NETWORK INTRUSION DETECTION SYSTEM USING MACHINE LEARNING

for

INFORMATION SECURITY ANALYSIS AND AUDIT (CSE3501)

in

B.Tech – Computer Science and Engineering

By

Jay Nitin Kaoshik
(20BCT0258)

Utkarsh Singh
(20BCI0284)

Mridul Madnani
(20BDS0191)

Rajeev Prakash
(20BCI0062)

In

Slot - F2

FACULTY: **Dr. SIVAKUMAR. N**

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING (SCOPE)

Abstract

Intrusion Detection System (IDS) is defined as a Device or software application which monitors the network or system activities and finds out if there is any malicious activities occurring. Exponential growth and usage of internet raises concerns about how to communicate and protect the digital information safely. In today's world hackers use different types of attacks for getting valuable information. The main objective of this project is to detect types of attacks and ensure detection and prevention of intrusion from the intruder. In this project we are using the IDS-2018 Dataset and NSL KDD dataset to test our models usingdeep learning algorithms and some machine learning algorithms.

Keywords: Machine learning, Deep learning, NSL KDD, IDS 2018, Feature extraction, Datapreprocessing

Introduction

Intrusion detection plays a vital role in the network defense process by alarming security administrators and forewarning them about malicious behaviors such as intrusions, attacks, and malware. Research in IDS has, hence, flourished over the years. An Intrusion detection system orIDS is a system developed to monitor for suspicious activity and issues alerts when such activity is discovered. The primary aim of IDS is to detect anomalous activities, but some systems are also able to act against these intrusions like blocking traffic from the suspicious IP address. An IDS can also be used to help analyze the quantity and types of attacks. Organizations can use this data to change their security systems or implement more effectivesystems. An IDS can also help organizations to identify bugs or problems with their network device configurations.

Base Paper Analysis

Problem Addressed:

A high-level view of the 5G threat landscape highlights security challenges and network segments that are at risk. Threats can be broken into categories based on which parts ofthe network they are impacting like: User Equipment Threats, Cloud Radio Access Network Threats, Core Network Threats, Network Slicing Threats, SDN (Software Defined Networking) Threats

Techniques used:

This paper proposes **SDS (Software Defined Security)** as a means to provide an automated, flexible and scalable network defense system. SDS will control current advances in machine learning to design a CNN using NAS to detect anomalous network traffic. SDS can be applied toan intrusion detection system to create a more proactive and end-to-end defense for a 5G network.

Experimental setup (Platform/datasets):

The **IDS-2018 data set** from the **Canadian Institute of Cybersecurity** is a data set derived from a simulated environment that attempts to address these shortcomings. This data set is used to for a systematic approach to generate a diverse and comprehensive benchmark data set for intrusion detection based on the creation of benign traffic and malicious traffic profiles.
Platform- AWS cloud platform, Windows, Linux.

Advantages of this Study:

CNN architecture using NAS gives the best accuracy levels when compared with other techniques. It is efficient, fast to implement, and also reduces a lot of pre-processing work. Thefactorized hierarchical search space in MnasNet adds layer diversity throughout the network and attempts to balance the size of the total search space, bringing more flexibility to NAS so that they can be designed to balance speed vs. accuracy.

Disadvantages of this Study:

The encryption of network traffic can be a disadvantage as malicious users can employ encryption to evade detection and secure their malicious activities. The issue then in terms of security is that the majority of organisations do not have the tools or solutionsto manage potentially malicious encrypted traffic and systems are not in place that have
the ability to effectively detect malicious encrypted traffic without performance impacts tothe network.

LITERATURE SURVEY:

[1] S. Zwane, P. Tarwireyi and M. Adigun, "Performance Analysis of Machine Learning Classifiers for Intrusion Detection," *2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC)*, 2018, pp. 1-5, doi: 10.1109/ICONIC.2018.8601203.

Machine algorithm methods have shown to be prominent way of detection of false and malicious nodes in a hostile environment. The paper proposes the following machine learning models that are used: Multi-Layer Perceptron, Bayesian Network, SVM, Adaboost, Random forest , Bootstrap Aggregation and Decision Trees. The dataset used in this paper is the UNSW-NB15 network datasets which contains nine different attacks fields upon which a model can be trained.

These attack vectors are: Fuzzers, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, Worms, among others. The dataset also consists of 45 different fields like duration of attack, prototype used, rate, data bytes, synack, ackdat, etc.

The proposed Machine Learning models were trained and tested upon the WEKA software. Weka is a collection of machine learning algorithms for data mining tasks. It contains tools for data preparation, classification, regression, clustering, association rules mining, and visualization. The models were trained upon WEKA using the database 6 times for each classifier and the average is used for consistency.

The results were then analyzed which showed that ensemble classifiers performed better than the single learning methods although the latter proved to be more efficient for model development and testing.

[2] S. Kumar, A. Viinikainen and T. Hamalainen, "Machine learning classification model for Network based Intrusion Detection System," 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST), 2016, pp. 242-249, doi: 10.1109/ICITST.2016.7856705.

This research paper focuses on Android based malwares. The dataset used in this paper is created using real traffic and attributes obtained from the bidirectional flow of malicious communications and interactions. The traffic was created using infected samples including threats such as bots, fake antivirus programs, backdoor, span transmitter, etc.

The paper further discusses some of the common datasets that are used to create Android malware traffic which acts as a foundation for a Network Intrusion and Detection System. The author then begins to describe his methodology which describes the preprocessing of dataset and feature extraction.

For the construction of the Machine learning classifiers, rule-based algorithms like decision trees were selected. The ML algorithms particularly selected were J48, PART, Random Forest Algorithm and RIDOR. The parameters for consideration of these algorithms were time to train and effectiveness in other domains. The author performed various experiments on eight different datasets.

The first experiment was cross-validation which was performed on dataset 1 to measure the efficiency when the training subset was used for testing as well. The second experiment involved splitting dataset into test and train sections for improvised training. The third and fourth experiments involved training the classifiers on a labelled dataset. Both percentage split and cross-validation were performed on dataset 2. Experiment 5 involved creating a new dataset. This experiment checked the performance of Random forest and PART algorithms. Experiment 6 involved testing the classifiers on unseen dataset to compare their accuracy against
Anti-Malware solutions. The results were then discussed.

[3] Das, Saikat & Mahfouz, Ahmed & Venugopal, Deepak & Shiva, Sajjan. (2019). DDoS Intrusion Detection through Machine Learning Ensemble. 10.1109/QRS-C.2019.00090.

A Network Intrusion Detection System is implemented that specifically aims at identifying Distributed Denial of Service Attacks (DDoS). The proposed methodology involves combining different classifiers using ensemble models and each classifier would further help in detection of specific attributes of the Intrusions.

The task of ensemble machine learning refers to a process which combines multiple base classifying models to produce one enhanced model. The proposed methodology used 3 ML classifiers to produce an ensemble model which are: Multi- Layered Perceptron (MLP), Sequential minimal optimization (SMO), IBK and J48 (decision tree-based classifying algorithm). Each classifier used builds a different model of the data.

The dataset used by the authors was the NSL-KDD dataset which is an improvised version of the KDD’99 dataset. Preprocessing is further performed to convert the non-numeric values into numerical feature sets. The attributes that were relevant to the DDoS attacks like: Back, Land, Neptune, Smurf and Teardrop. Each model was tested using the 10-folds cross validation method. 9 subsets were used for model training and the last one was used for testing. A comparison analysis was first established upon individual classifiers. J48 provided the best accuracy in this case. A graphical tabulation showed that the ensemble classifier had a better prediction rate (99.77%) than the individual classifiers. The RoC curves were also analyzed for both type of classification mechanisms.

CONCEPTS USED

DECISION TREE APPROACH

A decision tree model can be built to partition the data using information gain until instancesin each leaf node have uniform class labels. This is a very simple but yet an effective hierarchical method for supervised learning (classification or regression) whereby the local space (region) is recognized in a sequence of repetitive splits in a reduced number of steps (small). At each test, a single feature is used to split the nodeaccording to the feature values.

The generation process of a decision tree done by recursively splitting on features is equivalent to dividing the original training set into smaller sets recursively until the entropyof every one of these subsets is zero (i.e everyone will have instances from a single class target).

A Decision Tree is made up of internal decision nodes and terminal leaves. A test functionis implemented by each decision node with discrete results labelling the branches. Providing an input, at every node, a test is constructed and based on the outcome, one ofthe branches will be considered. Here the learning algorithm starts at the root and until a leaf node is reached.

Algorithm:

Generating a decision tree form training tuples of data partition D, Algorithm : Generate_decision_tree

Input:
Data partition, D, which is a set of training tuples and their associated class labels.attribute_list, the set of candidate attributes. Attribute selection method, a procedure to determine the splitting criterion that bestpartitions that the data tuples into individual classes. This criterion includes a splitting attribute and either a splitting point or splitting subset.

Output:
A Decision Tree

Method
```
create a node N;

if tuples in D are all of the same class, C then
   return N as leaf node labeled with class C;
```

```
if attribute_list is empty then
    return N as leaf node with labeled
    with majority class in D;|| majority voting

  apply attribute_selection_method(D, attribute_list) tofind
  the best splitting_criterion;
  label node N with splitting_criterion;

  if splitting_attribute is discrete-valued and
    multiway splits allowed then // no restricted to binary trees

  attribute_list = splitting attribute; // remove splitting attribute for
  each outcome j of splitting criterion

    // partition the tuples and grow subtrees for each partition
    let Dj be the set of data tuples in D satisfying outcome j; // a partition

    if Dj is empty then
      attach a leaf labeled with the majorityclass in
      D to node N;
    else
      attach the node returned by Generate decision
      tree(Dj, attribute list) to node N;
    end for
  return N;
```

<u>RANDOM FOREST CLASSIFIER</u>:

This algorithm is a modification of decision trees. A certain number of new datasets are derived from the original dataset containing the same number of records as the original one. The records in each derived dataset is chosen randomly. A decision tree is constructed for each dataset separately. The process of creating new datasets is called bootstrapping. While testing, when the input data is provided, it runs on each of the derived decision tree and the class label with the majority occurrences is voted to be the resulting label. A random forest algorithm consists of many decision trees. The ‘forest’ generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms.It predicts by taking the average or mean of the output from various trees.

<u>Pseudocode:</u>

```
1) Randomly select “k” features from total “m” features.
                    Where k << m

2) Among the “k” features, calculate the node “d” using the best split point.
3) Split the node into daughter nodes using the best split.
4) Repeat 1 to 3 steps until “l” number of nodes has been reached.
5) Build forest by repeating steps 1 to 4 for “n” number times to create “n”
   number of trees.
```

Random forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features.

NAÏVE BAYES:

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. Bayes' theorem states the following relationship, given class variable y and dependent feature vector x1 through xn:

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) P(x_1, \dots, x_n \mid y)}{P(x_1, \dots, x_n)}$$

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of P(xi | y).

In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters. (For theoretical reasons why naive Bayes works well, and on which types of data it does, see the references below.)

Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality.

ADABOOST CLASSIFIER:

An AdaBoost [1] classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

The principle behind boosting algorithms is first we built a model on the training dataset, then a second model is built to rectify the errors present in the first model. This procedure is continued until and unless the errors are minimized, and the dataset is predicted correctly. AdaBoost is a boosting algorithm. What this algorithm does is that it builds a model and gives equal weights to all the data points. It then assigns higher weights to points that are wrongly classified. Now all the points which have higher weights are given more importance in the next model. It will keep training models until and unless a lowe error is received.

