**Feature Engineering and Machine Learning Framework for DDoS Attack Detection in the Standardized Internet of Things**

**ABSTRACT**

**ABSTRACT**

Over the last decade, there has been a massive increase in the number of Internet of Things (IoT) devices and networks, which are often characterized by resource limits on energy, memory, communication, and compute power, and therefore the integration of security measures in these networks is sometimes ignored. As the number of attacks increases, it is critical to protect networks using machine learning (ML)-based intrusion detection systems (IDS) for greater efficiency and fewer false alarms. These systems, however, depend heavily on data and optimal attributes for extrapolation and attack detection. Existing benchmark data sets are obsolete and lack IoT compatible traffic data and, therefore, to address this issue, we explore a novel data set, cross layer intrusion detection data set for IoT (IoT-CIDDS) data set with 21 features and single labeling attribute. In this work, we propose a feature engineering and ML framework to detect Distributed Denial-of-Service (DDoS) attacks of IoT-CIDDS data set. The framework consists of two phases. In the first phase, we develop algorithms for data set enrichment and emphasize on advanced feature engineering for statistically analyzing the data set with probability distribution and correlation among features. In the second phase, we propose an ML model and perform complexity analysis of the feature engineered data set with five ML techniques by creating training, validation, and testing data sets from IoT-CIDDS. The ML models are evaluated in terms of accuracy, precision, recall, area under curve, false positive rate, and computational time for training the classifiers. The experimental results show that substantial feature reduction optimizes the performance of ML-based IDS for detecting DDoS attacks in standardized IoT networks employing 6LoWPAN stack.

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| S.NO | CONTENT | PGNO |
| 1 | Introduction   * 1. Objective   2. Problem Statement   3. Software requirements   4. Hardware requirements |  |
| 2 | Feasibility study  2.1 Economic feasibility  2.2 Technical feasibility  2.3 Social feasibility |  |
| 3 | Literature survey |  |
| 4 | System analysis  4.1 Existing system  4.1.1 Disadvantages of existing system  4.2 Proposed system  4.2.1 Advantages of proposed system  4.3 Functional requirements  4.4 Non-Functional requirements |  |
| 5 | System design  5.1 System architecture  5.2 UML diagrams |  |
| 6 | Implementation  6.1 Modules  6.2 Sample code |  |
| 7 | Software environment |  |
| 8 | System testing  8.1 Testing strategies  8.2 Test cases |  |
| 9 | Screens |  |
| 10 | Conclusion |  |
| 11 | References |  |

**INTRODUCTION**

**1. INTRODUCTION**

The Internet of Things (IoT) enables a vision where devices with sensing, actuating, computing, and communication capabilities can connect with each other. The IoT offers realization of a number of modern applications available today like smart cities, smart transportation, smart health care, etc. Other influencing factors include the availability of cheap Internet connections, development of built-in sensors in devices, the mobile revolution, and a multitude of companies developing necessary IoT applications and software. Techniques of DDoS detection include many approaches including machine learning (ML). Due to the huge data generated by IoT devices, ML techniques are quite helpful for continuous observations and in-depth analysis of IoT networks. Therefore, developing ML-based IDS seems to be a promising solution to secure IoT networks. ML-based algorithms, however, require data to be trained on and, thus, there is a need for IoT network traffic data set. As IoT consists newer protocols, there is a lack of appropriate data sets for training and evaluating ML-based IDS. Due to the deficiency of authentic IoT data sets, ML-based IDS lacks accurate and uniform performance advancements. In this work, we emphasize on securing the IoT networks from DDoS attacks with IDS.

* 1. **OBJECTIVE**

The main objective of this the application machine algorithms to detect DDOS Attacks by using different types of Machine learning and Deep learning Models.

* 1. **PROBLEM STATEMENT**

The proposed Project for prediction of DDOS attacks including the dataset, pre-processing, feature extraction and feature selection, algorithms, framework, and evaluation metrics, is presented and discusses the evaluation results of the experiments performed, and finally concludes the project with framework predict of credit card fraud.

* 1. **SOFTWARE REQUIREMENTS**

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

**Platform –** In computing, a platform describes some sort of framework, either in hardware or software, which allows software to run. Typical platforms include a computer’s architecture, operating system, or programming languages and their runtime libraries.

Operating system is one of the first requirements mentioned when defining system requirements (software). Software may not be compatible with different versions of same line of operating systems, although some measure of backward compatibility is often maintained. For example, most software designed for Microsoft Windows XP does not run on Microsoft Windows 98, although the converse is not always true. Similarly, software designed using newer features of Linux Kernel v2.6 generally does not run or compile properly (or at all) on Linux distributions using Kernel v2.2 or v2.4.

**APIs and drivers –** Software making extensive use of special hardware devices, like high-end display adapters, needs special API or newer device drivers. A good example is DirectX, which is a collection of APIs for handling tasks related to multimedia, especially game programming, on Microsoft platforms.

**Web browser –** Most web applications and software depending heavily on Internet technologies make use of the default browser installed on system. Microsoft Internet Explorer is a frequent choice of software running on Microsoft Windows, which makes use of ActiveX controls, despite their vulnerabilities.

1. **Software : Anaconda**
2. **Primary Language : Python**
3. **Frontend Framework : Flask**
4. **Back-end Framework : Jupyter Notebook**
5. **Database : Sqlite3**
6. **Front-End Technologies : HTML,CSS,JavaScript and Bootstrap4**

**1.2 HARDWARE REQUIREMENTS**

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

**Architecture –** All computer operating systems are designed for a particular computer architecture. Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture. See also a list of common operating systems and their supporting architectures.

**Processing power –** The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 architecture define processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored. This definition of power is often erroneous, as AMD Athlon and Intel Pentium CPUs at similar clock speed often have different throughput speeds. Intel Pentium CPUs have enjoyed a considerable degree of popularity, and are often mentioned in this category.

**Memory –** All software, when run, resides in the random access memory (RAM) of a computer. Memory requirements are defined after considering demands of the application, operating system, supporting software and files, and other running processes. Optimal performance of other unrelated software running on a multi-tasking computer system is also considered when defining this requirement.

**Secondary storage –** Hard-disk requirements vary, depending on the size of software installation, temporary files created and maintained while installing or running the software, and possible use of swap space (if RAM is insufficient).

**Display adapter –** Software requiring a better than average computer graphics display, like graphics editors and high-end games, often define high-end display adapters in the system requirements.

**Peripherals –** Some software applications need to make extensive and/or special use of some peripherals, demanding the higher performance or functionality of such peripherals. Such peripherals include CD-ROM drives, keyboards, pointing devices, network devices, etc.

**1) Operating System : Windows Only**

**2) Processor : i5 and above**

**3) Ram : 8gb and above**

**4) Hard Disk : 25 GB in local drive**

**FEASIBILITY STUDY**

**2. FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**2.1 ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### **2.2 TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**2.3 SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**LITERATURE SURVEY**

**3. LITERATURE SURVEY**

**3.1 Boosting-Based DDoS Detection in Internet of Things Systems:**

[[PDF] Boosting-Based DDoS Detection in Internet of Things Systems | Semantic Scholar](https://www.semanticscholar.org/paper/Boosting-Based-DDoS-Detection-in-Internet-of-Things-Cviti%C4%87-Perakovi%C4%87/d8c0629659164ae943e3f472a0bcf29ebfbd5698)

**ABSTRACT:** Distributed Denial-of-Service (DDoS) attacks remain challenging to mitigate in the existing systems, including in-home networks that comprise different Internet of Things (IoT) devices. In this article, we present a DDoS traffic detection model that uses a boosting method of logistic model trees for different IoT device classes. Specifically, a different version of the model will be generated and applied for each device class since the characteristics of the network traffic from each device class may have subtle variation(s). As a case study, we explain how devices in a typical smart home environment can be categorized into four different classes (and in our context, Class 1—very high level of traffic predictability, Class 2—high level of traffic predictability, Class 3—medium level of traffic predictability, and Class 4—low level of traffic predictability). Findings from our evaluations show that the accuracy of our proposed approach is between 99.92% and 99.99% for these four device classes. In other words, we demonstrate that we can use device classes to help us more effectively detect DDoS traffic.

**3.2 IoT-Sentry: A Cross-Layer-Based Intrusion Detection System in Standardized Internet of Things:**

[IoT-Sentry: A Cross-Layer-Based Intrusion Detection System in Standardized Internet of Things | Semantic Scholar](https://www.semanticscholar.org/paper/IoT-Sentry%3A-A-Cross-Layer-Based-Intrusion-Detection-Kamaldeep-Malik/8c4907aa71ceb9d717e4a79bd042d2e233cdbf80)

**ABSTRACT:** With the widespread expansion of internet connected devices, securing the Internet of Things (IoT) has become one of the biggest challenges and a crucial issue for ensuring a secure and robust IoT vision. The usage of adhoc topologies and the resource constraints of IoT devices and networks makes first line of defense mechanisms like cryptography unsuitable to implement. Therefore, second line of defense mechanisms such as the intrusion detection system become fundamental to detect attacks. In this work, we propose IoT-Sentry, a cross-layer intrusion detection system to detect five different attacks with zero additional overhead. Also, there is a lack of rich, illustrative and concise public datasets in IoT for evaluation of intrusion detection models for IoT networks. Our research work fills this research gap by (1) developing a novel cross-layer IoT dataset which contains four simulation instances for the five attacks in static IoT networks with up to 100 nodes, (2) utilize novel features of cross-layer attacks, and (3) employ ensemble learning model to detect IoT attacks. In our work, the Cooja IoT simulator has been utilized for generating malicious and benign traffic. The IoT attack dataset is then analyzed and fed into centralized detection module implemented at the 6LoWPAN border router. On evaluation, IoT-Sentry achieves on average an accuracy of 99% for 4 out of 5 attacks. To the best of our knowledge, this is the first time that cross layer intrusion detection system catering to five attacks from different layers is developed for securing standardized IoT networks.

