Network Anomaly Detection Using LightGBM: A Gradient Boosting Classifier

Md. Khairul Islam[1], Prithula Hridi[1], Md. Shohrab Hossain[1], Husnu S. Narman[2]
[1]Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Bangladesh
[2]Weisberg Division of Computer Science, Marshall University, Huntington, WV, USA
Email: khairulislamtanim@gmail.com, prithula5117@gmail.com, mshohrabhossain@cse.buet.ac.bd, narman@marshall.edu

***Abstract*—Anomaly detection systems are significant in recognizing intruders or suspicious activities by detecting unseen and unknown attacks. In this paper, we have worked on a benchmark network anomaly detection dataset UNSW-NB15, that reflects modern-day network traffic. Previous works on this dataset either lacked a proper validation approach or followed only one evaluation setup which made it difficult to compare their contributions with others using the same dataset but with different validation steps. In this paper, we have used a machine learning classifier LightGBM to perform binary classification on this dataset. We have presented a thorough study of the dataset with feature engineering, preprocessing, feature selection. We have evaluated the performance of our model using different experimental setups (used in several previous works) to clearly evaluate and compare with others. Using ten-fold cross-validation on the train, test, and combined (training and test) dataset, our model has achieved 97.21%, 98.33%, and 96.21% f1_scores, respectively. Also, the model fitted only on train data, achieved 92.96% f1_score on the separate test data. So our model also provides significant performance on unseen data. We have presented complete comparisons with the prior arts using all performance metrics available on them. And we have also shown that our model outperformed them in most metrics and thus can detect network anomalies better.**

***Index Terms*—anomaly detection, machine learning, network security.**

I. Introduction

Web applications are getting increasingly popular, and the Internet has become an important part in our daily life. As a consequence, network systems are being targeted more by attackers [1] with malicious intent [2]. To detect intruders in a network system, there are generally two approaches: signature-based and anomaly-based detection. Signature-based systems maintain a database of previously known attacks and raise alarms when any match is found with the analyzed data. However, they are vulnerable to zero-day attacks.

An anomaly in a network means a deviation of traffic data from its normal pattern. Thus, anomaly detection techniques have the advantage of detecting zero-day attacks. However, in a complex and large network system, it is not easy to define a set of valid requests or normal behavior of the endpoints. Hence, anomaly detection faces the disadvantage of having a high false-positive error rate.

According to Ahmed et al. [3], the main attack types are DoS, Probe, User to Root (U2R), and Remote to User (R2U) attacks. Ahmed et al. [3] mapped the point anomaly with the U2R and the R2U attacks, the DoS attack to the collective anomaly, and the Probe attack to the contextual anomaly.

Machine learning techniques, in many cases they have outperformed the previous state-of-the-art models. As UNSW-NB15 [4] is a benchmark network anomaly detection dataset, numerous studies have been done on it. However, to evaluate the same dataset many different setups were adopted (Section II). For example, train and evaluate on train data [5], [6] . Ten-fold cross-validation on train data [7], [8]), test data [9], combined (train+test) data [10], [11]. Five-fold cross-validation on train and test data [12] . Train on train data and evaluate on test data [13], [14].

With so many different experimental setups, it is difficult to find the single best work on this dataset. Moreover, works that followed the same experimentation setup did not compare their results with prior works in some cases (for example, Kanimozhi et al. [6] and Nawir et al. [10] did not compare their results with Koroniotis et al. [11]). Therefore, it is difficult to validate improvements. Mogal et al. [5] and Kanimozhi et al. [6] mentioned near-perfect detection scores. However, they did not mention the significant technical flaws regarding their approaches, which we have explained in Section IV-B. Some other works [6], [10], [12] followed only one validation setup. Hence, it is impossible to compare those works with the ones, which have worked on the same dataset but with different validation setups.

The novelty and contributions of this work are as follows:

- We have provided a thorough study of the UNSW-NB15 dataset with feature engineering, preprocessing, selection.
- We have explored the performance of a boosting algorithm in binary classification on the dataset following all experimentation setups found from prior studies whereas each of the previous works focused on only one setup.
- We have compared our results to prior state-of-the-art works using all related performance metrics.