GRADIENT BOOSTING CLASSIFIER:

This algorithm builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage n_classes_ regression trees are fit on the negative gradient of the loss function, e.g. binary or multiclasss log loss. Binary classification is a special case where only a single regression tree is induced. Gradient boosting is a method standing out for its prediction speed and accuracy, particularly with large and complex datasets. Gradient boosting has a fixed base estimator i.e., Decision Trees whereas in AdaBoost we can change the base estimator according to our needs.

The first step in gradient boosting is to build a base model to predict the observations in the training dataset. For simplicity we take an average of the target column and assume that to be the predicted value. The next step is to calculate the pseudo residuals which are (observed value – predicted value). In the next step, we build a model on these pseudo residuals and make predictions. We then find the output values for each leaf of our decision tree. That means there might be a case where 1 leaf gets more than 1 residual, hence we need to find the final output of all the leaves. In the final step, we have to update the predictions of the previous model. It can be updated as:

$$F_m(x) = F_{m-1}(x) + \nu_m h_m(x)$$

Here Fm-1(x) is the prediction of the base model (previous prediction).
Vm is the learning rate and Hm(x) is the recent DT made on the residuals.

<u>MULTI LAYER PERCEPTRON</u>:

Multilayer Perceptron is a Neural Network that learns the relationship between linear and non-linear data. A Multilayer Perceptron has input and output layers, and one or more hidden layers with many neurons stacked together. And while in the Perceptron the neuron must have an activation function that imposes a threshold, like ReLU or sigmoid, neurons in a Multilayer Perceptron can use any arbitrary activation function. Multilayer Perceptron falls under the category of feedforward algorithms, because inputs are combined with the initial weights in a weighted sum and subjected to the activation function, just like in the Perceptron. But the difference is that each linear combination is propagated to the next layer.

Each layer is feeding the next one with the result of their computation, their internal representation of the data. This goes all the way through the hidden layers to the output layer. Backpropagation is the learning mechanism that allows the Multilayer Perceptron to iteratively adjust the weights in the network, with the goal of minimizing the cost function.

There is one hard requirement for backpropagation to work properly. The function that combines inputs and weights in a neuron, for instance the weighted sum, and the threshold function, for instance ReLU, must be differentiable. These functions must have a bounded derivative, because Gradient Descent is typically the optimization function used in MultiLayer Perceptron. In each iteration, after the weighted sums are forwarded through all layers, the gradient of the Mean Squared Error is computed across all input and output pairs. Then, to propagate it back, the weights of the first hidden layer are updated with the value of the gradient. That’s how the weights are propagated back to the starting point of the neural network.

<u>XGBOOST CLASSIFIER</u>:

XGBoost is an implementation of Gradient Boosted decision trees. In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems. XGBoost is a scalable and highly accurate implementation of gradient boosting that pushes the limits of computing power for boosted tree algorithms, being built largely for energizing machine learning model performance and computational speed. With XGBoost, trees are built in parallel, instead of sequentially like GBDT. It follows a level-wise strategy, scanning across gradient values and using these partial sums to evaluate the quality of splits at every possible split in the training set.

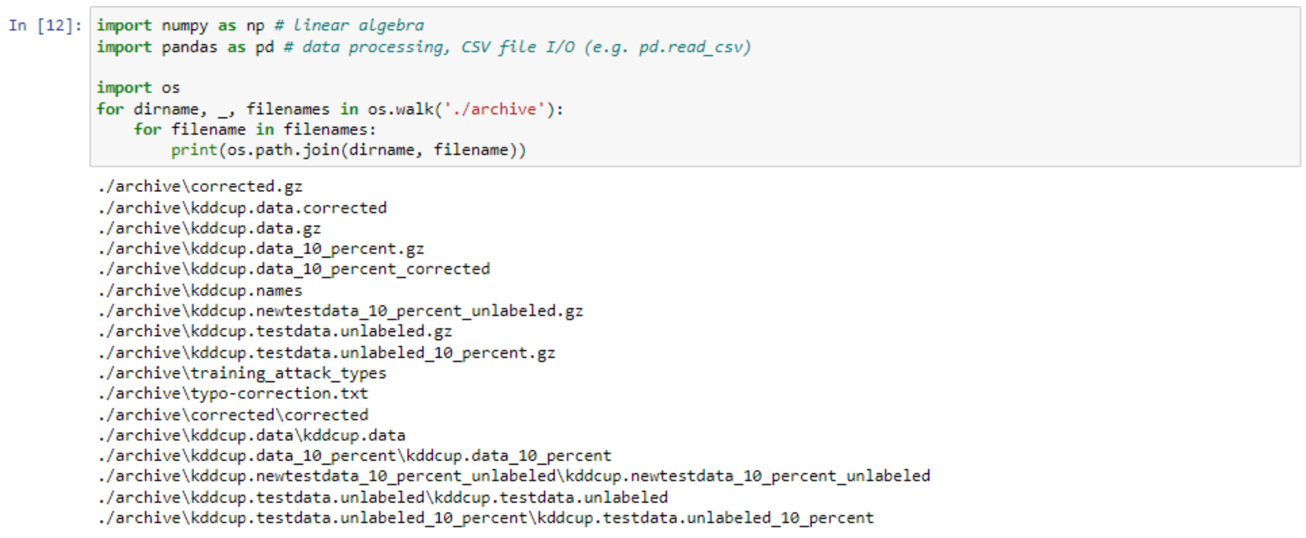
Implementation

A security mechanism can be implemented using an **Intrusion Detection System (IDS)** which can be described as a collection of software or hardware devices able to collect, analyze and detect any unwanted, suspicious, or malicious traffic either on a particular computer host or network. Therefore to achieve its task, an IDS should use some statistical or mathematical method to read and interpret the information itcollects and subsequently reports anymalicious activity to the network administrator.

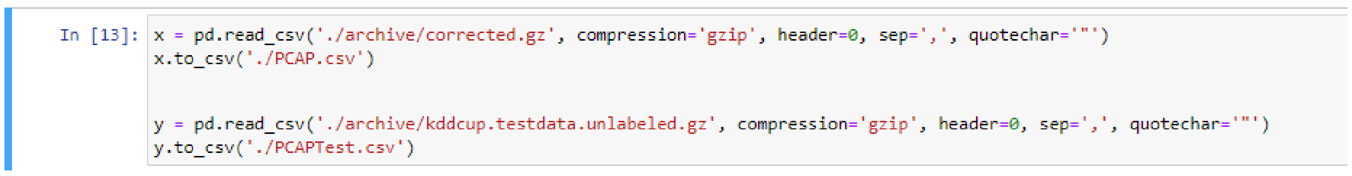
<u>Using Machine Learning Approach</u>

A data clean process can require a tremendous human effort, which is an extensive time consuming and expensive. A machine learning approach and data mining technique whichis the application of machine learning methods to large databases are widely known and used to reduce or eliminate the need of a human interaction.

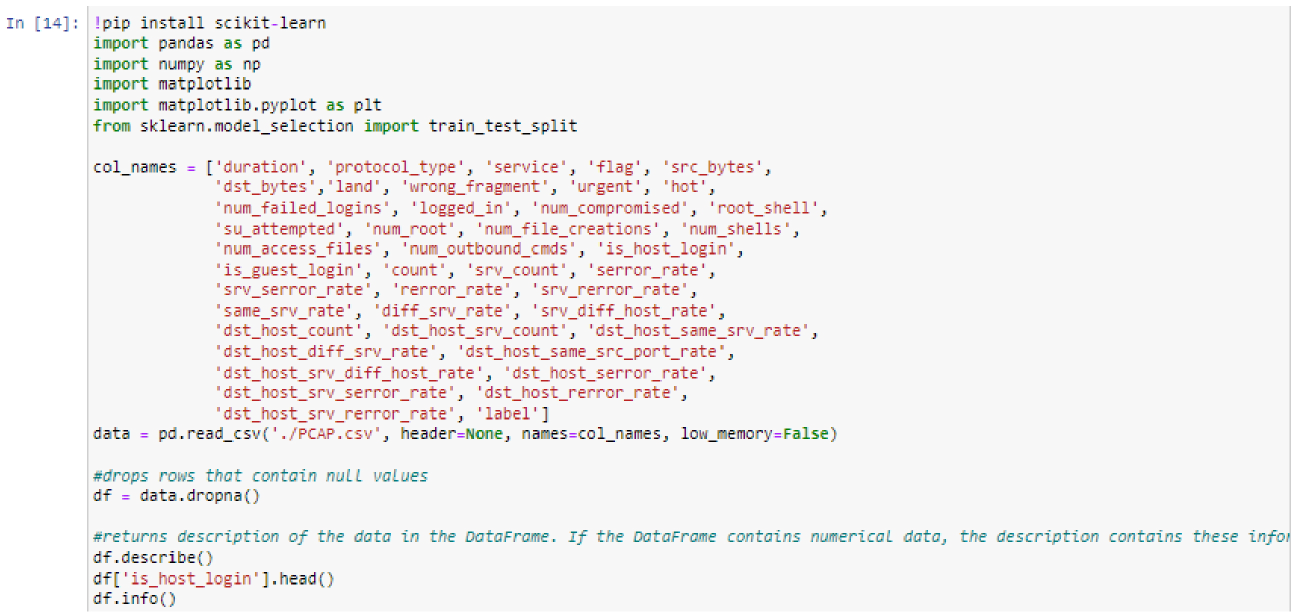
Machine learning helps to optimize performance criterion using example data or past experience using a computer program, models are defined with some parameters, and learning is the execution of the programming computer to optimize the parameters of the model using training data. The model can be predictive to make predictions in the future, or descriptive to gain knowledge from data.