**3.3 Learning-Driven Detection and Mitigation of DDoS Attack in IoT via SDN-Cloud Architecture:**

[Learning-Driven Detection and Mitigation of DDoS Attack in IoT via SDN-Cloud Architecture | IEEE Journals & Magazine | IEEE Xplore](https://ieeexplore.ieee.org/document/8993716)

**ABSTRACT:** The Internet-of-Things (IoT) network is growing big owing to its utility in smart applications. An IoT network is susceptible to security breaches, in majority due to the resource-constrained nature of IoT. Of the various breaches, the Distributed Denial-of-Service (DDoS) attack can snip off the network service to the users in various ways, such as consumption of server's resources, saturating link bandwidth, etc. These types of DDoS breaches can turn out to be a catastrophe in critical IoT use cases. This article delves into tackling the DDoS attack triggered by malicious wireless IoT on IoT servers. Our security scheme leverages the cloud and software-defined network (SDN) paradigm to mitigate the DDoS attack on IoT servers. We have proposed a novel mechanism named learning-driven detection mitigation (LEDEM) that detects DDoS using a semisupervised machine-learning algorithm and mitigates DDoS. We tested LEDEM in the testbed and emulated topology, and compared the results with state-of-the-art solutions. We achieved an improved accuracy rate of 96.28% in detecting DDoS attack.

**3.4 Evaluation of Network Intrusion Detection Systems for RPL Based 6LoWPAN Networks in IoT:**

[Evaluation of Network Intrusion Detection Systems for RPL Based 6LoWPAN Networks in IoT | SpringerLink](https://link.springer.com/article/10.1007/s11277-019-06485-w)

**ABSTRACT:** Over the past few years, Internet of Things security has attracted the attention of many researchers due to its challenging and constrained nature. Particularly in the development of Network Intrusion Detection Systems which act as first line of defense for the networks. Due to the lack of reliable Internet of Things based datasets, intrusion detection approaches are suffering from uniform and accurate performance advancements. Existing benchmark datasets like KDD99, NSL-KDD cup 99 are obsolete and unfit for the evaluation of Network Intrusion Detection Systems developed for RPL based 6LoWPAN networks. To address this issue, the RPL-NIDDS17 dataset has recently been generated. This dataset consists seven types of modern routing attack patterns along with normal traffic patterns. In the proposed dataset we consider twenty two attributes that comprise of flow, basic, time type of features and two additional labelling attributes. In this study, we have shown the effectiveness of RPL-NIDDS17 by statistically analysing the probability distribution of features, correlation between features. Complexity analysis of the developed dataset is done by evaluating five machine learning techniques on the dataset. Evaluation results are shown in terms of two prominent metrics accuracy and false alarm rate, and compared with the results of KDD99, UNSW-NB15, WSN-DS datasets. The experimental results are presented to show the suitability of our proposed RPL-NIDDS17 dataset for the evaluation of Network Intrusion Detection Systems in Internet of Things.

**3.5 The Internet of Things: a movement, not a market:**

[The Internet of Things: a movement, not a market / the-internet-of-things-a-movement-not-a-market.pdf / PDF4PRO](https://pdf4pro.com/view/the-internet-of-things-a-movement-not-a-market-3ed060.html)

**ABSTRACT:** The Internet of things: a movement, not a market COMPUTE CONNECT COLLECTCREATE2 IoT is more than just the latest buzzword about the future of business and technology. IoT is transforming everyday business practices and opening new windows of opportunity. IoT is here, and the future is happening Internet of things : revolutionizing the competitive landscape The number of connected IoT devices worldwide will jump 12% on average annually, from nearly 27 billion in 2017 to 125 billion in you know? Global data transmissions are expected to increase from 20-25% annually to 50% per year, on average, in the next 15 years. 2 COMPUTECONNECTCOLLECTCREATE3 Despite the revolutionary potential of IoT, many companies have trouble identifying a consistent IoT strategy. The Internet of Things: a movement, not a market. CONNECT COLLECT COMPUTE CREATE 2 IoT is more than just the latest buzzword about the future of business and technology. IoT is transforming everyday business practices and opening new windows of opportunity.

**SYSTEM ANALYSIS**

**4. SYSTEM ANALYSIS**

**4.1 EXISTING SYSTEM:**

In literature they focused on developing techniques for automatically detecting distributed denial of service (DDoS) attacks in consumer Internet of Things (IoT) devices. They demonstrate that using IoT-specific network behaviors, such as limited number of endpoints and regular time intervals between packets, can inform feature selection and result in high accuracy DDoS detection in IoT network traffic. They use a variety of machine learning algorithms, including neural networks, to achieve this. These results indicate that home gateway routers or other network middle boxes could automatically detect local IoT device sources of DDoS attacks using low-cost machine learning algorithms and traffic data that is flow-based and protocol-agnostic.

**4.1.1 DISADVANTAGES OF EXISTING SYSTEM:**

* The existing work relies on "correlated features," which might not be as comprehensive or informative as the network properties and behaviors used in the our work.
* The existing work uses the CICDDoS2019 dataset, The choice of dataset can impact the generalization and real-world applicability of the DDoS detection model. This dataset is not widely used for intrusion detection research and may not provide a more diverse and representative set of data.
* The existing work highlights only one AdaBoost as their algorithm.
* While the another existing work mentions HTTP flood, SID DoS, and normal traffic, it might lack the diversity of attack types and scenarios present in our work.

# 4.2 Proposed System:

We propose a machine learning method for detecting Distributed Denial of Service (DDoS) attacks that involves data acquisition, feature extraction and classification, and binary classification. The proposed method utilizes network properties such as packet length, inter-packet intervals, and protocol, as well as network behaviors as features. We evaluate the performance of various attack detection classifiers, including Logistic Regression, Random Forests, and K-Nearest Neighbor. To validate our proposed method, we use the NSL KDD dataset in our experiments.

# 4.2.1 Advantages of proposed system:

* The existing work primarily relies on IoT-specific network behaviors for feature selection. This approach might not capture all the intricate patterns and features that could contribute to accurate DDoS detection, potentially leading to false negatives or false positives.
* The existing work mentions the use of "low-cost machine learning algorithms," but it doesn't specify the variety or complexity of the algorithms used. This could limit the overall performance and adaptability of the detection system to evolving attack strategies.
* The existing work doesn't explicitly mention cross-layer analysis, which could result in missing important insights that could be leveraged for more effective detection.
* The existing work does not used hyper parameter optimization.

### **4.3 FUNCTIONAL REQUIREMENTS**

1. Data Collection

2. Data Pre-processing

3. Training and Testing

4. Modiling

5. Predicting

### **4.4 NON FUNCTIONAL REQUIREMENTS**

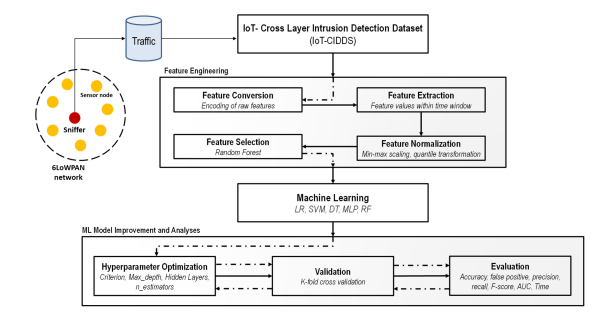
NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *“how fast does the website load?”* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non- functional Requirements allow you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users is > 10000. Description of non-functional requirements is just as critical as a functional requirement.

