Applications of Artificial Intelligence in Cyber Security

Elmurad Abbasov
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
abbas070@morris.umn.edu

ABSTRACT

Since 2018, there have been 10.5 billion malware attacks that government and private organizations, as well as individual users, are experiencing. Cyber security has been around for as long as user privacy became an important concern in our society, and the more users are using the internet and interacting with each other, the more malware attacks there are going to be. On a good note, some applications of artificial intelligence, such as machine learning can handle a good amount of those attacks and leave more decision-making issues to cyber security professionals.

This research is going to look into specific machine learning models that are able to automate some of the identifying protocols that prevent malware attacks. To be specific, machine Learning models such as KNN (k-nearest neighbors) and decision tree C 4.5 will be discussed with possible results on their efficiency compared to other machine Learning models.

Keywords

Artificial Intelligence, Machine Learning, Intrusion Detection System, Cyber Security, Cyber Threats

1. INTRODUCTION

The Internet has been a large part of our lives in the last two decades. Nearly on an everyday basis, millions of people are using smart devices, such as computers and phones, to communicate with each other. Internet and other technologies have brought many benefits to our society, but unfortunately, some internet users have negative intentions. Considering that the amount of new users increases daily, with such massive use of the internet and amounts of personal data stored, some individuals started developing malicious software to threaten other users, their privacy, and sensitive information.

To emphasize the problem of cyber threats, some reports suggest that there have been about 7.9 billion data breaches around the world in 2019 alone [4]. This was also 112 percent more data breaches than in 2018. In addition, the International Data Corporation predicts that data breaches by 2022 will be worth 133.7 billion dollars [1]. Cyber security is a term when professionals are trained to detect, prevents, and trouble-solve cyber attacks.

To deal with those problems, there are several machine learning related methods that will be discussed in this paper. Currently, there are several ways of detecting anomalies and mitigating malware, such as using Intrusion Detection System (IDS), which is a system that analyzes the incoming network traffic and monitors important operations in system files. Although cyber security professionals are able to manage important incidents, the amount of cyber threats is growing every year, and this is the time when several Artificial Intelligence applications will be very useful for not just automating the process of detection and prevention of malware, but also improving the efficiency. This way, automated algorithms, and models will be able to perform low and mid-level tasks, whereas cyber security professionals will be left with more challenging, decision-making, and intellectual work.

For the purposes of this study, we are going to focus on a subcategory of Artificial Intelligence called machine learning, and discuss two of the six algorithm that were used in the experiment by Mahfouz et al. [7]: KNN (k nearest neighbor) and decision tree C 4.5, as these two showed better performance comparing to other algorithms. The other four machine learning algorithms that the researchers Mahfouz et al. tested on identifying cyber attacks are Naive Bayes, Logistic, Neural Network (NN), and Support Vector Machine (SVM). The experiment itself has been done to evaluate the performance of machine learning algorithm on detecting cyber attacks, and the process has been divided into three phases. In the first phase, the researchers Mahfoiz et al. [7] have used the raw NSL-KDD data set without any modifications and default settings for their algorithms. For the second phase, the NSL-KDD data set was modified to reduce dimentionality and hyperparameter optimization was applied. In the third phase, the imbalance issue in the NSL-KDD data set was resolved. An overview of machine learning and related sections will be discussed in the future sections.

2. BACKGROUND

Because this study is built on explaining how Artificial Intelligence algorithms can improve the detection and classification of cyber attacks, it is necessary to underline several definitions before stepping into methodology part of this paper. To be specific, a description of machine learning and relevant methods will be discussed further in sections 2.1.1 through 2.2.1.

UMM CSci Senior Seminar Conference, Fall 2021 Morris, MN.

2.1 Machine Learning

Machine Learning (ML) is a branch of Artificial Intelligence (AI) that is focused on statistical models and algorithms that are meant to learn from data patterns, and such learning process in machine learning is called training. By training on data patterns and features in the data, ML models gradually improve their accuracy [5].