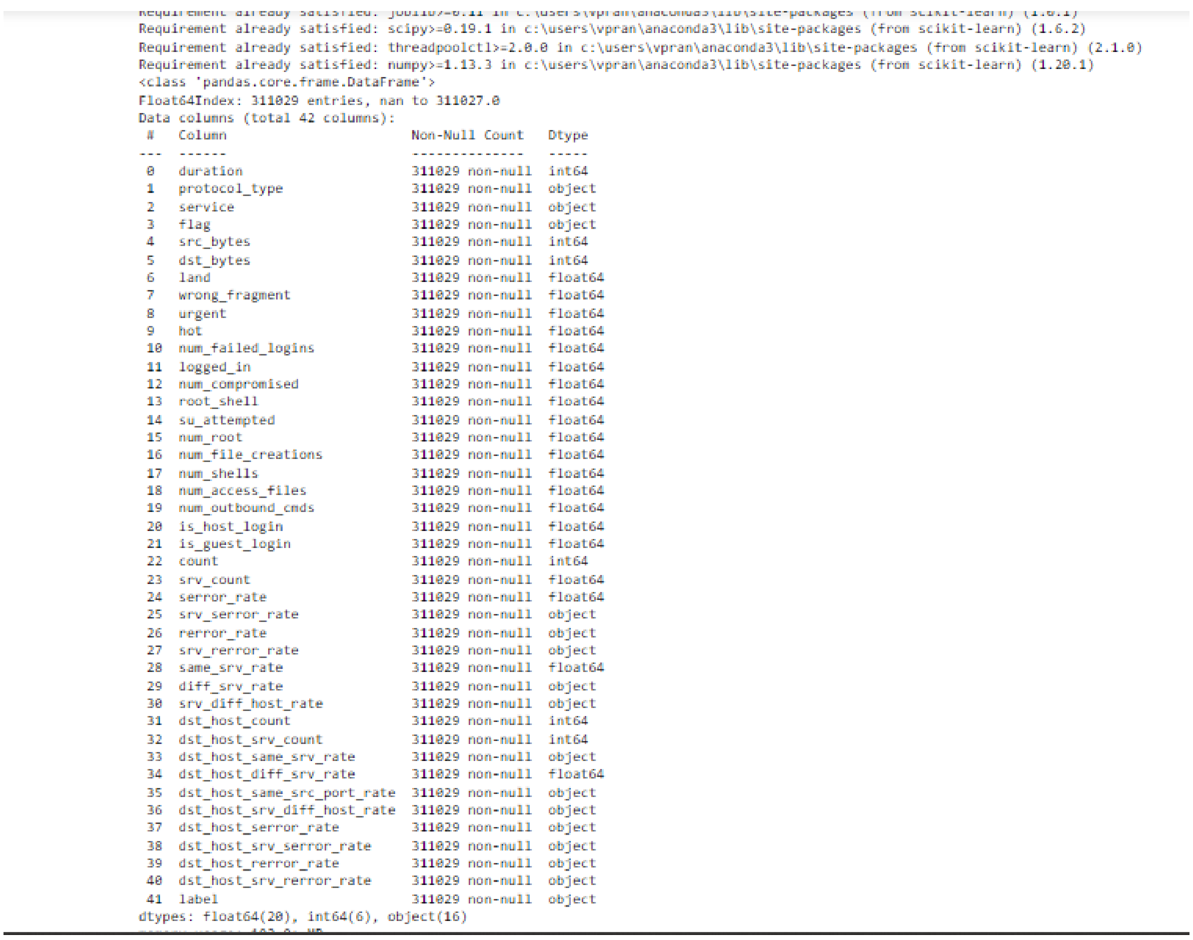
```
In [12]: import numpy as np # linear algebra
         import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

         import os
         for dirname, _, filenames in os.walk('./archive'):
             for filename in filenames:
                 print(os.path.join(dirname, filename))

./archive\corrected.gz
./archive\kddcup.data.corrected
./archive\kddcup.data.gz
./archive\kddcup.data_10_percent.gz
./archive\kddcup.data_10_percent_corrected
./archive\kddcup.names
./archive\kddcup.newtestdata_10_percent_unlabeled.gz
./archive\kddcup.testdata.unlabeled.gz
./archive\kddcup.testdata.unlabeled_10_percent.gz
./archive\training_attack_types
./archive\typo-correction.txt
./archive\corrected\corrected
./archive\kddcup.data\kddcup.data
./archive\kddcup.data_10_percent\kddcup.data_10_percent
./archive\kddcup.newtestdata_10_percent_unlabeled\kddcup.newtestdata_10_percent_unlabeled
./archive\kddcup.testdata.unlabeled\kddcup.testdata.unlabeled
./archive\kddcup.testdata.unlabeled_10_percent\kddcup.testdata.unlabeled_10_percent
```



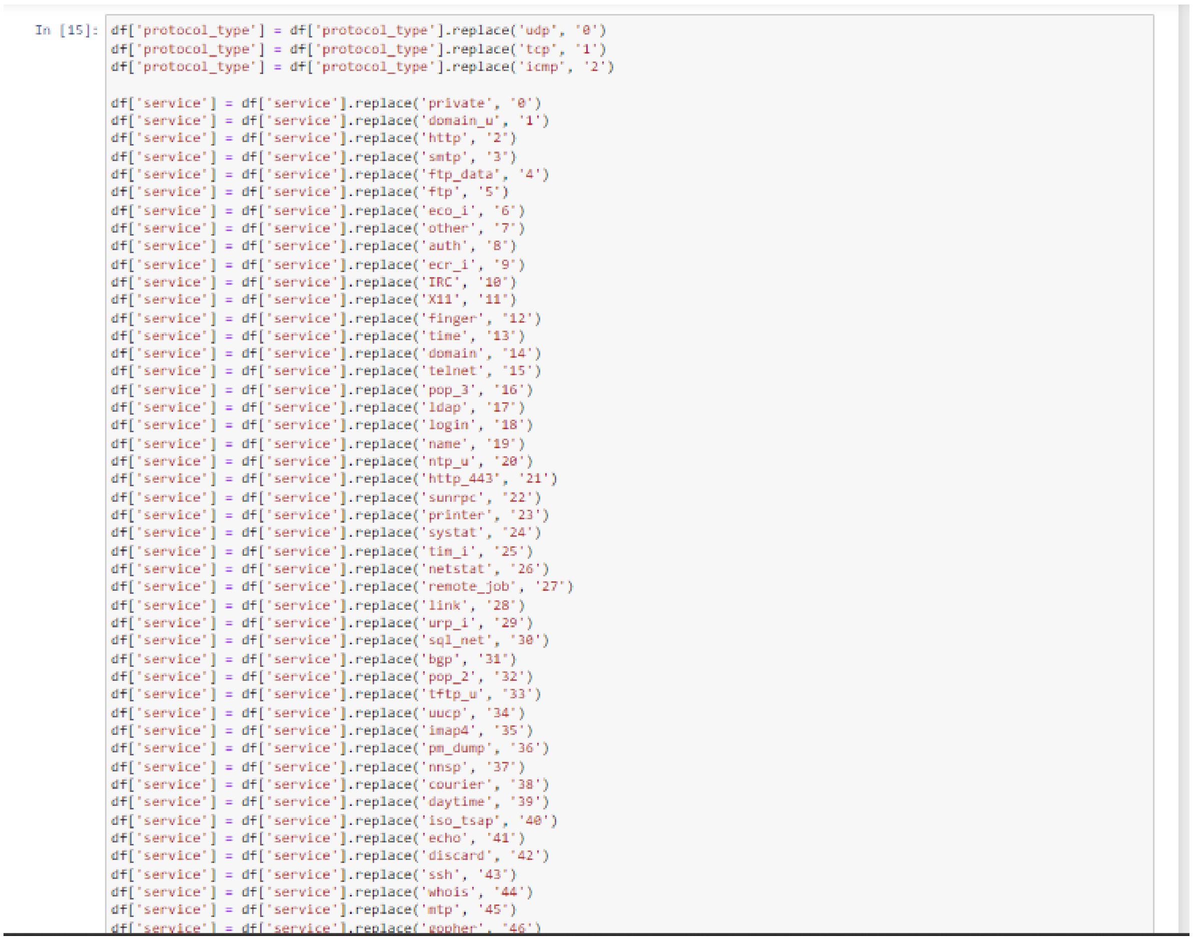
```
In [13]: x = pd.read_csv('./archive/corrected.gz', compression='gzip', header=0, sep=',', quotechar='"')
         x.to_csv('./PCAP.csv')


         y = pd.read_csv('./archive/kddcup.testdata.unlabeled.gz', compression='gzip', header=0, sep=',', quotechar='"')
         y.to_csv('./PCAPTest.csv')
```