* Usability requirement
* Serviceability requirement
* Manageability requirement
* Recoverability requirement
* Security requirement
* Data Integrity requirement
* Capacity requirement
* Availability requirement
* Scalability requirement
* Interoperability requirement
* Reliability requirement
* Maintainability requirement
* Regulatory requirement
* Environmental requirement

**SYSTEM DESIGN**

**5. SYSTEM DESIGN**

**5.1 SYSTEM ARCHITECTURE:**

****

**Fig.5.1.1 System architecture**

**DATA FLOW DIAGRAM:**

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

**Import libraries**

**VERIFY**

**NO PROCESS**

**Yes NO**

**Importing the dataset**

**Data processing**

**Feature Selection**

**Splitting the data into train & test**

**Building the model - Decision Tree, Logistic Regression, MLP, SVM, Random Forest, Stacking Classifier - RF + MLP with LightGBM, Voting CLassifier - RF + AB**

**Training the model**

**Signup & sign in**

**User input**

**Final outcome**

**End process**

**5.2 UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

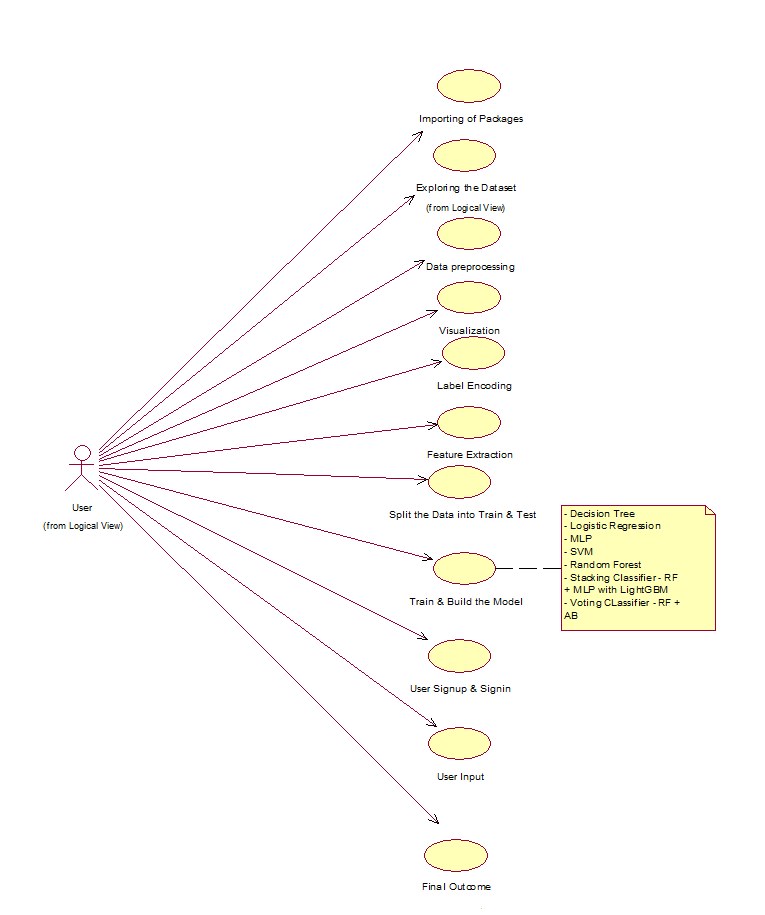
**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

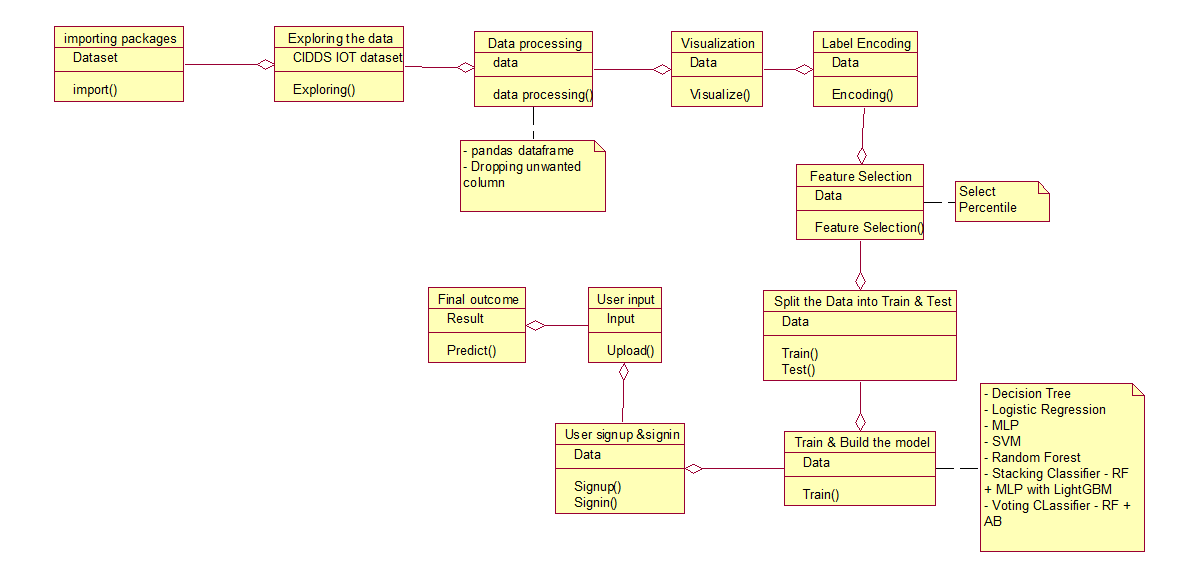
**Use case diagram:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



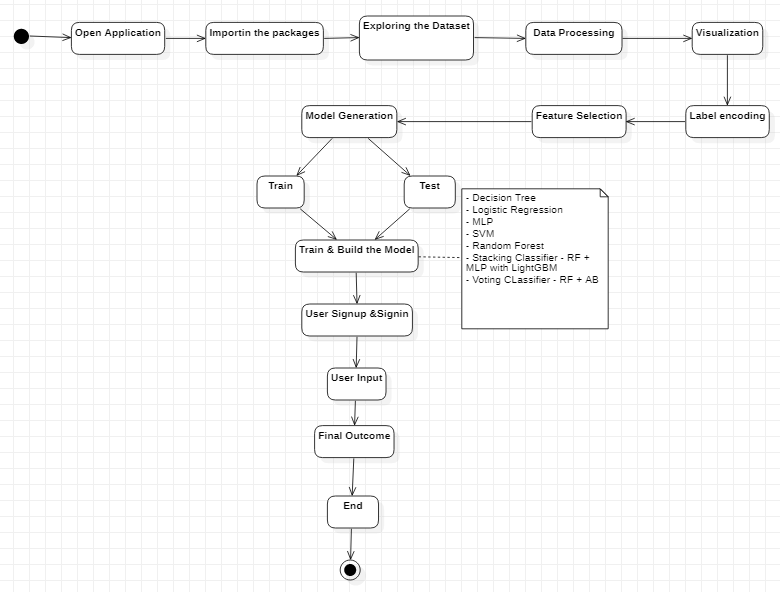
**Class diagram:**

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

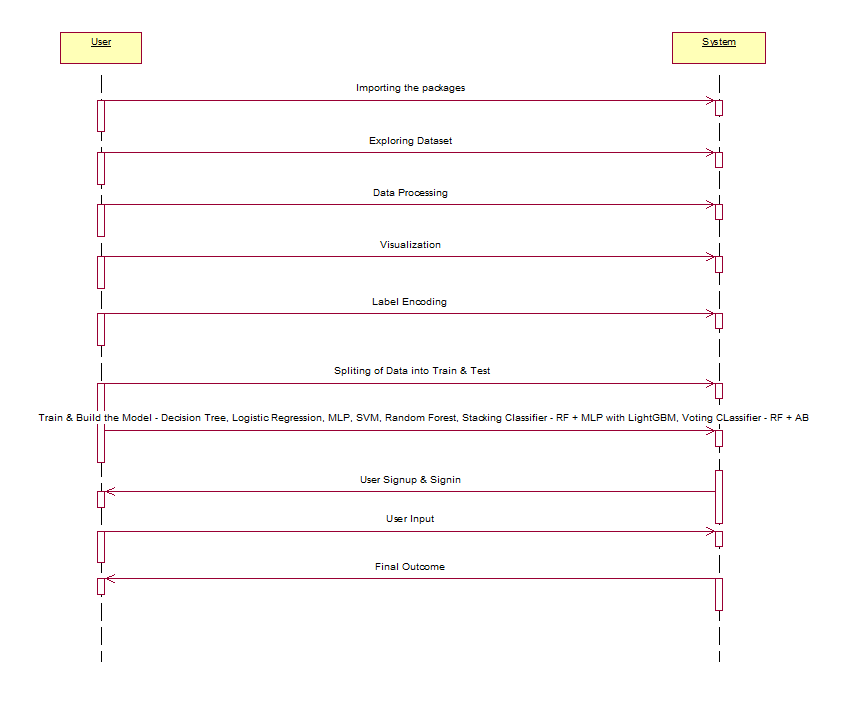


**Activity diagram:**

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.

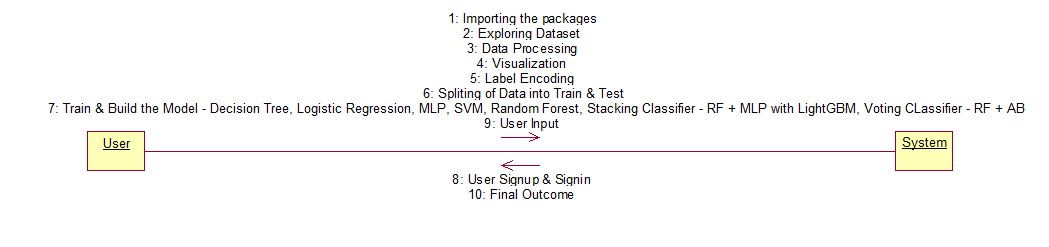
** Sequence diagram:**

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".