2.1.1 Types of Machine Learning

In **Supervised Learning**, the data that is used for training ML models is already labeled and defined. Furthermore, a feature in data sets represents a measurable piece of data that can be used for analysis. There are two main categories of supervised machine learning - classification, which is an algorithm that assigns test data into specific categories, and regression, which is a set of statistical processes for estimating the relationship between dependent and independent variables. For supervised learning, since the data set is already labeled and includes inputs as well as correct outputs, the model is able to learn over time and estimate accuracy through the loss function until the error is minimized. This type of machine learning is also known to be task driven, because after enough training, a model will be able to observe new data and predict a label for it.

The above mentioned machine learning type requires training for their models to learn patterns. Usually, the data set is divided into two different parts, where one part is used for training itself, and the other part is used for testing. After the ML model has gone through training, we want to test our model on the data that our model has not seen before to compare the performance and make sure that it can still solve out problem even with some different data.

2.1.2 Hyperparameter optimization

For us to define **hyperparameter optimization**, we first need to define what is a **parameter** and a **hyperparameter**.

- Model parameters are the set of configuration variables that are internal to the model and can be learned from the training data, and the value of those parameters is estimated from the input data. Model parameters specify how input data is transformed into the desired output.
- Model hyperparameters are the set of configuration variables that are external to the model, they have a value that is used to control the learning process in the model, and they cannot be directly trained from the input data. Model hyperparameters define the structure of the model.
- Hyperparameter optimization is the process of finding the most optimal hyperparameters for the learning in machine learning algorithm. The optimization process locates the hyperparameters tuple, which is a finite ordered list, and then produces a model that minimizes the predefined loss function on the given data.

2.1.3 Cross Validation

Cross Validation is a statistical method used to compare and evaluate machine learning algorithms. It is works by separating the data set into K equally sized folds, where K-1 folds are used to train the model, and the last fold is left for model testing. This process reiterates until every fold gets the chance to act as the test data set, and the capability of the model is estimated by averaging the performance measures across all folds. For comparing those six ML algorithms, the researchers Mahfouz et al. [7] have used Cross-Validation of 10-folds.

2.2 Malware

For us to better understand the relationship between machine learning and cyber security, meaning what role does ML play in detecting cyber attacks, it is important to introduce an idea behind Intrusion Detection System (IDS).

2.2.1 Intrusion Detection System (IDS)

An Intrusion Detection System is a software application that is searching for malicious activity in the entire network or between a server and a user, and an intrusion itself is an act of entering a virtual space without a proper permission. There are currently two major detection methods that can be used in an IDS. Signature-based detection, for example, searches for patterns in the network and compares them with pre-determined attack patterns, which are also known as signatures. Statistical anomaly-based detection is when an IDS detects a suspicious network traffic and compares it to an already established baseline, which is usually a data set of "normal" or "attack" files, and this type of an Intrusion Detection System is used in this study [13].

3. METHODOLOGY

In this section the machine learning method will be described and how it was applied on the NSL-KDD data set for intrusion detection. For the main part, the researchers Mahfouz et al. [7] are performing three different experiments, or three different phases, that share the same purpose. As it was mentioned earlier, the experiment was divided into three parts, where in the first part the researchers Mahfouz et al. [7] loaded raw NSL-KDD data set into six machine learning algorithms with default settings. In the second phase, a feature selection process has been done to reduce dimentionality of the data set, as well as hyperparameter optimization was applied. In the third phase, the imbalance issue on the NSL-KDD data set was resolved. For a visual illustration of those three phases, please refer to figure 1.