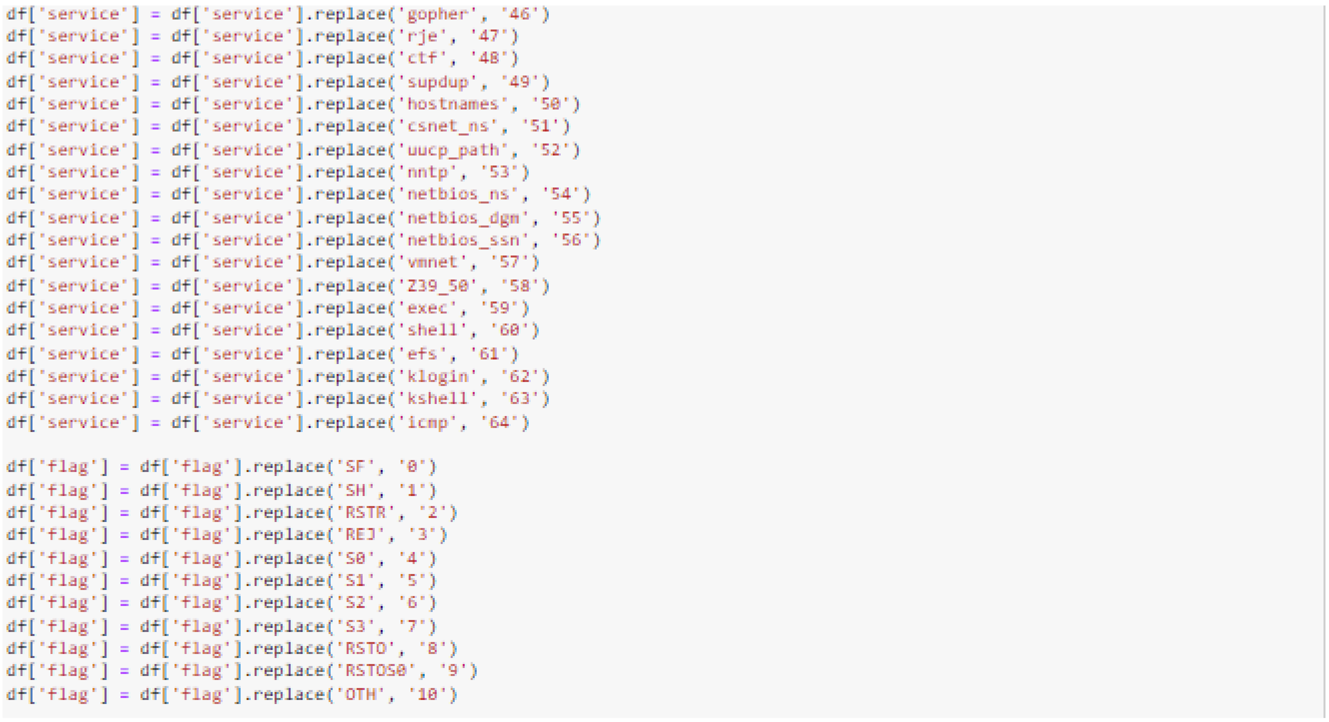
```
In [14]: !pip install scikit-learn
         import pandas as pd
         import numpy as np
         import matplotlib
         import matplotlib.pyplot as plt
         from sklearn.model_selection import train_test_split

         col_names = ['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
                      'dst_bytes','land', 'wrong_fragment', 'urgent', 'hot',
                      'num_failed_logins', 'logged_in', 'num_compromised', 'root_shell',
                      'su_attempted', 'num_root', 'num_file_creations', 'num_shells',
                      'num_access_files', 'num_outbound_cmds', 'is_host_login',
                      'is_guest_login', 'count', 'srv_count', 'serror_rate',
                      'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate',
                      'same_srv_rate', 'diff_srv_rate', 'srv_diff_host_rate',
                      'dst_host_count', 'dst_host_srv_count', 'dst_host_same_srv_rate',
                      'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
                      'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
                      'dst_host_srv_serror_rate', 'dst_host_rerror_rate',
                      'dst_host_srv_rerror_rate', 'label']
         data = pd.read_csv('./PCAP.csv', header=None, names=col_names, low_memory=False)

         #drops rows that contain null values
         df = data.dropna()

         #returns description of the data in the DataFrame. If the DataFrame contains numerical data, the description contains these infor
         df.describe()
         df['is_host_login'].head()
         df.info()
```



```
Requirement already satisfied: joblib>=0.11 in c:\users\vpran\anaconda3\lib\site-packages (from scikit-learn) (1.0.1)
Requirement already satisfied: scipy>=0.19.1 in c:\users\vpran\anaconda3\lib\site-packages (from scikit-learn) (1.6.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\vpran\anaconda3\lib\site-packages (from scikit-learn) (2.1.0)
Requirement already satisfied: numpy>=1.13.3 in c:\users\vpran\anaconda3\lib\site-packages (from scikit-learn) (1.20.1)
<class 'pandas.core.frame.DataFrame'>
Float64Index: 311029 entries, nan to 311027.0
Data columns (total 42 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   duration                     311029 non-null  int64
 1   protocol_type                311029 non-null  object
 2   service                      311029 non-null  object
 3   flag                         311029 non-null  object
 4   src_bytes                    311029 non-null  int64
 5   dst_bytes                    311029 non-null  int64
 6   land                         311029 non-null  float64
 7   wrong_fragment               311029 non-null  float64
 8   urgent                       311029 non-null  float64
 9   hot                          311029 non-null  float64
 10  num_failed_logins            311029 non-null  float64
 11  logged_in                    311029 non-null  float64
 12  num_compromised              311029 non-null  float64
 13  root_shell                   311029 non-null  float64
 14  su_attempted                 311029 non-null  float64
 15  num_root                     311029 non-null  float64
 16  num_file_creations           311029 non-null  float64
 17  num_shells                   311029 non-null  float64
 18  num_access_files             311029 non-null  float64
 19  num_outbound_cmds            311029 non-null  float64
 20  is_host_login                311029 non-null  float64
 21  is_guest_login               311029 non-null  float64
 22  count                        311029 non-null  int64
 23  srv_count                    311029 non-null  float64
 24  serror_rate                  311029 non-null  float64
 25  srv_serror_rate              311029 non-null  object
 26  rerror_rate                  311029 non-null  object
 27  srv_rerror_rate              311029 non-null  object
 28  same_srv_rate                311029 non-null  float64
 29  diff_srv_rate                311029 non-null  object
 30  srv_diff_host_rate           311029 non-null  object
 31  dst_host_count               311029 non-null  int64
 32  dst_host_srv_count           311029 non-null  int64
 33  dst_host_same_srv_rate       311029 non-null  object
 34  dst_host_diff_srv_rate       311029 non-null  float64
 35  dst_host_same_src_port_rate  311029 non-null  object
 36  dst_host_srv_diff_host_rate  311029 non-null  object
 37  dst_host_serror_rate         311029 non-null  object
 38  dst_host_srv_serror_rate     311029 non-null  object
 39  dst_host_rerror_rate         311029 non-null  object
 40  dst_host_srv_rerror_rate     311029 non-null  object
 41  label                        311029 non-null  object
dtypes: float64(20), int64(6), object(16)
```



```
In [15]: df['protocol_type'] = df['protocol_type'].replace('udp', '0')
         df['protocol_type'] = df['protocol_type'].replace('tcp', '1')
         df['protocol_type'] = df['protocol_type'].replace('icmp', '2')

         df['service'] = df['service'].replace('private', '0')
         df['service'] = df['service'].replace('domain_u', '1')
         df['service'] = df['service'].replace('http', '2')
         df['service'] = df['service'].replace('smtp', '3')
         df['service'] = df['service'].replace('ftp_data', '4')
         df['service'] = df['service'].replace('ftp', '5')
         df['service'] = df['service'].replace('eco_i', '6')
         df['service'] = df['service'].replace('other', '7')
         df['service'] = df['service'].replace('auth', '8')
         df['service'] = df['service'].replace('ecr_i', '9')
         df['service'] = df['service'].replace('IRC', '10')
         df['service'] = df['service'].replace('X11', '11')
         df['service'] = df['service'].replace('finger', '12')
         df['service'] = df['service'].replace('time', '13')
         df['service'] = df['service'].replace('domain', '14')
         df['service'] = df['service'].replace('telnet', '15')
         df['service'] = df['service'].replace('pop_3', '16')
         df['service'] = df['service'].replace('ldap', '17')
         df['service'] = df['service'].replace('login', '18')
         df['service'] = df['service'].replace('name', '19')
         df['service'] = df['service'].replace('ntp_u', '20')
         df['service'] = df['service'].replace('http_443', '21')
         df['service'] = df['service'].replace('sunrpc', '22')
         df['service'] = df['service'].replace('printer', '23')
         df['service'] = df['service'].replace('systat', '24')
         df['service'] = df['service'].replace('tim_i', '25')
         df['service'] = df['service'].replace('netstat', '26')
         df['service'] = df['service'].replace('remote_job', '27')
         df['service'] = df['service'].replace('link', '28')
         df['service'] = df['service'].replace('urp_i', '29')
         df['service'] = df['service'].replace('sql_net', '30')
         df['service'] = df['service'].replace('bgp', '31')
         df['service'] = df['service'].replace('pop_2', '32')
         df['service'] = df['service'].replace('tftp_u', '33')
         df['service'] = df['service'].replace('uucp', '34')
         df['service'] = df['service'].replace('imap4', '35')
         df['service'] = df['service'].replace('pm_dump', '36')
         df['service'] = df['service'].replace('nnsp', '37')
         df['service'] = df['service'].replace('courier', '38')
         df['service'] = df['service'].replace('daytime', '39')
         df['service'] = df['service'].replace('iso_tsap', '40')
         df['service'] = df['service'].replace('echo', '41')
         df['service'] = df['service'].replace('discard', '42')
         df['service'] = df['service'].replace('ssh', '43')
         df['service'] = df['service'].replace('whois', '44')
         df['service'] = df['service'].replace('mtp', '45')
         df['service'] = df['service'].replace('gopher', '46')
```



```
df['service'] = df['service'].replace('gopher', '46')
df['service'] = df['service'].replace('rje', '47')
df['service'] = df['service'].replace('ctf', '48')
df['service'] = df['service'].replace('supdup', '49')
df['service'] = df['service'].replace('hostnames', '50')
df['service'] = df['service'].replace('csnet_ns', '51')
df['service'] = df['service'].replace('uucp_path', '52')
df['service'] = df['service'].replace('nntp', '53')
df['service'] = df['service'].replace('netbios_ns', '54')
df['service'] = df['service'].replace('netbios_dgm', '55')
df['service'] = df['service'].replace('netbios_ssn', '56')
df['service'] = df['service'].replace('vmnet', '57')
df['service'] = df['service'].replace('Z39_50', '58')
df['service'] = df['service'].replace('exec', '59')
df['service'] = df['service'].replace('shell', '60')
df['service'] = df['service'].replace('efs', '61')
df['service'] = df['service'].replace('klogin', '62')
df['service'] = df['service'].replace('kshell', '63')
df['service'] = df['service'].replace('icmp', '64')

df['flag'] = df['flag'].replace('SF', '0')
df['flag'] = df['flag'].replace('SH', '1')
df['flag'] = df['flag'].replace('RSTR', '2')
df['flag'] = df['flag'].replace('REJ', '3')
df['flag'] = df['flag'].replace('S0', '4')
df['flag'] = df['flag'].replace('S1', '5')
df['flag'] = df['flag'].replace('S2', '6')
df['flag'] = df['flag'].replace('S3', '7')
df['flag'] = df['flag'].replace('RSTO', '8')
df['flag'] = df['flag'].replace('RSTOS0', '9')
df['flag'] = df['flag'].replace('OTH', '10')
```

```
df['label'] = df['label'].replace(['back.', 'land.', 'neptune.', 'pod.','smurf.', 'teardrop.', 'apache2.','udpstorm.','processta
#10 DoS

df['label'] = df['label'].replace(['satan.', 'ipsweep.', 'nmap.', 'portsweep.', 'mscan.', 'saint.'], '2')
#6 Probe

df['label'] = df['label'].replace(['guess_passwd.', 'ftp_write.', 'imap.', 'phf.', 'multihop.', 'warezmaster.', 'warezclient.',
#16 R2L

df['label'] = df['label'].replace(['buffer_overflow.', 'loadmodule.', 'rootkit.', 'perl.', 'sqlattack.', 'xterm.', 'ps.'], '4')
#7 U2R

df['label'] = df['label'].replace('normal.', '0')
#1 Normal

df = df.drop(df.index[:2])

#Gives the information about first 10 records.
df.head(10)

#converts non-essential non-numeric datafields into numeric datafields.
df["protocol_type"] = pd.to_numeric(df["protocol_type"])
df["service"] = pd.to_numeric(df["service"])
df["flag"]= pd.to_numeric(df["flag"])
df["label"] = pd.to_numeric(df["label"])
df["dst_host_srv_rerror_rate"] = pd.to_numeric(df["dst_host_srv_rerror_rate"])
df["dst_host_rerror_rate"] = pd.to_numeric(df["dst_host_rerror_rate"])
df["dst_host_srv_serror_rate"] = pd.to_numeric(df["dst_host_srv_serror_rate"])
df["dst_host_serror_rate"] = pd.to_numeric(df["dst_host_serror_rate"])
df["dst_host_srv_diff_host_rate"] = pd.to_numeric(df["dst_host_srv_diff_host_rate"])
df["dst_host_same_src_port_rate"] = pd.to_numeric(df["dst_host_same_src_port_rate"])
df["dst_host_same_srv_rate"] = pd.to_numeric(df["dst_host_same_srv_rate"])
df["srv_diff_host_rate"] = pd.to_numeric(df["srv_diff_host_rate"])
df["diff_srv_rate"] = pd.to_numeric(df["diff_srv_rate"])
df["srv_serror_rate"]= pd.to_numeric(df["srv_serror_rate"])
df["srv_rerror_rate"]= pd.to_numeric(df["srv_rerror_rate"])
df["rerror_rate"]= pd.to_numeric(df["rerror_rate"])

#displays the number of columns corresponding to a particular datatype.
df.info()

# df.head(10)
```

```
<class 'pandas.core.frame.DataFrame'>
Float64Index: 311027 entries, 1.0 to 311027.0
Data columns (total 42 columns):
 #   Column                       Non-Null Count   Dtype  
---  ------                       --------------   -----  
 0   duration                     311027 non-null  int64  
 1   protocol_type                311027 non-null  int64  
 2   service                      311027 non-null  int64  
 3   flag                         311027 non-null  int64  
 4   src_bytes                    311027 non-null  int64  
 5   dst_bytes                    311027 non-null  int64  
 6   land                         311027 non-null  float64
 7   wrong_fragment               311027 non-null  float64
 8   urgent                       311027 non-null  float64
 9   hot                          311027 non-null  float64
 10  num_failed_logins            311027 non-null  float64
 11  logged_in                    311027 non-null  float64
 12  num_compromised              311027 non-null  float64
 13  root_shell                   311027 non-null  float64
 14  su_attempted                 311027 non-null  float64
 15  num_root                     311027 non-null  float64
 16  num_file_creations           311027 non-null  float64
 17  num_shells                   311027 non-null  float64
 18  num_access_files             311027 non-null  float64
 19  num_outbound_cmds            311027 non-null  float64
 20  is_host_login                311027 non-null  float64
 21  is_guest_login               311027 non-null  float64
 22  count                        311027 non-null  int64  
 23  srv_count                    311027 non-null  float64
 24  serror_rate                  311027 non-null  float64
 25  srv_serror_rate              311027 non-null  float64
 26  rerror_rate                  311027 non-null  float64
 27  srv_rerror_rate              311027 non-null  float64
 28  same_srv_rate                311027 non-null  float64
 29  diff_srv_rate                311027 non-null  float64
 30  srv_diff_host_rate           311027 non-null  float64
 31  dst_host_count               311027 non-null  int64  
 32  dst_host_srv_count           311027 non-null  int64  
 33  dst_host_same_srv_rate       311027 non-null  float64
 34  dst_host_diff_srv_rate       311027 non-null  float64
 35  dst_host_same_src_port_rate  311027 non-null  float64
 36  dst_host_srv_diff_host_rate  311027 non-null  float64
 37  dst_host_serror_rate         311027 non-null  float64
 38  dst_host_srv_serror_rate     311027 non-null  float64
 39  dst_host_rerror_rate         311027 non-null  float64
 40  dst_host_srv_rerror_rate     311027 non-null  float64
 41  label                        311027 non-null  int64  
dtypes: float64(32), int64(10)
memory usage: 102.0 MB
```

