Attack and Anomaly Detection in IoT Sites Using Boosting Based Algorithms

Sayma Akter

Queen Mary University of London
email: saymaakter.cse@gmail.com

Abstract—**Detection of attacks and anomalies in Internet of Things (IoT) infrastructure is a growing concern in the IoT domain. As the use of IoT infrastructure increases in every domain, threats and attacks in this infrastructure also grow proportionally. In this paper, three different boosting-based algorithms are applied to the anomalous data set and compared with the supervised Random Forest model in the same data set. The main objective of this paper is to overcome the problem of overfitting in case of the Random Forest model application on the anomalous dataset using three different boosting based algorithms and achieving the highest accuracy compared to Random Forest without overfitting. The three boost-based algorithms used here are Adaboost, GradientBoost, and XGBoost. The machine learning supervised algorithm used here for comparison with the boosting algorithms is Random Forest. The evaluation metrics used in the performance comparisons are accuracy, precision, recall, f1 score, and five fold testing mean values. Random Forest provides greater accuracy but with overfitting issues for imbalanced datasets. On the other hand, the system achieves higher training and testing accuracy for boostig based algorithms than Random Forest. This experiment shows that the boosting-based algorithms performs relatively better than Random Forest for dataset with anomalies without overfitting.**

Keywords: Anomaly, Random Forest,AdaBoost, GradientBoost, XGBoost.

I INTRODUCTION

Anomaly detection methods are widely used to identify anomalous observations in the data.Anomaly detection is a crucial part of intrusion detection, in which changes in normal behaviour indicate the presence of malicious or unintentional attacks, faults, or defects [1]. However, using supervised algorithm to detect outliers is not trivial, as outliers in data usually make up a small proportion of their data set. Outlier detection, unlike traditional classification methods, often fails to provide fundamental truths. For the supervised algorithm, a highly imbalanced data set and insufficiently labeled data have limited its generalizability.

Boosting algorithms have been developed over the years as a way to detect outliers. This method specializes in exploring information associated with outliers, such as local density, global correlation, and hierarchical relationships for labeled data. Scientists have carried out a lot of research in order to improve ensemble algorithms. Different improved boosting algorithms have been applied based on the distribution pattern of data, and then performance validation has been carried out, in order to achieve better accuracy. Nikunj C. Oza and Stuart Russell have proposed an average version of AdaBoost in which the distribution pattern of samples after each iteration is compared to the distribution pattern after the previous iteration in the paper named online bagging and boosting [2].

Random Forest is supervised learning algorithms. The "Forest" they create is an ensemble of decision trees, usually trained using the "bagging" method. The idea is that using a combination of learning models improves the final result. As it uses a rule-based approach, there is no need to normalize the data because it is flexible to both classification and regression problems [3]. It also works with both categorical and continuous values. Even with these advantages, a Random Forest algorithm does have some complexities. It requires a great deal of computational power and resources since it builds numerous trees in order to combine their outputs. In addition, it requires a long time for training since it combines a lot of decision trees to determine the class. Due to the ensemble of decision trees, it lacks interpretability and doesn’t determine the significance of each variable [3]. Boosting algorithm was first proposed by Schapire, and then improved by Freund [4], is an ensemble learning algorithm that firstly initializes each sample with the same probability. Secondly, after completing the training of the neural network, adjust the probability of each sample dynamically based on the learning error of samples in the neural network, by which the misclassified samples judged by the already trained neural network will be included in the training sets for the next neural network with a high probability. Boosting algorithms work by augmenting the existing data model result and helping to correct errors. They use the concept of the weak learner and strong learner by using weighted average values and a higher score for predictions. The Boosting Algorithm creates one strong prediction rule out of each weak learning algorithm [4]. They are based on decision stamping, margin and classification. Some examples of boosting algorithms are: **AdaBoost**, **GradientBoost**, and **XGBoost**. Figure 1 shown basic structure of bagging and boosting method.

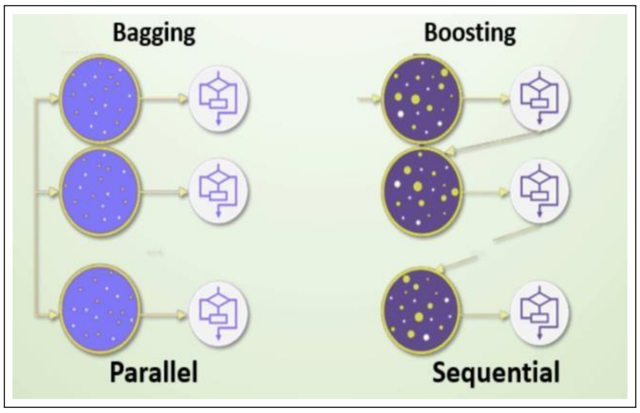


Fig. 1. Bagging and Boosting methods

Some boosting-based research papers have been described here. Firstly, In 2021, a research paper titled as “Anomaly Detection of Dust Removal System through Gradient Boosting Decision Tree Algorithm” by Tao Yang, Liang Chen, Jigang Wang, Zenghao Cui Used clustering to convert unsupervised problems into supervised, and the GradientBoost algorithm can handle various types of features well [4]. While the role of features is only used to split nodes, the data cannot be normalized. Also,due to the small sample size of the dataset used, which led to a lack of sufficient generalization ability. Secondly, how we can select a unique classifier for AdaBoostM1 could be a upcoming research issue in the paper titled “Spam Classification Using Adaptive Boosting Algorithm” [1]. Again, since XGBoost works in transformed outlier scores (TOS) selection method describrd in XGBoost paper of Yue Zhao, there were some difficulties regarding TOS selection. Lastly, research paper titled as “Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches” achieved 0.99 percent accuracy for Random Forest but it had overfitting problem like good training accuracy and poor testing accuracy [2]

The motivation of my work is to detect anomalies using boosting based algorithms without overfitting problem like Random Forest algorithm since it generates and combines a lot of decision trees to determine the class. To achieve this purpose, I used three different boosting based algorithms and compare with Random Forest. Finally, after result analysis, I can see that the same training and testing accuracies for three different boosting algorithms have achieved. As a result, we can overcome the overfitting problem like Random Forest.

