International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 1, February 2020, pp. 978~986
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i1.pp978-986

Performance analysis of binary and multiclass models using azure machine learning

Smitha Rajagopal, Katiganere Siddaramappa Hareesha, Poornima Panduranga Kundapur
Department of Computer Applications, Manipal Institute of Technology, Manipal Academy of Higher Education, India

Article Info

Article history:

Received Apr 17, 2019
Revised Sep 26, 2019
Accepted Oct 3, 2019

Keywords:

Azure machine learning
Decision forest
Intrusion detection
Locally deep SVM
Mutual information
UNSW NB-15

ABSTRACT

Network data is expanding and that too at an alarming rate. Besides, the sophisticated attack tools used by hackers lead to capricious cyber threat landscape. Traditional models proposed in the field of network intrusion detection using machine learning algorithms emphasize more on improving attack detection rate and reducing false alarms but time efficiency is often overlooked. Therefore, in order to address this limitation, a modern solution has been presented using Machine Learning-as-a-Service platform. The proposed work analyses the performance of eight two-class and three multiclass algorithms using UNSW NB-15, a modern intrusion detection dataset. 82,332 testing samples were considered to evaluate the performance of algorithms. The proposed two class decision forest model exhibited 99.2% accuracy and took 6 seconds to learn 1,75,341 network instances. Multiclass classification task was also undertaken wherein attack types like generic, exploits, shellcode and worms were classified with a recall percentage of 99%, 94.49%, 91.79% and 90.9% respectively by the multiclass decision forest model that also leapfrogged others in terms of training and execution time.

Corresponding Author:

Poornima Panduranga Kundapur,
Department of Computer Applications,
Manipal Institute of Technology, Manipal Academy of Higher Education,
Manipal, India.
Email: poornima.girish@manipal.edu

1. INTRODUCTION

The increased use of devices associated with internet generate huge volumes of network data [1]. This is also accompanied by advanced level of cyber-attacks that severely hamper the confidentiality, integrity and availability of computer resources [2, 3]. Robust network intrusion detection systems are the need of the hour to safeguard confidential information against malicious activities [4]. Machine learning algorithms are commonly used to address the problem of network intrusion detection [5]. Whenever machine learning algorithms are employed in the field of network intrusion detection, two recurring problems are commonly encountered by security experts, i.e., prolonged training and prediction time. The training time of algorithms span from seconds to hours [6, 7]. The longer training time taken by the Intrusion Detection Systems to analyse the data leads to substantial delays in generating alerts [8, 9],obviously considered unfavourable in the field of intrusion detection research. The problem, however, persists because network intrusion detection involves big data investigation too given its mammoth complexity [9, 10]. According to [11], 1 Gigabits per second (GBPs) of network traffic alone can introduce big data challenges. Traditional data mining tools like Weka, Scikitlearn and conventional numerical environments like Matlab may not be able to address the ever increasing issues of distributed data settings [12]. Performance and scalability are the two major considerations for conducting network intrusion detection study. Big data processing platforms like Pig [13], Spark machine learning [14] and Azure machine learning [15] are the preferred choices in

the modern scenario given their ability to uphold memory requirements and implementation essentials [16]. Going by these considerations, it is imperative to introduce radical advancements to intrusion detection infrastructure. Azure Machine Learning is one such Machine learning as-a Service initiative by Microsoft that can be employed to develop predictive models. The proposed work is an illustration to highlight the advantages of this initiative by considering a network intrusion detection use case implemented through supervised machine learning techniques. Given the existence of diverse algorithms in machine learning study, it is often advisable to investigate the performance of individual algorithms so that optimal predictions can be achieved. It is worthwhile to mention that algorithms perform differently for a given dataset. Therefore, such a comparative study as proposed, becomes indispensable in the field of machine learning research.

The objective of the proposed work is to analyse the performance of various algorithms and investigate their training time, prediction time, attack detection rate and false alarm rate by considering network instances of UNSW NB-15 dataset on a sophisticated Machine learning as a service (MLaaS) platform called Microsoft Azure Machine Learning Studio(MAMLS).A modern and a comprehensive dataset is essential to evaluate the effectiveness of the proposed approach and UNSW NB-15 dataset serves the purpose [17-19]. A significant advantage of any MLaaS offering is its ability to save computational resources that involve exceesive costs [20, 21].The novelty of the proposed approach is that the false alarm rate generated by two class decision forest model is quite negligible and the attack detection capability of multiclass decision forest model is definitely desirable. It is worthwhile to mention that the results of classification tasks are quite superior than existing state of the art techniques.Some existing studies in the literature have explored the performance of different machine learning algorithms on UNSW NB-15 dataset as elucidated below.