Our results show that feature engineering can make the model more generalized. So our model performance improved in cross-validation experiments, as well as when evaluated on separate test data. We have also shown a very small false alarm rate (1.83% - 4.81%). Our work can help in detecting unseen anomaly attacks better having very few false alarms. And Our different experimentation setups will help visualize the impact of validation strategies on the model performance of this dataset.

The rest of the paper has been organized as follows. SectionII describes the recent works related to NIDS (Network Intrusion Detection Systems) on the UNSW-NB15 dataset. Our proposed methodology has been explained in Section III. Section IV describes the experimentation setups and our results as well as some comparisons with the prior state-of-the-art. The rest of the comparisons regarding evaluating on train and test data, cross-validation approaches have been shown in Section V. Finally, SectionVI has the concluding remarks.

II. Related works

For network intrusion detection KDDCUP99, NSL-KDD, DARPA, UNSW-NB15 are among the benchmark dataset. As a popular dataset, we focus on binary classification of the UNSW-NB15 dataset [4] which is used in several anomaly detection works. Based on the model evaluation process, we have divided them into several parts.

A. Random train test

Moustafa et al. [15] used central points of attribute values and Association Rule Mining for feature selection on a high level of abstraction from datasets UNSW-NB15 and NSL-KDD. They have partitioned the datasets into train and test sets following an equation. Then evaluated performance using Expectation-Maximisation clustering (EM), Logistic Regression (LR), and Nave Bayes (NB). Moustafa et al. [16] also proposed a beta mixture model-based anomaly detection system on the UNSW-NB15 dataset. They first selected eight features from the dataset, then randomly selected samples from it. In another work, Moustafa et al. [17] selected random samples from the UNSW-NB15 dataset and ran ten-fold cross-validation on it.

B. Validation on same data used for training

Mogal et al. [5] used machine learning classifiers on both UNSW-NB15 and KDDCUP99 datasets. They achieved nearly 100% accuracy on both datasets using Naive Bayes and Logistic Regression on train data. Kanimozhi et al. [6] choose the best four features of the UNSW-NB15 dataset using the RandomForest classifier. They also used a Multi-Layer Perceptron to show how neural networks would perform on this dataset.

C. Cross validation

Koroniotis et al. [11] selected the top ten features of the UNSW-NB15 combined (train+test) dataset using Information Gain Ranking Filter. Then they ran ten-fold cross-validations using machine learning techniques. Among the techniques applied, DT (Decision Tree C4.5 Classifier) performed the best at distinguishing between Botnet and normal network traffic.

Suleiman et al. [7] explored the performance of machine learning classifiers on benchmark and new datasets (UNSW-NB15, NSL-KDD, and Phishing) using ten-fold cross-validation. They found the RandomForest classifier performs the best. Nawir et al. [10] applied ten-fold cross-validation on the binary classification of the combined (train+test) dataset by using the WEKA tool. They also compared centralized and distributed AODE algorithms based on accuracy against the number of nodes.

Meftah et al. [8] applied both binary and multiclass classification on the UNSW-NB15 dataset. They found that for binary classification SVM performs the best in ten-fold cross-validation and decision tree (C5.0) for multiclass classification. Hanif et al. [9] used ANN(Artificial Neural Network) on the same dataset. They compared their performance with prior works on the NSL-KDD dataset, instead of works on the same dataset. Meghdouri et al. [12] applied feature preprocessing and principal component analysis on the UNSW-NB15 dataset. Then performed five-fold cross-validation using a RandomForest classifier.