In my research paper it overcomes the overfitting problem using boosting algorithms with maximum accuracy. Data scientists refer to this as overfitting, which happens when a statistical model matches its training data exactly. Unfortunately, when this occurs, the algorithm is unable to accurately execute on unseen data, negating its goal. If the training data has a low error rate or higher training accuracies and the test data has a high error rate or lower testing accuracies, it signals overfitting. Random forests have been observed to overfit for the dataset used in this project with noisy classification/regression tasks. When use random Forest classifier over DAD dataset,a large number of trees may make the algorithm slow for result prediction. For the dataset including categorical variables with different number of levels, random forests are biased in favor of those attributes with more levels. Therefore, the variable importance scores from random forest are not reliable for this type of dataset. Since the data contain groups of correlated features of similar relevance for the output, that smaller groups are favored over larger groups. Hence, Random Forest overfits. Since boosted trees are derived by optimizing an objective function, basically GradientBoosting can be used to solve almost all objective function that we can write gradient out. This includes tasks that require RF, such as ranking and position regression. Moreover, Random Forest classifier have higher training accuracies but lower testing accuracies for each five folds cross validaions. Alternatively, boosting based classifiers have

[1]The article is available at https://ieeexplore.ieee.org/document/

[2]This article is available at https://www.sciencedirect.com/science/article and is submitted to the "Internet of Things" journal on 2019.

almost same accuracies for both train and test folds. This work provides a considerably more extensive description of the dataset. It also explains the procedures involved in preprocessing the dataset. The paper focuses on multiclass classification, which is more difficult than binary class classification. Also, this work used a highly imbalanced dataset. Finally, this work provides a detailed description for each classifier.

II Dataset and Method

The overall framework is composed of several independent processes. Figure 2 shows the overall framework.

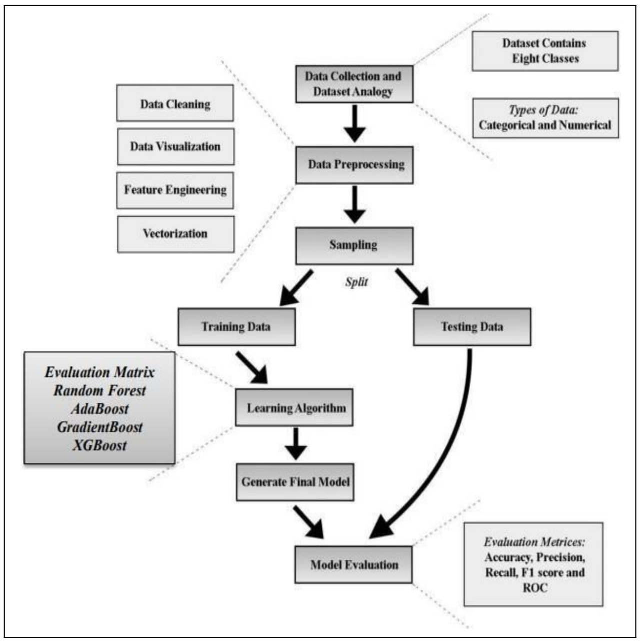

Data Collection and Dataset Analogy
Dataset Contains Eight Classes
Types of Data: Categorical and Numerical
Data Cleaning
Data Visualization
Feature Engineering
Vectorization
Data Preprocessing
Sampling
Split
Training Data
Testing Data
Evaluation Matrix Random Forest AdaBoost GradientBoost XGBoost
Learning Algorithm
Generate Final Model
Model Evaluation
Evaluation Metrices: Accuracy, Precision, Recall, F1 score and ROC

Fig. 2. Workflow diagram of model evaluation

The first process consists of the collection and observation of the dataset. This process, which takes considerable time, is carried out with meticulous care. Also, preprocessing steps were applied to the dataset, such as cleaning, visualizing, oversampling and under sampling. These steps converted the data into feature vectors. The training and testing set of these feature vectors were then split in 80–20 ratio for use in Learning Algorithm and a final model was produced by optimizing the different strategies like Random Forest, AdaBoost, GradientBoost and XGBoost classifiers. Evaluation of the final model was conducted based on different evaluation metrics.In Figure 2 shows the working method of the whole process in this project.

II-A Source of dataset

In table I the statistical summary of the whole DAD dataset shown. The source of the dataset is the DAD repository-Dataset for Anomaly Detection in IoT Network [3]. In this work DAD is presented as a complete and labeled IoT dataset with sufficient trace size, diverse anomaly scenarios and concrete feature extraction which can be used for the detection of traffic anomalies in IoT sensor networks.