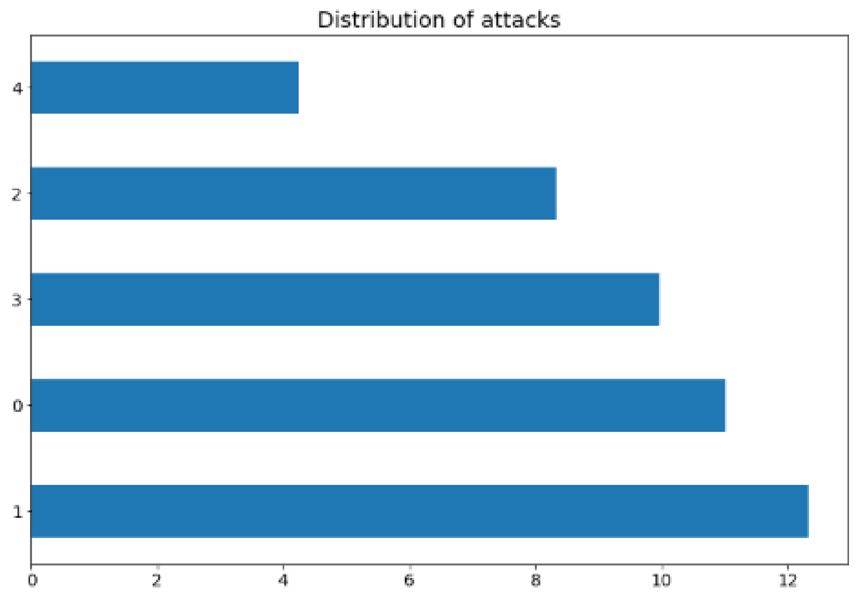
In [16]:
```
#drops the label entry from the dataframe
data = df.drop('label', axis=1)
label = pd.DataFrame(df['label'])

#dividing the data into testing and training sets.
X_train, X_test, y_train, y_test = train_test_split(data, label,test_size=0.2)
df.to_csv("./clean_labeled.csv",
          index=False)
X_train.to_csv("./train_data.csv",
           index=False)
y_train.to_csv("./train_label.csv",
           index=False)
X_test.to_csv("./test_data.csv",
           index=False)
y_test.to_csv("./test_label.csv",
           index=False)
```

In [17]:
```
#finds unique value counts of label.
df['label'].value_counts()
```

Out[17]:
```
1    224855
0     60591
3     21345
2      4166
4        70
Name: label, dtype: int64
```

In [18]:
```
#clears current figure
plt.clf()

#sets the figure size
plt.figure(figsize=(12, 8))
params = {'axes.titlesize':'18',
          'xtick.labelsize':'14',
          'ytick.labelsize':'14'}
matplotlib.rcParams.update(params)
plt.title('Distribution of attacks')

#Y axis denotes the label type/ type of attack and the x-axis denotes the log base e of the number of attacks.
df['label'].value_counts().apply(np.log).plot(kind='barh')

plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



In [19]:
```
col_names = ['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
             'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
             'num_failed_logins', 'logged_in', 'num_compromised', 'root_shell',
             'su_attempted', 'num_root', 'num_file_creations', 'num_shells',
             'num_access_files', 'num_outbound_cmds', 'is_host_login',
             'is_guest_login', 'count', 'srv_count', 'serror_rate',
             'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate',
             'same_srv_rate', 'diff_srv_rate', 'srv_diff_host_rate',
             'dst_host_count', 'dst_host_srv_count', 'dst_host_same_srv_rate',
             'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
             'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
             'dst_host_srv_serror_rate', 'dst_host_rerror_rate',
             'dst_host_srv_rerror_rate']
data = pd.read_csv('./PCAPTest.csv',
                   header=None, names=col_names, low_memory=False)
df = data.dropna()
df.head(10)
```

Out[19]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_count	dst_host_srv_count	dst_host_same_s
NaN	0	udp	private	SF	105	146	0.1	0.2	0.3	0.4	...	1.2	1.3	
0.0	0	udp	private	SF	105	146	0.0	0.0	0.0	0.0	...	255.0	254.0	
1.0	0	udp	private	SF	105	146	0.0	0.0	0.0	0.0	...	255.0	254.0	
2.0	0	udp	private	SF	105	146	0.0	0.0	0.0	0.0	...	255.0	254.0	
3.0	0	udp	private	SF	105	146	0.0	0.0	0.0	0.0	...	255.0	254.0	
4.0	0	udp	private	SF	105	146	0.0	0.0	0.0	0.0	...	255.0	254.0	
5.0	0	udp	private	SF	105	146	0.0	0.0	0.0	0.0	...	255.0	254.0	
6.0	0	udp	private	SF	105	146	0.0	0.0	0.0	0.0	...	255.0	253.0	
7.0	0	udp	private	SF	105	146	0.0	0.0	0.0	0.0	...	255.0	253.0	
8.0	0	udp	private	SF	105	146	0.0	0.0	0.0	0.0	...	255.0	254.0	

In [20]:
```python
df['protocol_type'] = df['protocol_type'].replace('udp', '0')
df['protocol_type'] = df['protocol_type'].replace('tcp', '1')
df['protocol_type'] = df['protocol_type'].replace('icmp', '2')

df['service'] = df['service'].replace('private', '0')
df['service'] = df['service'].replace('domain_u', '1')
df['service'] = df['service'].replace('http', '2')
df['service'] = df['service'].replace('smtp', '3')
df['service'] = df['service'].replace('ftp_data', '4')
df['service'] = df['service'].replace('ftp', '5')
df['service'] = df['service'].replace('eco_i', '6')
df['service'] = df['service'].replace('other', '7')
df['service'] = df['service'].replace('auth', '8')
df['service'] = df['service'].replace('ecr_i', '9')
df['service'] = df['service'].replace('IRC', '10')
df['service'] = df['service'].replace('X11', '11')
df['service'] = df['service'].replace('finger', '12')
df['service'] = df['service'].replace('time', '13')
df['service'] = df['service'].replace('domain', '14')
df['service'] = df['service'].replace('telnet', '15')
df['service'] = df['service'].replace('pop_3', '16')
df['service'] = df['service'].replace('ldap', '17')
df['service'] = df['service'].replace('login', '18')
df['service'] = df['service'].replace('name', '19')
df['service'] = df['service'].replace('ntp_u', '20')
df['service'] = df['service'].replace('http_443', '21')
df['service'] = df['service'].replace('sunrpc', '22')
df['service'] = df['service'].replace('printer', '23')
df['service'] = df['service'].replace('systat', '24')
df['service'] = df['service'].replace('tim_i', '25')
df['service'] = df['service'].replace('netstat', '26')
df['service'] = df['service'].replace('remote_job', '27')
df['service'] = df['service'].replace('link', '28')
df['service'] = df['service'].replace('urp_i', '29')
df['service'] = df['service'].replace('sql_net', '30')
df['service'] = df['service'].replace('bgp', '31')
df['service'] = df['service'].replace('pop_2', '32')
df['service'] = df['service'].replace('tftp_u', '33')
df['service'] = df['service'].replace('uucp', '34')
df['service'] = df['service'].replace('imap4', '35')
df['service'] = df['service'].replace('pm_dump', '36')
df['service'] = df['service'].replace('nnsp', '37')
df['service'] = df['service'].replace('courier', '38')
df['service'] = df['service'].replace('daytime', '39')
df['service'] = df['service'].replace('iso_tsap', '40')
df['service'] = df['service'].replace('echo', '41')
df['service'] = df['service'].replace('discard', '42')
df['service'] = df['service'].replace('ssh', '43')
df['service'] = df['service'].replace('whois', '44')
df['service'] = df['service'].replace('mtp', '45')
```

```python
df['service'] = df['service'].replace('rje', '47')
df['service'] = df['service'].replace('ctf', '48')
df['service'] = df['service'].replace('supdup', '49')
df['service'] = df['service'].replace('hostnames', '50')
df['service'] = df['service'].replace('csnet_ns', '51')
df['service'] = df['service'].replace('uucp_path', '52')
df['service'] = df['service'].replace('nntp', '53')
df['service'] = df['service'].replace('netbios_ns', '54')
df['service'] = df['service'].replace('netbios_dgm', '55')
df['service'] = df['service'].replace('netbios_ssn', '56')
df['service'] = df['service'].replace('vmnet', '57')
df['service'] = df['service'].replace('Z39_50', '58')
df['service'] = df['service'].replace('exec', '59')
df['service'] = df['service'].replace('shell', '60')
df['service'] = df['service'].replace('efs', '61')
df['service'] = df['service'].replace('klogin', '62')
df['service'] = df['service'].replace('kshell', '63')
df['service'] = df['service'].replace('icmp', '64')
#df['service'] = df['service'].replace('harvest', '64')
#df['service'] = df['service'].replace('http_2784', '64')
#df['service'] = df['service'].replace('http_8001', '64')
#df['service'] = df['service'].replace('urh_i', '64')
#df['service'] = df['service'].replace('aol', '64')
df = df[df.service != 'harvest']
df = df[df.service != 'http_2784']
df = df[df.service != 'http_8001']
df = df[df.service != 'urh_i']
df = df[df.service != 'aol']
df['flag'] = df['flag'].replace('SF', '0')
df['flag'] = df['flag'].replace('SH', '1')
df['flag'] = df['flag'].replace('RSTR', '2')
df['flag'] = df['flag'].replace('REJ', '3')
df['flag'] = df['flag'].replace('S0', '4')
df['flag'] = df['flag'].replace('S1', '5')
df['flag'] = df['flag'].replace('S2', '6')
df['flag'] = df['flag'].replace('S3', '7')
df['flag'] = df['flag'].replace('RSTO', '8')
df['flag'] = df['flag'].replace('RSTOS0', '9')
df['flag'] = df['flag'].replace('OTH', '10')

df.head()
```

\

Out[20]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_count	dst_host_srv_count	dst_host_same_s
NaN	0	0	0	0	105	146	0.1	0.2	0.3	0.4	...	1.2	1.3	
0.0	0	0	0	0	105	146	0.0	0.0	0.0	0.0	...	255.0	254.0	
1.0	0	0	0	0	105	146	0.0	0.0	0.0	0.0	...	255.0	254.0	
2.0	0	0	0	0	105	146	0.0	0.0	0.0	0.0	...	255.0	254.0	
3.0	0	0	0	0	105	146	0.0	0.0	0.0	0.0	...	255.0	254.0	

5 rows × 41 columns

In [21]:

```
df = df.drop(df.index[:2])

df["protocol_type"] = pd.to_numeric(df["protocol_type"])
df["service"] = pd.to_numeric(df["service"])
df["flag"]= pd.to_numeric(df["flag"])
df["dst_host_srv_rerror_rate"]= pd.to_numeric(df["dst_host_srv_rerror_rate"])
df["dst_host_rerror_rate"]= pd.to_numeric(df["dst_host_rerror_rate"])
df["dst_host_srv_serror_rate"]= pd.to_numeric(df["dst_host_srv_serror_rate"])
df["dst_host_serror_rate"]= pd.to_numeric(df["dst_host_serror_rate"])
df["dst_host_srv_diff_host_rate"]= pd.to_numeric(df["dst_host_srv_diff_host_rate"])
df["dst_host_same_src_port_rate"]= pd.to_numeric(df["dst_host_same_src_port_rate"])
df["dst_host_diff_srv_rate"]= pd.to_numeric(df["dst_host_diff_srv_rate"])
df["dst_host_same_srv_rate"]= pd.to_numeric(df["dst_host_same_srv_rate"])
df["srv_diff_host_rate"]= pd.to_numeric(df["srv_diff_host_rate"])
df["diff_srv_rate"]= pd.to_numeric(df["diff_srv_rate"])
df["srv_rerror_rate"]= pd.to_numeric(df["srv_rerror_rate"])
df["rerror_rate"]= pd.to_numeric(df["rerror_rate"])
df["srv_serror_rate"]= pd.to_numeric(df["srv_serror_rate"])

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Float64Index: 2984143 entries, 1.0 to 2984152.0
Data columns (total 41 columns):
 #   Column                       Dtype  
---  ------                       -----  
 0   duration                     int64  
 1   protocol_type                int64  
 2   service                      int64  
 3   flag                         int64  
 4   src_bytes                    int64  
 5   dst_bytes                    int64  
 6   land                         float64
 7   wrong_fragment               float64
 8   urgent                       float64
 9   hot                          float64
 10  num_failed_logins            float64
 11  logged_in                    float64
 12  num_compromised              float64
 13  root_shell                   float64
 14  su_attempted                 float64
 15  num_root                     float64
 16  num_file_creations           float64
 17  num_shells                   float64
 18  num_access_files             float64
 19  num_outbound_cmds            float64
 20  is_host_login                float64
 21  is_guest_login               float64
 22  count                        int64  
 23  srv_count                    float64
 24  serror_rate                  float64
 25  srv_serror_rate              float64
 26  rerror_rate                  float64
 27  srv_rerror_rate              float64
 28  same_srv_rate                float64
 29  diff_srv_rate                float64
 30  srv_diff_host_rate           float64
 31  dst_host_count               float64
 32  dst_host_srv_count           float64
 33  dst_host_same_srv_rate       float64
 34  dst_host_diff_srv_rate       float64
 35  dst_host_same_src_port_rate  float64
 36  dst_host_srv_diff_host_rate  float64
 37  dst_host_serror_rate         float64
 38  dst_host_srv_serror_rate     float64
 39  dst_host_rerror_rate         float64
 40  dst_host_srv_rerror_rate     float64
dtypes: float64(34), int64(7)
memory usage: 956.2 MB
```

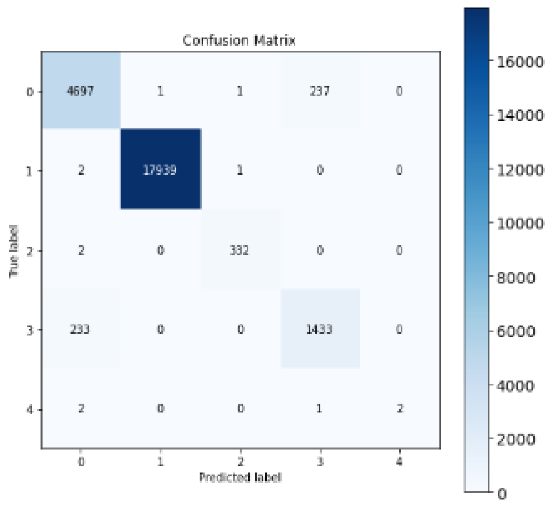
```
In [22]: #1 Random Forest Classifier
         import pandas as pd
         import numpy as np
         import pickle as p
         from sklearn.preprocessing import Normalizer
         from sklearn.model_selection import train_test_split
         from sklearn.ensemble import RandomForestClassifier
         from sklearn import metrics

         X = pd.read_csv('./train_data.csv')
         y = pd.read_csv('./train_label.csv')

         #Normalizes the data of each row to fit it into a range of 0-1
         scaler = Normalizer().fit(X)
         x = scaler.transform(X)

         X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.1,
                                                             random_state=0)
         y_train = np.array(y_train)

         #n_estimators denotes the number of decision trees to be derived
         #max_depth denotes the depth of the trees.
         #random_states controls the randomness of the bootstrapping of the samples used when building trees
         model = RandomForestClassifier(n_estimators=20, max_depth=20, random_state=0)
         model.fit(X_train, y_train)

         <ipython-input-22-3dc22f54a6da>:25: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please ch
         ange the shape of y to (n_samples,), for example using ravel().
           model.fit(X_train, y_train)

Out[22]: RandomForestClassifier(max_depth=20, n_estimators=20, random_state=0)
```

```
In [23]: y_pred = model.predict(X_test)

         print("Accuracy =", metrics.accuracy_score(y_test, y_pred))
         print("Confusion Matrix =\n", metrics.confusion_matrix(y_test, y_pred, labels=None,
                                                                 sample_weight=None))
         print("Recall =", metrics.recall_score(y_test, y_pred, labels=None,
                                                pos_label=1, average='weighted',
                                                sample_weight=None))
         print("Classification Report =\n", metrics.classification_report(y_test, y_pred,
                                                                           labels=None,
                                                                           target_names=None,
                                                                           sample_weight=None,
                                                                           digits=2,
                                                                           output_dict=False))

         Accuracy = 0.9807097214966041
         Confusion Matrix =
          [[ 4697     1     1   237     0]
          [    2 17939     1     0     0]
          [    2     0   332     0     0]
          [  233     0     0  1433     0]
          [    2     0     0     1     2]]
         Recall = 0.9807097214966041
         Classification Report =
                        precision    recall  f1-score   support

                    0       0.95      0.95      0.95      4936
                    1       1.00      1.00      1.00     17942
                    2       0.99      0.99      0.99       334
                    3       0.86      0.86      0.86      1666
                    4       1.00      0.40      0.57         5

             accuracy                           0.98     24883
            macro avg       0.96      0.84      0.88     24883
         weighted avg       0.98      0.98      0.98     24883
```

```
In [24]: !pip install scikit-plot

         Requirement already satisfied: scikit-plot in c:\users\vpran\anaconda3\lib\site-packages (0.3.7)
         Requirement already satisfied: scipy>=0.9 in c:\users\vpran\anaconda3\lib\site-packages (from scikit-plot) (1.6.2)
         Requirement already satisfied: matplotlib>=1.4.0 in c:\users\vpran\anaconda3\lib\site-packages (from scikit-plot) (3.3.4)
         Requirement already satisfied: scikit-learn>=0.18 in c:\users\vpran\anaconda3\lib\site-packages (from scikit-plot) (0.24.1)
         Requirement already satisfied: joblib>=0.10 in c:\users\vpran\anaconda3\lib\site-packages (from scikit-plot) (1.0.1)
         Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\vpran\anaconda3\lib\site-packages (from matplotlib>=1.4.0->scik
         it-plot) (1.3.1)
         Requirement already satisfied: pillow>=6.2.0 in c:\users\vpran\anaconda3\lib\site-packages (from matplotlib>=1.4.0->scikit-p
         lot) (8.2.0)
         Requirement already satisfied: python-dateutil>=2.1 in c:\users\vpran\anaconda3\lib\site-packages (from matplotlib>=1.4.0->s
         cikit-plot) (2.8.1)
         Requirement already satisfied: numpy>=1.15 in c:\users\vpran\anaconda3\lib\site-packages (from matplotlib>=1.4.0->scikit-plo
         t) (1.20.1)
         Requirement already satisfied: cycler>=0.10 in c:\users\vpran\anaconda3\lib\site-packages (from matplotlib>=1.4.0->scikit-pl
         ot) (0.10.0)
         Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\users\vpran\anaconda3\lib\site-packages (from
         matplotlib>=1.4.0->scikit-plot) (2.4.7)
         Requirement already satisfied: six in c:\users\vpran\anaconda3\lib\site-packages (from cycler>=0.10->matplotlib>=1.4.0->scik
         it-plot) (1.15.0)
```

```
In [25]: import scikitplot as skplt

         skplt.metrics.plot_confusion_matrix(
             y_test,
             y_pred,
             figsize=(8,8))
```

```
Out[25]: <AxesSubplot:title={'center':'Confusion Matrix'}, xlabel='Predicted label', ylabel='True label'>
```



```
In [26]: file = "./randomForest.pkl"
         with open(file, "wb") as f:
             p.dump(model, f)
```

```
In [27]: #2 Naive Bayes Algorithm
         from sklearn.naive_bayes import GaussianNB
```

```
In [28]: X = pd.read_csv('./train_data.csv')
         y = pd.read_csv('./train_label.csv')
         X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.1,
                                                             random_state=0)
         y_train = np.array(y_train)
```

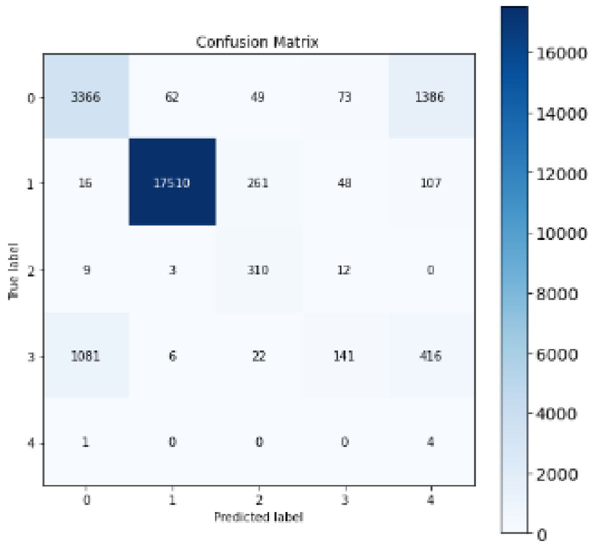
```
In [29]: model = GaussianNB()
         model.fit(X_train, y_train)

         C:\Users\vpran\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed
         when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
           return f(*args, **kwargs)

Out[29]: GaussianNB()
```

```
In [30]: y_pred = model.predict(X_test)
```

```
In [20]: print("Accuracy =", metrics.accuracy_score(y_test, y_pred))
         print("Confusion Matrix =\n", metrics.confusion_matrix(y_test, y_pred, labels=None,
                                                     sample_weight=None))
         print("Recall =", metrics.recall_score(y_test, y_pred, labels=None,
                                                     pos_label=1, average='weighted',
                                                     sample_weight=None))
         print("Classification Report =\n", metrics.classification_report(y_test, y_pred,
                                                                       labels=None,
                                                                       target_names=None,
                                                                       sample_weight=None,
                                                                       digits=2,
                                                                       output_dict=False))

         Accuracy = 0.8645259815938593
         Confusion Matrix =
          [[ 3400    67    34    67  1249]
          [   12 17715   240    36   117]
          [    9     0   279     6     0]
          [ 1106     0    25   111   403]
          [    0     0     0     0     7]]
         Recall = 0.8645259815938593
         Classification Report =
                        precision    recall  f1-score   support

                    0       0.75      0.71      0.73      4817
                    1       1.00      0.98      0.99     18120
                    2       0.48      0.95      0.64       294
                    3       0.50      0.07      0.12      1645
                    4       0.00      1.00      0.01         7

             accuracy                           0.86     24883
            macro avg       0.55      0.74      0.50     24883
         weighted avg       0.91      0.86      0.87     24883
```

