ACTA SCIENTIFIC COMPUTER SCIENCES

Volume 3 Issue 7 July 2021

Research Article

A Comparative Study of Machine Learning Techniques in Cyberthreat and Cyberattack Detection[1]

Balakrishnan Dasarathy*

University of Maryland Global Campus, Adelphi, MD, USA

***Corresponding Author:** Balakrishnan Dasarathy, University of Maryland Global Campus, Adelphi, MD, USA.

Received: May 01, 2021

Published: July 1, 2021

Abstract

This paper is on applying leading machine learning (ML) techniques to detect cyberthreats and cyberattacks. The paper begins with a short overview of three leading supervisory techniques, Logistic Regression, Neural Network (NN) and Support Vector Machine, and one unsupervised learning technique, the Multivariate Gaussian Distribution-based anomaly detection, that are being applied. The datasets used for training, validation and testing are then described. Potential alternatives to the datasets used are also discussed. Metrics used to assess the algorithms, precision, recall and accuracy, are then defined. An overview of bias vs. variance tradeoff to get optimal results with the algorithms is then provided. The performance of the ML algorithms applied is then described. The performance of the NN algorithm for two-class classification (normal vs. attack) and the anomaly detection using the Multivariate Gaussian Distribution function is encouraging. The results are summarized, and future directions are outlined. A main goal of the future research is to improve performance in classifying attacks into their constituent classes, i.e., to develop more accurate signatures or models for attack classes. An in-depth bias vs. variance analysis of the algorithms applied is also a major direction going forward.

Keywords: Intrusion Detection; Machine Learning; Cyberattacks; Cyberthreats; Machine Learning for Cyberthreat and Cyberattack Detection

1. Introduction

This paper is on applying machine learning (ML) techniques to detect cyberthreats and cyberattacks.

A question that begs an answer at the outset then is why ML for cybersecurity. As observed in [1], ML is growing in popularity because it has been phenomenally successful in various areas such as language understanding and image and voice recognition. Self-driving cars would not have been possible but for advances in image recognition. Cybersecurity attacks are growing both in frequency and sophistication over the years. Traditional methods of intrusion detection, while still largely used and recommended, are no longer keeping up with the demands of growing security threats. It takes time anywhere from hours to days to months to understand an attack, develop defensive techniques, such as patches, and distribute them. As computing power increases and

[1]A version of this paper was presented as a keynote talk at the ICCEDE 2020 conference: https://www.glbitm.org/iccede2020/Speakers.asp

cost drops, ML is seen as a supplemental or alternate technology to traditional intrusion detection and prevention techniques. ML techniques can help develop defensive mechanisms in a shorter time and proactively.

Three leading supervised ML techniques, Logistic Regression, Neural Network (also known as Artificial Neural Network) and Support Vector Machine, to classify normal and attack data into their classes and one unsupervised machine learning technique, Multivariate Gaussian Distribution-based anomaly detection, have been of focus. The work is ongoing, and this paper is a progress report.

Cyberthreats and cyberattacks

A distinction is being made between threats and attacks. A threat leads to an attack. A cyberthreat, given an opportunity will exploit a vulnerability, to cause an attack. We need to detect both threats and attacks. Proactively, we want to detect a threat so as neutralize it before it manifests into an attack. As an example, repeated scanning of our network by an adversary is a threat. If the adversary gets into the network and one or more hosts and accesses proprietary information, say because of weak remote login password or weak firewall rules, it becomes an attack. An intrusion can be just a threat or a full-blown attack. The ML techniques discussed here are to detect both types of intrusion.

This is the overview of the paper. Section 2 is on related work. Section 3 provides an overview of the ML techniques applied. Section 4, the crux of the paper, reports on the experimental results. Section 4.1 is on metrics used to assess the algorithms. Section 4.2 discusses the dataset used and the rationale for it. Alternate data-sets that can become viable soon are also discussed in Section 4.2. Section 4.3 is on performance of the algorithms using the metrics defined in Section 4.1. Section 5 summarizes the results and suggests future directions.