As described in [18], six different techniques were applied to classify the network instances of UNSW NB-15 dataset. The highest accuracy obtained was 85.56% using decision tree that also generated a false alarm rate of 15.78%. As discussed in [22], experimentation was conducted on Apache Spark to improve the accuracy and it can be noted that REP tree model achieved an accuracy of 93.56%. The training time taken was 7.92 seconds to learn 47,342 instances. A wrapper approach was implemented in [23] using genetic algorithm and various tree based classifiers by selecting different subsets of features. An accuracy of 81.42% and a false alarm rate of 6.39% was obtained using this approach but wrapper approaches are considered to be computationally exhaustive [24]. The performance of four classification algorithms like: Decision Tree, Random Forest, SVM and Naive Bayes were compared and Apache Spark was used as a processing paradigm [25]. It was noticed that Random Forest was the best performing classifier with 97.49% accuracy and the training time was reported as 5.69 seconds. Another insightful study was presented in [26] that focussed on the implementation of supervised machine learning techniques on UNSW NB-15 dataset to test their robustness. Empirical results revealed that logistic regression performed better than other algorithms like Tree-J48, SVM and Naive Bayes. An overall accuracy of 89.26% was reported by logistic regression model.

2. METHODOLOGY

This section describes in detail the various aspects of experimentation. This article focuses on eight two-class and three multiclass classification algorithms. Classification models were designed in four different stages: preprocessing, feature selection using mutual information, tuning of hyper- parameters and designing predictive workflows. Basically, UNSW NB-15 has 47 features and two class labels. The dataset has continuous, discrete and symbolic features in varied ranges thus subjected to pre-processing. During the experimentation, all nominal features were converted into integers. Numerical features with a wide range are difficult to handle. Hence logarithmic scaling was applied to decrease their range of values. Boolean features did not need any scaling. Min-max normalization was applied to determine the smallest and largest value of each feature in the range [0, 1].

$$V' = \frac{V - min_i}{max_i - min_i} \quad (1)$$

In (1), min and max refer to the minimum and maximum values of each feature "i". Each feature value V is scaled to V'. Feature scoring was used to prioritize the features followed by the design of workflows to perform classification tasks. Upon experimentation, mutual information yielded comparatively better results than other filter based feature selection methods. Mutual information, as the name suggests is a measure of information between a random feature 'x' and target variable 'y' or the label [27]. The mutual information between two variables is given by (2) as explained in (2) and (3).

$$I(X;Y) = \iint p(x,y) log \frac{p(x,y)}{p(x)\,p(y)}\, dxdy \quad (2)$$

$$S = argmax\, I\,(X_S\,;Y)|S| = n \quad (3)$$

In (2) includes p (x, y) which denotes the joint probability density function, p(x), p(y) are the marginal density functions. In the context of feature selection, 'n' refers to the number of selected features and is known as joint mutual information. The subset of selected features is referred to as X_S as given in (3). The distribution of training and test datasets is shown in Table 1. It can be noted that there was no redundancy found in training and testing distributions unlike benchmark datasets [17, 18]. As mentioned above, mutual information was used as feature scoring method available as a module on Azure Machine learning studio. The salient features, as listed in Table 2, were given as input to the various classifiers to obtain the best possible predictions from them. In order to aptly assess the performance of all machine learning algorithms considered in the study, 10-fold cross validation was applied and a separate testing set was considered for evaluation. Cross validation becomes important in machine learning research to control overfitting and corroborate the capability of algorithms to generalize on independent data (testing set) [28].