II-B Description of dataset

DAD is an open source, complete and labelled IoT dataset. The dataset was obtained from temperature sensors based on NFC smart passive sensor technology, which approximated a real environment. A virtual infrastructure consists of five virtual machines, a MQTT broker, and four client nodes, each with four refrigeration sensors connected to the internal IoT network. DAD presents a seven day network activity with three types of anomalies: duplication, interception, and modification on the MQTT messages over 5 days. At last, a feature description is made, which can then be used for various types of automatic classification or prediction. DAD contains a total of 101,583 packets. It presents TCP traffic on the transport layer and MQTT as the IoT protocol. The 96.9 percent of the traffic corresponds to TCP packets, and 3.4 percent to UDP traffic. The 63.3 percent of the total are MQTT packets and 16 percent of these packets are anomalies.

The node can be modified in one of three ways to implement the traffic anomalies:

- **Interception**: deleting some sent packets at random.
- **Modification**: changing the temperature without following the established pattern.
- **Duplication**: sending more tokens than originally planned.

Figure 3 presents the behavior of the broker with IP address 10.6.56.1, an InRow with IP address 10.6.56.50 which exhibits normal behavior, and an abnormal InRow with IP address 10.6.51.41. In this dataset there are five different ip-addresses.

II-C Dataset Pre-processing

To preprocess the dataset, We replace missing values by zeroes. Then we perform one-hot encoding to ip-src field to identify the ip addresses. We apply isolation forest over the preprocessed dataset. Figure 4 presents the Isolation Forest visual where we create a pivot on the dataframe to create a dataframe with all

[3]This dataset is available online source at https://github.com/dadrepository/dad

TABLE I
DAD DATASET SUMMERY OF SEVEN DAYS

Day	Src.byte	Dst.byte	S.pkt	D.pkt	Port.TCP	Port.UDP	Port.MQTT	Pack.nor	Pack.ano	label	ip.add
Monday	699527	348336	9526	4905	13936	495	9248	14431	0	0	10.6.56.1
Tuesday	696658	347846	9491	4907	13908	490	9184	14498	0	0	10.6.56.34
Wednesday	702714	350844	9574	4950	14032	492	9264	14316	208	1	10.6.56.41
Thursday	703431	351018	9585	4953	14048	490	9268	14426	112	1	10.6.56.41
Friday	704084	351124	9592	4952	14054	490	9292	14160	384	1	10.6.56.41
Saturday	707946	353310	9645	4985	14138	492	9339	14246	384	1	10.6.56.41
Sunday	702744	350682	9571	4947	14026	492	9277	14518	0	0	10.6.56.36
Total	4917104	2453160	66984	34599	98142	3441	64872	100495	1088	–	–

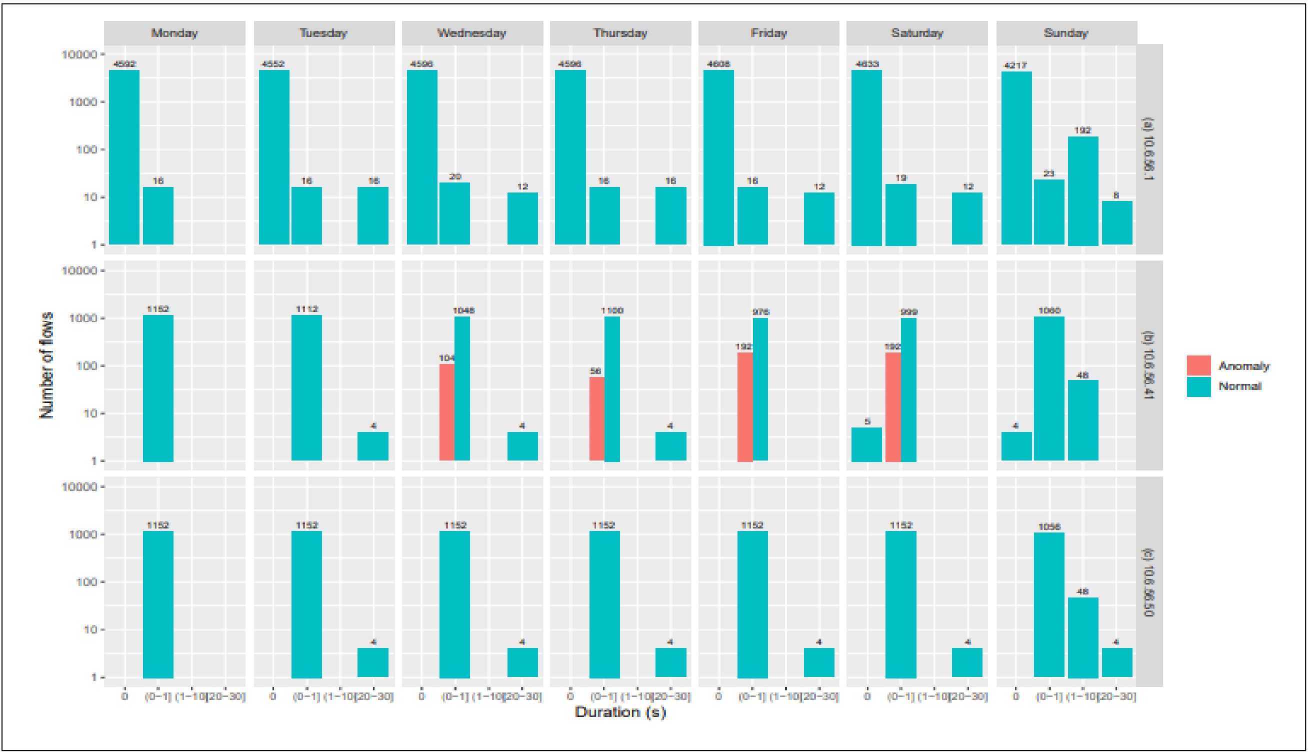


Fig. 3. Log-linear flow duration. (a) Broker. (b) An abnormal node. (c) A normal node

