PREDICTION OF ATTACK, ITS CATEGORY AND SUBCATEGORY IN BOT-IOT DATASET

COURSE PROJECT REPORT

Submitted by:

Rajan Kataria (18103076)
Rippendeep Kaur (18103078)
Ritika Goyal (18103080)
Sanyamdeep Singh (18103083)
Shreya (18103087)

Under the guidance of

Asst. Prof. Nonita Sharma
CSE Department, NIT Jalandhar

A report submitted for the partial fulfillment of the course:

DATA MINING AND DATA WAREHOUSING | CSPC-308/328

DEPT. OF COMPUTER SCIENCE AND ENGINEERING

DR B R AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY, JALANDHAR

ACADEMIC YEAR: 2020-2021

TABLE OF CONTENTS

GROUP MEMBERS' DETAILS

Sr. No.	Roll No.	Name	Course	Contact
1.	18103076	Rajan Kataria	B-Tech. (CSE)	rajank.cs.18@nitj.ac.in
2.	18103078	Rippendeep Kaur	B-Tech. (CSE)	rippendeepk.cs.18@nitj.ac.in
3.	18103080	Ritika Goyal	B-Tech (CSE)	ritikag.cs.18@nitj.ac.in
4.	18103083	Sanyamdeep Singh	B-Tech. (CSE)	sanyamdeeps.cs.18@nitj.ac.in
5.	18103087	Shreya	B-Tech. (CSE)	shreya.cs.18@nitj.ac.in

LINKS TO GOOGLE COLAB NOTEBOOKS

Algorithm	Code Link
Information Gain	Link to IG notebook
Gini Index	Link to GI notebook
Naïve Bayes	Link to NB notebook
KNN	Link to KNN notebook
Random Forest	Link to RF notebook
Gradient Boosting	Link to GB notebook

1. ABSTRACT

With the Internet of Things (IoT), millions of devices which are capable of interaction can be combined with each other with minimum user interaction. IoT is one of the fastest-emerging areas of computing; however, IoT is vulnerable to numerous types of cyberattacks. To address this challenge, practical and realistic countermeasures need to be developed to secure IoT networks, such as network anomaly detection and network intrusion detection. Although it is not possible to avoid the wholly forever, yet an early detection of an attack is crucial for practical defence. Intelligent network-based security solutions like machine learning solutions can provide an efficient solution to attack detection problems. In this project, our aim is to contribute to the literature by evaluating various machine learning algorithms that can be used to quickly and effectively detect IoT network attacks. The Bot-IoT dataset, is used to evaluate various detection algorithms. Six different machine learning algorithms are used in the implementation, and are compared in terms of their performance. Best features were extracted from the Bot-IoT dataset during the implementation.

2. KEYWORDS

Machine Learning, Internet of Things (IoT), Cyberattacks, Bot-IoT dataset, Network Forensics, Intrusion detection, Algorithms, Classification, Accuracy Precision, Recall, Sensitivity, Confusion Matrix

3. INTRODUCTION

3.1 REAL-LIFE APPLICATION

As mentioned earlier, this project aims to detect the cyberattacks on IoT based systems using different machine learning algorithms so as to prevent the attacks and develop secure networks.

With the growing communication of networks, concerns over security and privacy are increasing among people. Cyber security is nowadays the first requirement because of widespread involvement of computers in our lives. Internet of Things (IoT) has emerged as a result of the rapid growth of the Internet, with notable examples including smart homes and towns, as well as healthcare systems. IoT is a group of interconnected devices, having sensors and capability to communicate with each other with least human intervention. These devices are managed through web. Every new emerging technology attracts users as well as attackers. Same is the situation with IoT based systems. They are also vulnerable to cyberattacks. The hackers use complex techniques such as Botnets to exploit the network.

With the new emerging threats that are capable to destruct the IoT networks, it is the need of the hour to come up with some advanced methods of cyber forensics to identify the network anomalies. Network Forensics deal with analysing the network traffic and detecting the devices that are part of cyberattacks. We need quick and efficient methods attack detection. ML can provide an ideal solution in this scenario. With Machine Learning helpful knowledge can be inferred from data. Machine Learning algorithms have been useful in Intrusion detection systems. Thus, in an IoT network, we can deploy ML for ensuring security.

In this project we will train different ML models on the given (BOT-IoT) dataset to detect the various possible attacks so that they can be prevented. In this way, this project finds its applications in the IoT based networking system where it can be utilized to address security issues.

3.2 DATASET

The dataset that is given to us, BOT-IOT Dataset, has been made from the realistic network design. The environment has a combination of the botnet and the normal traffic. The original dataset has more than 7,20,00,000 records and about 46 attributes. We were supposed to work with the 5% of the original dataset. It has about 3 million records and 19 attributes. The Training Dataset is of 352 MB, while the Testing Dataset is of 88 MB The training set has 29,34,817 rows, the test set has 7,33,705 rows. There are no missing values in the provided training or testing dataset.