2. Related Work

Researchers have been applying Machine Learning (ML) techniques to detect cyber intrusions from early 2000. A catalyst for such work is the availability of DARPA 1998/99 dataset [2,3]. There are two survey papers, Buczak and Guven in 2016 [4] and Xin, *et al*. in 2018 [5], that provide a good coverage of ML techniques applied to cyber intrusion detection. The paper by Buczak and Guven is a survey of most used and cited Machine Learning (ML) and Data Mining (DM) techniques applied to cyber intrusion detection in wired networks. Buczak and Guven distinguish ML and DL techniques as follows: ML focuses on classification and prediction, whereas DM focuses on discovering properties in data. ML and DL, however, share a lot of techniques. Techniques discussed in [4] include Bayesian Network, Neural Network, K-means Clustering, Support Vector Machine and K-Nearest Neighbors. The paper by Xin., *et al*. is a survey of both ML techniques, specifically, K-means Clustering, Support Vector Machine, K-Nearest Neighbors and Decision Tree, and Deep Learning techniques, specifically Recurrent Neural Network and Convolutional Neural Network, toward addressing intrusion detection in both wired and wireless networks. (Deep Learning techniques employ several layers to understand complex patterns.) As alluded to earlier, the work presented here focuses on a leading set of ML (and DL) techniques, Logistic Regression, Neural Network (NN), Support Vector Machine (SVM) and Gaussian-Distribution-based anomaly detection. The Neural Network that is being applied is a Deep Learning technique, but a simple one. These two survey papers report on several works that apply NN or enhanced NN and SVM, alone, or in combination with other techniques to intrusion detection with varying success. Many of these works go to distance to make a particular technique or a particular application of technique work for a given dataset in a specific environment. These works often leverage the individual researcher's cybersecurity domain knowledge. Both these survey papers stress, rightly so, that no ML technique (algorithm) has been yet established as an effective method. The work reported here does not require much domain knowledge and thus easy to do technology transfer once viability has been demonstrated.

Logistic Regression (LR) is often used with other techniques. In [6], a host-based intrusion detection system for protecting virtual machines in the cloud environment using LR in conjunction with Neural Network, Decision Tree and Linear Discriminate Analysis. [7] proposes an intrusion detection system that apples the concept of Genetic Algorithm for feature selection and LR for classification to detect network intrusions is put forward. The Multivariate Gaussian Distribution for anomaly detection, currently successfully applied in other domains, does not seem to have followers in the cybersecurity research community.

3. A Brief Introduction to the Machine Learning (ML) Techniques Used

A modern definition of ML is due to Mitchell [8], a pioneer in

the field, is as follows: A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. Suppose you have an ML program that watches you marking some emails as spams and others as not spams and learn to classify a mail as a spam or not. Experience E by the computer is watching you classify email samples; each sample in E consists of meta information such as source IP address and sender's name and terms such as payout and reward in content and subject line of each mail. Task T is classifying an email as a spam or not, and the fraction of emails classified correctly is P.

An overview of ML techniques is provided in [9]. Ng [10] is Lecture Notes of a course at Stanford University and Coursera that provides an in-depth introduction to the field that includes techniques described here and has vastly influenced the direction of this work! There are two broad categories to learning: (1) supervised and (2) unsupervised. In supervised learning, we are given a dataset (samples) with "labels;" that is, the algorithm is provided witha relationship between the input and the output in the samples. The machine learning is to find a generalized correlation between input and output. Here are a few examples:

- Given a dataset of patients with their weight, height, age, etc. and diagnosed as either having diabetes or not, learn to classify new patients as having diabetes or not based on their weight, height, age, etc.
- Given emails labeled as spam/not spam, learn a spam filter.
- Given a dataset of traffic that are clearly marked as attack data or normal data, develop a classifier to detect a new attack.
- Given a dataset of sex, age, smoking history, height and weight of a population, learn to quantify the risk for heart attack or stroke for a specific individual.

The first three examples are characterized with discreate output, whereas the last example, the output is continuous, i.e., the heart attack or stroke risk can take any value from 0 to 1. We will focus on only discreate output value ML techniques, known as classification techniques. Figure 1 illustrates supervised learning for classification.

In Figure 1, there are two labeled classes of data given to the ML algorithm, depicted as red X's and green circles. The algorithm figures the blue line that separates them. Any new sample plotted to the left of the blue line will belong to the red X category and anything to the right will belong to the class depicted by green circles. The line that is generated is often known as the pattern or model. In this case, we have less than 20 samples in two dimensions. Today's ML algorithms routinely deal with data in 1000's of dimensions or features and millions of samples or observations in their training chore or experience E!

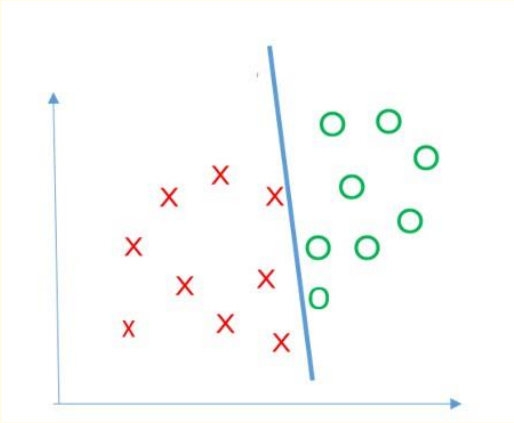

Figure 1: Illustration of supervised learning.

In unsupervised learning, the machine studies data to identify patterns; no labels are provided, i.e., no hint on how input and output might be related. A few examples of unsupervised learning are:

- Given a database of customer data, group customers into different market segments (this is often known as clustering).
- Given a set of good traffic data, identify rules/patterns/ distribution to detect attack/threat data.