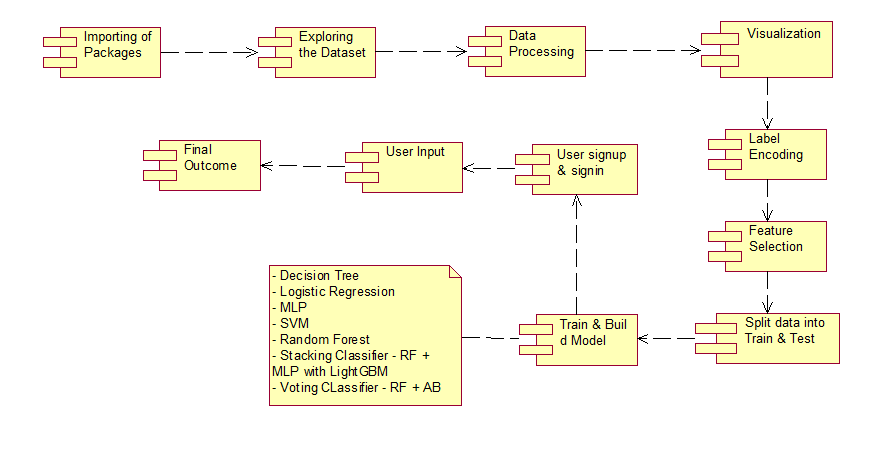
**Collaboration diagram:**

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.

****

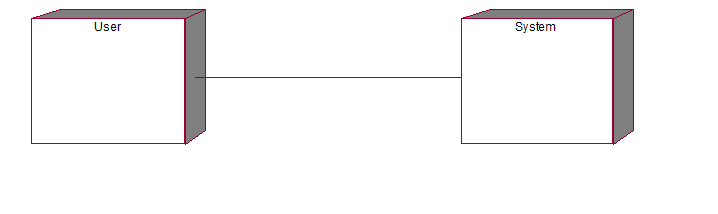
**Component diagram:**

The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase.

v

**Deployment diagram:**

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.

****

**IMPLEMENTATION**

1. **IMPLEMENTATION**

MODULES:

* + Data exploration: using this module we will load data into system
  + Processing: Using the module we will read data for processing
  + Splitting data into train & test: using this module data will be divided into train & test
  + Model generation: Model building - Decision Tree, Logistic Regression, MLP, SVM, Random Forest, Stacking Classifier - RF + MLP with LightGBM, Voting CLassifier - RF + AB
  + User signup & login: Using this module will get registration and login
  + User input: Using this module will give input for prediction
  + Prediction: final predicted displayed

**Note:** As an extension we applied an ensemble method combining the predictions of multiple individual models to produce a more robust and accurate final prediction.

However, we can further enhance the performance by exploring other ensemble techniques such as Voting Classifier and Stacking Classifier got 99% and 100% of accuracy.

**Algorithms:**

Random Forest: Random forest is a commonly-used machine learning algorithm trademarked by Leo Breiman and Adele Cutler, which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.

KNN: K-Nearest Neighbors Algorithm. The k-nearest neighbors’ algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

Logistic Regression: Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

Voting Classifier (RF + AdaBoost): Voting Classifier (AB + RF): A voting classifier is a machine learning estimator that trains various base models or estimators and predicts on the basis of aggregating the findings of each base estimator. The aggregating criteria can be combined decision of voting for each estimator output.

Stacking Classifier (RF + MLP with LightGBM ): Stacking Classifier: Stacking is a way of ensembling classification or regression models it consists of two-layer estimators. The first layer consists of all the baseline models that are used to predict the outputs on the test datasets. The second layer consists of Meta-Classifier or Regressor which takes all the predictions of baseline models as an input and generates new predictions.

**6.2 SAMPLE CODE:**

# Necessary imports

import numpy as np

import pandas as pd

import tensorflow as tf

import matplotlib.pyplot as plt

import seaborn as sns

import time

import multiprocessing

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from collections import Counter

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.model\_selection import train\_test\_split

from imblearn.over\_sampling import SMOTE, ADASYN

from sklearn.metrics import confusion\_matrix, r2\_score, mean\_squared\_error

from sklearn.metrics import precision\_score, recall\_score, f1\_score, roc\_auc\_score, accuracy\_score, classification\_report, precision\_recall\_curve

import warnings

warnings.filterwarnings("ignore")

df = pd.read\_csv("/kaggle/input/network-anamoly-detection/Train.txt",sep=",",names=["duration","protocoltype","service","flag","srcbytes","dstbytes","land", "wrongfragment","urgent","hot","numfailedlogins","loggedin", "numcompromised","rootshell","suattempted","numroot","numfilecreations", "numshells","numaccessfiles","numoutboundcmds","ishostlogin",

"isguestlogin","count","srvcount","serrorrate", "srvserrorrate",

"rerrorrate","srvrerrorrate","samesrvrate", "diffsrvrate", "srvdiffhostrate","dsthostcount","dsthostsrvcount","dsthostsamesrvrate", "dsthostdiffsrvrate","dsthostsamesrcportrate",

"dsthostsrvdiffhostrate","dsthostserrorrate","dsthostsrvserrorrate",

"dsthostrerrorrate","dsthostsrvrerrorrate","attack", "lastflag"])

df.head()

df.drop(['land','urgent','numfailedlogins','numoutboundcmds'],axis=1,inplace=True)

df.isna().sum()

df.select\_dtypes(exclude=[np.number])

df['attack'].loc[df['attack']!='normal']='attack'

le=LabelEncoder()

df['protocoltype']=le.fit\_transform(df['protocoltype'])

df['service']=le.fit\_transform(df['service'])

df['flag']=le.fit\_transform(df['flag'])

df['attack']=le.fit\_transform(df['attack'])

plt.figure(figsize=(20,15))

sns.heatmap(df.corr())

X=df.drop(['attack'],axis=1)

y=df['attack']

sns.countplot(df['attack'])

print("Class distribution: {}".format(Counter(y)))

scaler = StandardScaler()

scaler.fit(X)

X\_transformed = scaler.transform(X)

lr=LogisticRegression()

lr.fit(X\_transformed,y)

lr\_pred=lr.predict(X\_transformed)

lr\_df=pd.DataFrame()

lr\_df['actual']=y

lr\_df['pred']=lr\_pred

lr\_df.head()

print(accuracy\_score(y, lr\_pred))

print(classification\_report(y, lr\_pred))

rf=RandomForestClassifier()

rf.fit(X\_transformed,y)

rf\_pred=rf.predict(X\_transformed)

rf\_df=pd.DataFrame()

rf\_df['actual']=y

rf\_df['pred']=rf\_pred

rf\_df.head()

print(accuracy\_score(y, lr\_pred))

print(classification\_report(y, lr\_pred))

test\_df = pd.read\_csv("/kaggle/input/network-anamoly-detection/Test.txt",sep=",",names=["duration","protocoltype","service","flag","srcbytes","dstbytes","land", "wrongfragment","urgent","hot","numfailedlogins","loggedin", "numcompromised","rootshell","suattempted","numroot","numfilecreations", "numshells","numaccessfiles","numoutboundcmds","ishostlogin",

"isguestlogin","count","srvcount","serrorrate", "srvserrorrate",

"rerrorrate","srvrerrorrate","samesrvrate", "diffsrvrate", "srvdiffhostrate","dsthostcount","dsthostsrvcount","dsthostsamesrvrate", "dsthostdiffsrvrate","dsthostsamesrcportrate",

"dsthostsrvdiffhostrate","dsthostserrorrate","dsthostsrvserrorrate",

"dsthostrerrorrate","dsthostsrvrerrorrate","attack", "lastflag"])

test\_df.head()

test\_df.select\_dtypes(exclude=[np.number])

test\_df['attack'].loc[test\_df['attack']!='normal']='attack'

test\_df['protocoltype']=le.fit\_transform(test\_df['protocoltype'])

test\_df['service']=le.fit\_transform(test\_df['service'])

test\_df['flag']=le.fit\_transform(test\_df['flag'])

test\_df['attack']=le.fit\_transform(test\_df['attack'])

test\_df.drop(['land','urgent','numfailedlogins','numoutboundcmds'],axis=1,inplace=True)

X\_test=test\_df.drop(['attack'],axis=1)

y\_test=test\_df['attack']

sns.countplot(test\_df['attack'])

X\_test\_transformed = scaler.transform(X\_test)

test\_pred=rf.predict(X\_test\_transformed)

rf\_test\_df=pd.DataFrame()

rf\_test\_df['actual']=y\_test

rf\_test\_df['pred']=test\_pred

rf\_test\_df.head()

target\_names=["attack","normal"]

print(classification\_report(y\_test, test\_pred,target\_names=target\_names))

tn, fp, fn, tp = confusion\_matrix(y\_test, test\_pred).ravel()

print("True Negatives:",tn)

print("False Positives:",fp)

print("False Negatives:",fn)

print("True Positives:",tp)

**SOFTWARE ENVIRONMENT**

**7. SOFTWARE ENVIRONMENT**

**MACHINE LEARNING:**

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain.Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

## Challenges in Machines Learning:-

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

**Quality of data** − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

**Time-Consuming task** − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

**Lack of specialist persons** − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

**No clear objective for formulating business problems** − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

**Issue of over fitting & under fitting** − If the model is over fitting or under fitting, it cannot be represented well for the problem.