The experiment was done using machine learning tool called Weka (Waikato Environment for Knowledge Analysis) that provides access to work with many algorithms and can be used for tasks such as classification. In Weka, some of the machine learning algorithms are named differently, but the original name of the algorithms will be always provided in this paper. For example, the six machine learning algorithms that were tested on classifying cyber attacks by the researchers Mahfouz et al. [7] are named in Weka as Naive Bayes, Logistic, MultilayerPerception, SMO, IBK, and J48, whereas their original names are Naive Bayes, Logistic, Neural Network, SVM, KNN, DT C 4.5. Before diving into the experiment setup, it is essential to cover the details about NSL-KDD data set.

3.1 Data Description

Data, in the context of machine learning, as well as in cyber security research, is the most important unit which is used to predict and classify models. With a sufficient data set, machine learning algorithms can be trained based on the similar features or patterns in the data. Typically, the

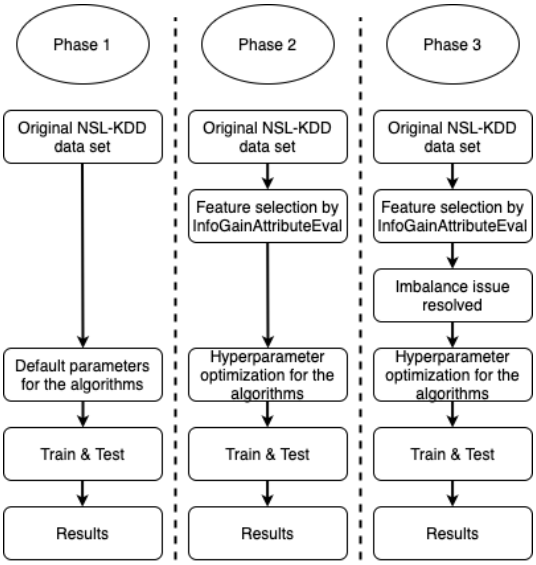


Figure 1: Illustration of three phases in the experiment

size of the data is an important factor that determines the accuracy of the ML algorithm.

3.1.1 NSL-KDD

For this paper, the Network Security Laboratory - Knowledge Discovery in Databases (NSL-KDD) data set has been used to train and evaluate machine learning models and their accuracy. The origin of NSL-KDD data set is the KDD Cup'99 data set, which is the benchmark in the research of Intrusion Detection System. This data set contains records of malware connections, labeled as intrusions or attacks, and safe connections. Unlike the original KDD Cup'99 data set, which contains a lot of redundant (78%) including many duplicate (75%) records, the updated, revised, and cleaned-up NSL-KDD version is more accurate. The NSL-KDD data set is less than KDD Cup'99 data set mainly because of removed redundant records.

This data set contains 42 features per record, where 41 of those features are about the traffic input, and the remaining feature is a label of either a safe connection, or a threat connection which is labeled as an attack type. There are three feature sets in NSL-KDD data set, such as basic feature of individual TCP connection, content features, and traffic features.

Furthermore, in this data set, we have four different classes of attacks:

- **Denial of Service (DoS)** - An attack which overloads a server with abnormal amount of traffic that shuts down the connection to and from the target system
- **Probe** - An attack that is extracting specific and often personal information from the target system
- **Remote to Local (R2L)** - An attack that tries to gain local access to a remote machine
- **User to Root (U2R)** - An attack that tries to gain root access to a system of interest or a network

Overall, NSL-KDD dataset contains 125973 patterns in the training set and 22544 in the testing set. For the breakdown of the patterns based on the attack category, please refer to the following table 1 [7]. One concern that the researchers Mahfouz et al. [7] have brought up is that the number of records for R2L and UR2 classes is abnormally small compared to other attack classes. Normal and DoS classes can be considered as majority classes, and R2L and U2R can be considered to be minority classes. Because of such an imbalance between the classes in the NSL-KDD data set, a machine learning model can potentially create biased results toward the records from the majority classes, and the classification accuracy would be better for Normal and DoS classes than for R2L and U2R. The researchers, however, have provided a solution to that imbalance issue that will be discussed later.