One application of unsupervised learning is in the area of anomaly detection whereby only "normal" or "good" data is used in training. The training is to identify a pattern or model for normal data. Anything that is not consistent with the pattern identified is an anomaly. The anomaly detection technique applied here is based on identifying a Multivariate Gaussian Distribution function fitting the training data. An anomaly detection algorithm is typically used when there are not that many "bad" samples available or the avail-able data is skewed toward good or normal data.

Figure 2 illustrates unsupervised learning. Figure 2a is an example of clustering. The clustering ML algorithm figures out there are 3 clusters in the given (unlabeled) data. Elements in a cluster are closer to each other than to elements in another cluster. Figure 2b illustrates that, given only good samples, the unsupervised algorithm fits the data using a Multivariate Gaussian Distribution. The algorithm generates the mean and variance of the distribution. Any data that is too far away from the good data (illustrated as green X in figure 2b), using a threshold, is an anomaly. The threshold is generated using a validation process by the algorithm so as to maximize accuracy, balancing false positives and false negatives, described formally, in a later section.

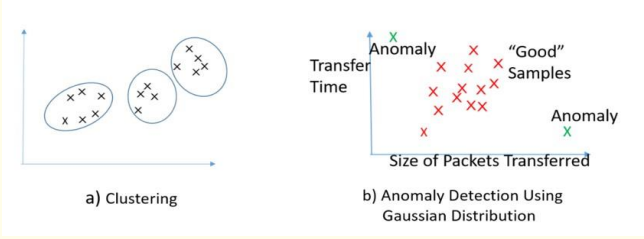

Figure 2: Unsupervised learning.

Three supervisory learning techniques, (1) Logistic Regression, (2) Support Vector Machine, and (3) Neural Network, that are of focus are described next.

3.1 Logistic Regression (LR)

Logistic Regression is a classification technique to assign samples or observations to discrete classes. Binary Logistic Regression is a special case with two classes. One way to extend any binary classification to classify a sample to multiple classes is to repeatedly apply the binary classification. For example, at the first level, one may classify a sample as "normal" and "attack". At the next level, an attack sample may be classified as a Denial of Service (DoS) or "Other" attack. A DoS attack can further be classified as a single machine-based or Distributed DoS attack, for example.

Logistic Regression is named for the function used at the core of the method, the logistic or sigmoid function. The function returns a probability value, 0 to 1, as shown in Figure 3 (taken from [11]). Using 0.5 as the threshold value, if the function were to return a value of 0.7, then we would classify an email sample, for example, as a spam. If this predictor function were to return a value of 0.2, then we would classify the email as not a spam or good.

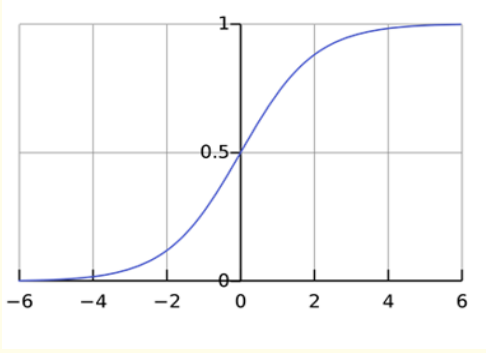

Figure 3: Sigmoid function.

3.2 Neural Network (NN)

A Neural Network models human brain's neuron connection. A Neural Network is a multilayer directed network. Because of the multiple layers employed in learning, it is also known as "Deep Learning." Neural networks have been lately applied very successfully in handwritten text recognition and image recognition with thousands of features and millions of samples. Each node in the network or neuron receives a signal from a node that precedes it, processes the signal and can signal neurons connected to it downstream. The output of each neuron is computed by some non-linear function of the sum of its inputs, as shown in Figure 4 (taken from [12]). The activation function used in the experiments reported here is the sigmoid function. Each edge has a weight that adjusts as learning proceeds and that weight determines the strength of the signal on that edge and hence the output at the downstream node. In essence, learning in a neural network is determining the weights on its edges.

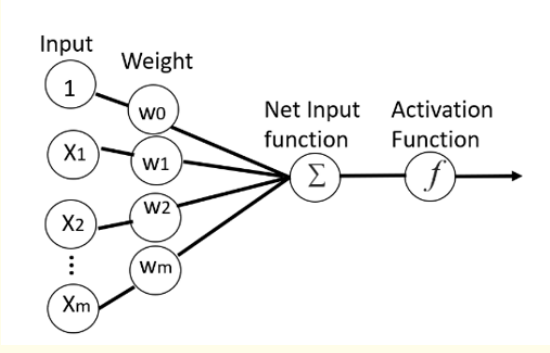

Figure 4: Neural Network basis.

A three-layer network, illustrated in Figure 5 [10], is used in the NN experiments. The $h_\theta(X)$ function is the predictor function (See section 3.4).