**Curse of dimensionality** − another challenge ML model faces is too many features of data points. This can be a real hindrance.

**Difficulty in deployment** − Complexity of the ML model makes it quite difficult to be deployed in real life.

**DEEP LEARNING**

Deep learning is a branch of machine learning which is based on artificial neural networks. It is capable of learning complex patterns and relationships within data. In deep learning, we don’t need to explicitly program everything. It has become increasingly popular in recent years due to the advances in processing power and the availability of large datasets. Because it is based on artificial neural networks (ANNs) also known as deep neural networks (DNNs). These neural networks are inspired by the structure and function of the human brain’s biological neurons, and they are designed to learn from large amounts of data.

## What is Anaconda for Python?

Anaconda Python is a free, open-source platform that allows you to write and execute code in the programming language Python. It is by continuum.io, a company that specializes in Python development. The Anaconda platform is the most popular way to learn and use Python for scientific computing, data science, and machine learning. It is used by over [thirty](https://www.anaconda.com/blog/10-years-of-data-science-innovation-anacondas-commitment-to-the-open-source-python-community)[million people](https://www.anaconda.com/blog/10-years-of-data-science-innovation-anacondas-commitment-to-the-open-source-python-community) worldwide and is available for Windows, macOS, and Linux.

People like using Anaconda Python because it simplifies package deployment and management. It also comes with a large number of libraries/packages that you can use for your projects. Since Anaconda Python is free and open-source, anyone can contribute to its development.

**What is Anaconda for Python?**

Anaconda software helps you create an environment for many different versions of Python and package versions. Anaconda is also used to install, remove, and upgrade packages in your project environments. Furthermore, you may use Anaconda to deploy any required project with a few mouse clicks. This is why it is perfect for beginners who want to learn Python.

Now that you know what Anaconda Python is, let's look at how to install it.

**How to install Anaconda for Python?**



To install Anaconda, just head to the Anaconda Documentation website and follow the instructions to download the installer for your operating system. Once the installer successfully downloads, double-click on it to start the installation process.

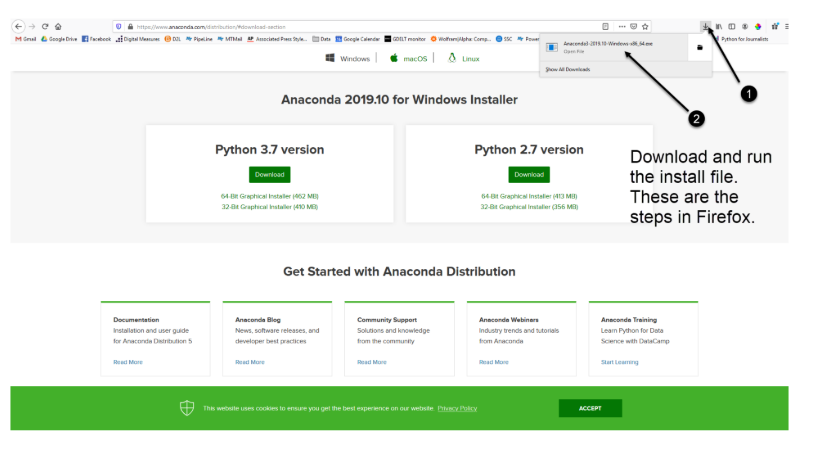
Follow the prompts and agree to the terms and conditions. When you are asked if you want to "add Anaconda to my PATH environment variable," make sure that you select "yes." This will ensure that Anaconda is added to your system's PATH, which is a list of directories that your operating system uses to find the files it needs.

Once the installation is complete, you will be asked if you want to "enable Anaconda as my default Python." We recommend selecting "yes" to use Anaconda as your default Python interpreter.

### **Python Anaconda Installation**

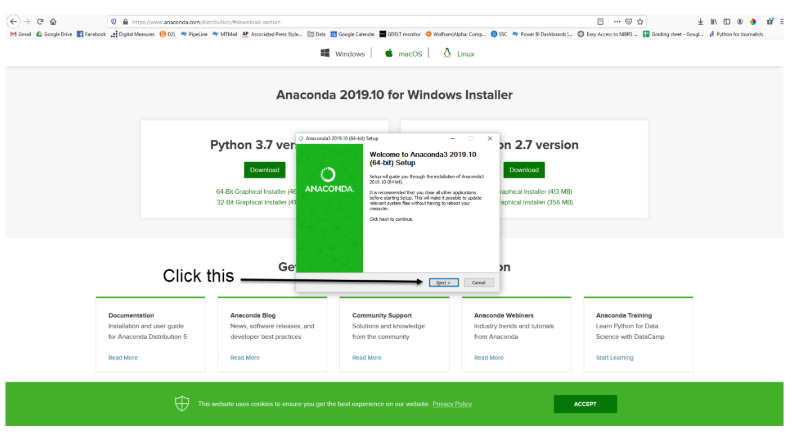
Next in the Python anaconda tutorial is its installation. The latest version of Anaconda at the time of writing is 2019.10. Follow these steps to download and install Anaconda on your machine:

1. Go to this link and download Anaconda for Windows, Mac, or Linux: – [Download anaconda](https://www.anaconda.com/distribution/)

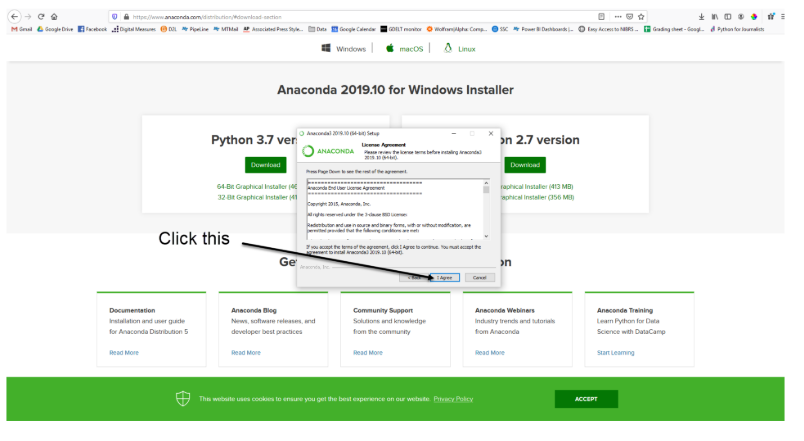


You can download the installer for Python 3.7 or for Python 2.7 (at the time of writing). And you can download it for a 32-bit or 64-bit machine.

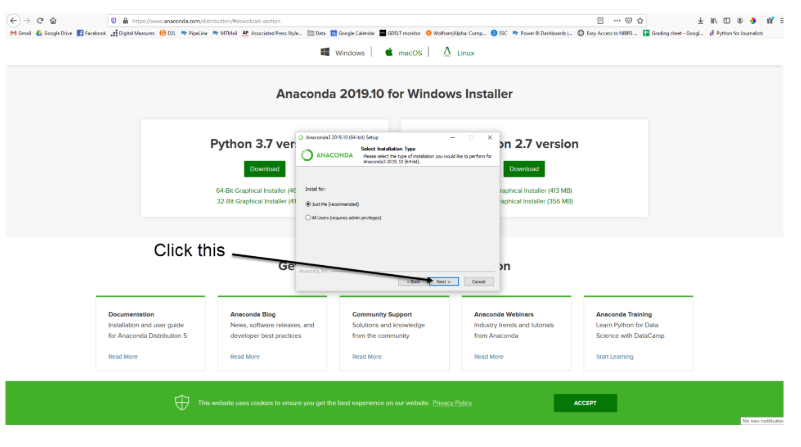
2. Click on the downloaded .exe to open it. This is the Anaconda setup. Click next.



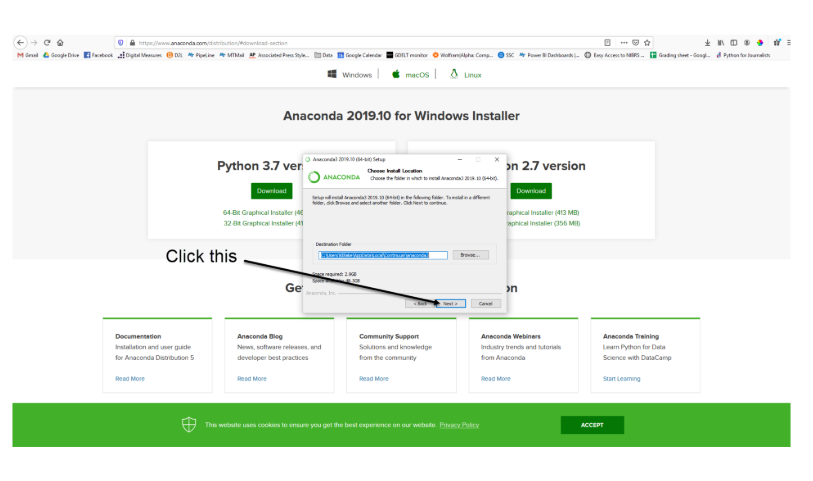
3. Now, you’ll see the license agreement. Click on ‘I Agree’.