```
In [31]: import scikitplot as skplt

         skplt.metrics.plot_confusion_matrix(
             y_test,
             y_pred,
             figsize=(8,8))
```

Out[31]: `<AxesSubplot:title={'center':'Confusion Matrix'}, xlabel='Predicted label', ylabel='True label'>`



In [32]:
```
#3 Adaboost Classifier
from sklearn.ensemble import AdaBoostClassifier
```

In [33]:
```
X = pd.read_csv('./train_data.csv')
y = pd.read_csv('./train_label.csv')
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.1,
 random_state=0)
y_train = np.array(y_train)
```

In [34]:
```
model = AdaBoostClassifier(n_estimators=50)
model.fit(X_train, y_train)
```

```
C:\Users\vpran\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
```

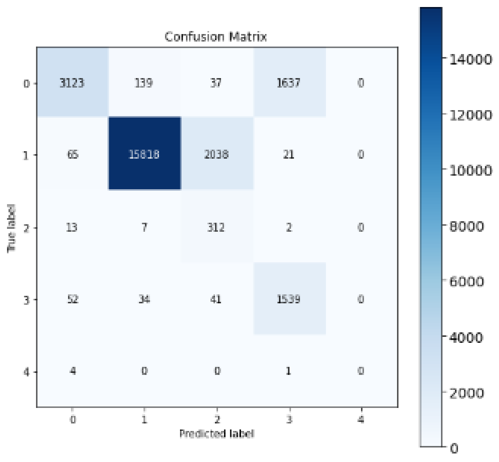
Out[34]: `AdaBoostClassifier()`

In [35]:
```
y_pred = model.predict(X_test)
```

In [36]:
```
print("Accuracy =", metrics.accuracy_score(y_test, y_pred))
print("Confusion Matrix =\n", metrics.confusion_matrix(y_test, y_pred, labels=None,
                                                        sample_weight=None))
print("Recall =", metrics.recall_score(y_test, y_pred, labels=None,
                                        pos_label=1, average='weighted',
                                        sample_weight=None))
print("Classification Report =\n", metrics.classification_report(y_test, y_pred,
                                                                  labels=None,
                                                                  target_names=None,
                                                                  sample_weight=None,
                                                                  digits=2,
                                                                  output_dict=False))
```

```
Accuracy = 0.8355905638387654
Confusion Matrix =
 [[ 3123   139    37  1637     0]
 [   65 15818  2038    21     0]
 [   13     7   312     2     0]
 [   52    34    41  1539     0]
 [    4     0     0     1     0]]
Recall = 0.8355905638387654
Classification Report =
               precision    recall  f1-score   support

           0       0.96      0.63      0.76      4936
           1       0.99      0.88      0.93     17942
           2       0.13      0.93      0.23       334
           3       0.48      0.92      0.63      1666
           4       0.00      0.00      0.00         5

    accuracy                           0.84     24883
   macro avg       0.51      0.67      0.51     24883
weighted avg       0.94      0.84      0.87     24883
```

```
In [37]: import scikitplot as skplt
         skplt.metrics.plot_confusion_matrix(
          y_test,
          y_pred,
          figsize=(8,8))

Out[37]: <AxesSubplot:title={'center':'Confusion Matrix'}, xlabel='Predicted label', ylabel='True label'>
```



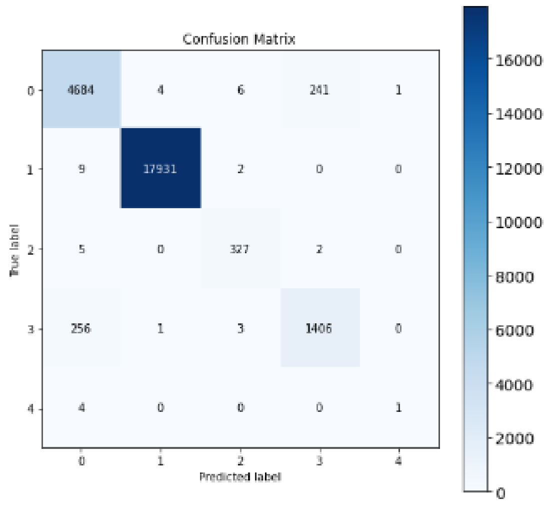
```
In [38]: #4 GRADIENT BOOSTING CLASSIFIER
         from sklearn.ensemble import GradientBoostingClassifier

In [40]: X = pd.read_csv('./train_data.csv')
         y = pd.read_csv('./train_label.csv')
         X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.1,
          random_state=0)
         y_train = np.array(y_train)

In [41]: model = GradientBoostingClassifier(n_estimators=50)
         model.fit(X_train, y_train)

         C:\Users\vpran\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed
         when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
           return f(*args, **kwargs)

Out[41]: GradientBoostingClassifier(n_estimators=50)

In [42]: y_pred = model.predict(X_test)

In [44]: print("Accuracy =", metrics.accuracy_score(y_test, y_pred))
         print("Confusion Matrix =\n", metrics.confusion_matrix(y_test, y_pred, labels=None,
                                                          sample_weight=None))
         print("Recall =", metrics.recall_score(y_test, y_pred, labels=None,
                                                      pos_label=1, average='weighted',
                                                      sample_weight=None))
         print("Classification Report =\n", metrics.classification_report(y_test, y_pred,
                                                      labels=None,
                                                      target_names=None,
                                                      sample_weight=None,
                                                         digits=2,
                                                      output_dict=False))
```

```
Accuracy = 0.9785395651649721
Confusion Matrix =
 [[ 4684     4     6   241     1]
 [     9 17931     2     0     0]
 [     5     0   327     2     0]
 [   256     1     3  1406     0]
 [     4     0     0     0     1]]
Recall = 0.9785395651649721
Classification Report =
               precision    recall  f1-score   support

           0       0.94      0.95      0.95      4936
           1       1.00      1.00      1.00     17942
           2       0.97      0.98      0.97       334
           3       0.85      0.84      0.85      1666
           4       0.50      0.20      0.29         5

    accuracy                           0.98     24883
   macro avg       0.85      0.79      0.81     24883
weighted avg       0.98      0.98      0.98     24883
```

```
import scikitplot as skplt
skplt.metrics.plot_confusion_matrix(
 y_test,
 y_pred,
 figsize=(8,8))
```

```
In [45]: import scikitplot as skplt
         skplt.metrics.plot_confusion_matrix(
          y_test,
          y_pred,
          figsize=(8,8))
```

```
Out[45]: <AxesSubplot:title={'center':'Confusion Matrix'}, xlabel='Predicted label', ylabel='True label'>
```



```
In [46]: # 5 Multi-Layer Perceptron
         from sklearn.neural_network import MLPClassifier
```

```
In [47]: X = pd.read_csv('./train_data.csv')
         y = pd.read_csv('./train_label.csv')
         X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.1,
          random_state=0)
         y_train = np.array(y_train)
```

```
In [48]: model = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(15,), random_state=1)
         model.fit(X_train, y_train)
```

```
C:\Users\vpran\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
C:\Users\vpran\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:500: ConvergenceWarning: lbfgs fail
ed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Out[48]: MLPClassifier(alpha=1e-05, hidden_layer_sizes=(15,), random_state=1,
                       solver='lbfgs')
```

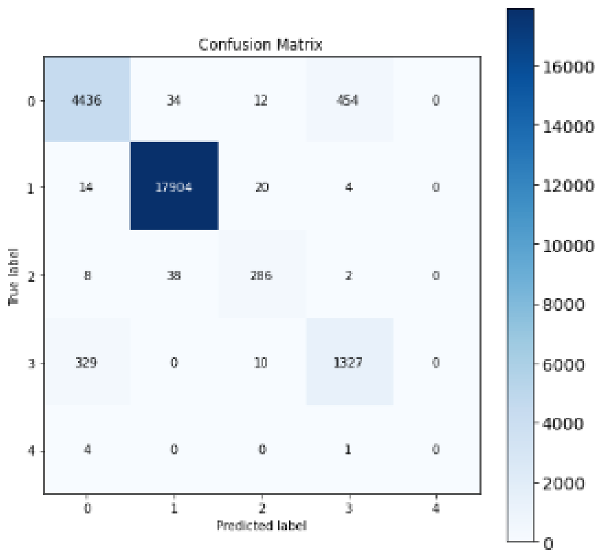
```
In [49]: y_pred = model.predict(X_test)
```

```
In [50]: print("Accuracy =", metrics.accuracy_score(y_test, y_pred))
         print("Confusion Matrix =\n", metrics.confusion_matrix(y_test, y_pred, labels=None,
                                                      sample_weight=None))
         print("Recall =", metrics.recall_score(y_test, y_pred, labels=None,
                                                      pos_label=1, average='weighted',
                                                      sample_weight=None))
         print("Classification Report =\n", metrics.classification_report(y_test, y_pred,
                                                     labels=None,
                                                      target_names=None,
                                                      sample_weight=None,
                                                      digits=2,
                                                      output_dict=False))
```

```
Accuracy = 0.9626250853996705
Confusion Matrix =
 [[ 4436    34    12   454     0]
 [   14 17904    20     4     0]
 [    8    38   286     2     0]
 [  329     0    10  1327     0]
 [    4     0     0     1     0]]
Recall = 0.9626250853996705
Classification Report =
               precision    recall  f1-score   support

           0       0.93      0.90      0.91      4936
           1       1.00      1.00      1.00     17942
           2       0.87      0.86      0.86       334
           3       0.74      0.80      0.77      1666
           4       0.00      0.00      0.00         5

    accuracy                           0.96     24883
   macro avg       0.71      0.71      0.71     24883
weighted avg       0.96      0.96      0.96     24883
```

In [51]:
```
import scikitplot as skplt
skplt.metrics.plot_confusion_matrix(
 y_test,
 y_pred,
 figsize=(8,8))
```

Out[51]:
```
<AxesSubplot:title={'center':'Confusion Matrix'}, xlabel='Predicted label', ylabel='True label'>
```



In [52]:
```
#6 XGBoost Classifier
!pip install xgboost
```