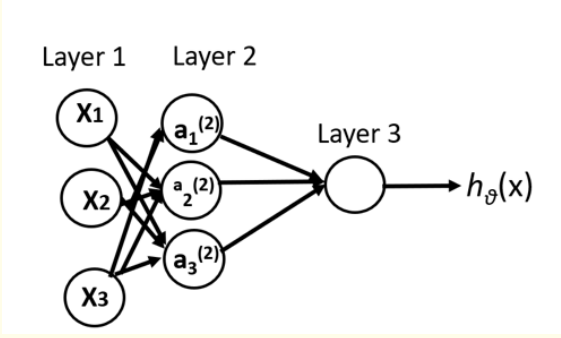

Figure 5: Three-layer Neural Network.

3.3 Support Vector Machine (SVM)

SVM is often a preferred technique for its high accuracy with less complexity. The idea behind the technique is finding a frontier which separates samples into classes and that is maximally far away from any sample. This distance from that decision boundary to the closest sample(s) determines the margin of the classifier. Maximizing the margin distance provides a degree of freedom in classifying future samples. In the case of linear separation, the hyperplane that provides the maximum margin is appropriately called the maximal margin hyperplane. Referring to Figure 6, three hyperplanes in brown, blue and black color separate the two labeled samples, shown as red X's and green circles. The hyperplane in black is the maximal margin hyperplane.

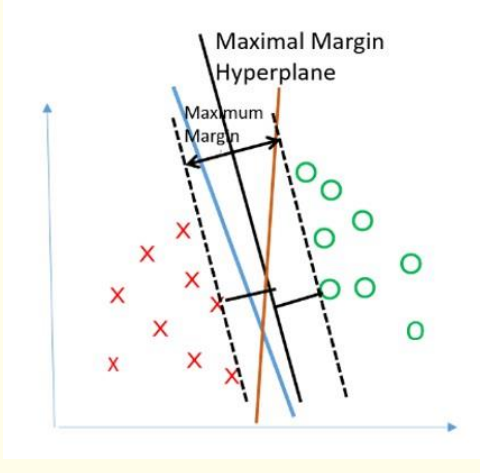

Figure 6: Support Vector Machine.

In addition to performing linear or hyperplane classification, as illustrated above, an SVM can also efficiently carry out non-linear classifications using a kernel function or kernel trick that implicitly maps the input into high-dimensional feature space. The kernel trick allows the algorithm to fit the maximal margin hyperplane in a transformed feature space [10]. The classifier is a hyperplane in the transformed feature space, it is, however, nonlinear in the original input space. The specific kernel function used in the SVM experimentation is the Gaussian radial basis function (RBF), a commonly used and a packaged solution/API in ML languages such as MATLAB and Gnu Octave.

3.4 Predictor and cost functions

Typically, a supervisory machine learning model has a function $h_\theta(X)$ (θ is the parameter or weight and X is the sample space of size m x n with m samples and n features) associated with it for prediction, i.e., what the output or label is going to be for a given sample. During training, the model evaluates how well this function predicts over the training samples. A cost function is associated with this predictor function that basically penalizes a wrong prediction. The main function of an ML algorithm then is to minimize this cost function using a well-known minimization algorithm such as the gradient descent.

Neural networks have two classes of computations. During the forward propagate mode, intermediate node predictions are propagated forward to get the predictions in the output nodes and they are compared with the real label value of each sample to get the error. By propagating forward, we estimate how well the neural network is behaving and find the error or cost of wrong predictions over the samples. Now, to minimize the error, in the propagate backward mode, the derivative of the error with respect to each weight is computed and the weights adjusted accordingly. This will go on for a pre-determined number of iterations or until some minimum for the error value is reached.

3.5 Underfitting and overfitting data

A predictor function can underfit or overfit the data. Underfitting, or high bias, is when the hypothesis function *h* maps poorly to the trend of the data. It is usually caused by a function that is too simple or uses too few samples or features of samples, as illustrated in Figure 7a. It is biased toward the very few samples or features used. Overfitting, or high variance, on the other hand, is caused by a hypothesis function that fits the available data rather two well, as illustrated in Figure 7b, but does not generalize well; on another dataset, the predictive function can be quite different.

The predictive function varies too much based on the training data, hence the predictor function is called high variance. Figure 7c illustrates the right fit.

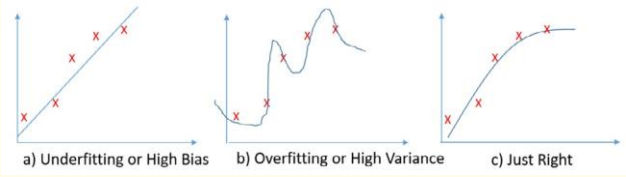

Figure 7: Underfitting and overfitting the data.

Adjusting overfitting/underfitting using regularization parameter

It is important to understand whether the implementation of an algorithm is suffering from high variance or high bias. An implementation suffers from high bias, if

- The predictive function graph has an overly low order polynomial (as in Figure 7a),
- Many samples or their features are not used in fitting the data,
- The training set and validation set errors (costs) are both high (more on the various types of datasets in Section 4.2), and
- The cross-validation error is close to or slightly higher than that of the training set.