D. Validation on separate test data

Moustafa et al. [14] analyzed the statistical properties of the UNSW-NB15 dataset and showed that it is more complex compared to the KDD99 dataset. Vinaykumar et al. [18] used classical machine learning classifiers, and deep neural networks on several intrusion detection datasets. The classical models performed much better than the neural network models. Dahiya et al. [19] applied feature reduction techniques on both larger and smaller versions of the UNSW-NB15 dataset. Bhamare et al. [13] tested the robustness of machine learning models in cloud scenarios. They trained classifiers on the UNSW-NB15 dataset and tested them on a cloud security dataset ISOT. They found that these models did not perform well in the cloud environment.

E. Gap analysis

To the best of our knowledge, there has been no work that has provided a thorough study of the UNSW-NB15 dataset with feature engineering to improve results. Moreover, most of the previous works on this dataset focused on only one evaluation process. So it is difficult to make a proper comparison among them. For example, the performance achieved in five-fold cross-validation, can not be compared with that achieved in ten-fold cross-validation. So, we have evaluated our model performance using all possible experimentation setups found in prior arts and provided a thorough comparison with prior state-of-the-art techniques. We have also used feature engineering to reduce overfitting, thereby providing a more generalized model.

III. Proposed Methodology

We have targeted only to perform binary classification on the dataset. We have used Kaggle kernels for running our models. It provided us with 4 CPU cores, 16 Gigabytes of RAM when this work was done. In the following subsections, we have described how the dataset was prepared for experimentation and the performance metrics used for evaluation.

A. Dataset Description

We have used the UNSW-NB15 dataset [4] which is a recent benchmark dataset for NIDS (Network Intrusion Detection

TABLE I
UNSW-NB15 DATASET DESCRIPTION

Type	Train	Test
Normal	56,000	37,000
Anomaly	119,341	45,332
Total	**175,341**	**82,332**

Systems). The dataset was created at the Cyber Range Lab of the Australian Center of Cyber Security. Compared to other existing datasets (such as KDDCup99, NSL-KDD, DARPA), the UNSW-NB15 dataset is more recent and better reflects modern network traffic. UNSW-NB15 represents nine major families of attacks by utilizing the IXIA PerfectStorm tool. The main data set contains 2,540,044 observations. The authors divided a part of this data set was divided into train and test sets, which has been used in this work. The dataset description is shown in Table I. We have considered binary classification for this study. Hence, we have only predicted whether the record is attack type or normal. The dataset labels class 0 for normal and 1 for attack records. From Table I we can see the train data is imbalanced. The majority of the records are anomalies. However, in the test data, they are nearly balanced.

In the following list, we have described the 43 features used by us from the UNSW-NB15 dataset. Similar features for source and destination are described together.

- proto : Transaction protocol.
- state: The state and its dependent protocol
- dur : Record total duration.
- sbytes & dbytes: Source to destination and destination to source transaction bytes.
- sttl & dttl: Live value of source to destination and destination to source time.
- sloss & dloss: Source and destination packets retransmitted or dropped.
- service: http, ftp, smtp, ssh, dns, ftp-data, irc and (-) if not much-used service.
- sload & dload: Source and destination bits per second.
- spkts & dpkts: Source to destination and destination to source packet count.
- swin & dwin : Source and destination TCP window advertisement value.
- stcpb & dtcpb: Source and destination TCP base sequence number.
- smean & dmean: Mean of the packet size transmitted by the source and destination.
- trans_depth: The pipelined depth into the connection of http request/response transaction.
- response_body_len: The actual uncompressed content size of the data transferred from the servers http service.
- sjit & djit: Source and destination jitter (mSec).
- rate: a feature based on record start and end time.
- sinpkt & dinpkt : Source and destination interpacket arrival time (mSec).
- synack: The time between the SYN and the SYN_ACK packets.
- ackdat: The time between the SYN_ACK and the ACK packets.
- tcprtt: The sum of synack and ackdat.
- is_sm_ips_ports: If the source and destination IP addresses are equal and port numbers equal then, this variable takes value 1 else 0.
- is_ftp_login: If the ftp session is accessed by user and password then 1 else 0.

The following features are calculated in the last 100 connections from the current record.