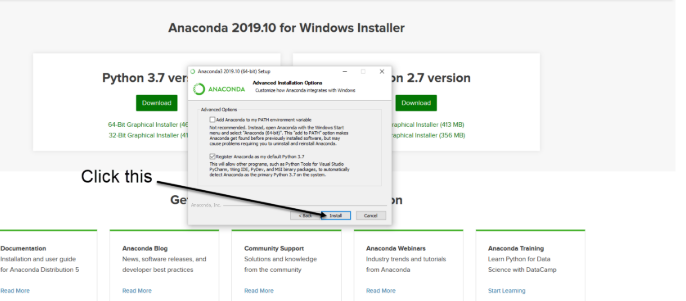
4. You can install it for all users or just for yourself. If you want to install it for all users, you need administrator privileges.



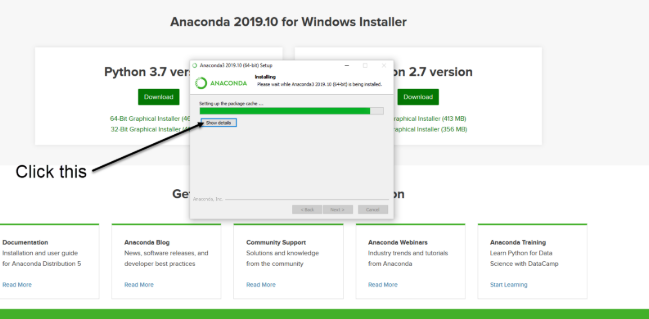
5. Choose where you want to install it. Here, you can see the available space and how much you need.



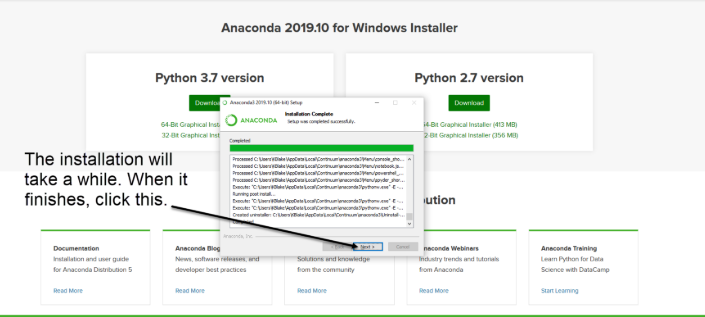
6. Now, you’ll get some advanced options. You can add Anaconda to your system’s PATH environment variable, and register it as the primary system Python 3.7. If you add it to PATH, it will be found before any other installation. Click on ‘Install’.



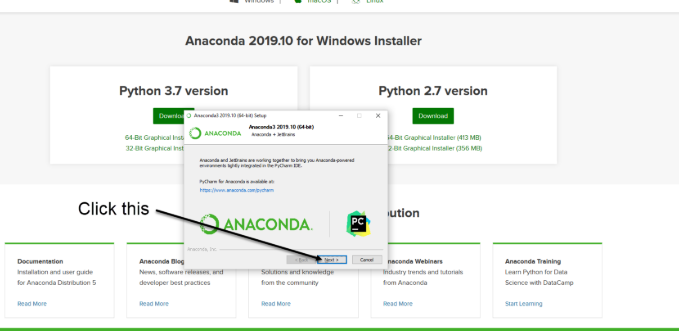
7. It will unpack some packages and extract some files on your machine. This will take a few minutes.



8. The installation is complete. Click Next.



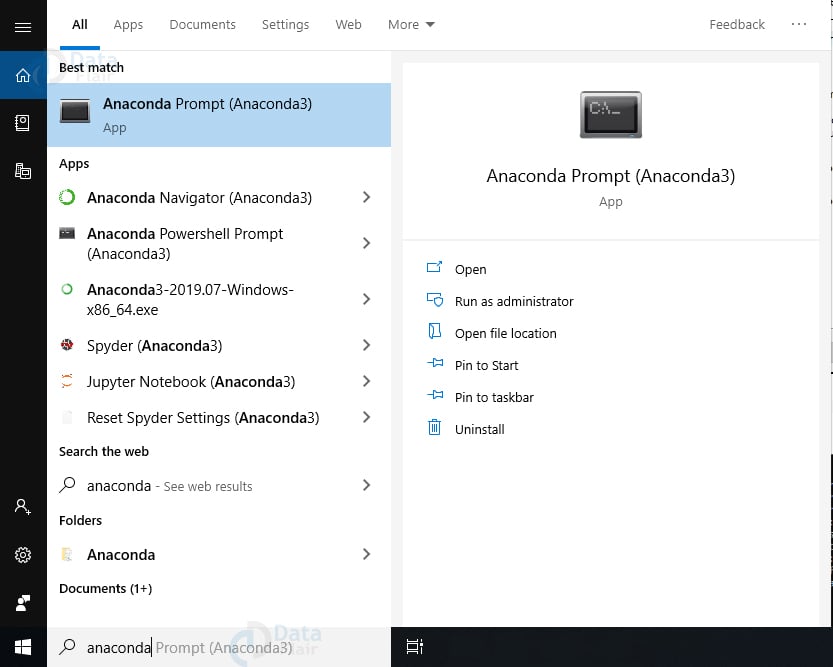
9. This screen will inform you about PyCharm. Click Next.



10. The installation is complete. You can choose to get more information about Anaconda cloud and how to get started with Anaconda. Click Finish.



11. If you search for Anaconda now, you will see the following options:



**PYTHON LANGUAGE:**

Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding; make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Python is a dynamic, high-level, free open source, and interpreted programming language. It supports object-oriented programming as well as procedural-oriented programming. In Python, we don’t need to declare the type of variable because it is a dynamically typed language. For example, x = 10 Here, x can be anything such as String, int, etc.

## Features in Python:

There are many features in Python, some of which are discussed below as follows:

### **1. Free and Open Source**

[Python](https://www.geeksforgeeks.org/python-programming-language/)language is freely available at the official website and you can download it from the given download link below click on the **Download Python** keyword. [Download Python](https://www.python.org/downloads/) Since it is open-source, this means that source code is also available to the public. So you can download it, use it as well as share it.

### **2. Easy to code**

Python is a [high-level programming language](https://www.geeksforgeeks.org/difference-between-high-level-and-low-level-languages/). Python is very easy to learn the language as compared to other languages like C, C#, JavaScript, Java, etc. It is very easy to code in the Python language and anybody can learn Python basics in a few hours or days. It is also a developer-friendly language.

### 3. Easy to Read

As you will see, learning Python is quite simple. As was already established, Python’s syntax is really straightforward. The code block is defined by the indentations rather than by semicolons or brackets.

### **4. Object-Oriented Language**

One of the key features of [Python is Object-Oriented programming](https://www.geeksforgeeks.org/python-oops-concepts/). Python supports object-oriented language and concepts of classes, object encapsulation, etc.

### **5. GUI Programming Support**

Graphical User interfaces can be made using a module such as [PyQt5](https://www.geeksforgeeks.org/pyqt5-qaction/), PyQt4, wxPython, or [Tk in python](https://www.geeksforgeeks.org/python-gui-tkinter/). PyQt5 is the most popular option for creating graphical apps with Python.

### **6. High-Level Language**

Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture, nor do we need to manage the memory.

### **7. Extensible feature**

Python is an **Extensible** language. We can write some Python code into C or C++ language and also we can compile that code in C/C++ language.

### 8. Easy to Debug

Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program’s issues once you understand how to [interpret](https://www.geeksforgeeks.org/difference-between-compiled-and-interpreted-language/)Python’s error traces. Simply by glancing at the code, you can determine what it is designed to perform.

### **9. Python is a Portable language**

Python language is also a portable language. For example, if we have Python code for windows and if we want to run this code on other platforms such as [Linux](https://www.geeksforgeeks.org/introduction-to-linux-operating-system/), Unix, and Mac then we do not need to change it, we can run this code on any platform.

### **10. Python is an integrated language**

Python is also an integrated language because we can easily integrate Python with other languages like C, [C++](http://www.geeksforgeeks.org/c-plus-plus/), etc.

### **11. Interpreted Language:**

Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, [Java](https://www.geeksforgeeks.org/java/), etc. there is no need to compile Python code this makes it easier to debug our code. The source code of Python is converted into an immediate form called **byte code**.

### **12. Large Standard Library**

Python has a large [standard library](https://www.geeksforgeeks.org/libraries-in-python/) that provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in Python such as [regular expression](https://www.geeksforgeeks.org/regular-expression-python-examples-set-1/)s, [unit-testing](https://www.geeksforgeeks.org/unit-testing-software-testing/), web browsers, etc.

### **13. Dynamically Typed Language**

Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don’t need to specify the type of variable.