An implementation suffers from high variance, if:

- The graph has too large degree of polynomial (as in Figure 7b) to fit the data,
- Many samples or their features are getting nearly equal weight, and
- The training set error is low, but the validation set error is much larger than the training error.

One way to trade off bias against variance in LR and NN algorithm experiments is through varying the regularization parameter, denoted as λ (lambda), in the cost function. The higher it is, the higher the bias and the lower the variance and hence moreunderfitting; the lower it is, the lower the bias and the higher the variance and more overfitting. In essence, λ controls or varies the importance of a feature or the term involving a feature variable. SVM also has a regularization parameter, denoted as C; C is inverseof λ. Hence the higher it is, the lower the bias, the higher the variance and overfitting; the lower the C value is, the higher the bias and lower the variance and underfitting. For an SVM with a kernel function, another variable that can be controlled is the standard deviation σ used in the kernel function. The larger it is, the higher the bias and the lower it is, the higher the variance. For more details on overfitting and underfitting data see [10], Sections 7 and 10.

4. Experimentation, Results and Discussion

4.1 Metrics used

		Actual	
		1	0
Predicted	1	True positive	False positive
	0	False negative	True negative

Table 1: Metrics basics.

Referring to table 1,

True Positives (TP) - These are the correctly predicted positive values

True Negatives (TN) - These are the correctly predicted negative values

False Positives (FP) - When the actual class is No, but the predicted class is Yes

False Negatives (FN) - When the actual class is Yes, but the predicted class is No

With N as the size of the dataset or number of samples, we define the following metrics to evaluate the performance of our algorithms:

$$\text{Precision (P)} = \frac{TP}{TP + FP}$$

$$\text{Recall (R)} = \frac{TP}{TP + FN}$$

$$\text{Accuracy} = \frac{TP + TN}{N}$$

$$F1 = 2 * \frac{P * R}{P + R}$$

Precision (P) is a measure of false positives (the lower it is for an algorithm, the higher the false positives reported by the algorithm), whereas Recall (R) is a measure of false negatives (the lower it is for an algorithm, the higher the false negatives reported by it). In cybersecurity, high recall is often more important than high precision, because missing an attack or threat is more problematic than identifying a normal situation as an attack or a threat. Accuracy is a measure of both false positives and false negatives. F1 is yet another measure that takes into account both false positives and false negatives and is used during validation (see Section 4.2).

4.2 Datasets

The experiments reported here were using the NSL-KDD dataset, one of the few, publicly available datasets that is heavily used for benchmarking ML algorithms in cybersecurity [4,5]. It is a carefully selected subset of DARPA KDD 1999 [13] dataset which itself derived from DARPA 98/99 dataset [2,3], mostly removing the redundancy in the DARPA KDD 1999 dataset and thus minimizing the class bias. The size of the dataset is more manageable with the NSL- KDD dataset. A few of the highlights of the NSL-KDD dataset are:

- Contains both normal data and (simulated) attack data
- Contains both training and test datasets
- 41 features (TCP packet data and higher-layer (e.g., failed logins) features), collected from the network and operating system.
- categories of attacks and threats: Denial of Service (DoS) (attack), Probe or Scan (threat), Remote to Local (R2L) (attack) and User to Root (U2R) (attack)
- Contains a 20% subset for quick evaluation: KDDTrain+_20Percent (will be referred to as NSL-KDD-Training-20%) for training (and validation) and KDDTest-21 (will be referred to as NSL-KDD-Test-20%) for testing (The experiments reported here actually employ these 20% subsets.)

Many experiments in machine learning employ only two datasets: (1) training and (2) testing, and a few make use of an additional, validation, dataset. The training dataset is used to fit the model and compute the parameters of the model. During validation, parameters of an algorithm such as λ (lambda), are varied so as to maximize F. The testing dataset is used to evaluate the model obtained from training or validation to determine how well the model would work in the unforeseen real world.

The more recent CIC-IDS2017 dataset [14] has many attractive features including:

- Data on more recent attacks (e.g., Brute Force FTP, Brute Force SSH, DoS, Heartbleed, XXS and SQL attacks, Infiltration, Botnet and DDoS)
- Data collected over HTTP, HTTPS, FTP, SSH and email protocols
- Data from several network elements such as gateways, routers and switches
- Data using two major platforms today: Windows and Ubuntu hosts
- Large feature set (78 features as opposed to 41 features in KDD and NSL-KDD datasets)

This dataset is not quite ready just out-of-the-box. The files are too large with more than 3 million records in all. The data also has class imbalance that can lead a classifier to bias towards majority classes. The dataset also has missing values in fields and missing labels (see [15]). We are planning to employ this dataset in our future experiments, as the community gets more experience with the dataset and the dataset gets accepted, and smaller subsets are created.

Another potential dataset for our experimentation is the CTU-13 dataset, a dataset of botnet traffic that was captured at the CTU University, Czech Republic [16]. This dataset also has issues that require pre-processing for data cleaning and to address class imbalance [17]. Both the survey papers referred to in Section 2 [4,5] do emphasize the lack of publicly available dataset in the cybersecurity field capturing modern day attacks as a major issue in the field.