- ct_state_ttl: According to specific range of values of source/destination time to live for each stat .
- ct_flw_http_mthd :Number of flows that has Get and Post methods in http service.
- ct_ftp_cmd: Number of flows that have a command in ftp session.
- ct_srv_src & ct_srv_dst: Number of connections that contain the same address for service and source or destination.
- ct_src_ltm & ct_dst_ltm: Number of connections of the same source and destination address.
- ct_src_dport_ltm & ct_dst_sport_ltm: Number of connections of the same source address and the destination port or same destination address and the source port.
- ct_dst_src_ltm: Number of connections of the same source and the destination address.

And our target column is 'label', which is 0 for normal and 1 for attack records.

B. Preprocessing

We have performed the preprocessing on the data set using the following steps :

1) Feature engineer categorical columns: We have found many categorical labels to have a very low frequency. To make it easier for the model to learn from these categorical features, labels with low frequency were converted into a single label.

- For state feature, except the top five labels by frequency ('FIN', 'INT', 'CON', 'REQ', 'RST') other labels were converted into label 'others'.
- For service columns, labels except '-', 'dns', 'http', 'smtp', 'ftp-data', 'ftp', 'ssh', 'pop3' were converted into 'others' labels.
- For the proto column, 'igmp', 'icmp', 'rtp' labels were combined into the label 'igmp_icmp_rtp'. Then labels except 'tcp', 'udp', 'arp', 'ospf', 'igmp_icmp_rtp' were converted into label 'others'.

Before this, test data had new categorical values present. However, after this feature engineering, categorical value sets for train and test data became the same. That enabled us to use one-hot encoding on the dataset. If any categorical value present in test data was never present in train data. Then in one-hot encoding, there will be no column for it in the train, but it will in test data. So the column mismatch will make the prediction difficult.

TABLE II
FEATURE IMPORTANCE

Feature	Importance	Feature	Importance
sttl	16.53	sjit	1.7
ct_state_ttl	11.06	dloss	1.22
dload	7.2	proto	1.21
dttl	4.93	djit	1.14
dmean	4.19	sloss	0.9
ackdat	3.8	ct_src_ltm	0.83
rate	3.79	ct_dst_ltm	0.83
dinpkt	3.51	stcpb	0.81
sbytes	3.23	ct_dst_sport_ltm	0.8
smean	2.85	dtcpb	0.75
sload	2.72	swin	0.62
state	2.71	is_sm_ips_ports	0.57
dpkts	2.59	ct_src_dport_ltm	0.57
tcprtt	2.49	service	0.51
ct_srv_dst	2.49	spkts	0.47
ct_dst_src_ltm	2.43	ct_flw_http_mthd	0.17
sinpkt	2.41	response_body_len	0.16
ct_srv_src	2.2	trans_depth	0.14
dbytes	1.93	dwin	0.02
synack	1.76	ct_ftp_cmd	0.01
dur	1.76	is_ftp_login	0.01

2) Scaling: We have applied StandardScaler from the scikit-learn preprocessing library on all non-categorical features. It was fitted on the train data, then the fitted scaler was used to convert both train and test data. It converts values using the following equation:

$$x = \frac{x - \mu}{\sigma} \tag{1}$$

where μ is the mean value and σ is the standard deviation.

3) Feature Selection: We have used the RandomForest classifier of sklearn with default parameters to calculate feature importance on the train dataset. We have first preprocessed the dataset using previous steps. Then averaged feature importance over ten-fold cross-validation. We have converted the values into percentages, for easier understanding. Then sorted them in descending order. From there we have chosen to drop features with less than 0.5% importance value. The dropped 7 features are response_body_len, spkts, ct_flw_http_mthd, trans_depth, dwin, ct_ftp_cmd, is_ftp_login. In Table II we have shown the chosen features with corresponding importance.

4) OneHotEncoding: We have used the pandas library to OneHotEncode all the categorical features. It became possible as after using feature engineering, categorical value sets became the same in train and test datasets. The final number of features in our dataset is 53.