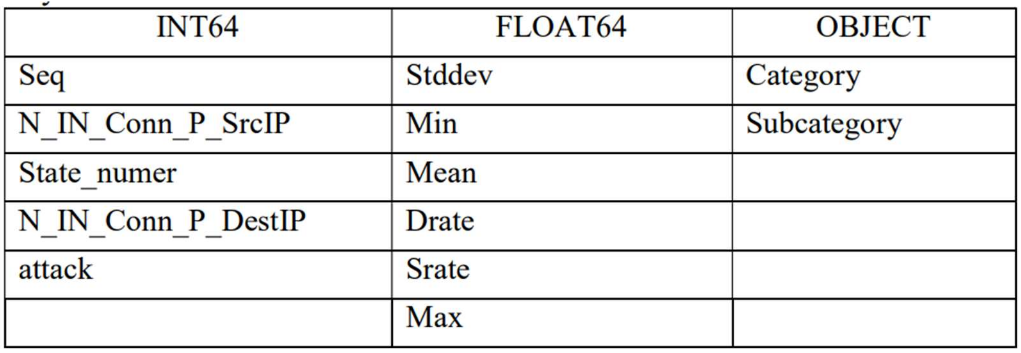
All the 46 features with their description are given as follows:

Feature	Description	Feature	Description
pkSeqID	Row Identifier	Spkts	Source-to-destination packet count
Stime	Record start time	Dpkts	Destination-to-source packet count
Flgs	Flow state flags seen in transactions	Sbytes	Source-to-destination byte count
flgs_number	Numerical representation of feature flags	Dbytes	Destination-to-source byte count
Proto	Textual representation of transaction protocols present in network flow	Rate	Total packets per second in transaction
proto_number	Numerical representation of feature proto	Srate	Source-to-destination packets per second
Saddr	Source IP address	Drate	Destination-to-source packets per second
Sport	Source port number	TnBPSrcIP	Total Number of bytes per source IP
Daddr	Destination IP address	TnBPDstIP	Total Number of bytes per Destination IP.
Dport	Destination port number	TnP_PSrcIP	Total Number of packets per source IP.
Pkts	Total count of packets in transaction	TnP_PDstIP	Total Number of packets per Destination IP.

Bytes	Totan number of bytes in transaction	TnP_PerProto	Total Number of packets per protocol.
State	Transaction state	TnP_Per_Dport	Total Number of packets per dport
state_number	Numerical representation of feature state	AR_P_Proto_P_SrcIP	Average rate per protocol per Source IP. (calculated by pkts/dur)
Ltime	Record last time	AR_P_Proto_P_DstIP	Average rate per protocol per Destination IP.
Seq	Argus sequence number	N_IN_Conn_P_SrcIP	Number of inbound connections per source IP.
Dur	Record total duration	N_IN_Conn_P_DstIP	Number of inbound connections per destination IP.
Mean	Average duration of aggregated records	AR_P_Proto_P_Sport	Average rate per protocol per sport
Stddev	Standard deviation of aggregated records	AR_P_Proto_P_Dport	Average rate per protocol per dport
Sum	Total duration of aggregated records	Pkts_P_State_P_Protocol_P_DestIP	Number of packets grouped by state of flows and protocols per destination IP.
Min	Minimum duration of aggregated records	Attack	Class label: 0 for Normal traffic, 1 for Attack Traffic
Max	Maximum duration of aggregated records	Category	Traffic category
Pkts_P_State_P_Protocol_P_SrcIP	Number of packets grouped by state of flows and protocols per source IP.	Subcategory	Traffic subcategory

We were provided with 19 best features, out of which we chose the best 13 (10 predictor variables and 3 target variables).

They are:



INT64	FLOAT64	OBJECT
Seq	Stddev	Category
N_IN_Conn_P_SrcIP	Min	Subcategory
State_numer	Mean	
N_IN_Conn_P_DestIP	Drate	
attack	Srate	
	Max	

The prediction is to be made on the following three target attributes:

1. *Attack* – It has *2* types.
   Normal - 0, BotNet - 1
2. *Category* - It has *5* unique categories.
   DDoS, DoS, Reconnaissance, Normal, Theft
3. *Subcategory* – It has *8* unique categories.
   UDP, TCP, OS_Fingerprint, Service_Scan, HTTP, Normal, Keylogging, Data_Exfiltration

3.3 CLASSIFICATION

3.3.1 What is Classification?

Classification, in simple words, mean to categorize data into various categories or classes. The process of classification starts with the prediction of class of given record points. The classes are also known as target, label or categories. Classification in particular includes two steps - training and prediction, firstly the model is trained on the training dataset and afterward the target attribute is predicted for the testing dataset using the trained model.

The assignment of approximating the mapping function from discrete info factors to discrete yield factors is classified predictive modelling. The primary point is to sort out which gathering or class the new information has a place with. Classification in machine learning falls under the "Supervised Learning" in which the new data points are classified into predefined classes or target variables. The target variables are also provided with input data. A classifier is the algorithm that performs the classification on a dataset.

3.3.2 Types of Classification

- Binary Classification: The data is to be classified in only two categories or the target variable has only two values.
  Examples: NO/YES, MALE or FEMALE, MARRIED/UNMARRIED, SPAM/NOT SPAM, CAT/DOG, etc.
- Non-binary Classification: The target variable has multiple values.
  Example: Types of movies, Types of cyberattacks etc.

3.3.3 Types of Learners in Classification

The sorts of learners in classification are: lazy learners and eager learners.

- Lazy Learners – The training data is simply stored and no pre-model is designed using training data and when testing data is encountered, classification is finishes utilizing the most related information in the put away training data. They take in additional in prediction as contrasted with eager learners. They are mostly suitable for datasets with few features.
  e.g. – k-nearest neighbour
- Eager Learners – Before getting data for predictions, eager learners develop a classification model dependent on the provided training data. When the testing dataset arrives, the prediction is done on the basis of classification model. The time taken in training the model is more and the prediction time is very less.
  e.g. – Decision Tree, Naive Bayes, Random Forest

3.3.4 Methods of Classification

There are a lot of classification algorithms available but it is very tough and even not right to say that which one is superior. All the classification algorithms have their own advantages and disadvantages. In this report itself we will study in detail about some of the classification algorithms and compare their performances on the given dataset (BOT-IoT).