In my experiments, data normalized, and features with zero variance removed. Several of the experiments also leveraged Principal Component Analysis (PCA) [18]. PCA reduces the dimensionality of the dataset. The number of dimensions can be adjusted as a function of the variation to be retained in the dataset.

Performance results

All the ML algorithms here are implemented in GNU Octave [19]. Octave is free software, and its syntax is largely compatible with MATLAB. C and Microsoft Excel were used in preparing the datasets for the experiments.

Anomaly detection using multivariate gaussian distribution

The NSL-KDD-Training-20% dataset has both normal data and attack data. 80% of the "normal" data from this dataset went to training; the remaining 20% of "normal" data went to validation. Since the algorithm is for anomaly detection, the training is done only on the normal data. 100% of the attack (and threat) data in this training set went to validation. All attack data are marked 1 and all normal data are marked 0. For testing data, NSL-KDD-Test-20% dataset was used. All data were normalized, and zero-variance features were upfront eliminated. The size of datasets used are: training: 10,797; validation: 14395; and test: 11,850.

Table 2 lists the result. (In the tables below, rows having bold text correspond to the best observed result(s).)

Variance Kept	No of Features	Validation	Test Details
1	37	Best F1: 0.954048	Precision: 0.828628 Recall: 0.898948 Accuracy: 0.765148 F1: 0.862357
0.95	**25**	**Best F1: 0.959831**	**Precision: 0.862262** **Recall: 0.85595** **Accuracy: 0.770211** **F1: 0.859094**
0.9	21	Best F1: 0.953622	Precision: 0.875447 Recall: 0.782017 Accuracy: 0.730549 F1: 0.826099

Table 2: Anomaly Detection using Multivariate Gaussian Distribution (with or without PCA).

With no reduction in feature dimensionality (see the first row), close to 90% recall was observed. Maximum accuracy of 77% with corresponding precision and recall at 86% and 85.6%, respectively, was observed with the feature space reduced to 25.

Logistic Regression (LR)

The NSL-KDD-Training-20% dataset was used for training as it had both normal and attack data. Logistic Regression (LR) was applied to classify this data into 2 classes, normal (0 class) vs. attack (attack and threat) (1 class). The NSL-KDD-Test-20% dataset was used for testing. All data were normalized, and zero-variance features were upfront eliminated. The performance was studied as a function of λ, the regularization parameter, to tradeoff bias and variance, as described in Section 3.5.

The number examples and features, respectively, in the training dataset after removing features with no variance are 25192 and 39, and for the test dataset, these values are 11850 and 39. Table 3 and Table 4 highlight the results without and with PCA applied, respectively. 90% variance is kept when applying PCA resulting inreducing the feature dimension to 18.

As can be seen from table 3 and table 4, the best recall and accuracy values were both less than 60%. As such, applying LR to further classify the attack class was not carried out. It is worth noting that reducing features to 18 (roughly in half) (so the algorithm can converge faster) didn't affect the performance appreciably and the application of PCA does not require cybersecurity domain knowledge.

λ (Lambda)	Training Data			Test Data		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall
1,000	95.530327	96.5046	93.8091	55.341772	90.1275	51.0208
5,000	**94.5578**	**95.9589**	**92.2081**	**58.2363**	**95.6722**	**51.2889**
7,500	93.9822	96.0717	90.803	57.6034	95.5911	50.5259
10,000	93.287552	96.127	89.1936	57.088608	95.6823	49.8144

Table 3: Logistic Regression in normal vs. attack classification (without PCA).

λ (Lambda)	Training Data			Test Data		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall
1,000	94.899174	95.6203	93.3322	54.9282	89.7754	50.7012
5,000	**94.2402**	**95.8645**	**91.595**	**57.519**	**95.5469**	**50.4434**
7,500	93.621	95.9808	90.0877	57.1814	95.6196	49.9691
10,000	93.136710	96.1475	88.8359	56.708861	95.7899	49.2679

Table 4: Logistic Regression in normal vs. attack classification (With PCA applied).

Neural Network (NN)

As in Logistic Regression, performance as a function of λ (Lambda), the regularization parameter, was studied. In all the NN experiments, a three-layer network was employed; the hidden layer had 10 nodes; the number of nodes in the output layer is the number of classes; and the number of nodes in the input layer is 40, the number of features or dimensions in our input data, plus 1 for the bias term.

Here are the salient features of the NN-based binary classification (Normal Vs. Attack) experiment:

- The NSL-KDD-Training-20% dataset was used for training; the number of examples and features in the training dataset: 25192, 39.
- The NSL-KDD-Test-20% dataset was used for testing; the number of examples and features in the test dataset: 11850, 39
- The number of optimization iterations: 100.