metrics of preprocessed date levels and treat anomaly with 0. Each point in the data is attempted to be separated by the isolation forest, which in the case of 2D randomly draws a line and attempts to isolate a point. Here, a point that is anomalous may be separated in a few steps, whereas closer typical locations may require a lot more steps to be separated. We also used heatmap, pair plot and other plots to visualize our dataset features. Figure 6 presents the Heatmap visual on preprocessed dataset. The corr( ) function on the dataset plotted on the heatmap which shows the correlation between variables on the dataset.

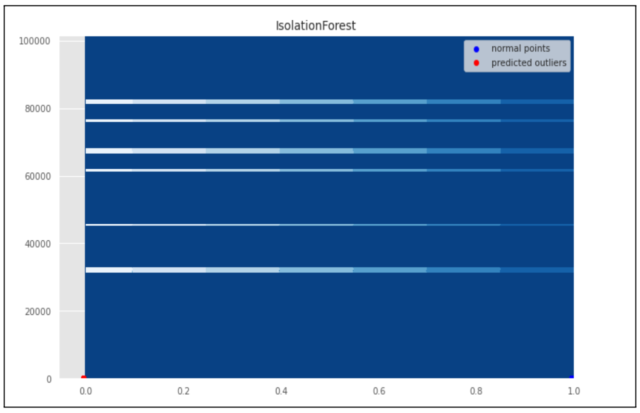


Fig. 4. Isolation Forest to preprocessd data

Finally, we split the whole preprocessed dataset into 80 percent training and 20 percent testing set and

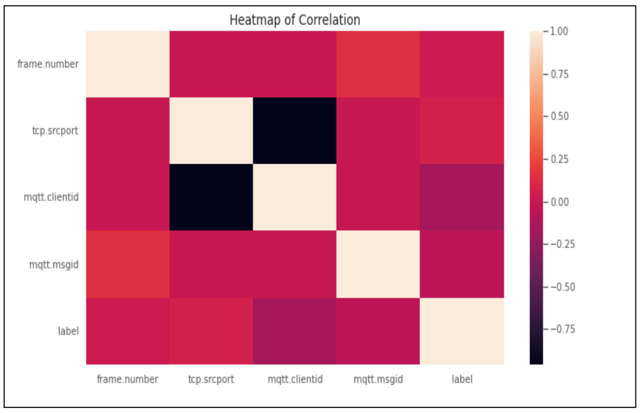


Fig. 5. Heatmap of preprocessed dataset

resample the preprocessed dataset using oversampling and under sampling method.

III Theoretical Considerations

Several machine learning algorithms were used for the data analysis. First, Random Forest was used over the splitting dataset. After that, three boosting based algorithms (AdaBoost, GradientBoost and XGBoost) were used and compared with the RF accuracy. The following is a list of the algorithms and their descriptions.

III-A Random Forest

Random forests are machine learning techniques used for regression and classification problems. To solve complex problems, ensemble learning is used. Ensemble learning involves the use of more than one classifier. Random forest algorithms consist of many decisions trees. The "forest" that was created by the bootstrap aggregation or bagging techniques are used to train the random forest algorithm. An ensemble meta- method called bagging increases the precision of machine learning systems. Based on the predictions of the decision trees, the (random forest) algorithm determines the result. It makes predictions by averaging or averaging out the results from different trees. The accuracy of the result grows as the number of trees increases.

Given an ensemble of classifiers $h1(x)$, $h2(x)$,. . . ., $hk(x)$ and with the training set drawn at random from the distribution of the random vector Y , X define the margin function as follows-

$$mg(\mathbf{X},Y) = av_k\, I(h_k(\mathbf{X}) = Y) - \max_{j \neq Y} av_k\, I(h_k(\mathbf{X}) = j) \quad (1)$$

Where I is the indicator function. The margin measures the extent to which the average number of votes at X,Y for the right class exceeds the average vote for any other class. The larger the margin, the more confident in the classification. The generalization error is given by the following equation where the subscripts X,Y indicate that the probability is over the X,Y space-

$$PE* = P_{\mathbf{X},Y}(mg(\mathbf{X},Y) < 0) \quad (2)$$

III-B AdaBoost (Adaptive Boosting)

Schapire and Freund presented boosting methods first [4].They developed AdaBoost.M1 and AdaBoost.M2 from their AdaBoost algorithm. The difference between the two binary classification problems is how they handle problems with more than two classes. AdaBoost.M1 has access to a learning algorithm to call distributions over the training set periodically. The formula below shows how AdaBoostM1 considers the Dt distribution Where Zt is a normalization constant chosen so that Dt+1 will be distribution.

$$D_{t+1(i)} = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases} \quad (3)$$

Weak Learner calculates a hypothesis that attempts to correctly classify all instances of the test data. Incorrectly classified learners are given greater weighting for the next pass. Finally, the boost algorithm combines all the hypotheses into one final hypothesis.