Below are listed various classification algorithms:

- Decision Tree Classifier
- Random Forest Classifier
- KNN (k- nearest neighbours) Classifier
- Naïve Bayes Classifier
- Logistic Regression
- SVM (Support Vector Machines)

There are various methods available for the evaluation of classification models such as Confusion Matrix, Precision and Recall, ROC Curve etc.

3.3.5 Scope and Applications of Classification

Classification has a wide scope and is used very widely used. Some of the areas where classification finds its applications are listed below:

- NETWORK ATTACK/ANOMALY DETECTION: In this project itself, we are providing various classification models that are trained using BOT-IoT dataset which identify the different kinds of attacks. The attacks are further classified into five major categories.
- DISEASE DETECTION: Classification models are used to detect based on the records of the past patients that whether a given patient suffers from a disease or not.
- SPAM FILTERS: Spam filters classify the calls/messages/e-mails into spam or not spam.

Some other applications include:

- Word categorisation
- Climate forecasting
- Voting Ensembling
- Real time object prediction from image
- Real time object detection from videos

- Stock pricing
- Sentiment/Mood Analysis
- Handwriting detection

3.4 STRUCTURE OF PROJECT REPORT

The structure of remainder report is given as follows:

- Section 4 provides a detailed study of all the six algorithms (Information Gain, Gini Index, Naïve Bayes, KNN, Random Forest, Gradient Boost).
- Experimentation and Implementation details are discussed in Section 5.
- Section 6 deals with the evaluation of experimental results. Results of different classification algorithms are compared in the form of Graphical visualizations, confusion matrices and other measures such as Accuracy, Precision, Recall, Sensitivity and F1-Score.
- Finally, the report is concluded with the discussion of the future scope in Section 7.

4. METHODS OF CLASSIFICATION

4.1 DECISION TREE ALGORITHM (INFORMATION GAIN)

Rectangular nodes and edges make up the majority of decision trees. Each node represents a result or makes a decision. The edges serve as a connection between the two nodes. There are three types of nodes in the decision tree: Root node, Inner nodes, and Leaf nodes. The decision tree is typically used in the form of a questionnaire, in which each node asks a question relevant to a specific condition, whose answer is in the form of Yes/No, and hence it splits the tree into subtrees according to the response received.

The complete process of building the decision tree is mentioned using the below algorithm:

- The tree is started with the root node, that contains the tuples of the complete training dataset.
- Using the Attribute Selection Measure, we have to find the best attribute in the dataset on which the split is to be made.
- Sub-divide the set into subsets that contain the best attribute's possible values.
- Create the node of the decision tree mentioning the best attribute.
- Now, sub decision trees are created in a recursive manner with the help of subsets of the dataset that were created in step 3. Keep on applying this method until you can no longer divide the nodes into subtrees, at which point the final node is referred to as a leaf node.

The decision trees can be used to build both classification and regression models. However, classification trees are generated using the ID3 (Iterative Dichotomiser 3) algorithm. Ross Quinlan is the creator of this algorithm. To fabricate a decision tree, this calculation utilizes a hierarchical avaricious technique. The top-down methodology implies that working of the tree has begun from the top and the greedy methodology implies that at every single cycle, the best component is chosen at that point to make a hub.

Information Gain is the quality determination measure utilized in the ID3 calculation.

The difference in entropy after segmenting a dataset based on an attribute is known as information gain. It's used to figure out how much data a function gives us about a class. The node is split and the D-tree is made based on the importance of knowledge gained. The highest information gain node/attribute is split first in a decision tree algorithm, which always seeks to maximise the amount of IG.

Let us say, for a given column X, the information gain is calculated as:
$IG (A, X) = Entropy(A) - \sum ((|A_v| / |A|) * Entropy(A_v))$
Where,
A_v is the set of rows in A for which the feature column X has value v,
$|A_v|$ is the number of rows in A_v
and likewise |A| is the number of rows in A.

And, Entropy is a measure that is used to measure the impurity in a given attribute. It signifies the randomness of that very attribute. Entropy can be calculated as:
$Entropy(A) = - \sum (prob_i * log_2(prob_i))$; i = 1 to m
where,
m = Total number of classes in the target column
$prob_i$ = The probability of class ‘i’ or “the ratio of number of rows with class i in the target column to the total number of rows in the dataset.”

Algorithm for making decision tree using ID3 is:

- Calculate the value of Information Gain for each feature.
- The dataset is split on the basis of the attribute which is having the maximum Information Gain. And, the attribute which has the maximum information gain is made the node in the decision tree.
- In case you observe that, all the rows belong to the same class, make that node as the leaf node and give it the class name.
- Repeat the same for the remaining features until the decision tree made by you has all leaf nodes or we run out of features.

4.2 DECISION TREE ALGORITHM (GINI INDEX)

As talked about above, ID3 calculation depends on the Information Gain as the standards to part nodes, there is another calculation called "CART" calculation that makes utilize another model called Gini to part the nodes.
While ID3, as an "Iterative Dichotomiser" is for classification only; CART, or "Classification And Regression Trees" is not only limited to, classification tree learning; it can be used to create regression trees too.
“The Gini index measures the extent to which the distribution deviates from a perfectly equal distribution. It calculates the likelihood of a given feature being labelled incorrectly when it is chosen at random.”
The Gini index ranges from 0 to 1, with 0 indicating complete classification and 1 indicating complete classification. And 1 denotes a random distribution of elements among different groups.

The Gini Index value of 0.5 indicates that elements are distributed evenly through certain groups.
The Gini Index is determined by deducting the quantity of squared probabilities of each class from one. Gini Index can be addressed numerically as Gini Index = $1-\sum(Prob_i^2)$
For a split, a function with a lower Gini index is selected.