Table 1. Dataset distribution

Class	Training samples	Testing samples
Normal	56000	37000
Analysis	2000	677
Backdoor	1746	583
Reconnaissance	10491	3496
Shellcode	1133	378
Worms	130	44
DOS	12264	4089
Fuzzers	18184	6062
Generic	40000	18871
Exploits	33393	11132
Total	1,75,341	82,332

Table 2. List of salient features

Sl.No	Name of the feature	Feature score
1	ct_state_ttl	0.686
2	dttl	0.56
3	Sttl	0.27
4	dinpkt	0.23
5	smean	0.20
6	rate	0.199
7	ct_dst_sport_ltm	0.196
8	sload	0.190
9	state	0.1875
10	dload	0.1872
11	sbytes	0.185
12	dpkts	0.175
13	dbytes	0.171
14	dur	0.158
15	ackdat	0.156
16	dmean	0.147
17	synack	0.138
18	tcprtt	0.131

2.1. Averaged perceptron

Averaged Perceptron is a simplified form of neural network that uses a linear function to classify the samples. MAMLS offers an option of setting a single value or multiple values as learning rates in order to test the proficiency of two class Averaged Perceptron model. Different parameter values like 0.1, 0.5 and 1.0 were set as learning rate to determine the optimal configuration of the stochastic gradient descent optimizer. The advantage of using a parameter range is that the model reprises over several combinations eventually producing the optimal model.

2.2. Bayes point machine

Bayes point machine is based on Bayesian principle to efficiently classify network instances by choosing a Bayes point (average). Typically, iterations are set in the range of 5 to 100. This value indicates the number of times the algorithm iterates over the training data. Numerous trials were conducted by varying the number of iterations within the given range but the results were not convincing enough and longer training time was observed during those trials. However, on setting the number of training iterations as 30 for two-class Bayes Point Machine, the results obtained were satisfactory and this was the basis for retaining 30 as the number of training iterations for the experiment.

2.3. Boosted decision tree

Boosted decision tree is an ensemble model primarily aimed at rectifying the errors of previously built trees. The four critical hyper-parameter values as shown in Table 3 were used to examine the competence of Two- class Boosted Decision Tree. Here, maximum number of leaves indicate the maximum leaves that can be created in any tree. The size of the tree can be increased by varying this value but overfitting and prolonged training time were encountered by increasing the number of leaves. Minimum number of samples per leaf node refers to the number of cases considered to create a leaf node. The value 10 signifies that the training data contains 10 cases meeting the same condition as the rules formulated. The initial learning rate was assigned a value 0.2 which basically hints at the rate of convergence. Further, 100 decision trees were created in the ensemble. There is also a provision to create more than 100 trees but again, the training time becomes considerably longer, hence considered inadvisable.

Table 3. Critical parameters used for configuring boosted decision trees

Max. leaves per tree	Minimum number of samples per leaf node	Learning rate	Number of trees constructed
20	10	0.2	100

2.4. Decision forest

Two-class Decision forest, as recommended by Team Azure [15] is one of the most preferred models to perform binary classification. There are two resampling methods namely replicate and bagging available to design a two class decision forest model. Replicate method trains each tree on the same training data whereas bootstrap aggregating or bagging allows each tree to be grown on a new sample. It can be noted that the values as shown in Table 4, when assigned bestowed the optimal results.

Table 4. Critical parameters used for configuring decision forest

Number of decision trees	Maximum depth of decision tree	Number of random splits per node	Minimum number of samples per leaf node
8	32	128	1

Raising maximum depth led to a maximum precision of 1 but overfitting was noted which also resulted in a longer training time (not desirable). The number of random splits signifies the number of splits generated per node from which the optimal split could be chosen. Minimum number of samples per leaf node refers to the number of cases needed to create the leaf. Attempts were made to ascertain whether better results could be obtained by varying the values of critical parameters but were not effectual.

2.5. Decision jungle

Unlike decision trees that allow only one path to every node, a decision jungle allows multiple paths from root to each leaf. Unlike decision forest which uses tree as the base learner, decision jungle employs Directed Acyclic Graph (DAG) as the base learner. Shotton et al. [29] introduced the concept of decision jungle to conserve memory and improve generalization. Number of optimization steps per decision DAG layer indicates the number of steps to be used to enhance each level of the DAG. The values as enumerated in Table 5 were used to build the model and variations introduced resulted in unsatisfactory predictions.