$$h_{fin}(x) = \underset{y \in Y}{\operatorname{argmax}} \underset{t h_t(x)=y}{\times} \log \frac{1}{\beta_t} \quad (4)$$

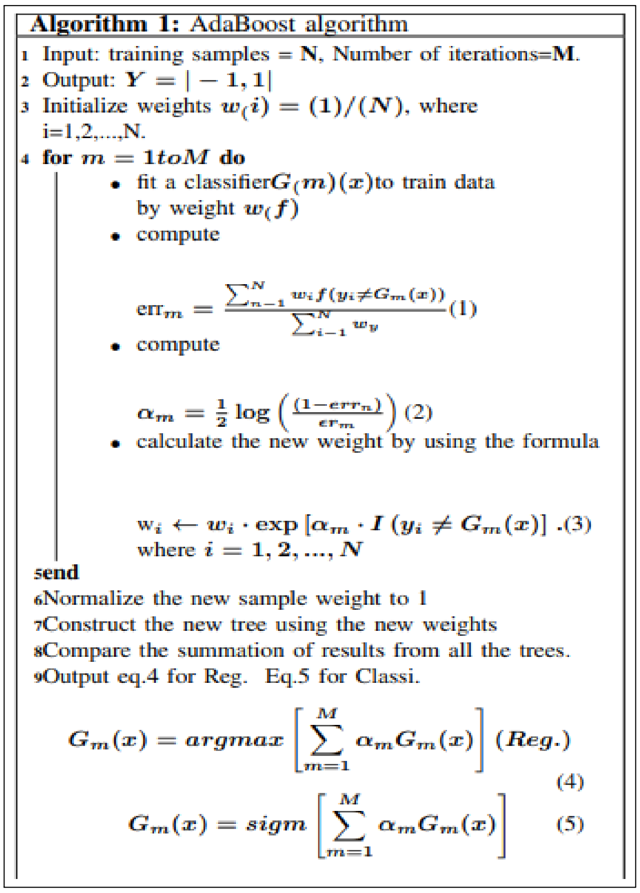
AdaBoost is described [5] in Algorithm 1 below-

III-C Gradient Boosting

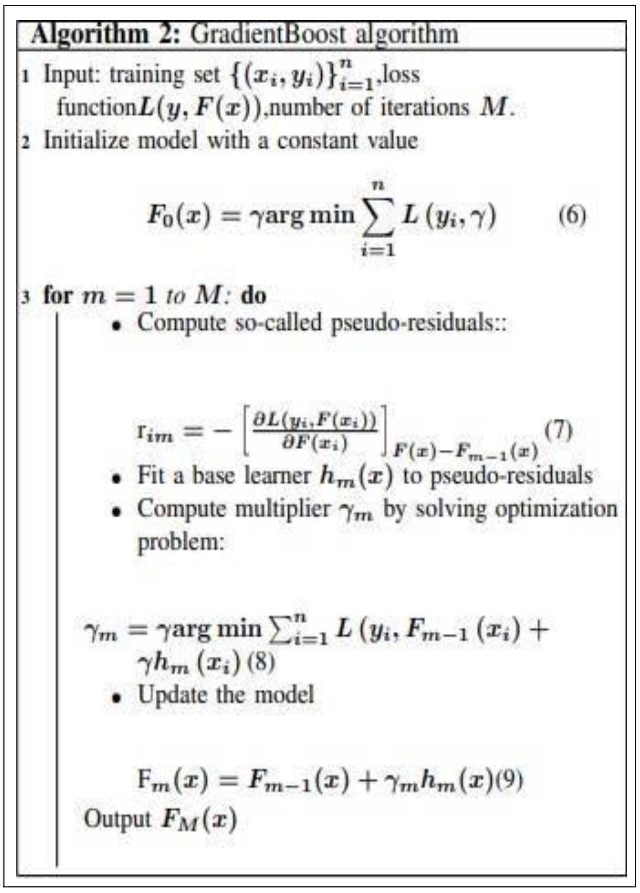
Gradient boosting was inspired by Leo Breiman’s remark that boosting can be viewed as an optimization technique on an appropriate cost function [6]. Gradient boosting is a machine learning technique for regression, classification and other tasks, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. When a decision tree is the weak learner, the resulting algorithm is called gradient boosted trees, which usually outperforms random forest. It builds the model in a

[4]An article titled-The strength of weak learnability,year-1990
[5]https://en.wikipedia.org/wiki/AdaBoost

[6]From Wikoipedia [online]Available https://en.wikipedia.rg/wiki/Gradient$_b$oosting .



```
Algorithm 1: AdaBoost algorithm
1 Input: training samples = N, Number of iterations=M.
2 Output: $Y = | - 1, 1|$
3 Initialize weights $w_(i) = (1)/(N)$, where
  i=1,2,...,N.
4 for $m = 1 to M$ do
      • fit a classifier$G_(m)(x)$to train data
        by weight $w_(f)$
      • compute
        $\mathrm{err}_m = \frac{\sum_{n-1}^{N} w_i f(y_i \neq G_m(x))}{\sum_{i-1}^{N} w_y}$(1)
      • compute
        $\alpha_m = \frac{1}{2}\log\left(\frac{(1-err_n)}{er_m}\right)$ (2)
      • calculate the new weight by using the formula
        $\mathrm{w}_i \leftarrow w_i \cdot \exp\left[\alpha_m \cdot I\left(y_i \neq G_m(x)\right]\right.$ .(3)
        where $i = 1, 2, ..., N$
5 end
6 Normalize the new sample weight to 1
7 Construct the new tree using the new weights
8 Compare the summation of results from all the trees.
9 Output eq.4 for Reg.  Eq.5 for Classi.
  $G_m(x) = argmax\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$ (Reg.)  (4)
  $G_m(x) = sigm\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$  (5)
```