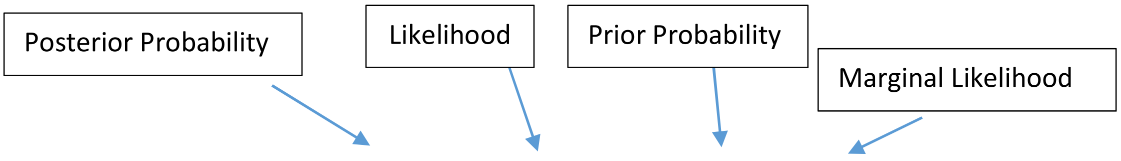
Algorithm for making decision tree using ID3 is:
Step-1: Calculate the value of Gini index for each feature which is calculated by weighted sum of gini index for that feature.
$Gini(A, X) = \sum_i(A_i / A)*Gini(Ai)$
Step-2: The dataset is split on the basis of the attribute which is having the minimum Gini Index. And, the attribute which has the minimum Gini Index is made the node in the decision tree.
Step-3: Make a decision tree node using the feature with the minimum Gini index.
Step-4: In case you observe that, all the rows belong to the same class, make that node as the leaf node and give it the class name.
Step-5: Repeat the same for the remaining features until the decision tree made by you has all leaf nodes or we run out of features.

4.3 NAÏVE BAYES ALGORITHM

The Naïve Bayes Algorithm is a Supervised Classification algorithm. The backbone of this algorithm is the Bayes Theorem. As it is a classification algorithm, it calculates the probability of an instance of belonging to a particular class based on its features. In short, it calculates the conditional probability.
The Naïve Bayes assumes that there is no correlation between the features i.e., they are independent of each other. Also, each variable has equal weightage in the final outcome in Naïve Bayes. It only takes in discrete variables. Hence the continuous ones have to be converted to discrete. It is popular for multiclass classification.
Some of its advantages are:

- It is fast, versatile, flexible and an easy-to-understand algorithm. It works well for document classification, email classification into spam/normal.
- It does not require a large dataset for training.
- There is no risk of overfitting in Naïve Bayes.
- It can be used for both binomial and multiclass classification.

Mathematics behind Naïve Bayes:



Bayes' Law: $P(G/H) = (P(H/G) * P(G)) / P(H)$

P(G/H) = probability of G given that H is true
P(H) = Independent probability of H.
P(G) = Independent probability of G.

Steps involved in the Naïve Bayes Theorem:

- Find all the probabilities needed for the Bayes Theorem for the calculation of posterior probability.
- Calculate the posterior probabilities.
- Compare the posterior probabilities of the classes to which we want to classify our value. It will be classified into the one with greater probability.

There are 3 types of Naïve Bayes Algorithms:

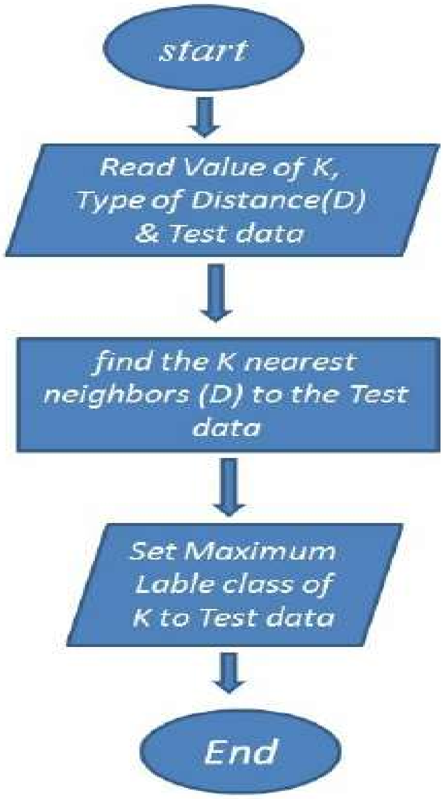
- Gaussian Naïve Bayes – The likelihood of the attributes is considered to be Normal Distribution. It is symmetric about the mean of the attributes.
- Multinomial Naïve Bayes – The attribute vectors represent count of occurrence of certain events. This type of model is used for the document classification.
- Bernoulli Naïve Bayes – Here the feature vector represents whether or not the word occurs in the document rather than its frequency.

4.4 K- NEAREST NEIGHBOURS ALGORITHM

KNN is a straightforward Machine Learning calculation which depends on Supervised learning. KNN calculation expects the likeness between the new information and accessible information and put the new information into the class that is like the comparable classifications. This implies when the new information shows up then it very well may be effectively ordered into an appropriate classification utilizing KNN calculation. KNN calculation can be utilized both for relapse and grouping yet generally is utilized for arrangement issues. KNN is a non-parametric calculation, which implies it doesn't make any suspicions on fundamental information. It's otherwise called a sluggish student calculation since it doesn't gain from the preparation set straight away; all things being equal, it stores the dataset and plays out a procedure on it at the hour of order. Model, assume we have a picture to perceive that seems to be like cat and a dog, yet we need to know whether it is a cat or a dog. So for this task, we can utilize the KNN calculation which chips away at a likeness measure. Our model will discover the component like new dataset to the pictures of cat and dog dependent on the most comparable highlights, it will place in one or the other, cat or a dog classification. The KNN working can be explained dependent on the underneath computation:

- First of all, the number K is selected fr the number of neighbours.
- Then, the Euclidean distance or any other distance, say, Manhatten distance is calculated for K neighbours.
- Based on the distances calculated, K nearest data points are considered.
- Among the considered k neighbours, we record the amount of the data accentuations in each characterization.
- Assign the new data accentuations to that characterization for which the amount of the neighbour is generally outrageous.