Table 5. Critical parameters used for configuring decision jungle

Number of DAG	Maximum depth of DAG	Maximum width of DAG	Number of optimization steps per decision DAG layer
8	32	128	2048

2.6. Locally deep SVM (support vector machine)

Locally deep kernel can be beneficial in producing better classification accuracy than Radial Basis Function (RBF) kernels due to their capability to increase feature embedding and attain consistent speed [30]. Depth of the tree is a hyper-parameter used to configure two class locally deep SVM which indicates the maximum tree depth. Choosing an appropriate value of tree depth becomes important since the training cost increases sequentially with tree depth. Thus, three regularization parameters were used to control overfitting namely lambda (W), lambda theta and lambda theta prime set at 0.1, 0.01 and 0.01 respectively. Lambda indicates the weight to be assigned to the regularization term. Lambda theta defines the space between a region boundary and the nearest data point. Lambda theta prime, a parameter needed to control the curvature in decision boundaries is also an integral component required to build the two class locally deep SVM. Usually, lambda theta and lambda theta prime will be one tenth of lambda, if chosen otherwise causes overfitting. Sigmoid sharpness refers to the scaling parameter. Sigmoid kernel is quite favorable due to its genesis from neural networks. However, its usage is not encouraged widely due to its non-positive semi definite properties [31]. Sigmoid kernel does not satisfy Mercer's theorem. Therefore, large values cannot be assigned to sigmoid sharpness. Smaller values like 1 when used can control the threshold. Table 6 illustrates the critical default parameters tuned to model the two class locally deep SVM.

Table 6. Critical parameters used for configuring locally deep SVM

Depth of the tree	Lambda	Lambda theta	Lambdatheta prime	Sigmoid sharpness
3	0.1	0.01	0.01	1

2.7. Support vector machine

Two class SVM uses L1 (Lasso) regularization to control overfitting. The default value of Lambda W=0.001 was set as weight since it is preferable to use a non-zero value to control the degree of overfitting.

2.8. Logistic regression

Optimization tolerance is a threshold that is normally specified while designing two class logistic regression model using L-BFGS (limited memory Broyden-Fletcher-Goldfarb-Shanno) optimization [15]. This model necessitates proper tuning of L1 and L2 values set as 1 and 1 respectively. The memory size in megabytes used by L-BFGS optimizer was set as 20 which indicates the past gradients stored in memory for the execution of successive steps. If the memory size is higher, then in all possibilities, it slows down the training process and the model ends up being flawed. Three significant parameters were used to build and test the effectiveness of two class logistic regression model as mentioned in Table 7. Regularization is often applied to classification problems in order to minimize overfitting.

Table 7. Critical parameters used for configuring logistic regression

L1 Regularization	L2 Regularization	Memory size used by L-BFGS
1	1	20

3. EXPERIMENTAL RESULTS AND DISCUSSION

This section presents the results obtained through experimentation using MAMLS. The four classification possibilities of any intrusion detection study are: True positives(TP), True negatives (TN), False Negatives (FN) and False Positives(FP) that determine the significant performance metrics namely Accuracy(A), Precision(P), Recall(R), F1-score(F1), Area under the curve(AUC) and false alarm rate. Additionally, the training and execution time of each model is also reported. Execution time refers to the time taken by the model to output the predictions (Class labels with respect to binary and attack type with respect to multiclass models). True Positive (Sensitivity) defines the number of positive samples correctly classified as positive. False Negative (FN) is the number of positive examples wrongly classified as negative. False Positive (FP) is the number of negative examples wrongly classified as positive and True Negative (Specificity) (TN) is the number of negative examples correctly classified as negative. The (4) to (10) define these various performance metrics:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (4)$$

$$Precision = \frac{TP}{TP+FP} \quad (5)$$

$$Recall = \frac{TP}{TP+FN} \quad (6)$$

$$F\ 1-score = 2\frac{Precision\ X\ Recall}{Precision+Recall} \quad (7)$$

False Positive Rate *(FPR)* = $\frac{FP}{FP+TN}$ (8)

False Negative Rate *(FNR)* = $\frac{FN}{FN+TP}$ (9)

False Alarm Rate *(FAR)* = $\frac{FPR+FNR}{2}$ (10)

In the proposed work, eight two-class classification algorithms were considered and their performance was analysed. Results are enumerated in Table 8. The Confusion matrix shown below represents the results of classification obtained from three classifiers namely multiclass decision forest, multiclass decision jungle and multiclass logistic regression. The actual (A) versus predicted (P) classifications presented in the confusion matrix pertain to the ten classes wherein the top most row signifies the name of the class: A (Analysis), B (Backdoor), D (DOS), E (Exploits), F (Fuzzers), G (Generic), N (Normal), R (Reconnaissance), S (Shellcode) and W (Worms). The Results obtained using Multiclass Decision Forest shown in Table 9 (Confusion Matrix 1), Table 10 (Confusion Matrix 2), and Table 11 (Confusion Matrix 3).