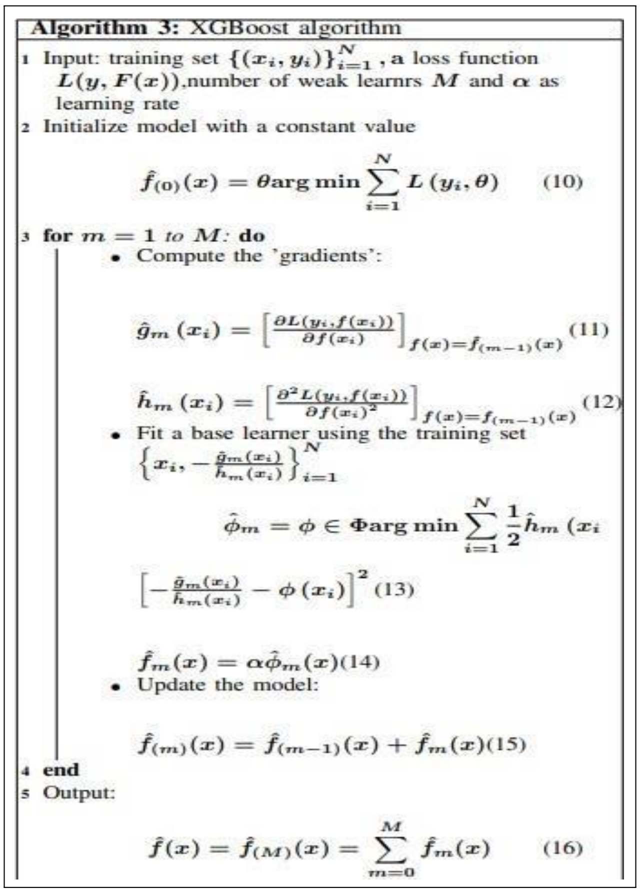


```
Algorithm 2: GradientBoost algorithm
1 Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$,loss
    function$L(y, F(x))$,number of iterations $M$.
2 Initialize model with a constant value
    $F_0(x) = \gamma \arg\min \sum_{i=1}^{n} L(y_i, \gamma)$  (6)
3 for $m = 1$ to $M$: do
      • Compute so-called pseudo-residuals::
        $\mathrm{r}_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)-F_{m-1}(x)}$ (7)
      • Fit a base learner $h_m(x)$ to pseudo-residuals
      • Compute multiplier $\gamma_m$ by solving optimization
        problem:
        $\gamma_m = \gamma \arg\min \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$ (8)
      • Update the model
        $\mathrm{F}_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$(9)
    Output $F_M(x)$
```

stage-wise fashion like other boosting methods do, and it generalises them by allowing optimization of an arbitrary differentiable loss function. In pseudocode, the generic gradient boosting method is:

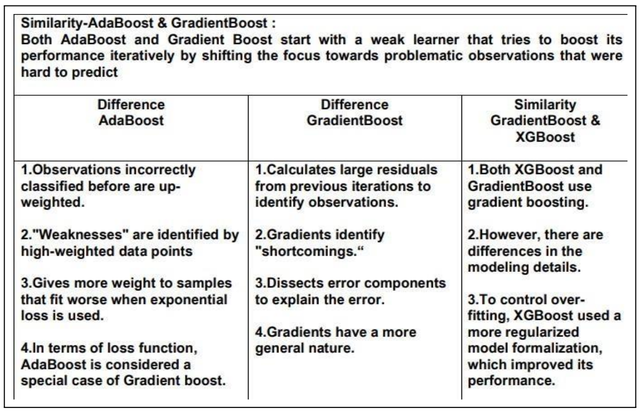
III-D XGBoost

XGBoost is a three-section framework. In the primary section, new facts representations are generated. Specifically, numerous unsupervised outlier detection methods are implemented to the authentic facts to get converted outlier scores as new facts representations. The second phase involves selecting the useful outlier scores from the newly generated outlier scores.Finally, an XGBoost classifier is trained on the new feature space, and its output is regarded as the prediction result.A generic unregularized xgboost algorithm is [7]: The difference of the three boosting algorithms [8] is shown in figure 6



```
Algorithm 3: XGBoost algorithm
1 Input: training set $\{(x_i, y_i)\}_{i=1}^{N}$ , a loss function
    $L(y, F(x))$,number of weak learnrs $M$ and $\alpha$ as
    learning rate
2 Initialize model with a constant value
    $\hat{f}_{(0)}(x) = \theta \arg\min \sum_{i=1}^{N} L(y_i, \theta)$  (10)
3 for $m = 1$ to $M$: do
      • Compute the 'gradients':
        $\hat{g}_m(x_i) = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}_{(m-1)}(x)}$ (11)
        $\hat{h}_m(x_i) = \left[\frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2}\right]_{f(x)=f_{(m-1)}(x)}$ (12)
      • Fit a base learner using the training set
        $\left\{x_i, -\frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)}\right\}_{i=1}^{N}$
        $\hat{\phi}_m = \phi \in \Phi \arg\min \sum_{i=1}^{N} \frac{1}{2}\hat{h}_m(x_i$
        $\left[-\frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)} - \phi(x_i)\right]^2$ (13)
        $\hat{f}_m(x) = \alpha\hat{\phi}_m(x)$(14)
      • Update the model:
        $\hat{f}_{(m)}(x) = \hat{f}_{(m-1)}(x) + \hat{f}_m(x)$(15)
4 end
5 Output:
    $\hat{f}(x) = \hat{f}_{(M)}(x) = \sum_{m=0}^{M} \hat{f}_m(x)$  (16)
```