### **14. Frontend and backend development**

With a new project py script, you can run and write Python codes in HTML with the help of some simple tags <py-script>, <py-env>, etc. This will help you do frontend development work in Python like JavaScript. Backend is the strong forte of Python it’s extensively used for this work cause of its frameworks like [Django](https://www.geeksforgeeks.org/django-tutorial/)and [Flask](https://www.geeksforgeeks.org/flask-creating-first-simple-application/).

### 15. Allocating Memory Dynamically

In Python, the variable data type does not need to be specified. The memory is automatically allocated to a variable at runtime when it is given a value. Developers do not need to write int y = 18 if the integer value 15 is set to y. You may just type y=18.

**LIBRARIES/PACKGES:-**

**Tensor flow**

Tensor Flow is a [free](https://en.wikipedia.org/wiki/Free_software) and [open-source](https://en.wikipedia.org/wiki/Open-source_software) [software library for dataflow and differentiable programming](https://en.wikipedia.org/wiki/Library_(computing)) across a range of tasks. It is a symbolic math library, and is also used for [machine learning](https://en.wikipedia.org/wiki/Machine_learning) applications such as [neural networks](https://en.wikipedia.org/wiki/Neural_networks). It is used for both research and production at [Google](https://en.wikipedia.org/wiki/Google).‍

TensorFlow was developed by the [Google Brain](https://en.wikipedia.org/wiki/Google_Brain) team for internal Google use. It was released under the [Apache 2.0](https://en.wikipedia.org/wiki/Apache_License) [open-source license](https://en.wikipedia.org/wiki/Open-source_license) on November 9, 2015.

**Numpy**

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

**Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and [IPython](http://ipython.org/) shells, the [Jupyter](http://jupyter.org/) Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the [sample plots](https://matplotlib.org/tutorials/introductory/sample_plots.html) and [thumbnail gallery](https://matplotlib.org/gallery/index.html).

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

**Scikit – learn**

Scikit-learn provide a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

**SYSTEM TESTING**

**8. SYSTEM TESTING**

System testing, also referred to as system-level tests or system-integration testing, is the process in which a quality assurance (QA) team evaluates how the various components of an application interact together in the full, integrated system or application. System testing verifies that an application performs tasks as designed. This step, a kind of black box testing, focuses on the functionality of an application. System testing, for example, might check that every kind of user input produces the intended output across the application.

Phases of system testing:

A video tutorial about this test level. System testing examines every component of an application to make sure that they work as a complete and unified whole. A QA team typically conducts system testing after it checks individual modules with functional or user-story testing and then each component through integration testing.

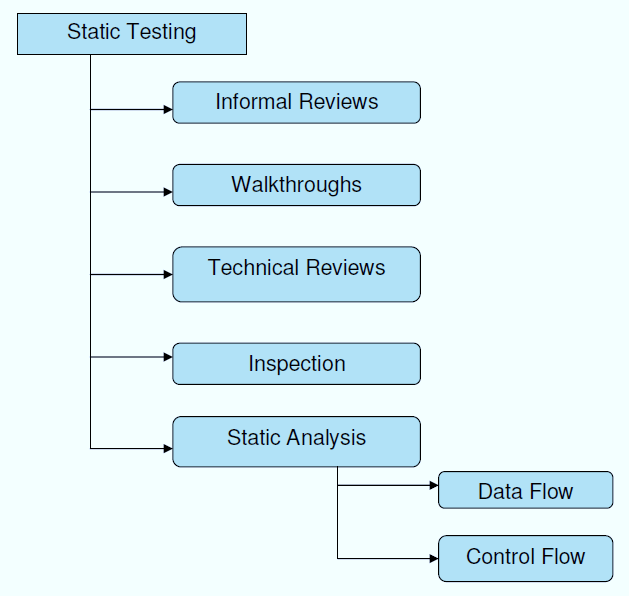
If a software build achieves the desired results in system testing, it gets a final check via acceptance testing before it goes to production, where users consume the software. An app-dev team logs all defects, and establishes what kinds and amount of defects are tolerable.

**8.1 Software Testing Strategies:**

Optimization of the approach to testing in software engineering is the best way to make it effective. A software testing strategy defines what, when, and how to do whatever is necessary to make an end-product of high quality. Usually, the following software testing strategies and their combinations are used to achieve this major objective:

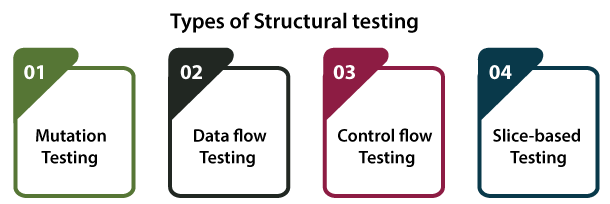
Static Testing:

The early-stage testing strategy is static testing: it is performed without actually running the developing product. Basically, such desk-checking is required to detect bugs and issues that are present in the code itself. Such a check-up is important at the pre-deployment stage as it helps avoid problems caused by errors in the code and software structure deficits.



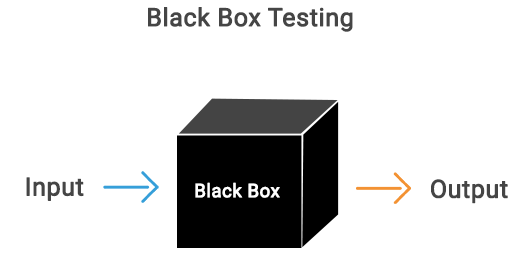
**Structural Testing:**

It is not possible to effectively test software without running it. Structural testing, also known as white-box testing, is required to detect and fix bugs and errors emerging during the pre-production stage of the software development process. At this stage, unit testing based on the software structure is performed using regression testing. In most cases, it is an automated process working within the test automation framework to speed up the development process at this stage. Developers and QA engineers have full access to the software’s structure and data flows (data flows testing), so they could track any changes (mutation testing) in the system’s behavior by comparing the tests’ outcomes with the results of previous iterations (control flow testing).



**Behavioral Testing:**

The final stage of testing focuses on the software’s reactions to various activities rather than on the mechanisms behind these reactions. In other words, behavioral testing, also known as black-box testing, presupposes running numerous tests, mostly manual, to see the product from the user’s point of view. QA engineers usually have some specific information about a business or other purposes of the software (‘the black box’) to run usability tests, for example, and react to bugs as regular users of the product will do. Behavioral testing also may include automation (regression tests) to eliminate human error if repetitive activities are required. For example, you may need to fill 100 registration forms on the website to see how the product copes with such an activity, so the automation of this test is preferable.



**8.2 TEST CASES:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **INPUT** | **If available** | **If not available** |
| 1 | User signup | User get registered into the application | There is no process |
| 2 | User sign in | User get login into the application | There is no process |
| 3 | Enter input for prediction | Prediction result displayed | There is no process |

**SCREENS**

1. **SCREENSHOTS**

SCREENS:

**CONCLUSION**

**10. CONCLUSION**

In this work, we analyze and evaluate IoT-CIDDS and compare it with existing data sets in IoT domain. We perform analysis of the data set in three phases: statistical analysis using kurtosis, skewness and correlation, feature engineering to extract the most relevant features and finally, an experimental analysis with supervised ML classifiers. The results are compared against performance metrics of accuracy, false positive rate, and precision, recall, F-score, AUC, and computation time. The optimal parameters for each classifier was identified using cross-validation and grid search technique. The outcome of the evaluation demonstrated that RF outperformed amongst all classifiers with highest detection rate and minimum false positive and computation time. As future work, we aim to evaluate the IoT-CIDDS data set with more experiments rooted on supervised, unsupervised, and semi-supervised models and the impact of feature engineering and cross validation on their prediction performance. We also aim to extend the algorithms for multiclass classification of attacks.

**BIBILOGRAPHY**

**11. REFERENCES**

[1] I. Markit. “The Internet of Things: A movement, not a market,” 2018. [Online]. Available: https://cdn.ihs.com/www/pdf/IoTebook.pdf

[2] M. R. Palattella et al., “Standardized protocol stack for the Internet of (important) Things,” IEEE Commun. Surveys Tuts., vol. 15, no. 3, pp. 1389–1406, 3rd Quart., 2013.

[3] Kamaldeep, M. Malik, M. Dutta, and J. Granjal, “IoT-sentry: A cross-layer-based intrusion detection system in standardized Internet of Things,” IEEE Sensors J., vol. 21, no. 24, pp. 28066–28076, Dec. 2021.

[4] A. Sivanathan et al., “Characterizing and classifying IoT traffic in smart cities and campuses,” in Proc. IEEE Conf. Comput. Commun. Workshops, 2017, pp. 559–564.

[5] A. Verma and V. Ranga. “RPL-NIDDS17: A data set for intrusion detection in RPL based 6LoWPAN networks (Internet of Things).” [Online]. Available: <https://zenodo.org/record/1406034#.XLfrI-gzbIU>

[6] V. H. Bezerra, V. G. T. da Costa, R. A. Martins, S. Barbon, R. S. Miani, and B. B. Zarpelao, “Providing IoT host-based datasets for intrusion detection research,” in Proc. 18th Braz. Symp. Inf. Security Comput. Syst., 2018, pp. 15–28.