Table 8. Classification results obtained by applying eight two-class algorithms considered in the work

Algorithm	Testing Accuracy (%)	Precision (%)	Recall (%)	F1-score	AUC	FAR (%)	Training Time (seconds)	Execution Time (seconds)	Training accuracy (%)
Average Perceptron	77.2	80.3	88.1	0.84	0.885	28.9	9	2.5	73.5
Bayes point machine	91	90	97	0.936	0.948	12.5	8	2.3	89.4
Boosted Decision Tree	95.9	96.6	97.4	0.97	0.994	4.9	7	2.7	92.2
Decision Forest(Using Bagging as Resampling)	99.2	99%	99.6	0.994	1	1%	6	2	97
Decision Forest(Using Replicate as Resampling)	99.5	99.4	99.8	0.996	1	0.7	6	2	96.6
Decision Jungle	94.6	94.3	98	0.961	0.9	7.2	6.5	2.9	92.2
Locally deep SVM	93.3	91.6	99.3	0.953	0.975	10	7.8	3.3	91.0
SVM	85.7	89.4	89.6	0.895	0.917	16.5	7.9	3.5	83.8
Logistic Regression	91.6	90.8	97.6	0.941	0.952	11.7	7	3	88.8

Table 9. Results obtained using multiclass decision forest (confusion matrix 1)

A \ P	A	B	D	E	F	G	N	R	S	W	Recall (%)
Analysis	197	13	74	385	0	3	3	2	0	0	29
Backdoor	9	127	76	365	3	0	0	2	1	0	21.78
DOS	0	0	1300	2670	3	0	0	113	3	0	31.79
Exploits	0	0	558	10519	27	0	3	22	0	3	94.49
Fuzzers	0	0	79	467	5383	3	115	6	6	3	88.79
Generic	0	0	57	113	0	18701	0	0	0	0	99
Normal	0	0	0	24	666	0	36310	0	0	0	98.13
Reconnaissance	0	0	112	521	7	0	0	2856	0	0	81.69
Shellcode	0	0	0	13	7	0	2	9	347	0	91.79
Worms	0	0	0	4	0	0	0	0	0	40	90.9
Precision (%)	95.63	90.7	57.62	69.75	88.3	99.96	99.66	94.88	97.19	86.95	

The time taken by multiclass models to learn numerous network instances and subsequently distinguish between attack categories and normal patterns ranged between 16 to 20 seconds. The least training time of 16 seconds was taken by multiclass decision forest, followed by multiclass decision jungle that took 18 seconds to recognize the patterns belonging to different classes. The maximum training time of 20 seconds was taken by multiclass logistic regression model. It is worthwhile to mention that the execution time of Multiclass decision forest and Multiclass decision jungle was reported as 6 and 6.5 seconds respectively whereas Multiclass logistic regression took 7 seconds to output the class-wise predictions.

Table 10. Result obtained using multiclass decision jungle (confusion matrix 2)

A \ P	A	B	D	E	F	G	N	R	S	W	Recall (%)
Analysis	15	0	2	498	7	2	141	11	0	1	2.2
Backdoor	0	37	0	504	16	0	4	19	3	0	6.34
DOS	0	0	107	3667	137	0	22	142	10	4	2.6
Exploits	0	1	22	10352	366	2	111	267	11	0	92.9
Fuzzers	10	1	1	723	4455	0	757	103	12	0	73.49
Generic	0	0	28	339	47	18455	0	0	1	1	97.79
Normal	10	0	0	430	3848	0	32597	111	0	4	88.1
Reconnaissance	0	4	10	1010	36	0	49	2387	0	0	68.27
Shellcode	0	0	0	121	89	0	9	106	53	0	14
Worms	0	6	0	5	2	0	1	0	0	30	68
Precision (%)	42.85	75.5	62.94	58.65	49.48	99.97	96.75	75.87	58.88	75	

