

MACHINE LEARNING TECHNIQUES FOR CYBER ATTACK DETECTION

A Project Report Submitted

In

Partial Fulfillment of the Requirements

For the Award of the Degree of

Bachelor of Technology

In

COMPUTER SCIENCE AND ENGINEERING

Submitted By

L.Pavan Kumar (19541A0571)

P.Sai Dhanaraj (19541A0581)

A.Rukmini(19541A0501)

K.Tejaswi (19541A0515)

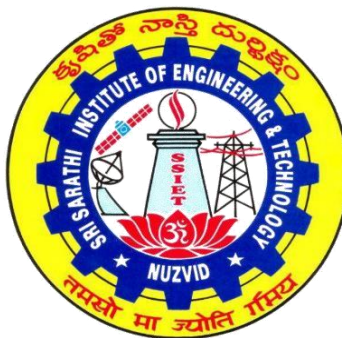
N.Sravani (19541A0533)

K.Durga Prasad (16541A0569)

Under the Guidance of

Mr. P. Sateesh Babu , M.Tech

Assistant Professor of CSE.



2019-2023

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SRI SARATHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Approved by AICTE and Affiliated to JNTUK, Kakinada)

Nuzvid – 521 201, KRISHNA, A.P., INDIA

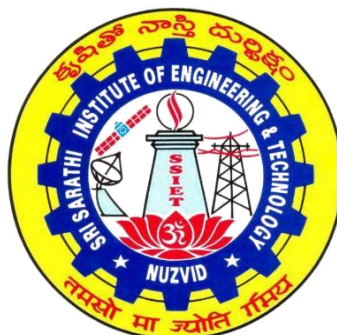
2019-2023

SRI SARATHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Approved by AICTE Affiliated to JNTUK, ISO 9001:2008 Certified)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

2019-2023



CERTIFICATE

This is to certify that the Project Report entitled “Machine Learning Techniques For Cyber Attack Detection” That is being Submitted by L.Pavan Kumar (19541A0571), P.Sai Dhanaraj (19541A0581), A.Rukmini (19541A0501), K.Tejaswi (19541A0515), N.Sravani (19541A0533), K.Durga Prasad (16541A0569) in partial fulfilment of requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering to the Jawaharlal Nehru Technological University, Kakinada is a record of bona fide work carried out by him/her under our guidance and supervision.

The results embodied in this project report have not been submitted to any other university for the award on any degree or diploma by any student of this college.

Signature Of The Internal Guide

Mr. P. Sateesh Babu , M.Tech

Assistant Professor of CSE

Signature Of The HOD

Dr. K. Nageswara Rao , Ph.D

Head Of The CSE Dept.

Signature Of The External Examiner

DECLARATION

We hereby declare that this Project Report entitled “**Machine Learning Techniques For Cyber Attack Detection**” to **Jawaharlal Nehru Technological University, Kakinada**, is a genuine work carried out by us, for the partial fulfilment of the degree of Bachelor of Technology in Computer Science and Engineering during the academic year 2019-2023 under the supervision of our guide Mr. P.Sateesh Babu, Assistant professor of CSE, in **SRI SARATHI INSTITUTE OF ENGINEERING AND TECHNOLOGY, NUZVID**.

L.Pavan Kumar (19541A0571).

P.Sai Dhanaraj (19541A0581).

A.Rukmini (19541A0501).

K.Tejaswi (19541A0515).

N.Sravani (19541A0533).

K. Durga Prasad (16541A0569).

ACKNOWLEDGEMENT

We thank our honourable chairman **Sri. S.V. SUDHEER** garu, for his moral support and the excellent facilities provided. We are grateful to **Sri. S. SRIDHAR** garu, Principal, **Sri Sarathi Institute of Engineering & Technology, Nuzvid**, for his support throughout the project for given the permission to undergo the project work.

We would also like to thank **Sri Dr. K. NAGESWARA RAO** garu, **Head of the Department, Department of computer science engineering and technology** for having given us the opportunity to complete the project successfully.

We sincerely thanks to **Mr. P. SATEESH BABU** garu, **Assistant Professor of Computer Science and Engineering, Sri Sarathi Institute of Engineering and Technology, Nuzvid** for his valuable guidness and support.

I thank for timely suggestions and constant encouragement that boosted up our moral and let to the accomplishment of this project.

Firstly, we indebted to my parents, friends and relatives for the physical, psychological and moral extended by them at all time.

L.Pavan Kumar (19541A0571).

P.Sai Dhanaraj (19541A0581).

A.Rukmini (19541A0501).

K.Tejaswi (19541A0515).

N.Sravani (19541A0533).

K. Durga Prasad (16541A0569).

ABSTRACT

Cyber-crime is proliferating everywhere exploiting every kind of vulnerability to the computing environment. Ethical Hackers pay more attention towards assessing vulnerabilities and recommending mitigation methodologies.

The development of effective techniques has been an urgent demand in the field of the cyber security community. Most techniques used in today's IDS are not able to deal with the dynamic and complex nature of cyber-attacks on computer networks.

Machine learning for cyber security has become an issue of great importance recently due to the effectiveness of machine learning in cyber security issues. Machine learning techniques have been applied for major challenges in cyber security issues like intrusion detection, malware classification and detection, spam detection and phishing detection.

Although machine learning cannot automate a complete cyber security system, it helps to identify cyber security threats more efficiently than other software-oriented methodologies, and thus reduces the burden on security analysts.

Hence, efficient adaptive methods like various techniques of machine learning can result in higher detection rates, lower false alarm rates and reasonable computation and communication costs.