Class	Training Set	Occurrences (%)	Testing Set	Occurrences (%)
Normal	67343	53.46 %	9711	43.08 %
DoS	45927	36.46 %	7460	33.08 %
Probe	11656	9.25 %	2421	10.74 %
R2L	995	0.79 %	2885	12.22 %
UR2	52	0.04 %	67	0.89 %
Total	125973	100.0 %	22544	100.0 %

Table 1: Number of patterns per class

3.1.2 Features description

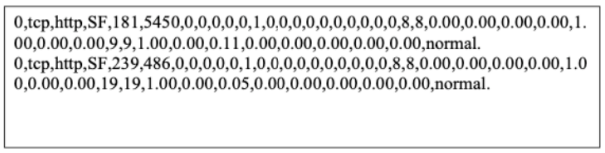
For us to better understand the feature selection process, it is important to take a look at some of the features provided in the NSL-KDD data set and understand what they stand for and what they represent. On the example below we can see most of the features listed from NSL-KDD data set.

Each 41 feature in the NSL-KDD data set has its meaning and certain role that it plays in the scenario where a cyber criminal is attacking a user. To outline some features, **Logged_in** is if logged in then logged_in = 1, else 0. This would mean that if a user successfully logged in into a user's system then it would be counted as 1, and if a user could not log in then it would be counted as 0. **Is_guest_login** would mean that if a cyber criminal would be able to log in into user's system as a guest then it would be counted as 1, otherwise it is 0. With the same logic we can have **Root_shell** = if root shell is obtained then root_shell is 1, else is 0. To provide another example that is not a boolean, **Count No.** is a feature that counts number of connections to the same host in last 2 seconds. **Src Bytes** is a feature that denotes the number of data bytes transferred from source to destination in single connection. **Num Root** feature represents a number of root accesses or number of operations performed as a root in the connection. **Land** denotes if source and destination IP addresses and port numbers are equal then it is counted as 1, otherwise it is counted as 0. Such features are either binary, such as *Is_guest_login and Land*, continuous, such as *Num Root*, or discrete, such as *Count No.* and carry a value type of integers [10].

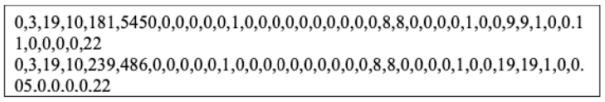
Aside from features that are counted in numerical values, NSL-KDD contains a few categorical features that are not represented in numerical values, such as **Flag** feature that denotes whether a connection was successful or error, or a **Protocol Type** feature that denotes type of the protocol such as TCP, UDP, etc. In addition, there are 2 more categorical features - **Service** and **Class**. [3] Such categorical features carry a value type of strings.

3.1.3 Data Manipulation

For the nature of machine learning algorithms used in the research by Mahfouz et al., categorical features need to be converted to numeric features so that ML algorithms can identify and train on such data. Although Mahfouz et al. did not specify the exact method they have used for converting the features, one of the most popular ways that has been used in similar research is to assign specific numerical values to specific categorical features. For example, the *normal* class if NSL-KDD data set could be assigned to the value of 22, TCP could be a value of 3, HTTP could be a value of 19, and so on. In the example below of figure 2 we can see an illustration on such a conversion, where some of the features were assigned specific numerical values.



Original samples from NSL-KDD dataset.



Results after data transformation.

Figure 2: Transformation of categorical features to numerical values

3.2 Three phases of the experiment

The researchers Mahfouz et al. did an experiment over three phases, where in the first one they used default Weka settings for all six machine learning algorithms, in the second phase feature selection and hyperparameter optimization was done to improve accuracy, and in the third phase imbalance issue was mitigated to help ML algorithms detect and classify attacks types of U2R and R2L.

In the first phase, the researchers are applying a raw NSL-KDD data set without any changes on six machine learning algorithms with default settings.