Table 11. Result obtained using multiclass logistic regression (confusion matrix 3)

A \ P	A	B	D	E	F	G	N	R	S	W	Recall (%)
Analysis	30	0	11	338	52	0	134	23	39	50	4.43
Backdoor	0	90	8	300	86	0	38	61	0	0	15.4
DOS	0	0	101	2900	466	29	258	335	0	0	2.47
Exploits	0	0	67	8539	957	22	812	735	0	0	76.7
Fuzzers	0	0	36	1660	3469	97	624	176	0	0	57.22
Generic	5	0	8	302	38	18455	20	38	0	5	97.79
Normal	20	11	40	1850	4403	0	30599	77	0	0	82.7
Reconnaissance	0	0	14	1304	696	10	77	1395	0	0	39.9
Shellcode	0	0	0	30	98	0	10	210	30	0	7.93
Worms	0	0	0	33	0	0	0	0	0	11	25
Precision (%)	54.54	89	35.43	49.48	33.79	99.15	93.9	45.73	43.47	16.66	

Typically, any Intrusion Detection System (IDS) aims at improving the attack detection rate and reducing false alarms. Technically, it is very challenging to achieve a lower false alarm rate in spite of a satisfactory recall percentage. The proposed study demonstrates that the decision forest models are quite robust. A rigorous investigation of all the models considered in the study made some interesting revelations as elaborated in this section. False Alarm Rate (FAR) has been considerably low (<= 10%) with respect to four binary classifiers namely boosted decision tree, decision forest, decision jungle and locally deep SVM as mentioned in Table 8. Particularly, two class decision forest surpassed other classifiers with highest recall rate of 99.8% and lowest FAR of 1% with bagging and 0.7% with replicate as re-sampling techniques respectively as mentioned in Table 8. Although other performance measures have been used to validate the effectiveness of the proposed models, Recall and FAR are the two standard metrics widely employed in intrusion detection research and the remaining metrics serve as supplementary. Two-class locally deep SVM has performed well with seemingly good attack detection rate of 99.3% and false alarm rate as low as 10%. Two-class Bayes point machine (BPM) and Boosted Decision Tree (BDT) models have been consistent in their performance with 97% recall. However, BPM has recorded a higher (12.5%) FAR as compared to BDT model (4.9% as FAR). The FAR reported by Averaged perceptron is seemingly high i.e., 28.9%. There is a substantial difference between the recall percentage of two-class SVM and two-class locally deep SVM (89.6% and 99.3% respectively). Two class SVM's capability to detect false alarms has not been impressive since its FAR is reported to be as high as 16.5%. On the other hand, locally deep SVM has been comparatively better in reducing false alarms due to the application of sigmoid kernel. Two class Logistic regression has been mediocre in its performance with a reasonable recall percentage of 97.6% and apparently a higher FAR of 11.7%.

Network samples in any dataset are not uniformly distributed across various classes and machine learning practitioners often encounter the problem of imbalanced datasets in real time [32]. Binary classification alone may not be insightful because two class algorithms cannot classify the samples into a particular attack type or category. In view of the above mentioned limitation of binary classification, three algorithms were employed to perform multiclass classification tasks. Empirical investigation demonstrated that multiclass decision forest outperformed others in identifying various attack types.The recall percentage of seven classes including normal are quite appealing except Analysis, Backdoor and Denial of Service(DOS) as predicted by multiclass decision forest (as enumerated in confusion matrix 1). On the multiclass classification front, the results obtained from both decision jungle and logistic regressions were trivial. Both these classifiers reported a good recall percentage, i.e., above 90% with respect to only two attack categories like generic and exploits. This can be attributed to the presence of larger samples in

the training set with respect to generic and exploits as observable from Table 1. It is discernible that the sequence of experimentation conducted on Azure Machine Learning Studio supported by an ingenious set of algorithms strengthened the implementation aspect since overall attack detection rate is visibly high and false alarm rate is apparently low. The current study considers substantial samples for experimentation (257,673 network instances inclusive of both training and testing datasets). It is worthwhile to mention that training time of all the eight two-class predictive models was found to be quite minimal as reported in the range of 6 to 9 seconds whereas multiclass classification models took relatively longer to get familiar with different attack categories.