C. Evaluation metrics

In this section, we have discussed the performance metrics we have used in our experiments on the UNSW-NB15 dataset [4]. We have also used them to compare the performance of our approach with previous works.

1) Accuracy: It is the ratio of the number of correct predictions to the total number of input samples.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

2) Precision: It is the ratio of the number of correct positive results to the number of positive results predicted by the classifier.

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

3) Recall or Detection Rate or True Positive Rate): It is the ratio of the number of correct positive results to the number of all positive samples.

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

4) F1_score: The harmonic means of precision and recall.

$$f1\_score = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}} \tag{5}$$

5) False Positive Rate (FPR): It is the proportion of incorrectly identified observations.

$$FPR = \frac{FP}{FP + TN} \tag{6}$$

6) False Alarm Rate (FAR): The probability that a record gets incorrectly classified.

$$FAR = \frac{FP + FN}{FP + FN + TP + TN} \tag{7}$$

7) ROC AUC: It computes the Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.

IV. EXPERIMENT AND RESULTS

For evaluating the UNSW-NB15 dataset, we have performed ten-fold cross-validation using Stratified KFold of the sklearn library with a random shuffle set to true. We have used several popular machine learning classifiers to measure the prediction performance. The models were run mostly with default parameters. We have set the random state to 1 for all of them so that the results are reproducible. All models except LightGBM [20], were from sklearn library version 0.23.0. During prediction, for LightGBM we used the best iteration. The used models are listed below with their important parameters.

1) LogisticRegression : penalty = l2, max_iter = 100
2) GradientBoosting: learning_rate = 0.1, n_estimators = 100, max_depth = 3

TABLE III
TEN-FOLD CROSS VALIDATION WITH DIFFERENT MODELS

Metrics(%)	Accuracy(%)	F1_score(%)
LogisticRegression	93.54	95.42
GradientBoosting	94.58	96.11
DecisionTree	94.99	96.32
RandomForest	96.08	97.14
LighGBM	96.18	97.21

3) DecisionTree: criterion = 'gini', max_depth = None, max_features = None,
4) RandomForest: n_estimators = 100, criterion = 'gini', max_depth = None, max_features = None
5) LightGBM: learning rate=0.1, metric=binary_logloss, num_round = 2000, early_stopping_rounds = 50.

The result of this experiment is shown in Table III. We have chosen the best model based on f1-score and accuracy. Here class labels are 0 for normal and 1 for attack records. So these metrics are the best choices to validate the model performance. As shown in Table III, LightGBM achieved the best performance in both accuracy (96.18%) and f1-score (91.21%). LightGBM (Ke et al. [20])is a highly efficient gradient boosting framework that uses tree-based learning algorithms. It follows a more complex leaf-wise split approach rather than a level-wise approach. Which reduces overfitting and improves the validation results.

A. *Handling class imbalance*

From Table I we can see the train dataset is slightly imbalanced. The ratio of normal and anomaly records is 56: 119. We used is_unbalance and scale_pos_weight parameters provided by LightGBM to test whether handling class imbalance will improve results. If is_unbalance is set to true, LightGBM will automatically try to balance the weight of the dominated label. Using scale_pos_weight, we can manually set weight for the positive class. During ten-fold cross-validation in our experimentation, we found using these parameters decreases the f1_score. So we have not used them finally. However, during predicting on separate test data, we have found setting is_unbalance to true improves the prediction performance slightly.

B. *Validation on same data used for training*

Mogal et al. [5], Kanimozhi et al. [6] evaluated model performance on the UNBSW-NB15 dataset without using any cross-validation approach. The same data used for training the model was used for validation too. To compare our model's performance with them, we have followed a similar setup. As evident from the results shown in Table IV, this experimentation setup does not truly reflect model performance. As the model overfits on train data, its performance will be very poor on a separate test set. For example, we have found our model when overfitted on train data, only achieved 86.88% accuracy