Table 5 highlights the performance results obtained. As relatively good performance especially with respect to recall (97% corresponding to λ = 200) was observed (see the highlighted row). The NN algorithm was then applied to study how well it could classify attacks into their constituent classes: DoS (Class 1), Probe (Class 2), R2L (Class 3), and U2R (Class 4) (Class 0 is the Normal class).

Λ (Lambda)	Training Data			Test Data		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall
0	92.846936	97.7244	86.6729	66.489451	80.721	77.583
0.1	95.581931	97.9261	92.4806	74.194093	82.0711	87.6057
10	93.636869	98.139	88.0184	71.915612	80.7077	86.3168
100	94.402985	97.8335	89.9855	77.848101	81.6352	94.1019
200	**94.692760**	**98.373**	**901047**	**80.270042**	**81.8504**	**97.515**
1000	92.656399	97.3485	86.6048	53.248945	79.2817	58.0429

Table 5: Neural Network (NN) in normal vs attack classification.

As can be seen from table 6, a three-layer network with one hidden layer with 10 nodes, is not quite sufficient to give an acceptable performance. The plan is to introduce one additional layer as well more node(s) in the hidden layer(s) in future experiments.

Λ (Lambda)	Training Data				Test Data			
	Accuracy for all 5 classes	Normal Vs. Attack			Accuracy for all 5 classes	Normal vs. Attack		
		Accuracy	Precision	Recall		Accuracy	Precision	Recall
0.0	87.642903	92.1999	97.6047	85.3615	36.991369	61.1097	76.2547	68.3451
0.1	86.174182	90.7391	98.5953	81.291	39.050555	58.6806	78.3431	60.4733
0.2	**86.698158**	**92.0967**	**99.2525**	**83.6754**	**34.895191**	**65.3021**	**76.314**	**76.519**
0.5	86.325024	91.7394	99.2557	82.9005	41.479655	57.6449	81.0791	55.2367
10	86.364719	91.7355	99.4675	82.7131	40.616523	58.0025	80.5995	56.4115
100	**86.317085**	**90.8463**	**99.2691**	**80.9589**	**43.107275**	**56.5228**	**85.9976**	**48.758**

Table 6: Neural Network (NN) in five-class classification.

Support Vector Machine (SVM)

As in the other cases, Support Vector Machine (SVM) was applied to classify normal vs. attack (attack and threat) traffic. The NSL-KDD-Training-20% and NSL-KDD-Test-20% datasets were used for training and testing, respectively. The number of examples and features in the training dataset were 25192 and 39, respectively, and the number of examples and features in the test dataset were 11850 and 39, respectively. In the experiments, the performance of SVM was studied/controlled using two parameters: C, the regularization parameter (the larger the C value, the higher the variance, the lower the bias) and σ (Sigma), the standard deviation, used in the kernel function (the larger the Sigma, the lower the variance and the higher the bias).

Sigma (σ)	C	Training Data			Test Data		
		Accuracy	Precision	Recall	Accuracy	Precision	Recall
1	0.001	98.4797	99.4343	97.292	52.4304	94.5187	44.4525
1	0.1	99.4205	99.3868	99.3698	61.4008	94.541	56.0734
1	1	99.7777	99.7446	99.7786	60.2025	94.1823	54.7536
1	10.	99.8928	99.8723	99.8978	60.5569	94.4287	55.0526
0.1	**10**	**99.9801**	**99.9745**	**99.983**	**68.7510**	**94.8124**	**65.3949**
0.1	**1**	**99.9484**	**99.9574**	**99.9319**	**68.7848**	**94.8691**	**65.3949**
0.5	1	99.8690	99.8298	99.8893	59.1898	94.2966	53.3615

Table 7: Support Vector Machine (SVM) in normal vs. attack classification.

As can be seen from the highlighted rows in table 7, the best accuracy and recall that were obtained are both under 70%. As such, the experiment to study SVM for multiclass classification is currently on hold.

5. Concluding Remarks

For the cybersecurity field, recall is more important than precision. High recall of 97.5%, and a little more than 80% accuracy and precision were observed with the Neural Network (NN) when applied to binary classification, normal vs. attack. This is the best result we have so far observed. The result should not be surprising as neural networks can better deal with a large number of features than either SVM or LR and the three-stage network employed facilitates learning of complex patterns. The NN classification has, however, so far fallen short (less than 50%

accuracy) in classifying attack classes in more granularity. A main goal of the future research is to improve performance in classifying attacks into their constituent classes, i.e., to develop more accurate signatures or models for attack classes. Toward that goal, a plan to study the applicability of a more complex NN, with one additional hidden layer as well as more nodes in the hidden layer(s).

The performance of the anomaly detection algorithm based on the Multivariate Gaussian Distribution is encouraging, although not quite acceptable. Recall values in the high 80% range are being observed. This is remarkable in the sense that the NSL-KDD data was not all that Gaussian (normally distributed)!