4. CONCLUSION AND PROSPECTS

In this study, eight two-class and three multiclass classification models were developed using UNSW NB-15 dataset. Based on empirical investigation, it can be stated that decision forest accomplished the best performance. Since it is extremely time consuming to execute the experiments on local systems, Microsoft Azure Machine Learning Studio (MAMLS) was chosen for experimentation. Apart from standard performance metrics like accuracy, precision, recall, f1-score and AUC, the proposed work also considered training time and execution time to evaluate the effectiveness of the algorithms. The proposed study has highlighted that MAMLS can serve as an expedient Integrated Development Environment (IDE) for handling large datasets. As a part of future work, it will be interesting to employ different intrusion detection datasets, subsequently gauge the performance of various classifiers. Experts have always urged the research community to experiment with different datasets and introduce novel techniques for network intrusion detection [33, 34]. Another avenue which can be explored in future can possibly include the deployment of predictive models as scalable web services thereby leveraging the capabilities of MAMLS. It will be technically challenging to implement a wrapper based approach on MAMLS. Such wrapper based approaches may be helpful to demonstrate the effectiveness of MAMLS, eventually resuting in a perceptive assessment of its computational performance.

REFERENCES

[1] Hakimi, Zahra, Karim Feaz, Morteza Barati, "A Flow-based Distributed Intrusion Detection System Using Mobile Agents," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 3, no. 6, pp. 732-740, 2013.
[2] Jang-Jaccard J, Nepal S., "A survey of emerging threats in cybersecurity," *Journal of Computer and System Sciences*, vol. 80, no. 5, pp. 973-93, 2014.
[3] Yan F, Jian-Wen Y, Lin C., "Computer network security and technology research," *In 2015 Seventh International Conference on Measuring Technology and Mechatronics Automation*, IEEE, pp. 293-296, 2015.
[4] Yazdani, Navid Moshtaghi, Masoud Shariat Panahi, and Ehsan Sadeghi Poor, "Intelligent Detection of Intrusion into Databases Using Extended Classifier System," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 3, no. 5, pp. 2088-8708, 2013.
[5] Aburomman, Abdulla Amin, and Mamun Bin Ibne Reaz, "Review of IDS Development Methods in Machine Learning," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 6, no. 5, pp. 2432-2436, 2016.
[6] Buczak, Anna L., and Erhan Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153-1176, 2015.
[7] Othman, Suad Mohammed, Fadl Mutaher Ba-Alwi, Nabeel T. Alsohybe, and Amal Y. Al-Hashida, "Intrusion detection model using machine learning algorithm on Big Data environment," *Journal of Big Data*, vol. 5, no. 1, pp. 34, 2018.
[8] Tchakoucht, Taha AIT, and Mostafa Ezziyyani, "Building a fast intrusion detection system for high-speed-networks: probe and DOS attacks detection," *Procedia Computer Science*, vol. 127, pp. 521-530, 2018.
[9] Zuech, Richard, Taghi M. Khoshgoftaar, and Randall Wald, "Intrusion detection and big heterogeneous data: a survey," *Journal of Big Data*, vol. 2, no. 1, pp. 3, 2015.
[10] Suthaharan, Shan, "Big data classification: Problems and challenges in network intrusion prediction with machine learning," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 4, pp. 70-73, 2014.
[11] Nassar, Mohamed, Bechara al Bouna, and Qutaibah Malluhi, "Secure outsourcing of network flow data analysis," *In 2013 IEEE International Congress on Big Data*, IEEE, pp. 431-432, 2013.
[12] Sparks, Evan R., Ameet Talwalkar, Virginia Smith, Jey Kottalam, Xinghao Pan, Joseph Gonzalez, Michael J. Franklin, Michael I. Jordan, and Tim Kraska, "MLI: An API for distributed machine learning," *In 2013 IEEE 13th International Conference on Data Mining*, IEEE, pp. 1187-1192, 2013.
[13] Casado, Ruben, and Muhammad Younas, "Emerging trends and technologies in big data processing," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 8, pp. 2078-2091, 2015.
[14] Zaharia, Matei, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng et al., "Apache spark: a unified engine for big data processing," *Communications of the ACM*, vol. 59, no. 11, pp. 56-65, 2016.