TABLE IV
EVALUATING MODEL ON DATA USED FOR TRAINING

Metrics(%)	Train	Test
Accuracy	99.60	99.98
Precision	99.52	99.97
Recall	99.89	99.98
F1_score	99.71	99.98
FPR	0.01	0.0004
AUC	99.99	99.99
Time(s)	243	237

TABLE V
TEN-FOLD CROSS VALIDATION

Metrics(%)	Train	Test	Combined
Accuracy	96.18	98.18	95.19
Precision	96.54	98.87	96.84
Recall	97.89	97.80	95.58
F1_score	97.21	98.33	96.21
FPR	7.47	1.37	5.51
FAR	3.82	1.83	4.81
AUC	99.44	99.81	99.26
Time(s)	628.1	281.1	838.8

and 89.14% f1_score on test data. So the models proposed by both of those prior works should not be used in reality.

C. *Ten-fold cross validation*

Ten-fold cross-validation on train, test or combined(train+test) dataset was performed by Meftah et al. [8], Suleiman et al. [7], Nawir et al. [10], Hanif et al. [9] . We have used the StratifiedKFold method of sklearn.model_selection module with shuffle enabled to perform the ten-fold cross-validation. Average scores achieved in that process are shown in Table V. Interestingly we see cross-validation on test data has the best results. This can be because the test data is more balanced.

D. *Validation on test data*

In this experiment, we have validated the model trained on train data using the separate test dataset of UNSW-NB15 following [13], [14], [18], [19]. As Meftah et al. [8] mentioned, some columns have new labels in test data. However, after our feature engineering process in section III-B, we were able to overcome it. For this evaluation specifically, we have found that setting parameters is_unbalance to True and learning rate to 0.05 in LightGBM improved prediction performance. The results are shown in Table VI along with comparisons with prior arts. Our FAR and AUC scores for this case are 8.05% and 98.67%. Our model outperforms the work of Vinayakumar et al [18] by both accuracy and f1_score. Though Dahiya et al

TABLE VI
VALIDATION ON TEST DATA

Metrics(%)	Ours	RF [18]	REP Tree [19]
Accuracy	91.95	90.3	93.56
Precision	89.59	98.8	83.3
Recall	96.60	86.7	83.2
F1_score	92.96	92.4	83.25
FPR	13.75	-	2.3

TABLE VII
PERFORMANCE COMPARISON WITH [7]

Metrics(%)	RF [7]	LightGBM
Accuracy	90.14	96.17
Precision	99.8	96.54
Recall	97.8	97.89
F1_score	98.7	97.20
FPR	0.10	7.48

[19] achieved better accuracy than ours, they had near a 10% drop in f1_score than our model. In an intrusion detection dataset where class distribution is imbalanced, f1_score is more important.

V. COMPARISON WITH STATE-OF-THE-ART MODELS

In this section, we have compared our model performance with prior state-of-the-art models on the same dataset. We have arranged this section into subsections based on different experimentation setups that were followed in those works.

A. Evaluation on train data

Mogal et al. [5] achieved 99.96% accuracy on the UNSW-NB15 dataset using Naive Bayes and Logistic Regression, which did not follow any cross-validation approach. A similar approach was taken by Kanimozhi et al. [6] with the best four features chosen using the RandomForest classifier. The model achieved 98.3% accuracy. We have shown in Table IV that in the same validation process, our model has achieved near-perfect scores on both train and test data.

B. Ten-fold cross validation

Suleiman et al. [7] evaluated performance using ten-fold cross-validation on train data. They found the Random Forest classifier to have the best accuracy and f1_score. We have mentioned our model performance using the same validation process in the train column of Table V. TPR and recall are the same. Hence, we have mentioned only recall.

Meftah et al. [8] applied ten-fold cross-validation on the train dataset and achieved the best accuracy of 82.11% using the SVM classifier. In the same validation process, our model accuracy is 96.17%. Hanif et al. [9] applied ten-fold cross-validation on the train and test dataset repeatedly using Artificial Neural Network(ANN) and achieved an average 84% accuracy, 8% false-positive rate. In a similar case, our model performance is better, 96.18% accuracy and 7.47% FPR as shown in Table V. Though Meftah et al. [8] and Hanif et al. [9] followed the same experimentation setup similar to Suleiman et al. [7], none of them presented any comparison with it.