The flowchart for KNN algorithm is as shown below:


start
Read Value of K, Type of Distance(D) & Test data
find the K nearest neighbors (D) to the Test data
Set Maximum Lable class of K to Test data
End

Benefits of K-NN Algorithm are listed as:

- Straightforward execution.
- It is lively to the vigorous getting ready data.
- It is regularly more impressive if the arrangement data is monstrous.

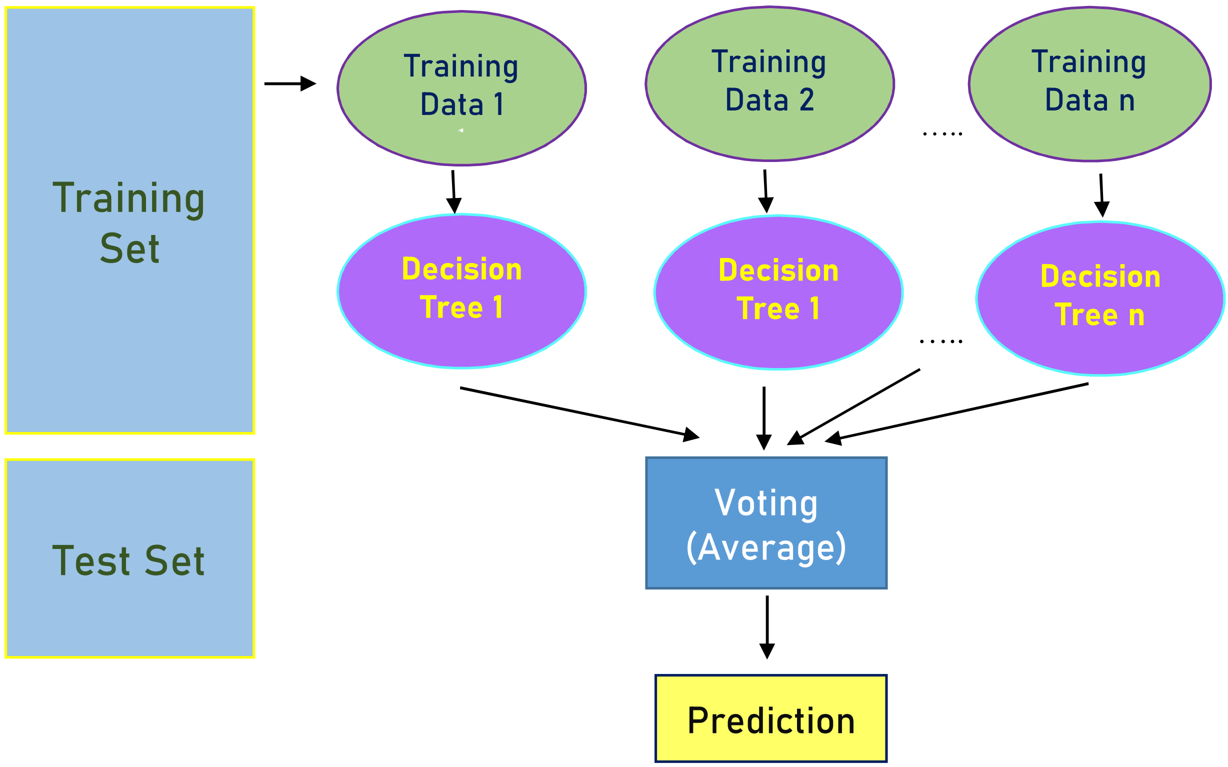
Drawbacks of KNN algorithm:

- It in every case needs to decide the 'K' which can be intricate at some point.
- The computation cost is high because of ascertaining the distance between the information focuses for all the preparation tests.

4.5 RANDOM FOREST ALGORITHM

Random Forest is an algorithmic AI program which is supervised i.e., it is provided with the correct labels in order to make the prediction for the unseen data. This algorithm can be utilized for each regression and classification problems. They are often referred to as the Black Box Models because one cannot see what is going on inside.
Random Forest is a classifier which contains a number of decision trees trained on different subsets from the training dataset.
Decision tree makes a greedy selection of the best split point (by discovering the trait and the estimation of that property brings about the most minimal expense) from the dataset at each progression. This makes them defenceless to high difference if the pruning isn't finished. To get rid of the high variance problem, we can make numerous trees with various examples of the training dataset and then combine the predictions to get a final result. However, a unique component of random forest is that it restricts the lines which

the greedy algorithm assesses at each split while making the tree. This is the thing that assists with getting a low variance in the final result.
Henceforth, to put it plainly, various examples from the training dataset are taken. Each tree is prepared in exceptional manner. At each point a split is made in the information and added to the tree by thinking about just a fixed number of features. The more the quantity of trees in the forest, the higher is the accuracy. This forestalls overfitting.



The Random Forest algorithm has the following steps:
Step 1: Select K data points from the training set at random.
Step 2: According to the selected data points, build a decision tree.
Step 3: Chose the number of decision trees we want to build (N)
Step 4: Repeat the process 1 and the process 2
Step 5: Find each decision tree's predictions and add new data points to the group with the most votes.

The Random Forest algorithm has the following advantages:

- It has less time to train the model.
- It runs efficiently for large datasets as well, and gives high accuracy.
- Even if there is some data missing, the accuracy is good.
- It can perform both Classification and Regression tasks.
- It prevents the overfitting issue.

4.6 GRADIENT BOOST ALGORITHM

Gradient Boosting is a word made by combining both Gradient Descent and Boosting. It is a powerful machine learning algorithm. It can do regression, classification, as well as ranking. Learning happens by optimizing the loss function. It uses two types of base estimators: Average-type model and Decision tree with full depth. Trees are bigger in size than the Adaboost, generally. The range of leaf nodes generally lie in a range of 8-32. For example, suppose we want to calculate height of a person from the age and BMI provided. Here, age and BMI are independent variables and age is target variable.