[15] Team, AzureML, "AzureML: Anatomy of a machine learning service," *In Conference on Predictive APIs and Apps*, pp. 1-13. 2016.
[16] Elshawi, Radwa, Sherif Sakr, Domenico Talia, and Paolo Trunfio, "Big data systems meet machine learning challenges: Towards big data science as a service," *Big data research*, vol. 14, pp. 1-11, 2018.
[17] Moustafa, Nour, and Jill Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," *In 2015 military communications and information systems conference (MilCIS)*, IEEE, pp. 1-6, 2015.
[18] Moustafa, Nour, and Jill Slay, "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Information Security Journal: A Global Perspective*, vol. 25, no. 1-3, pp. 18-31, 2016.
[19] Moustafa, Nour, and Jill Slay, "The significant features of the UNSW-NB15 and the KDD99 data sets for network intrusion detection systems," *In 2015 4th international workshop on building analysis datasets and gathering experience returns for security (BADGERS)*, IEEE, pp. 25-31, 2015.
[20] Ribeiro, Mauro, Katarina Grolinger, and Miriam AM Capretz, "Mlaas: Machine learning as a service," *In 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, IEEE, pp. 896-902, 2015.
[21] Tafti, Ahmad P., Eric LaRose, Jonathan C. Badger, Ross Kleiman, and Peggy Peissig, "Machine learning-as-a-service and its application to medical informatics," *In International Conference on Machine Learning and Data Mining in Pattern Recognition*, Springer, Cham, pp. 206-219, 2017.
[22] Dahiya, Priyanka, and Devesh Kumar Srivastava, "Network intrusion detection in big dataset using Spark," *Procedia computer science*, vol. 132, pp. 253-262, 2018.
[23] Khammassi, Chaouki, and Saoussen Krichen, "A GA-LR wrapper approach for feature selection in network intrusion detection," *computers & security*, vol. 70, pp. 255-277, 2017.
[24] Ambusaidi, Mohammed A., Xiangjian He, Priyadarsi Nanda, and Zhiyuan Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *IEEE transactions on computers*, vol. 65, no. 10, pp. 2986-2998, 2016.
[25] Belouch, Mustapha, Salah El Hadaj, and Mohamed Idhammad, "Performance evaluation of intrusion detection based on machine learning using Apache Spark," *Procedia Computer Science*, vol. 127, pp. 1-6, 2018.
[26] Bhamare, Deval, Tara Salman, Mohammed Samaka, Aiman Erbad, and Raj Jain, "Feasibility of supervised machine learning for cloud security," *In 2016 International Conference on Information Science and Security (ICISS)*, pp. 1-5, IEEE, 2016.
[27] Vergara, Jorge R., and Pablo A. Estevez, "A review of feature selection methods based on mutual information," *Neural computing and applications*, vol. 24, no. 1, pp. 175-186, 2014.
[28] Smith, Tony C., and Eibe Frank, "Introducing machine learning concepts with WEKA," *In Statistical genomics*, *Humana Press, New York, NY*, pp. 353-378, 2016.
[29] Shotton, Jamie, Toby Sharp, Pushmeet Kohli, Sebastian Nowozin, John Winn, and Antonio Criminisi, "Decision jungles: Compact and rich models for classification," *In Advances in Neural Information Processing Systems*, pp. 234-242, 2013.
[30] Jose, Cijo, Prasoon Goyal, Parv Aggrwal, and Manik Varma, "Local deep kernel learning for efficient non-linear svm prediction," *In International conference on machine learning*, pp. 486-494. 2013.
[31] Howley, Tom, and Michael G. Madden, "The genetic kernel support vector machine: Description and evaluation," *Artificial intelligence review*, vol. 24, no. 3-4, pp. 379-395, 2005.
[32] Krawczyk, Bartosz, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221-232, 2016.
[33] Moustafa, Nour, Jiankun Hu, and Jill Slay, "A holistic review of Network Anomaly Detection Systems: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 128, pp. 33-55, 2019.
[34] Ring, Markus, Sarah Wunderlich, Deniz Scheuring, Dieter Landes, and Andreas Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, 2019.