[7]From https://en.wikipedia.org/wiki/XGBoost
[8]Available https://www.analyticsvidhya.com/blog/



Similarity-AdaBoost & GradientBoost :
Both AdaBoost and Gradient Boost start with a weak learner that tries to boost its performance iteratively by shifting the focus towards problematic observations that were hard to predict

Difference AdaBoost	Difference GradientBoost	Similarity GradientBoost & XGBoost
1.Observations incorrectly classified before are up-weighted. 2."Weaknesses" are identified by high-weighted data points 3.Gives more weight to samples that fit worse when exponential loss is used. 4.In terms of loss function, AdaBoost is considered a special case of Gradient boost.	1.Calculates large residuals from previous iterations to identify observations. 2.Gradients identify "shortcomings." 3.Dissects error components to explain the error. 4.Gradients have a more general nature.	1.Both XGBoost and GradientBoost use gradient boosting. 2.However, there are differences in the modeling details. 3.To control over-fitting, XGBoost used a more regularized model formalization, which improved its performance.

Fig. 6. Difference between three boosting methods.

IV Evaluation Criteria

The following metrics were calculated to assess the developed system's performance. These metrics can be used to determine which technique is best suited for this task.

IV-A Confusion Matrix

The confusion matrix is a tool for visualizing a technique's performance. It's a table that's frequently used to describe a classification model's performance on a set of test data for which the true values are known. It enables for the quick discovery of class confusion. Almost all performance measurements are calculated from it the majority of the time. A definition of True Positive (TP), False Positive (FP), False Negative (FN) and True Negative (TN) for multiple classes can be given from confusion matrix. Let Ci be any class out of the four classes. Following are the definitions of TP, FP, FN, and TN for Ci:

- **TP(Co)** = All the instances of Co classified as Co.
- **FP(C1)** = All the non C1 instances classified as C1.
- **FN(C2)** = All the C2 instances not classified as C2.
- **TN(C3)** = All non C3 instances not classified as C3.

IV-B Accuracy

The accuracy of a model is merely one aspect of its overall performance. One of the criteria used to evaluate classification models is accuracy shown below-

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad (5)$$

IV-C Precision

Precision refers to the ability to forecast something with a high degree of accuracy. It's a measure of how many true positives the model claims vs how many positives it claims. The following equation gives the precision value for a single class:

$$Precision = \frac{TP}{TP + FP} \qquad (6)$$

IV-D Recall

The recall is also known as the real positive rate, which is the difference between the number of positives in the model claims and the total number of positives in the data. The following equation gives the recall value for a single class:

$$Recall = \frac{TP}{TP + FN} \qquad (7)$$

IV-E F1-score

A model's performance can also be measured using the F1 score. It is a weighted average of a model's precision and recall. Eq-17 gives the F1 Score value for a specific class:

$$F1Score = \frac{2 * TP}{2 * TP + FP + FN} \qquad (8)$$

V Result Analysis

The 5-fold mean values of train and test stages for different classifiers are given in the following table II: bar graph visual of the table II has shown in figure 7.

TABLE II
5-FOLD MEAN VALUES OF TRAIN AND TEST CLASSIFIERS

5-fold SD	RF	AdaB	GradientB	XGB
train-mean-grid-1	1	1	1	0.9999
train-mean-grid-2	1	1	1	0.9999
train-mean-grid-3	1	1	1	0.9999
train-mean-grid-4	1	1	1	0.9999
train-mean-grid-5	1	1	1	0.9999
test-mean-grid-1	1	1	1	0.9931
test-mean-grid-2	0.99999	1	1	0.9931
test-mean-grid-3	0.99999	1	1	0.9931
test-mean-grid-4	0.99999	1	0.851267	0.9931
test-mea-grid-5	0.99999	1	1	0.9931

In the theoretical considerations section, It has been described that Random Forest, AdaBoost, GradientBoost and XGBoost algorithms were applied to the