TABLE VIII
COMPARISON OF OUR MODEL WITH KORONIOTIS ET AL. [11]

Classifier	Accuracy (%)	FAR(%)
Decision Tree [11]	93.23	6.77
LightGBM	95.19	4.81

Koroniotis et al. [11] performed ten-fold cross-validation on the combined dataset. The best result was achieved using the Decision Tree C4.5. In Table VIII we have shown the comparison. Koroniotis et al. [11] presented model performance with two metrics only, accuracy and FAR (False Alarm Rate). Our model has shown better performance in both of them.

Nawir et al. [10] applied a similar ten-fold cross-validation evaluation on the combined (train + test) dataset, using the WEKA J48 classifier. They have mentioned achieving high accuracy of 98.71% using the default parameter. However, using exactly the same environment for multiple runs we have found that is not true. It achieves around 94.6% accuracy on average. That is lower than ours (95.19% accuracy).

C. Five-fold cross validation

We have found only Meghdouri et al. [12] to validate using five-fold cross-validation. Also, they have not mentioned any specific reason to not use ten-fold cross-validation like prior works. No performance comparison was also presented. Here we have not added any separate section for this. We have presented our model performance using same validation process in Table IX and X. Table IX shows our model performance compared to theirs on five-fold cross-validation of the train dataset. Their model achieved higher accuracy (99%) compared to ours (96.18%). However, for precision, recall, and f1_score our model performance is much higher. Using the same validation process on the test dataset, from Table X, our test accuracy is very close to theirs. However, as before our precision, recall and f1_score are much better than theirs. Our ROC-AUC scores are very close too. For intrusion detection techniques f1_score is very important, in which our model outperforms them by a large margin.

D. Validation on separate test data

Bhamare et al. [13] achieved accuracy 89.26%, 93.7% TP and 95.7% TN at prediction threshold 0.5. Increasing the prediction threshold to 0.7-0.8 their TPR improved to 97%, but TN dropped to 80%. Where our accuracy, TP, and TN are 91.95%, 97%, and 86% at threshold 0.5. Moustafa et al. [14] achieved 85.56% accuracy and 15.78% FAR using the Decision Tree technique built-in Visual Studio Business Intelligence 2008 with the default input parameters. Our model

TABLE IX
COMPARISON WITH MEGHDOURI ET AL. [12] (TRAIN DATA)

Metrics(%)	Train [12]	Train
Accuracy	99.0	96.18
Precision	85.9	96.56
Recall	85.1	97.87
F1_score	84.9	97.21
ROC AUC	99.8	99.43

TABLE X
COMPARISON WITH MEGHDOURI ET AL. [12] (TEST DATA)

Metrics(%)	Test [12]	Test
Accuracy	98.9	98.08
Precision	84.9	98.79
Recall	85.1	97.7
F1_score	84.9	98.24
ROC AUC	99.8	99.81

accuracy is 91.95% and 8.05% FAR, which outperforms them in this validation setup.

E. Results summary

The followings are the summary of our results:

- Feature engineering can make the model more generalized and improve performance on separate test data.
- Nearly 17 features have the importance of less than 1%.
- Our model can better predict network anomaly than normal records. This is due to the presence of more anomalies in the dataset than normal.

VI. CONCLUSION

In this paper, we have presented a boosting algorithm-based model for performing binary classification of the UNSW-NB15 dataset. Different experimentation setups were followed to compare our performance with prior works. Results show that our model outperforms state-of-the-art works in most metrics. We have shown why the experimental setups followed by some prior works are heavily overfitted and should be avoided. Even when using a different cross-validation approach, our model outperforms most prior arts. Our model is also found to perform well on test data when it is fitted on train data only, validating the generalization of our model. So we believe this will help the network security community in improving anomaly detection. This study only performs a binary classification. However, our proposed algorithm can be easily adapted to multiclass-classification by changing the LightGBM objective parameter to 'multiclass'. In the future, we intend to improve the performance of multiclass-classification on this dataset in a similar way.