[7] F. Y. Yavuz, D. Ünal, and E. Gül, “Deep learning for detection of routing attacks in the Internet of Things,” Int. J. Comput. Intell. Syst., vol. 12, no. 1, pp. 39–58, 2018.

[8] A. Agiollo, M. Conti, P. Kaliyar, T.-N. Lin, and L. Pajola, “DETONAR: Detection of routing attacks in RPL-based IoT,” IEEE Trans. Netw. Service Manag., vol. 18, no. 2, pp. 1178–1190, Jun. 2021.

[9] M. R. Shahid, “Deep learning for Internet of Things (IoT) network security,” Ph.D. dissertation, Télécom SudParis, Institut Polytechnique de Paris, Palaiseau, France, 2021.

[10] “NetSimv12.1.” Tetcos. 2019. [Online].Available: https://www.tetcos. com/netsim-pro.html

[11] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, “Crosslevel sensor network simulation with COOJA,” in Proc. 31st IEEE Conf. Local Comput. Netw., 2006, pp. 641–648.

[12] A. Verma and V. Ranga, “Evaluation of network intrusion detection systems for RPL based 6LoWPAN networks in IoT,” Wireless Pers. Commun., vol. 108, no. 3, pp. 1571–1594, 2019.

[13] Z. Shelby, K. Hartke, and C. Bormann, “The constrained application protocol (CoAP),” IETF, RFC 7252, Jun. 2014. [Online]. Available: http:// www.rfc-editor.org/rfc/rfc7252.txt

[14] J. Postel, “User datagram protocol,” IETF, RFC 768, Aug. 1980. [Online]. Available: <http://www.rfc-editor.org/rfc/rfc768.txt>

[15] S. Deering and R. Hinden, “Internet protocol, version 6 (IPv6) specification,” IETF, RFC 2460, Dec. 1998. [Online]. Available: https:// datatracker.ietf.org/doc/html/rfc2460

[16] T. Winter et al., “RPL: IPv6 routing protocol for low-power and lossy networks,” IETF, RFC 6550, Mar. 2012. [Online]. Available: http://www. rfc-editor.org/rfc/rfc6550.txt

[17] A. Conta, S. Deering, and M. Gupta, “Internet control message protocol (ICMPv6) for the Internet protocol version 6 (IPv6) specification,” IETF, RFC 4443, Mar. 2006. [Online]. Available: https://datatracker.ietf.org/ doc/html/rfc4443

[18] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, “Transmission of IPv6 packets over IEEE 802.15.4 networks,” IETF, RFC 4944, Sep. 2007. [Online]. Available: <http://www.rfc-editor.org/rfc/rfc4944.txt>

[19] IEEE Standard for Low-Rate Wireless Networks Corrigendum 1, IEEE 802.15.4-2015/Cor1-2018, 2015. [Online]. Available: https://standards. ieee.org/standard/802.15.4-2015-Cor1-2018.html

[20] R. Doshi, N. Apthorpe, and N. Feamster, “Machine learning DDoS detection for consumer Internet of Things devices,” in Proc. IEEE Security Privacy Workshops (SPW), San Francisco, CA, USA, 2018, pp. 29–35.

[21] K. Bhardwaj, J. C. Miranda, and A. Gavrilovska, “Towards IoTDDoS prevention using edge computing,” in Proc. USENIX Workshop Hot Topics Edge Comput. (HotEdge), Boston, MA, USA, 2018, pp. 1–7.

[22] N. Ravi and S. M. Shalinie, “Learning-driven detection and mitigation of DDoS attack in IoT via SDN-cloud architecture,” IEEE Internet Things J., vol. 7, no. 4, pp. 3559–3570, Apr. 2020.

[23] Y.-W. Chen, J.-P. Sheu, Y.-C. Kuo, and N. Van Cuong, “Design and implementation of IoT DDoS attacks detection system based on machine learning,” in Proc. Eur. Conf. Netw. Commun. (EuCNC), 2020, pp. 122–127.

[24] D. Yin, L. Zhang, and K. Yang, “A DDoS attack detection and mitigation with software-defined Internet of Things framework,” IEEE Access, vol. 6, pp. 24694–24705, 2018.

[25] I. Cvitic, D. Perakovi ´ c, M. Periša, and M. Botica, “Novel approach for ´ detection of IoT generated DDoS traffic,” Wireless Netw., vol. 27, no. 3, pp. 1573–1586, 2021.

[26] I. Cvitic, D. Perakovic, B. B. Gupta, and K.-K. R. Choo, “Boosting- ´ based DDoS detection in Internet of Things systems,” IEEE Internet Things J., vol. 9, no. 3, pp. 2109–2123, Feb. 2022.

[27] M. N. Napiah, M. Y. I. B. Idris, R. Ramli, and I. Ahmedy, “Compression header analyzer intrusion detection system (CHA-IDS) for 6LoWPAN communication protocol,” IEEE Access, vol. 6, pp. 16623–16638, 2018.

[28] I. Hafeez, M. Antikainen, A. Y. Ding, and S. Tarkoma, “IoT-KEEPER: Detecting malicious IoT network activity using online traffic analysis at the edge,” IEEE Trans. Netw. Service Manag., vol. 17, no. 1, pp. 45–59, Mar. 2020.

[29] I. Hafeez, M. Antikainen, A. Y. Ding, and S. Tarkoma. “IoT-Dataset.” 2019. [Online]. Available: <https://bit.ly/31uuNK7>

[30] E. Canbalaban and S. Sen, “A cross-layer intrusion detection system for RPL-based Internet of Things,” in Proc. Int. Conf. Ad-Hoc Netw. Wireless, 2020, pp. 214–227.

[31] A. Abbas, M. A. Khan, S. Latif, M. Ajaz, A. A. Shah, and J. Ahmad, “A new ensemble-based intrusion detection system for Internet of Things,” Arab. J. Sci. Eng., vol. 47, no. 16, pp. 1805–1819, 2022.

[32] “Intrusion detection evaluation dataset (CICIDS2017).” Canadian Institute for Cybersecurity. Accessed: Sep. 29, 2019. [Online]. Available: <https://www.unb.ca/cic/datasets/ids-2017.html>

[33] P. Verma et al., “A novel intrusion detection approach using machine learning ensemble for IoT environments,” Appl. Sci., vol. 11, no. 21, pp. 1–21, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/ 11/21/10268

[34] “Amazon’s AWS LAN network (CIC-AWS-2018).” Canadian Institute for Cybersecurity. Accessed: Jan. 30, 2022. [Online]. Available: https:// registry.opendata.aws/CSE-CIC-IDS2018-V2/

[35] A. Verma and V. Ranga, “ELNIDS: Ensemble learning based network intrusion detection system for RPL based Internet of Things,” in Proc. 4th Int. Conf. Internet Things Smart Innov. Usages, 2019, pp. 1–6.

[36] A. Dunkels, B. Gronvall, and T. Voigt, “Contiki—A lightweight and flexible operating system for tiny networked sensors,” in Proc. 29th Annu. IEEE Int. Conf. Local Comput. Netw., Nov. 2004, pp. 455–462.

[37] J. Crnogorac, J. Crnogorac, M. Vucini ˇ c, E. Ko ´ can, and T. Watteyne, ˇ “Dense multi-channel sniffing in large IoT networks,” IEEE Access, vol. 10, pp. 105101–105110, 2022.

[38] B. Mostafa, M. Molnar, M. Saleh, A. Benslimane, and S. Kassem, “Optimal proactive monitor placement & scheduling for IoT networks,” J. Oper. Res. Soc., vol. 73, no. 11, pp. 2431–2450, 2021.

[39] M. Malik, M. Dutta, and J. Granjal, “A survey of key bootstrapping protocols based on public key cryptography in the Internet of Things,” IEEE Access, vol. 7, pp. 27443–27464, 2019.

[40] K. V. Mardia, “Measures of multivariate skewness and kurtosis with applications,” Biometrika, vol. 57, no. 3, pp. 519–530, 1970.

[41] J. M. Bland and D. G. Altman, “Calculating correlation coefficients with repeated observations: Part 2—Correlation between subjects,” BMJ, vol. 310, no. 6980, p. 633, 1995.

[42] N. Moustafa, J. Slay, and G. Creech, “Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks,” IEEE Trans. Big Data, vol. 5, no. 4, pp. 481–494, Dec. 2019.

[43] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” J. Mach. Learn. Res., vol. 12, no. 85, pp. 2825–2830, 2011.

[44] “Tuning the hyper-parameters of an estimator.” 2020. [Online]. Available: <https://scikit-learn.org/stable/modules/gridsearch.html>

[45] Y. Jia, F. Zhong, A. Alrawais, B. Gong, and X. Cheng, “FlowGuard: An intelligent edge defense mechanism against IoT DDoS attacks,” IEEE Internet Things J., vol. 7, no. 10, pp. 9552–9562, Oct. 2020.

[46] M. H. Aysa, A. A. Ibrahim, and A. H. Mohammed, “IoT DDoS attack detection using machine learning,” in Proc. 4th Int. Symp. Multidiscip. Stud. Innov. Technol. (ISMSIT), Istanbul, Turkey, 2020, pp. 1–7.