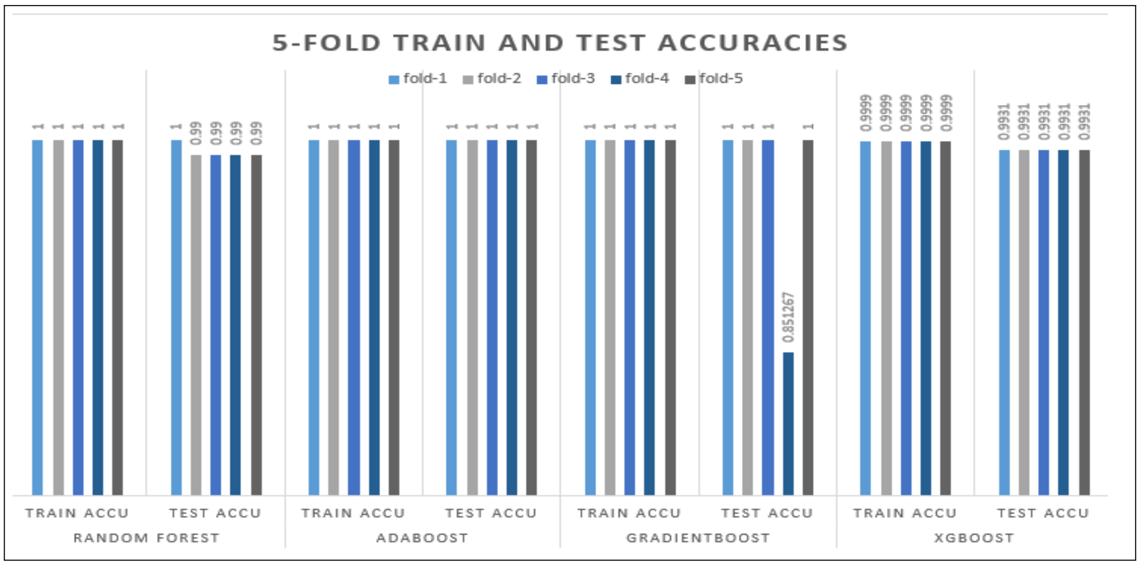


Fig. 7. 5-fold Train and Test accurecies of models-bar graph

dataset. Five-fold cross-validation was performed on the dataset using each of these techniques.

This figure shows how the accuracy results are converged after five-fold cross-validation. From the cross-validation, it can be inferred that AdaBoost and XGBoost have performed best both in training and testing accuracy since they have the same training and testing accuracies. GradientBoost performed with approximate similarity to AdaBoost in the case of training. In the case of testing, the GradientBoost performed better than XGBoost but weakly than AdaBoost on 4th fold. Random Forest performed well on each five training accuracies but in case of testing folds Random Forest has lower testing accuracies on every fold except 1st fold. In summary, we can say that boosting based algorithms were performed better than Random Forest and has recovered overfitting well than Random Forest.

Table III represents different testing evaluation metrics for different techniques trained on the dataset. From table IIIit can be seen that Random Forest, AdaBoost, GradientBoost and XGBoost have same accuracy, precision, recall, and F1 score values during testing phase. By considering the confusion metrics generated on the coding part, it can be said that all algorithms have similar accuracy after splitting the dataset. Out of 20317 samples which is the 20 percent of the whole dataset, the algorithms correctly classified 20316 as true positive. In table III the evaluation metrics has shown:

As a summary of result analysis we can say that If the training data has a low error rate or higher training accuracies and the test data has a high error rate or lower testing accuracies, it signals overfitting. This statement applies to only Random Forest case but not for three boosting algorithms case.

TABLE III
EVALUATION MATRIX FOR TRAIN AND TEST ACCURECIES

Eva.Matrix	RF	AdaB	GB	XGB
Precision(test)	1	1	1	1
Recall(test)	1	1	1	1
f1-score(test)	1	1	1	1
Support-0(test)	20316	20316	20316	20316
Support-1(test)	1	1	1	1
Accuracy(test)	1	1	1	1

VI Conclusion

According to the findings of the study, the boosting technique should be used on these types of anomalous datasets to solve overfitting on IoT sites. The aim of this research paper is to detect anomaly using boosting based algorithms with highest accuracy but not with overfitting problem like random forest. Boosting based algorithms are used to help in reducing variance and bias in a machine learning ensemble.

Since GradientBoostiong method have several hyperparameters that include the number of trees, the depth (or number of leaves), and the shrinkage (or learning rate) so GradientBoost is harder to tune. In future, we should work on ways of easy tuning in case of boosting based algorithms. Also, boosting is not the proper algorithm to workwith real time data environment. Hence, further study is needed for detecting ways of easy tuning and implemention of real time dataset. Lastly, we should find other advanced algorithm to work with anomilies in IoT sites. For example we can use Light GradientBoost which is a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm, used for ranking, classification and many other machine learning tasks. Again, for real time network traffic anomaly detection we can use big data processing frameworks such as Apache Hadoop, Apache Kafka, and Apache Storm in conjunction with machine learning algorithms [5].

Acknowledgment

I would like to thank associate professor of machine learning, dept of electronic engineering and computer science Jiadong Yu for her valuable comments and suggestions to improve the quality of the paper. I am also grateful to her for helping me review my work regularly.

References

[1] Aleksandar Lazarevic, Levent Ertoz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. *A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection*, pages 25–36.

[2] Nikunj C Oza and Stuart J Russell. Online bagging and boosting. In *International Workshop on Artificial Intelligence and Statistics*, pages 229–236. PMLR, 2001.

[3] D. Tanouz, R Raja Subramanian, D. Eswar, G V Parameswara Reddy, A. Ranjith Kumar, and CH V N M Praneeth. Credit card fraud detection using machine learning. In *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 967–972, 2021.

[4] Tao Yang, Liang Chen, Jigang Wang, Zenghao Cui, and Jian-peng Qi. Anomaly detection of dust removal system through gradient boosting decision tree algorithm. In *2021 International Conference on Communications, Information System and Computer Engineering (CISCE)*, pages 685–688, 2021.

[5] Shuai Zhao, Mayanka Chandrashekar, Yugyung Lee, and Deep Medhi. Real-time network anomaly detection system using machine learning. In *2015 11th International Conference on the Design of Reliable Communication Networks (DRCN)* , pages 267–270, 2015.