the security datasets discussed in earlier section. For each machine learning-based model, we use the same training and testing set of data, where 80% data are used to train the security model and the rest 20% data are used for testing the model. If we observe Figure 6, we see that our IntruDTree model has a better outcome than the traditional models. The reason is that in our IntruDTree model, we first determined the important features and then built the intrusion detection tree by taking into account the selected features. As a result, it minimizes the model variance and over-fitting issues, and generalizes the security model as well by considering the important features. Thus, the model is able not only to increase the prediction results for unseen test cases but also to minimize the computational complexity of the resultant data-driven security model. Therefore, according to the results shown in Figure 6 and the above experimental analysis, we can conclude that our IntruDTree model is more effective than the traditional machine learning classification based methods while analyzing the cybersecurity data.

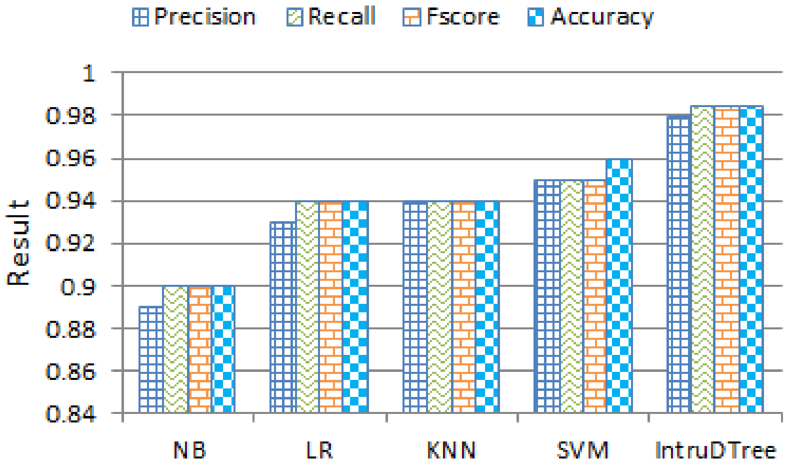

Figure 6. Effectiveness comparison in terms of precision, recall, fscore, and accuracy of different machine learning based security models.

5. Conclusions

In this paper, we have presented an intrusion detection tree IntruDTree machine-learning-based security model. In our approach, we have first taken into account the ranking of security features according to their importance and then built a tree-based generalized intrusion detection model based on the selected important features. We have done this to make the security model effective in terms of prediction accuracy for unseen test cases, and efficient by reducing the computational cost with the processing of less number of features while generating the resultant tree-like model. Finally, the effectiveness of our IntruDTree model was examined by conducting a range of experiments on cybersecurity datasets. We have also compared the outcome results of IntruDTree model with several traditional popular machine learning methods to analyze the effectiveness of the resulting security model.

Future work could assess the effectiveness of the IntruDTree model by collecting large datasets with more dimensions of security features in IoT security services, and measuring its effectiveness at the application level in the domain of cybersecurity.

Author Contributions: All authors agreed to publish this manuscript.

Funding: This research was funded by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, under Grant No. D-213-611-1441.