In the second phase, the data set NSL-KDD was modified to reduce its dimension, which was done through a *feature selection* process. For that, the researchers have used InfoGainAttributeEval algorithm provided by Weka tool that operates by measuring how each feature contributes in decreasing the overall entropy, or in other words the algorithm evaluates the worth of a feature by measuring the information gain with respect to the class. When computing the IG, entry values vary from 0, which is denoted as no information at all, and 1, which is the maximum information. The features that contribute more relevant information will have a higher IG value and can be selected. Features that do not add much information will have a lower score and can be removed. Ranker search method is applied in this algorithm, where we can specify a starting set of features that will be ignored during the ranking and threshold by which features may be discarded from the ranking. As a result, this algorithm selected 14 out of 41 features. In addition, a *hyperparameter optimization* was done by CVParameterSelection algorithm also provided by Weka that performed parameter selection by cross-validation.

In the third phase, the data set NSL-KDD was modified to solve the imbalance issue. Specifically, to deal with the majority classes, or Normal and DoS classes, the researchers Mahfouz et al. [7] applied under-sampling technique using Weka's resample filter that takes a random subsample using either sampling with or without replacement. To deal with the minority classes, or R2R and U2R classes, over-sampling technique was applied using Weka's SMOTE (Synthetic Minority Over-sampling Technique) filter that generates synthetic instances similar to the minority class and as a result increases the minority group. Weka allows to select the minority group and specify the percentage of increment, such that a percentage of 100% will double the minority group.

3.2.1 Feature Selection

Because of the large amounts of data, processing such data can be often inefficient. The feature selection in a process that is very important for high dimensional data sets, such as for our NSL-KDD data set that has 41 attributes with an additional label attribute.

During a feature selection process, an algorithm is selecting a subset of the original features so that the feature space is optimally reduced to the evaluation criteria; a feature selection method selects a subset of relevant features. Kohavi et al. [6] describes features to be either strongly relevant or weakly relevant, where in strongly relevant feature removal of data deteriorates the performance of the algorithm. A feature is called weakly relevant if removal of a subset of features containing s deteriorates the performance of the algorithm. In addition, if a feature is neither strongly nor weakly relevant, then it is irrelevant [7].

The InfoGainAttributeEval algorithm was applied by Mahfouz et.al [7]. The algorithm used a ranker system that ranked the attributes based on their evaluation, which is chosen according to the ranking method described earlier. As a result, the algorithm selected 14 out of 41 NSL-KDD based features.

3.3 Outlined ML Algorithms

As it was mentioned earlier, two machine learning algorithms with better classification performance were outlined by the researchers Mahfouz et al. Those two algorithms are KNN (k-nearest neighbors) and decision tree C 4.5.

3.3.1 KNN (k-nearest neighbors)

In the experiment, there were two machine learning algorithms that showed better performance comparing to other algorithms, and one of those is KNN, or k-nearest neighbors algorithm. This algorithm uses majority voting principle, where an object, a sample, a data point, or a record of a cyber attack as in our case is classified by the majority vote of its neighbors. The classification is based on a distance function that measures the difference or similarity between two instances. For instance, when k is 1, then the object is assigned to the class of the single nearest neighbor, and the neighbors are taken from a set of objects for which the class is known [14]. The distance can be calculated using the standard Euclidean distance d(x,y) between two instances x and y which is defined in the following equation, where x_i is the feature element of X, y_i is the feature element of Y, and

n is the total number of features in the data set.

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

In the context of machine learning, KNN algorithm would work with first loading the data to the model, then k would be chosen to represent the number of neighbors. The algorithm then would calculate the distance between the record of a cyber attack and its neighbors and further this distance would be stored in ascending order. Once the first k entries would be listed out the algorithm would assign a class of an attack based on the majority present in the neighbor points. In other words, since we are working with labeled data, we already have pre-classified training points that represent attack types, and the algorithm would essentially select the k pre-labelled cases closest to the unknown case.