REFERENCES

[1] M. I. Ashiq, P. Bhowmick, M. S. Hossain, and H. S. Narman, "Domain flux based dga botnet detection using feedforward neural network," in *IEEE Military Communications (MILCOM)*. Norfolk, VA, USA: IEEE, 12-14 Nov., 2019.

[2] M. A. Jonas, R. Islam, M. S. Hossain, H. S. Narman, and M. Atiquzzaman, "An intelligent system for preventing ssl stripping-based session hijacking attacks," in *IEEE Military Communications (MILCOM)*. Norfolk, VA, USA: IEEE, 12-14 Nov., 2019.

[3] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[4] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Military communications and information systems conference (MilCIS)*. Canberra, Australia: IEEE, 10-12 Nov. 2015, pp. 1–6.

[5] D. G. Mogal, S. R. Ghungrad, and B. B. Bhusare, "Nids using machine learning classifiers on unsw-nb15 and kddcup99 datasets," *International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE)*, vol. 6, no. 4, pp. 533–537, 2017.

[6] V. Kanimozhi and P. Jacob, "Unsw-nb15 dataset feature selection and network intrusion detection using deep learning," *International Journal of Recent Technology and Engineering*, vol. 7, pp. 443–446, 01 2019.

[7] M. Suleiman and B. Issac, "Performance comparison of intrusion detection machine learning classifiers on benchmark and new datasets," in *28th International Conference on Computer Theory and Application*, 10 2018, pp. 447–489.

[8] S. Meftah, T. Rachidi, and N. Assem, "Network based intrusion detection using the unsw-nb15 dataset," *International Journal of Computing and Digital Systems*, vol. 8, no. 5, pp. 478–487, 2019.

[9] S. Hanif, T. Ilyas, and M. Zeeshan, "Intrusion detection in iot using artificial neural networks on unsw-15 dataset," in *16th International Conference on Smart Cities: Improving Quality of Life Using ICT & IoT and AI)*. IEEE, 2019, pp. 152–156.

[10] M. Nawir, A. Amir, N. Yaakob, and O. B. Lynn, "Effective and efficient network anomaly detection system using machine learning algorithm," *Bulletin of Electrical Engineering and Informatics*, vol. 8, no. 1, pp. 46–51, 2019.

[11] N. Koroniotis, N. Moustafa, E. Sitnikova, and J. Slay, "Towards developing network forensic mechanism for botnet activities in the iot based on machine learning techniques," in *International Conference on Mobile Networks and Management*. Springer, 2017, pp. 30–44.

[12] F. Meghdouri, T. Zseby, and F. Iglesias, "Analysis of lightweight feature vectors for attack detection in network traffic," *Applied Sciences*, vol. 8, no. 11, 2018.

[13] D. Bhamare, T. Salman, M. Samaka, A. Erbad, and R. Jain, "Feasibility of supervised machine learning for cloud security," in *International Conference on Information Science and Security*. Pattaya, Thailand: IEEE, 2016, pp. 1–5.

[14] N. Moustafa and J. 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.

[15] ——, "A hybrid feature selection for network intrusion detection systems: Central points," *arXiv preprint arXiv:1707.05505*, 2017.

[16] N. Moustafa, G. Creech, and J. Slay, "Anomaly detection system using beta mixture models and outlier detection," in *Progress in Computing, Analytics and Networking*. Springer, 2018, pp. 125–135.

[17] N. Moustafa, J. Hu, and J. Slay, "A holistic review of network anomaly detection systems: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 128, pp. 33–55, 2019.

[18] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, 2019.

[19] P. Dahiya and D. K. Srivastava, "Network intrusion detection in big dataset using spark," *Procedia computer science*, vol. 132, pp. 253–262, 2018.

[20] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in neural information processing systems*, 2017, pp. 3146–3154.