This is a regression problem where Gradient Boost can be used to solve it.

Gradient Boost algorithm has three elements: A loss function, a weak learner and, an additive model. Three of these are explained as follows:

4.6.1 Loss Function

The loss function used depends on the type of problem which is to be solved.

It must be differentiable, but many standard loss functions are supported and we can define our own. For example, in regression, a squared error may be used, and in classification, a logarithmic loss may be used. The gradient boosting framework has the advantage of not requiring the creation of a new boosting algorithm for each loss function that may be used; instead, it is a general enough framework that any differentiable loss function may be used.

4.6.2 Weak Learner

In gradient boosting, decision trees are used as the weak learner. Regression trees are used specifically that output real values for splits and whose output can be added together, allowing subsequent models outputs to be added and correct the residuals in our predictions. Trees are built greedily, with the best split points chosen based on Gini scores or to mitigate the loss. It is collective to make the weak learners in specific ways, like a maximum number of layers, nodes, splits or leaf nodes. This is to make sure that the learners remain weak, but can still be constructed greedily.

4.6.3 Additive Model

Trees are added each in turn, and existing trees inside the model are not changed. A slope drop technique is utilized to limit the misfortune when adding trees. Customarily, inclination drop is utilized to limit set of boundaries, like the coefficients during a relapse of y on x condition or loads in a neural organization. Subsequent to computing blunder or misfortune, loads are refreshed to limit that mistake. Rather than boundaries, we've frail student sub-models or all the more explicitly the decision trees. Subsequent to ascertaining the misfortune, to play out the slope plummet methodology, we should add a tree to the model that lessens the lost (for example follow the angle). We do this by defining the tree, at that point changing the tree's boundaries and diminishing the leftover misfortune to step the correct way.

The gradient boost algorithm has the following benefits:

- It provides a lot of flexibility in approach as we can optimize different loss functions.
- There is no need of data pre-processing.
- It handles missing data as well i.e. imputation is not required.

5. EXPERIMENTATION

For experimenting and implementing different machine learning models to predict the results in the dataset, the Google Colab Platform has been used in this project. We have chosen this platform because multiple users can work on the same notebook at the same time and hence all the team members can participate in editing or discussing the code.

After importing the dataset to the Google drive, we have imported the training and testing dataset in the Google Colab python notebook using PyDrive library. After importing we analysed that in the dataset, the columns pkSeqID, proto, saddr, sport, daddr, dport were mentioned in the dataset for the identity of data points (packets to be transferred from the source to destination). After dropping these columns from the dataset, we were left with the ten best columns as mentioned in the dataset description.

During pre-processing, it was found that there was no missing value in the dataset. We also checked the number of unique values for each column in the dataset, and it was found that, all the variables that were to be predicted, i.e., attack, category and subcategory were categorical in nature having two, five and eight classes respectively. Then, we also checked the coefficient of correlation between different columns of the dataset. We also visualised the dataset using bar charts and histogram.

After pre-processing and visualization, our first target was to predict the attack. For the prediction of attack as either 0 or 1 (0 denoting Normal traffic and 1 denoting attack), we divided the training dataset into 2 different data-frames, X and Y, X having all the independent columns (having ten best columns as specified in dataset description), and Y having the outcome variable, i.e., attack here. Then, the training dataset was split in the ratio 0.7:0.3 into training set and validation set. And, then the machine learning model is trained on the training set and validated on the validation set. After the successful validation, the model is tested on the tested dataset imported from the Google drive. Note that before testing the model on the testing set, the columns that were dropped from the training set were also dropped from the testing set.

After the prediction of attack, the columns category and subcategory were to be predicted. And, for that, we needed to convert them from object data type to numerical data type using some encoding technique. We have used LabelEncoder from Sklearn.preprocessing for encoding category and subcategory variables to numerical values.

The next step is the prediction of category of attack. For this, X (i.e. set of predictor variables) had the ten best attributes and the attack variable, because the attack variable is essential to predict the category of attack. Y (i.e. predicted variable here) had the category column. And the machine learning model was trained on the newly built X and Y. Now, in the test dataset, the predicted value of attack in the first step was appended in the columns of testing X so that accurate prediction could be made, and then category was predicted.

Similarly, to predict the value of subcategory, the model was trained on the training set with category as an additional variable to the training set in the previous step because category variable is essential to predict the subcategory of the attack. And to predict the subcategory on the testing set, the predicted category in the previous step was appended to the testing set.

In the end, after predicting attack, category and subcategory, the actual accuracy of the machine learning model was calculated taking into account, that three of the predicted variables must match with the outcome given in the testing set for a successful prediction.

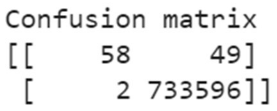
The same scheme of prediction was used on all the machine learning models and the accuracy of all the models was compared. All the results obtained from different classifier models and their comparison has been shown in section 6.

6. RESULTS AND DISCUSSION

We have constructed three confusion matrices for each model, the first one is for attack (either 0 or 1), second is for category of attack and last is for subcategory of attack.
Let us look at attack for each model.

6.1 Decision Tree

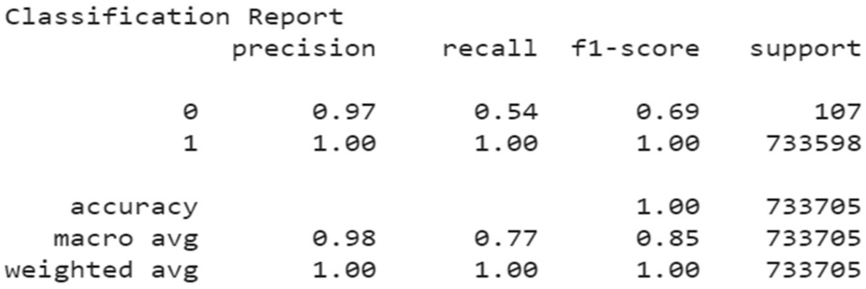
Confusion matrix:



```
Confusion matrix
[[    58     49]
 [     2 733596]]
```

If we consider the confusion matrix formed after predicting attack variable, 49 values for 0 have been predicted wrong.