3.3.2 Decision Tree C 4.5

Another successful machine learning algorithm with good performance outlined in the research by Mahfouz et al. [7] is the C 4.5 algorithm that generates a decision tree. This algorithm splits data recursively into subsets so that each subset contains more or less homogeneous states of the target variable. The creation of sub-nodes increases the homogeneity of resultant sub-nodes. When the recursive process is completed, a decision tree is formed which can be converted into simple If and Then rules, such as **if** a flower color is red **then** it is a rose. For example, when a node N is created by a DT, if records in the data set are all of the same class, C, then a DT would return N as a leaf node labeled with the class C.

A decision tree consists of a **root node** with no incoming edges and zero or more outgoing edges, **internal or test nodes** with exactly one incoming edge for each and two or more outgoing edges, and **leaf or terminal nodes** that represent the decision node and have exactly one incoming edge and no outgoing edges [12]. Each leaf node is assigned to a class label, and non-terminal nodes such as a root and other internal nodes, contain attribute test conditions to separate records that have different characteristics.

For the algorithm to start the tree development process, it first needs to determine which attribute can work as a root node. C 4.5 uses **information gain (IG)**, which is a measure of how much information a feature provides about a class and determines the order of features represented in the nodes, and **entropy**, which quantifies how much information there is in a random variable and determines how a decision tree splits data. IG provides a way to use entropy to calculate how a change to a data set impacts its purity, such that smaller entropy suggest more purity. To divide each input data, first the root node is chosen to separate the data [8], and a root node is the node that has the maximum IG. It is also important to note that information gain is inversely proportional to Entropy.

For the equation 1 representing entropy, the summation denotes the total number of C classes and $p(X)$ is the probability of randomly picking an element of class X. The information gain is computed for each feature and represented in equation 2. If S denotes a set of training examples, and a is the selected feature, then IG is calculated for a split by subtracting the original entropy for the data set before split, $H(S)$, from the conditional entropy, $H(S|a)$, for the data set given the feature a [2]. Conditional entropy is the amount of information in one random variable given we already know the other.

$$\text{Entropy: } H(X) = -\sum_{X=1}^{c} p(X) \log p(X) \tag{1}$$

$$\text{Information Gain: } I(S, a) = H(S) - H(S|a) \tag{2}$$

In the context of machine learning, DT C 4.5 algorithm would work by first calculating the information gain of each feature, then splitting the data set into subsets using the feature for which the information gain is maximum, and then a node is formed using a feature with the maximum IG. If all the records belong to the same class, a DT makes the current node as a leaf node with the class as its label. This process is repeated for all other selected features until DT has formed all leaf nodes [9].

In simple scenarios, a decision tree can develop a reasonable number of nodes that are easy to understand, but in some cases when working with large data sets, such as with NSL-KDD, a decision tree can contain up to several thousands of nodes, which could lead to low accuracy results. To reduce the number of nodes a DT can develop, pruning, which is the selective removal of certain parts of the tree, is commonly applied. For pruning to be applied, not all points in a leaf node should be in the same class. Although the researchers Mahfouz et al. did not specify if they have used pruning, a similar IDS study [11] that used DT C 4.5 with Weka on NSL-KDD data set show that without pruning, a DT C 4.5 produced a tree of 456 nodes and 400 leaves, but with pruning their DT produced a tree of 229 nodes size and 188 leaves.

4. RESULTS EVALUATION

For evaluation the results, metrics such as True Positive (TP), False Negative (FN), False Positive (FP) and True Negative (TN) were used.