Acknowledgments: This project was funded by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, under Grant No. D-213-611-1441. The authors, therefore, gratefully acknowledge DSR technical and financial support.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sun, N.; Zhang, J.; Rimba, P.; Gao, S.; Zhang, L.Y.; Xiang, Y. Data-driven cybersecurity incident prediction: A survey. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 1744–1772.
2. Dainotti, A.; Pescapé, A.; Ventre, G. Worm traffic analysis and characterization. In Proceedings of the 2007 IEEE International Conference on Communications, Glasgow, UK, 24–28 June 2007; pp. 1435–1442.
3. Qu, X.; Yang, L.; Guo, K.; Ma, L.; Sun, M.; Ke, M.; Li, M. A Survey on the Development of Self-Organizing Maps for Unsupervised Intrusion Detection. *Mob. Netw. Appl.* **2019**, doi:10.1007/s11036-019-01353-0.
4. IBM Security Report. Available online: https://www.ibm.com/security/data-breach (accessed on 20 October 2019).
5. Tsai, C.F.; Hsu, Y.F.; Lin, C.Y.; Lin, W.Y. Intrusion detection by machine learning: A review. *Expert Syst. Appl.* **2009**, *36*, 11994–12000.
6. Mohammadi, S.; Mirvaziri, H.; Ghazizadeh-Ahsaee, M.; Karimipour, H. Cyber intrusion detection by combined feature selection algorithm. *J. Inf. Secur. Appl.* **2019**, *44*, 80–88.
7. Tapiador, J.E.; Orfila, A.; Ribagorda, A.; Ramos, B. Key-recovery attacks on KIDS, a keyed anomaly detection system. *IEEE Trans. Dependable Secur. Comput.* **2013**, *12*, 312–325.
8. 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.* **2010**, *40*, 516–524.
9. Milenkoski, A.; Vieira, M.; Kounev, S.; Avritzer, A.; Payne, B.D. Evaluating computer intrusion detection systems: A survey of common practices. *ACM Comput. Surv. (CSUR)* **2015**, *48*, 1–41.
10. Buczak, A.L.; Guven, E. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Commun. Surv. Tutor.* **2015**, *18*, 1153–1176.
11. 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* **2018**, *6*, 35365–35381.
12. Moskovitch, R.; Elovici, Y.; Rokach, L. Detection of unknown computer worms based on behavioral classification of the host. *Comput. Stat. Data Anal.* **2008**, *52*, 4544–4566.
13. Sommer, R.; Paxson, V. Outside the closed world: On using machine learning for network intrusion detection. In Proceedings of the 2010 IEEE Symposium on Security and Privacy, Berkeley/Oakland, CA, USA, 16–19 May 2010; pp. 305–316.
14. Seufert, S.; O'Brien, D. Machine learning for automatic defence against distributed denial of service attacks. In Proceedings of the 2007 IEEE International Conference on Communications, Glasgow, UK, 24–28 June 2007; pp. 1217–1222.
15. Sarker, I.H.; et al. Cybersecurity Data Science: An Overview from Machine Learning Perspective, 2020..
16. Sarker, I.H.; Kayes, A.; Watters, P. Effectiveness Analysis of Machine Learning Classification Models for Predicting Personalized Context-Aware Smartphone Usage. *J. Big Data* **2019**, *6*, 57.
17. Sinclair, C.; Pierce, L.; Matzner, S. An application of machine learning to network intrusion detection. In Proceedings 15th Annual Computer Security Applications Conference (ACSAC'99), Phoenix, AZ, USA, 6–10 December 1999; pp. 371–377.
18. Alazab, A.; Hobbs, M.; Abawajy, J.; Alazab, M. Using feature selection for intrusion detection system. In Proceedings of the 2012 International Symposium on Communications and Information Technologies (ISCIT), Gold Coast, Australia, 2–5 October 2012; pp. 296–301.
19. Han, J.; Pei, J.; Kamber, M. *Data mining: Concepts and Techniques*; Elsevier: Amsterdam, The Netherlands, 2011.
20. Witten, I.H.; Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques*; Morgan Kaufmann: Burlington, MA, USA, 2005.
21. Agrawal, R.; Srikant, R. Fast algorithms for mining association rules. In Proceedings of the 20th International Conference on Very Large Data Bases, Santiago, Chile, 12–15 September 1994; Volume 1215, pp. 487–499.
22. Sarker, I.H.; Salim, F.D. Mining User Behavioral Rules from Smartphone Data through Association Analysis. In Proceedings of the 22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Melbourne, Australia, 3–6 June 2018; pp. 450–461.
23. Sarker, I.H. Context-aware rule learning from smartphone data: Survey, challenges and future directions. *J. Big Data* **2019**, *6*, 95.