Classification report:

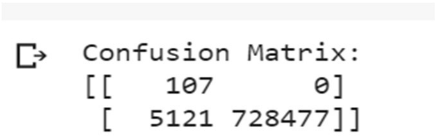


```
Classification Report
              precision    recall  f1-score   support

           0       0.97      0.54      0.69       107
           1       1.00      1.00      1.00    733598

    accuracy                           1.00    733705
   macro avg       0.98      0.77      0.85    733705
weighted avg       1.00      1.00      1.00    733705
```

It can be seen from the classification report, that the value of precision is good but recall is low.

6.2 Naïve Bayes

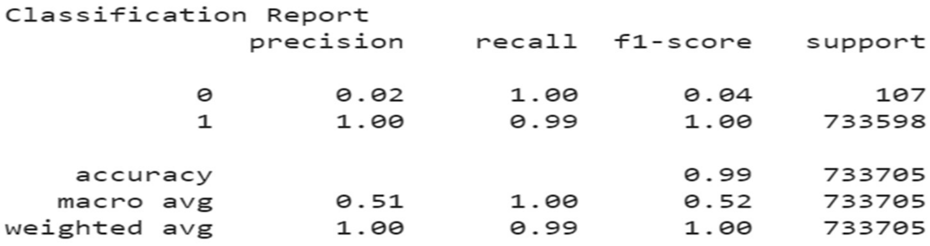
Confusion matrix:



```
Confusion Matrix:
[[   107      0]
 [  5121 728477]]
```

All zeroes are predicted correctly but accuracy for 1 has been reduced.

Classification report:

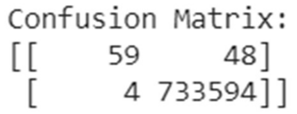


```
Classification Report
              precision    recall  f1-score   support

           0       0.02      1.00      0.04       107
           1       1.00      0.99      1.00    733598

    accuracy                           0.99    733705
   macro avg       0.51      1.00      0.52    733705
weighted avg       1.00      0.99      1.00    733705
```

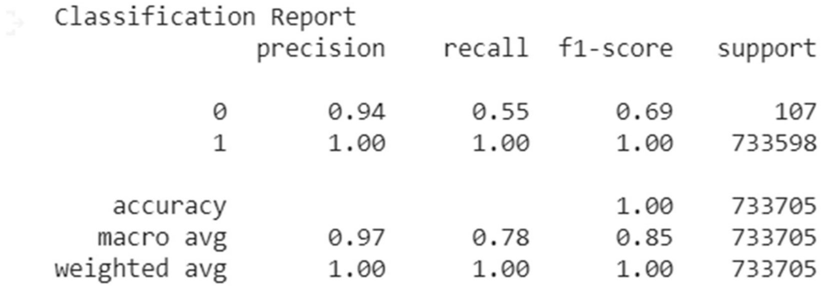
Thus, precision is substantially less, as can be seen from the above report.

6.3 KNN

The classification report and the confusion matrix for KNN is given as follows:



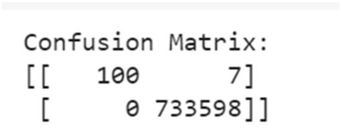
```
Confusion Matrix:
[[     59     48]
 [      4 733594]]
```



```
Classification Report
              precision    recall  f1-score   support

           0       0.94      0.55      0.69       107
           1       1.00      1.00      1.00    733598

    accuracy                           1.00    733705
   macro avg       0.97      0.78      0.85    733705
weighted avg       1.00      1.00      1.00    733705
```

The value of precision is good but recall is less for attack=0.

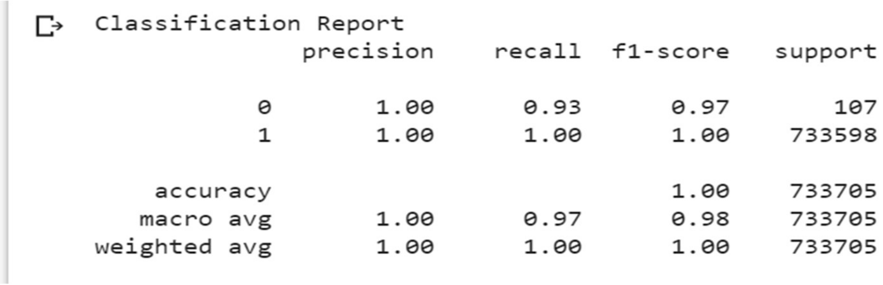
6.4 Random Forest



```
Confusion Matrix:
[[    100      7]
 [      0 733598]]
```

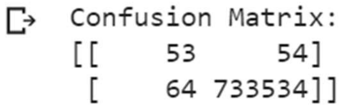
We can see that 0 and 1 are predicted quite accurately by Random forest classifier.