The training and test datasets used were 20% subset of the NSL-KDD complete dataset, a common dataset used in cyber experiments to compare ML techniques. It is 20 years old. As such, as alluded to earlier, the NSL-KDD dataset lacks attack data that are more common today such as phishing, buffer overflow and SQL injections. We desperately need realistic modern-day datasets of attacks and threats (and normal data); CIC-IDS2017 dataset [14] is promising, as observed in Section 4.2. This dataset has a few drawbacks. The files are also too large with more than3 million records in all. The data also has class imbalance that can lead a classifier to bias towards majority classes. The plan is to incorporate this dataset in our experiments, as the community gets more experience with the dataset and smaller subsets are created. A critical examination of the CTU-13 dataset [16] for its readiness and applicability is also planned. A research goal is also to be a catalyst in developing new datasets for a given environment. Assuch, planned directions include correlation analysis of features for common attacks, and development of format standards for feature data.

The future work will also focus on doing an in-depth analysis of bias vs. variance in applying the algorithms to classify a specific cyber dataset. The Logistic Regression for binary classification of NSL-KDD seems to suffer from high bias (high value of Lambda) and the Neural Network-based multiclass classification seems to suffer from high variance (low Lambda). We need to study learning curves, i.e., how error or cost functions vary with the size of training, test and validation datasets so as to be sure whether an ML algorithm suffers from bias or variance (see Section 3.5). Increasing the role of more features and higher polynomial terms in the predictive function will address the bias problem, whereas employing more training samples and the use of fewer features will address the variance problem.

Bibliography

1. Delplace A., *et al*. "Cyber Attack Detection thanks to Machine Learning Algorithms". COMS7507: Advanced Security Course Material. The University of Queensland, Australia (2019). https://arxiv.org/pdf/2001.06309v1.pdf, last accessed 2021/05/11.

2. R Lippmann., *et al*. "Evaluating intrusion detection systems: The 1998 DARPA offline intrusion detection evaluation". Proc. IEEE DARPA Inf. Surviv. Conf. Expo., (2000): 12-26.

3. Lippmann R., *et al*. "The 1999 DARPA offline intrusion detection evaluation". *Computer Networks* 34 (2000): 579-595.

4. Buczak AL and Guven E A. "A Survey of Data Mining and Machine Learning, Methods for Cyber Security Intrusion Detection". *IEEE Comm. Surveys and Tutorials* 18.2 (2016).

5. Xin,Y., *et al*. "Machine Learning and Deep Learning Methods for Cybersecurity". IEEE Access. date of current version 6 (2018): 36365-35381.

6. Besharati E., *et al*. "LR-HIDS: logistic regression host-based intrusion detection system for cloud environments". *Journal of Ambient Intelligence and Humanized Computing* 10 (2018): 3669-3692. https://doi.org/10.1007/s12652-018-1093-8, last accessed 2021/05/11.

7. Ghosh P and Mitra R. "Proposed GA-BFSS and logistic regression based intrusion detection system". Proc. 2015 Third Intl. and Information Technology (C3IT), Hooghly (2015): 1-6. doi: 10.1109/C3IT.2015.7060117 (2015).

8. Mitchell T. "Machine Learning". McGraw-Hill (1997).

9. Costa C.D. "A Tour of Machine Learning Algorithms" (2020).

10. Ng A. "Stanford University Machine Learning Course by Prof. Andrew Ng" (2011). https://www.holehouse.org/mlclass/index.html, last accessed 2021/05/11.

11. Pathmind. "A Beginner's Guide to Logistic Regression for Machine Learning". https://wiki.pathmind.com/logistic-regression, last accessed 2021/05/10.

12. Pathmind. "A Beginner's Guide to Neural Networks and Deep Learning". https://wiki.pathmind.com/neural-network, last accessed 2021/05/08

13. KDD Cup 1999 Data. "University of California Irvine, UCI KDD archive". http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, last accessed 2021/05/11.

14. Sharafaldin I., *et al*. "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization". 4th International Conference on Information Systems Security and Privacy (ICISSP) Portugal (2018). https:// www.sciencepubco.com/index.php/ijet/article/view/22797, last accessed 2021/05/11.

15. Panigrahi R and Borah, H. "A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection System". *International Journal of Engineering and Technology* 7.3.24 (2018). https:// www.sciencepubco.com/index.php/ijet/article/view/22797, last accessed2021/05/11.

16. The CTU-13 Dataset. "A Labeled Dataset with Botnet, Normal and Background traffic". https://www. stratosphereips.org/ datasets-ctu13, last accessed 2021/05/11.

17. Vishwakarma AR. Network Traffic Based Botnet Detection Using Machine Learning Master's Thesis, San Jose State University, San Jose, CA (2020). https://scholarworks.sjsu.edu/ cgi/ viewcontent.cgi?article=1917&context=etd_projects, last accessed 2021/05/11.

18. Brems M. "A One-Stop Shop for Principal Component Analysis". *Towards Data Science* (2017). https://towardsdatascience. com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c, last accessed 2021/05/11.

19. "GNU Octave". https://www.gnu.org/software/octave/index.

Volume 3 Issue 6 June 2021