```
Collecting xgboost
  Downloading xgboost-1.7.1-py3-none-win_amd64.whl (89.1 MB)
Requirement already satisfied: numpy in c:\users\vpran\anaconda3\lib\site-packages (from xgboost) (1.20.1)
Requirement already satisfied: scipy in c:\users\vpran\anaconda3\lib\site-packages (from xgboost) (1.6.2)
Installing collected packages: xgboost
Successfully installed xgboost-1.7.1
```

In [53]:
```
import xgboost as xgb
```

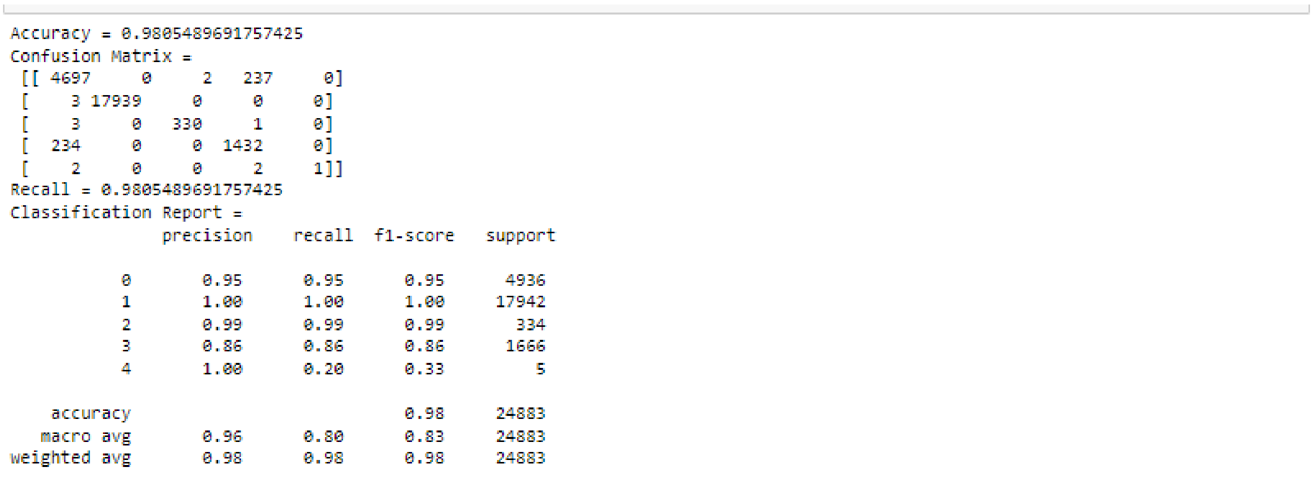
In [54]:
```
X = pd.read_csv('./train_data.csv')
y = pd.read_csv('./train_label.csv')
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.1,
 random_state=0)
y_train = np.array(y_train)
```

In [55]:
```
model = xgb.XGBClassifier(objective="binary:logistic", random_state=42)
model.fit(X_train, y_train)
```

Out[55]:
```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, feature_types=None, gamma=0, gpu_id=-1,
              grow_policy='depthwise', importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_bin=256, max_cat_threshold=64, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, objective='multi:softprob',
              predictor='auto', ...)
```

In [56]:
```
y_pred = model.predict(X_test)
```

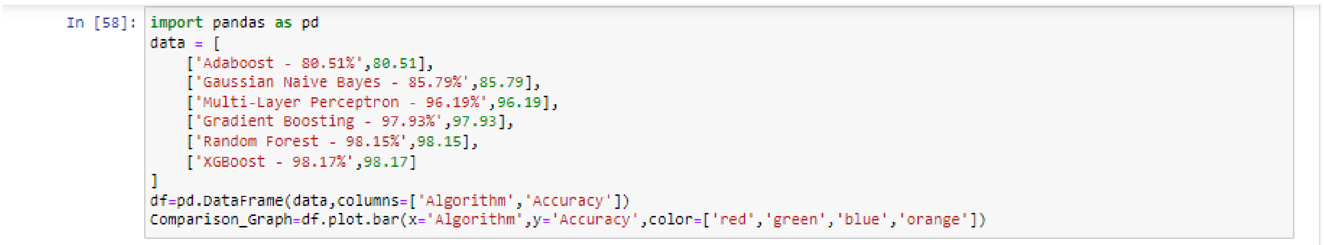
In [57]:
```
print("Accuracy =", metrics.accuracy_score(y_test, y_pred))
print("Confusion Matrix =\n", metrics.confusion_matrix(y_test, y_pred, labels=None,
                                                        sample_weight=None))
print("Recall =", metrics.recall_score(y_test, y_pred, labels=None,
                                                        pos_label=1, average='weighted',
                                                        sample_weight=None))
print("Classification Report =\n", metrics.classification_report(y_test, y_pred,
                                                        labels=None,
                                                        target_names=None,
                                                        sample_weight=None,
                                                        digits=2,
                                                        output_dict=False))
```



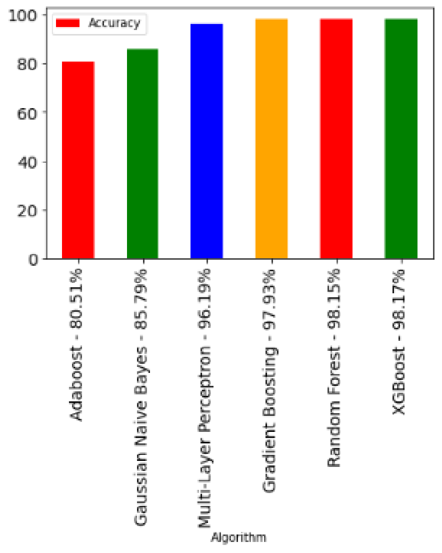
```
Accuracy = 0.9805489691757425
Confusion Matrix =
 [[ 4697     0     2   237     0]
 [     3 17939     0     0     0]
 [     3     0   330     1     0]
 [   234     0     0  1432     0]
 [     2     0     0     2     1]]
Recall = 0.9805489691757425
Classification Report =
               precision    recall  f1-score   support

           0       0.95      0.95      0.95      4936
           1       1.00      1.00      1.00     17942
           2       0.99      0.99      0.99       334
           3       0.86      0.86      0.86      1666
           4       1.00      0.20      0.33         5

    accuracy                           0.98     24883
   macro avg       0.96      0.80      0.83     24883
weighted avg       0.98      0.98      0.98     24883
```

RESULTS

The graph given below shows the accuracy comparison of the 6 ML classifiers used above. It can be clearly seen that XGBoost Classifier had the best accuracy (98.17%) among all the others whereas Adaboost Classifier had the worst accuracy (80.51%).



```
In [58]: import pandas as pd
         data = [
             ['Adaboost - 80.51%',80.51],
             ['Gaussian Naive Bayes - 85.79%',85.79],
             ['Multi-Layer Perceptron - 96.19%',96.19],
             ['Gradient Boosting - 97.93%',97.93],
             ['Random Forest - 98.15%',98.15],
             ['XGBoost - 98.17%',98.17]
         ]
         df=pd.DataFrame(data,columns=['Algorithm','Accuracy'])
         Comparison_Graph=df.plot.bar(x='Algorithm',y='Accuracy',color=['red','green','blue','orange'])
```



Conclusion

The aim of this project was to demonstrate the benefits of employing machine learning algorithms for the development of an Intrusion Detection System(IDS). The limitations, as found in the previous research conducted for the same, were eliminated to obtain better results with an efficient system.The result was a system with an accuracy of above 95 in predicting Dos,Probe ,U2R and R2L attacks. The dataset used was NSL KDD dataset whichis a new and improved dataset which included a great number of U2R and R2L attacks as compared to the old KDD dataset used in many research. The machine learning algorithm employed is also lightweight compared to many heavy computational cost algorithms . The data taken from NSL KDD dataset was preprocessed to remove redundancy and unnecessary features to provide a high accuracy model with low computational cost.
Pre-processing of data also provided better results in identifying U2R and R2Lattacks which were lacking in the previous research.

Precision score :

Algorithms	**Normal**	**DoS**	**Probe**	**R2L**	**U2R**
Random forest	95	100	99	86	100
Naïve Bayes	75	100	48	50	74
Adaboost Classifier	96	99	13	48	78
Gradient Boosting	94	100	97	85	50
Multi-Layer Perceptron	93	100	87	74	82
XGBoost Classifier	95	100	99	86	100

REFERENCES:

[1] Dawoud, A., Shahristani, S., & Raun, C. (2018, December). Deep learning for network anomalies detection. In *2018 International Conference on Machine Learning and Data Engineering (iCMLDE)* (pp.149-153). IEEE.

[2] Ariafar, E., & Kiani, R. (2017, December). Intrusion detection system using an optimized frameworkbased on data mining techniques. In *2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI)* (pp. 0785-0791). IEEE.

[3] Hasan, M., Islam, M. M., Zarif, M. I. I., & Hashem, M. M. A. (2019). Attack and anomaly detection inIoT sensors in IoT sites using machine learning approaches. *Internet of Things*, *7*, 100059.

[4] Alzahrani, A. O., & Alenazi, M. J. (2021). Designing a Network Intrusion Detection System Based on Machine Learning for Software Defined Networks. *Future Internet*, *13*(5), 111.

[5] Kwon, D., Natarajan, K., Suh, S. C., Kim, H., & Kim, J. (2018, July). An empirical study on network anomaly detection using convolutional neural networks. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)* (pp. 1595-1598). IEEE.

[6] Alrashdi, I., Alqazzaz, A., Aloufi, E., Alharthi, R., Zohdy, M., & Ming, H. (2019, January). Ad-iot: Anomaly detection of iot cyberattacks in smart city using machine learning. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)* (pp. 0305-0310). IEEE.

[7] Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference onComputer Vision and Pattern Recognition* (pp. 2820-2828).

[8] Nandhini, K., Pavithra, M., Revathi, K., & Rajiv, A. (2017, March). Anamoly detection for safety monitoring. In *2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)* (pp. 1-6). IEEE.

[9] H. M. Anwer, M. Farouk and A. Abdel-Hamid, "A framework for efficient network anomaly intrusion detection with features selection," 2018 9th International Conference on Information and CommunicationSystems (ICICS), 2018, pp. 157-162, doi: 10.1109/IACS.2018.8355459.

[10] Khraisat A., Gondal I., Vamplew P. (2018) An Anomaly Intrusion Detection System Using C5 Decision Tree Classifier. In: Ganji M., Rashidi L., Fung B., Wang C. (eds) Trends and Applications inKnowledge Discovery and Data Mining. PAKDD 2018. Lecture Notes in Computer Science, vol 11154. Springer, Cham.

[11] Ishaq, Kashif. "An Efficient Hybrid Classifier Model for Anomaly Intrusion Detection System." International Journal of Computer Science and Network Security (2018)

[12] P. Satam and S. Hariri, "WIDS: An Anomaly Based Intrusion Detection System for Wi-Fi (IEEE 802.11) Protocol," in IEEE Transactions on Network and Service Management, vol. 18, no. 1, pp. 1077-1091, March 2021, doi: 10.1109/TNSM.2020.3036138.

[13] WS, J. D. S., Parvathavarthini, B., & Arunmozhi, S. (2020, July). Machine Learning based Intrusion Detection Framework using Recursive Feature Elimination Method. In *2020 International Conference onSystem, Computation, Automation and Networking (ICSCAN)* (pp. 1-4). IEEE.

[14] Divakar, S., Priyadarshini, R., & Mishra, B. K. (2020, December). A Robust Intrusion Detection Systemusing Ensemble Machine Learning. In *2020 IEEE International Women in Engineering (WIE) Conferenceon Electrical and*

Computer Engineering (WIECON-ECE) (pp. 344-347). IEEE.

[15] Rokade, M. D., & Sharma, Y. K. (2021, March). MLIDS: A Machine Learning Approach for Intrusion Detection for Real Time Network Dataset. In *2021 International Conference on Emerging Smart Computing and Informatics (ESCI)* (pp. 533-536). IEEE.

[16] Schueller, Q., Basu, K., Younas, M., Patel, M., & Ball, F. (2018, November). A hierarchical intrusion detection system using support vector machine for SDN network in cloud data center. In *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)* (pp. 1-6). IEEE.

[17] Zoph, Barret, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. "Learning transferable architectures for scalable image recognition." In *Proceedings of the IEEE conference on computer visionand pattern recognition*, pp. 8697-8710. 2018.

[18Arfeen, Muhammad Asad, Krzysztof Pawlikowski, D. McNickle, and Andreas Willig. "The role of theweibull distribution in internet traffic modeling." In *Proceedings of the 2013 25th International TeletrafficCongress (ITC)*, pp. 1-8. IEEE, 2013.

[19] Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference onComputer Vision and Pattern Recognition* (pp. 2820-2828).

[20] Alrashdi, Ibrahim, et al. "Ad-iot: Anomaly detection of iot cyberattacks in smart city using machine learning." *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*.IEEE, 2019.