- **True Positive (TP)** - an outcome where the model correctly predicts the positive class (malware was identified as a threat)
- **False Negative (FN)** - an outcome where the model incorrectly predicts the negative class (a malware file was identified as a non-threat)
- **False Positive (FP)** - an outcome where the model incorrectly predicts the positive class (a clean file was identified as a threat)
- **True Negative (TN)** - an outcome where the model correctly predicts the negative class (a malware was identified as a threat)

For accuracy evaluation, parameters such as True Positive Rate (TPR), which is used to measure the percentage of positives, False Positive Rate (FPR), meaning a model falsely identified that there is a relationship between the two phenomena such as between two attack types. Precision, which is the quality of a positive prediction made by the model, and Recall, which measures the algorithm's ability to detect Positive samples. Generally, to know when a machine learning algorithm performs well, TPR should be high, where 1 is

the perfect condition, and FPR should be low, where 0 is the perfect condition. In addition, F-measure is a combined of both precision and recall that captures both properties, and is also known to be the harmonic mean of the precision and recall. These parameters will be seen in some of our tables later in the experiment section. Please refer to the following figure 3 for the confusion matrix overview that will help us understand the performance of the algorithm.

		Predicted	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

Figure 3: Confusion matrix

TPR is calculated as a ratio of true positives divided by the total number of true positives and false negatives, FPR is the ratio between the number of malicious files categorized as non-threats over the total number of negatives, Recall is the number of true positives over the number of true positives and false negatives, and Precision is calculated as a ratio of the number of true positives divided by the total number of positive predictions. Please refer to the following equations for the detailed description of the parameters.

$$TPR = \frac{TP}{TP + FP} \qquad FPR = \frac{FP}{FP + TN}$$

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

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

$$F = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN}$$

5. EXPERIMENT DISCUSSION

The results show us that there has been a decent improvement in the accuracy and other parameters of those six machine learning models between phase one, two, and three. In tables 2 and 3 we can see the performance improvement between phase one and two in test data set, as well as notice the accuracy of individual machine learning algorithms where KNN (IBK) and decision tree C 4.5 (J48) have better performance than other algorithms.

It is important to note that only results from the test data set are presented below. Nearly any machine learning model will demonstrate good results with enough training and time. The moment when we can see the actual performance of the machine learning model is when we test it on some unseen for that model data that has not been used for training. Thus, we refer to the results of the test data set to evaluate algorithm's performance.

Classifier	Accuracy	TPR	FPR	Precision	Recall	F-Measure
NB	76.12 %	0.916	0.337	0.673	0.916	0.776
Logistic	75.60 %	0.928	0.380	0.649	0.928	0.763
NN	77.60 %	0.929	0.393	0.642	0.929	0.759
SVM	75.39 %	0.926	0.393	0.641	0.926	0.758
KNN	79.35 %	0.927	0.353	0.665	0.927	0.775
C 4.5	81.69 %	0.972	0.318	0.698	0.972	0.813

Table 2: Classifiers results on test data of phase 1

Classifier	Accuracy	TPR	FPR	Precision	Recall	F-Measure
NB	78.15 %	0.782	0.083	0.821	0.782	0.794
Logistic	81.51 %	0.815	0.142	0.851	0.815	0.832
NN	78.15 %	0.782	0.173	0.818	0.782	0.799
SVM	79.83 %	0.798	0.161	0.832	0.798	0.814
KNN	84.35 %	0.824	0.134	0.860	0.824	0.841
C 4.5	82.67 %	0.807	0.157	0.837	0.807	0.821

Table 3: Classifiers results on test data of phase 2

In table 4, we can compare our two best algorithms and their performance in phases one, two, and three. First, we can see a noticeable improvement in accuracy between phase one and two, two and three, as well as a substantial improvement in accuracy between phases one and three. In addition, we can see that with the case of KNN (IBK) algorithm, the model was not able to classify, or in the context of Cyber Security detect any cyber attacks from R2L and U2L classes because of the imbalance issue. The decision tree C 4.5 algorithm was able to classify some of R2L attacks, although with poor accuracy. In phase three, however, when the researchers have resolved the imbalance issue, the KNN (IBK) algorithm has a substantial improvement from 0% accuracy to 53.2% accuracy in detecting R2L attacks, and from 0% to 41.5% in detecting U2R attacks. The C 4.5 algorithm also showed significant improvement in phase three as for classifying minority groups, such as an improvement from 18.9% accuracy in phase one to 55.1% accuracy in phase three in detecting R2L and from 0% to 39.3% detecting U2R attacks.