24. Sarker, I.H. A machine learning based robust prediction model for real-life mobile phone data. *Internet Things* **2019**, *5*, 180–193.
25. Li, Y.; Xia, J.; Zhang, S.; Yan, J.; Ai, X.; Dai, K. An efficient intrusion detection system based on support vector machines and gradually feature removal method. *Expert Syst. Appl.* **2012**, *39*, 424–430.
26. Amiri, F.; Yousefi, M.R.; Lucas, C.; Shakery, A.; Yazdani, N. Mutual information-based feature selection for intrusion detection systems. *J. Netw. Comput. Appl.* **2011**, *34*, 1184–1199.
27. Hu, W.; Liao, Y.; Vemuri, V.R. Robust Support Vector Machines for Anomaly Detection in Computer Security. In Proceedings of the International Conference on Machine Learning and Applications—ICMLA 2003, Los Angeles, CA, USA, 23–24 June 2003; pp. 168–174.
28. Wagner, C.; François, J.; Engel, T. Machine learning approach for ip-flow record anomaly detection. In Proceedings of the International Conference on Research in Networking, Valencia, Spain, 9–13 May 2011; pp. 28–39.
29. Moskovitch, R.; Nissim, N.; Stopel, D.; Feher, C.; Englert, R.; Elovici, Y. Improving the detection of unknown computer worms activity using active learning. In Proceedings of the Annual Conference on Artificial Intelligence, Osnabrück, Germany, 10–13 September 2007; pp. 489–493.
30. Kotpalliwar, M.V.; Wajgi, R. Classification of Attacks Using Support Vector Machine (SVM) on KDDCUP'99 IDS Database. In Proceedings of the 2015 Fifth International Conference on Communication Systems and Network Technologies, Gwalior, India, 4–6 April 2015; pp. 987–990.
31. Saxena, H.; Richariya, V. Intrusion detection in KDD99 dataset using SVM-PSO and feature reduction with information gain. *Int. J. Comput. Appl.* **2014**, *98*, 25–29.
32. Pervez, M.S.; Farid, D.M. Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs. In Proceedings of the 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014), 2014; pp. 1–6.
33. Shon, T.; Kim, Y.; Lee, C.; Moon, J. A machine learning framework for network anomaly detection using SVM and GA. In Proceedings of the Sixth Annual IEEE SMC Information Assurance Workshop, 2005; pp. 176–183.
34. Kokila, R.; Selvi, S.T.; Govindarajan, K. DDoS detection and analysis in SDN-based environment using support vector machine classifier. In Proceedings of the 2014 Sixth International Conference on Advanced Computing (ICoAC), 2014; pp. 205–210.
35. Kruegel, C.; Mutz, D.; Robertson, W.; Valeur, F. Bayesian event classification for intrusion detection. In Proceedings of the 19th Annual Computer Security Applications Conference, 2003; pp. 14–23.
36. Benferhat, S.; Kenaza, T.; Mokhtari, A. A naive bayes approach for detecting coordinated attacks. In Proceedings of the 2008 32nd Annual IEEE International Computer Software and Applications Conference, 2008; pp. 704–709.
37. Panda, M.; Patra, M.R. Network intrusion detection using naive bayes. *Int. J. Comput. Sci. Netw. Secur.* **2007**, *7*, 258–263.
38. Koc, L.; Mazzuchi, T.A.; Sarkani, S. A network intrusion detection system based on a Hidden Naïve Bayes multiclass classifier. *Expert Syst. Appl.* **2012**, *39*, 13492–13500.
39. Shapoorifard, H.; Shamsinejad, P. Intrusion detection using a novel hybrid method incorporating an improved KNN. *Int. J. Comput. Appl.* **2017**, *173*, 5–9.
40. Vishwakarma, S.; Sharma, V.; Tiwari, A. An intrusion detection system using KNN-ACO algorithm. *Int. J. Comput. Appl.* **2017**, *171*, 18–23.
41. Sharifi, A.M.; Amirgholipour, S.K.; Pourebrahimi, A. Intrusion detection based on joint of K-means and KNN. *J. Converg. Inf. Technol.* **2015**, *10*, 42.
42. Bapat, R.; Mandya, A.; Liu, X.; Abraham, B.; Brown, D.E.; Kang, H.; Veeraraghavan, M. Identifying malicious botnet traffic using logistic regression. In Proceedings of the 2018 Systems and Information Engineering Design Symposium (SIEDS), 2018; pp. 266–271.
43. Besharati, E.; Naderan, M.; Namjoo, E. LR-HIDS: Logistic regression host-based intrusion detection system for cloud environments. *J. Ambient Intell. Humaniz. Comput.* **2019**, *10*, 3669–3692.
44. Kumar, P.A.R.; Selvakumar, S. Distributed denial of service attack detection using an ensemble of neural classifier. *Comput. Commun.* **2011**, *34*, 1328–1341.
45. Dainotti, A.; Pescapé, A.; Ventre, G. A cascade architecture for DoS attacks detection based on the wavelet transform. *J. Comput. Secur.* **2009**, *17*, 945–968.

46. Quinlan, J.R. Induction of decision trees. *Mach. Learn.* **1986**, *1*, 81–106.
47. Quinlan, J.R. *C4.5: Programs for Machine Learning*; Morgan Kaufmann Publishers, Inc.: Burlington, MA, USA, 1993.
48. 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.* **2019**, doi:10.1007/s11036-019-01443-z.
49. Ingre, B.; Yadav, A.; Soni, A.K. Decision tree based intrusion detection system for NSL-KDD dataset. In Proceedings of the International Conference on Information and Communication Technology for Intelligent Systems, 2017; pp. 207–218.
50. Malik, A.J.; Khan, F.A. A hybrid technique using binary particle swarm optimization and decision tree pruning for network intrusion detection. *Clust. Comput.* **2018**, *21*, 667–680.
51. Relan, N.G.; Patil, D.R. Implementation of network intrusion detection system using variant of decision tree algorithm. In Proceedings of the 2015 International Conference on Nascent Technologies in the Engineering Field (ICNTE), 2015; pp. 1–5.
52. Rai, K.; Devi, M.S.; Guleria, A. Decision tree based algorithm for intrusion detection. *Int. J. Adv. Netw. Appl.* **2016**, *7*, 2828.
53. Puthran, S.; Shah, K. Intrusion detection using improved decision tree algorithm with binary and quad split. In Proceedings of the International Symposium on Security in Computing and Communication, 2016; pp. 427–438.
54. 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.* **2017**, *73*, 2881–2895.
55. Balogun, A.O.; Jimoh, R.G. Anomaly intrusion detection using an hybrid of decision tree and K-nearest neighbor, Journal of Advances in Scientific Research and Applications (JASRA), 2015.
56. Sangkatsanee, P.; Wattanapongsakorn, N.; Charnsripinyo, C. Practical real-time intrusion detection using machine learning approaches. *Comput. Commun.* **2011**, *34*, 2227–2235.
57. Network Intrusion Detection. Available online: https://www.kaggle.com/ (accessed on 12 March 2020).