Classification report:



```
Classification Report
              precision    recall  f1-score   support

           0       1.00      0.93      0.97       107
           1       1.00      1.00      1.00    733598

    accuracy                           1.00    733705
   macro avg       1.00      0.97      0.98    733705
weighted avg       1.00      1.00      1.00    733705
```

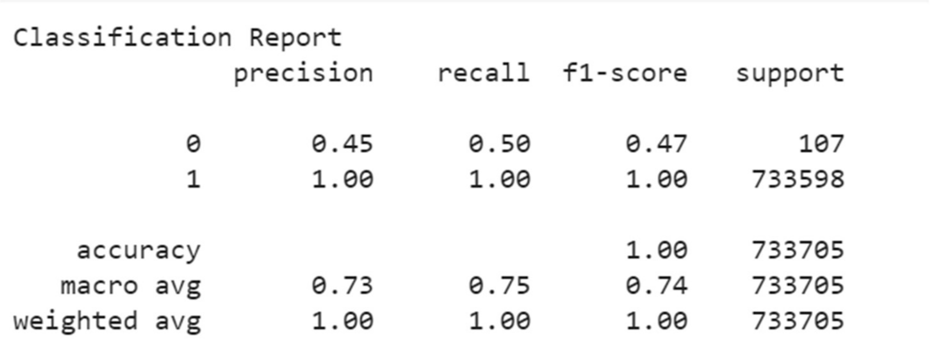
6.5 Gradient Boosting



```
Confusion Matrix:
[[     53     54]
 [     64 733534]]
```

There are 54 values which are wrongly predicted as 1 and 64 values wrongly predicted as 0.

Classification report:



```
Classification Report
              precision    recall  f1-score   support

           0       0.45      0.50      0.47       107
           1       1.00      1.00      1.00    733598

    accuracy                           1.00    733705
   macro avg       0.73      0.75      0.74    733705
weighted avg       1.00      1.00      1.00    733705
```

Thus, precision and recall values have been reduced.

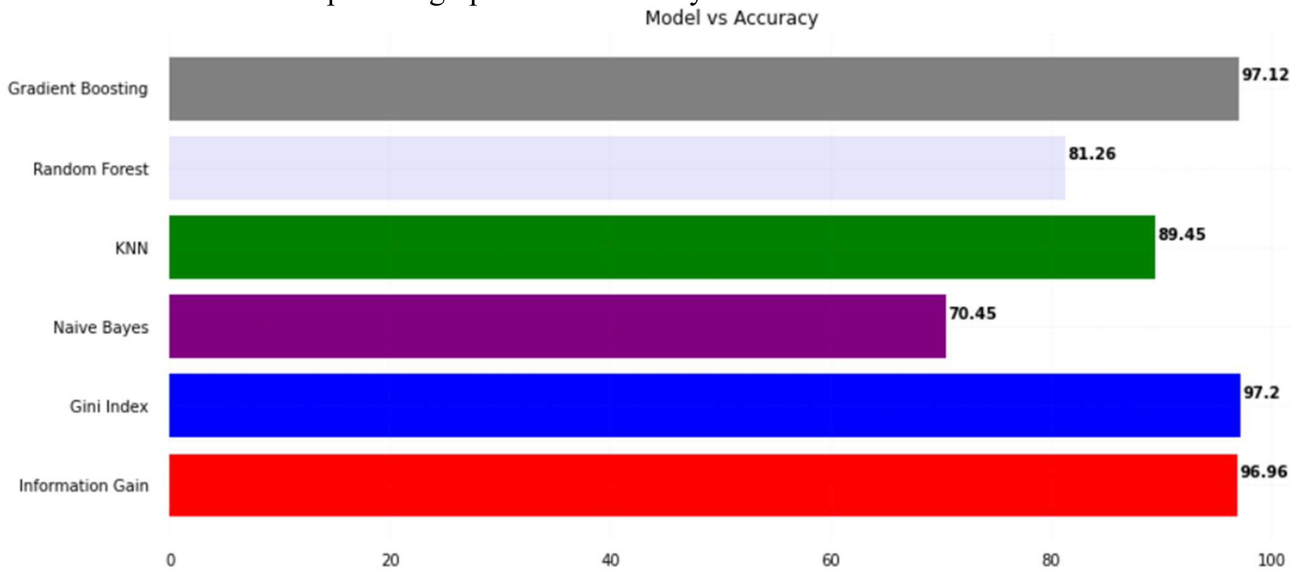
After predicting attack, category and subcategory were predicted using the method mentioned in experimentation section, and it can be seen that the net accuracy of every model is lesser than the individual accuracies of prediction of attack, category and subcategory as mentioned in the below written table.

Classifier	Attack Prediction	Category Prediction	Subcategory Prediction	Net Accuracy
Information Gain	99.96	97.26	99.73	96.96
Gini Index	99.99	97.45	99.74	97.20
Naïve Bayes	99.30	70.86	81.37	70.45*
K-NN	99.99	89.45	71.92	89.45*
Random Forest	99.99	81.26	98.16	81.26
Gradient Boosting	99.98	99.13	97.92	97.12

*The net accuracy is for the attack and category column only.

As we can see, Gini Index gives us highest accuracy followed by Gradient Boosting. Naïve Bayes gives us lowest accuracy.

Let us now look at the comparison graph of net accuracy of different models:



7. CONCLUSION AND FUTURE SCOPE

With the fast improvement of the IoT technologies, cyber-attacks are mostly focused on those sorts of devices. The botnets attacks are massively hard in those environments.

After infecting the botnets, the attackers can manipulate those devices. Our proposed prediction of assault approach can save you this from going on and it can be prolonged to locate any type of recent attacks as well. The Bot-IoT dataset consists of each everyday IoT-associated and different community visitors. Along with that, it additionally has numerous sorts of assault visitors which can be generally used by botnets.

The prediction of attack variable is a binary classification. While the category and subcategory prediction are multiclass classification.

In the future, with further optimization of those models, we consider that better outcomes may be achieved. We plan to work with different models inclusive of SVM. We additionally plan to develop a network forensic version using deep learning and evaluate its reliability using the BoT-IoT dataset.