Classifier	Class	Phase I	Phase II	Phase III
KNN	Normal	79.3 %	86.8 %	99.4 %
	DoS	80.5 %	90.7 %	99.5 %
	Probe	71.8 %	76.2 %	99.0 %
	R2L	00.0 %	00.0 %	53.2 %
	U2R	00.0 %	00.0 %	41.5 %
C 4.5	Normal	81.6 %	84.8 %	99.5 %
	DoS	80.1 %	89.2 %	99.2 %
	Probe	67.9 %	63.2 %	91.6 %
	R2L	18.9 %	18.2 %	55.1 %
	U2R	00.0 %	00.0 %	39.3 %

Table 4: Classifiers accuracy for different classes and phases

6. CONCLUSION

In conclusion, six different classifiers were evaluated on their performance to detect cyber attacks on the NSL-KDD data set. KNN (IBK) and decision tree C 4.5 (J48) showed good performance comparing to other algorithms. In addition, imbalance mitigation method improved limitations of the algorithms in detecting RL2 and U2L attacks.

Acknowledgments

Thank you to Elena Machkasova and Nic McPhee for their feedback and advice.

7. REFERENCES

[1] Budgeting for Cyber Attacks: Security spending to reach 133.7 billion by 2022. `https://www.digitalinformationworld.com/2019/05/the-future-of-cybersecurity-budgeting-infographic.html`. [Online; accessed 01-December-2021].

[2] A Simple Explanation of Information Gain and Entropy. https://victorzhou.com/blog/information-gain/. [Online; accessed 07-June-2019].

[3] A. Alazab, M. Hobbs, J. H. Abawajy, and M. Alazab. Using feature selection for intrusion detection system. *2012 International Symposium on Communications and Information Technologies (ISCIT)*, pages 296–301, 2012.

[4] C. Brooks. More Alarming Cybersecurity Stats For 2021. https://www.forbes.com/sites/chuckbrooks/2021/10/24/more-alarming-cybersecurity-stats-for-2021-/.

[5] IBM Cloud Education. What is Supervised Learning?, 2020. https://www.ibm.com/cloud/learn/supervised-learning.

[6] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1):273–324, 1997. Relevance.

[7] A. M. Mahfouz, D. Venugopal, and S. G. Shiva. Comparative analysis of ML classifiers for network intrusion detection. In *ICICT*, 2019.

[8] K. Rai, M. S. Devi, and A. Guleria. Decision Tree Based Algorithm for Intrusion Detection. *International Journal of Advanced Networking and Applications*, 07:2828–2834, 01 2016.

[9] Y. Sakkaf. Decision Trees for Classification: ID3 Algorithm Explained. https://towardsdatascience.com/decision-trees-for-classification-/id3-algorithm-explained-89df76e72df1, Sept. 2020.

[10] G. Saporito. A Deeper Dive into the NSL-KDD Data Set. https://towardsdatascience.com/a-deeper-dive-into-the-nsl-kdd-data-/set-15c753364657, Nov. 2019.

[11] A. Sheta and A. Alamleh. A professional comparison of C4.5, MLP, SVM for network intrusion detection based feature analysis. *ICGST Journal of Computer Networks and Internet Research*, 12 2015.

[12] Tan and Steinbach. Data Mining Classification: Basic Concepts, Decision Trees, and Model Evaluation. 2004.

[13] Wikipedia contributors. Intrusion detection system — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Intrusion_detection_system&oldid=1051927958, 2021. [Online; accessed 6-November-2021].

[14] Wikipedia contributors. K-nearest neighbors algorithm — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=K-nearest_neighbors_algorithm&oldid=1051590352, 2021. [Online; accessed 18-November-2021].