Our main goal is that the task of finding attacks is fundamentally different from these other applications, making it significantly harder for the intrusion detection community to employ machine learning effectively.

INDEX

1. Introduction	1
1.1 Intro	2
1.2 Overview Of Methods For Cyber Attack Detection	2
1.2.1 Signature-Based Methods	2
1.2.2 Anomaly-Based Methods	3
1.3 Machine learning	3
1.3.1 Machine Learning Methods	4
2. Literature Survey	5
2.1 Requirements	6
2.1.1 Input Design	6
2.1.2 Output Design	6
2.2 Existing System	6
2.2.1 Existing Works	7
2.2.2 Drawbacks	7
2.3 Proposed System	7
2.3.1 Advantages	7
3. System Requirements Analysis	8
3.1 Requirement Specification	9
3.2 Hardware Requirements	9
3.3 Software Requirements	9
3.4 Feasibility Study	9
3.4.1 Economic Feasibility	10
3.4.2 Technical Feasibility	10
3.4.3 Social Feasibility	10
3.5 Algorithms	10
4. System Design	12
4.1 Objectives	13
4.2 Modules	13
4.3 System Architecture	15
4.3.1 Network Traffic	15
4.3.2 Packet Capture	16
4.3.3 Classification	16

4.3.4 Response Module	17
4.4 UML Diagrams	17
4.4.1 Use Case Diagram	18
4.4.2 Class Diagram	19
4.4.3 Sequence Diagram	19
4.4.4 Activity Diagram	20
5.System Implementation	21
5.1 Implementation Details	22
5.2 Monitoring Levels	22
5.3 Sample Code	23
5.4 Technology Used	25
6.Screenshots	40
7.Testing	46
7.1 Testcases	47
8.Summary	50
8.1 summary Of Previous Work	51
9.Conclusion	53
10.Reference	55

LIST OF FIGURES

Figure No.	Figure Name	Page No.
1.3	Machine Learning	4
1.3.1	Machine Learning Methods	4
4.3	System Architecture	15
4.4.1	Use Case Diagram	18
4.4.2	Class Diagram	19
4.4.3	Sequence Diagram	19
4.4.4	Activity Diagram	20

LIST OF TABLES

Table No.	Table Name	Page No.
8.1	Summary Of Previous Work	51,52

INTRODUCTION

1.INTRODUCTION

1.1 INTRO

Today, political and commercial entities are increasingly engaging in sophisticated cyberwarfare to damage, disrupt, or censor information content in computer networks. In designing network protocols, there is a need to ensure reliability against intrusions of powerful attackers that can even control a fraction of parties in the network. The controlled parties can launch both passive (e.g., eavesdropping, nonparticipation) and active attacks (e.g., jamming, message dropping, corruption, and forging). Intrusion detection is the process of dynamically monitoring events occurring in a computer system or network, analysing them for signs of possible incidents and often interdicting the unauthorized access. This is typically accomplished by automatically collecting information from a variety of systems and network sources, and then analysing the information for possible security problems. Traditional intrusion detection and prevention techniques, like firewalls, access control mechanisms, and encryptions, have several limitations in fully protecting networks and systems from increasingly sophisticated attacks like denial of service. Moreover, most systems built based on such techniques suffer from high false positive and false negative detection rates and the lack of continuously adapting to changing malicious behaviours.

In the past decade, however, several Machine Learning (ML) techniques have been applied to the problem of intrusion detection with the hope of improving detection rates and adaptability. These techniques are often used to keep the attack knowledge bases up-to-date and comprehensive. In recent days, cyber-security and protection against numerous cyber-attacks are becoming a burning question. The main reason behind that is the tremendous growth of computer networks and the vast number of relevant applications used by individuals or groups for either personal or commercial use, especially after the acceptance of the Internet of Things (IoT).

1.2 OVERVIEW OF METHODS FOR CYBER ATTACK DETECTION

1.2.1 SIGNATURE-BASED METHODS

The Signature-based category of cyber attacks detection methods typically include Intrusion Prevention and Detection Systems (IDS and IPS) which use predefined set of patterns (or rules) in order to identify an attack. The patterns (or rules) are typically matched against a content of a packet (e.g. TCP/UDP packet header or payload). Commonly IPS and IDS are designed to increase the

security level of a computer network through detection (in case of IDS) and detection and blocking (in case of IPS) of network attacks.

One of the most popular IDS/IPS software, widely deployed worldwide, is Snort [7]. Since it is an open source project, its users are allowed to freely modify it as well as feed the Snort engine with rules obtained from different sources (e.g. not only from Snort homepage).

1.2.2 ANOMALY-BASED METHODS

The anomaly-based methods for a cyber attacks detection typically build a model that is intended to describe normal and abnormal behaviour of network traffic. Commonly such methods use two types of algorithms borrowed from machine learning theory, namely unsupervised and supervised approach.

For unsupervised learning commonly [8, 9, 10, 11, 12, 13, 14] clustering approaches are used that usually adapt algorithms like k-means, fuzzy c-means, QT, and SVM. The clustered network traffic established using mentioned approaches commonly requires decision whenever given cluster should be indicated as a malicious or not. Pure unsupervised algorithms use a majority rule telling that only the biggest clusters are considered normal. That means that network events that happen frequently have no symptoms of an attack. In practice, it is a human role to tell which cluster should be considered as an abnormal one.

1.3 MACHINE LEARNING

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.

Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves. The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers to learn automatically without human intervention or assistance and adjust actions accordingly.

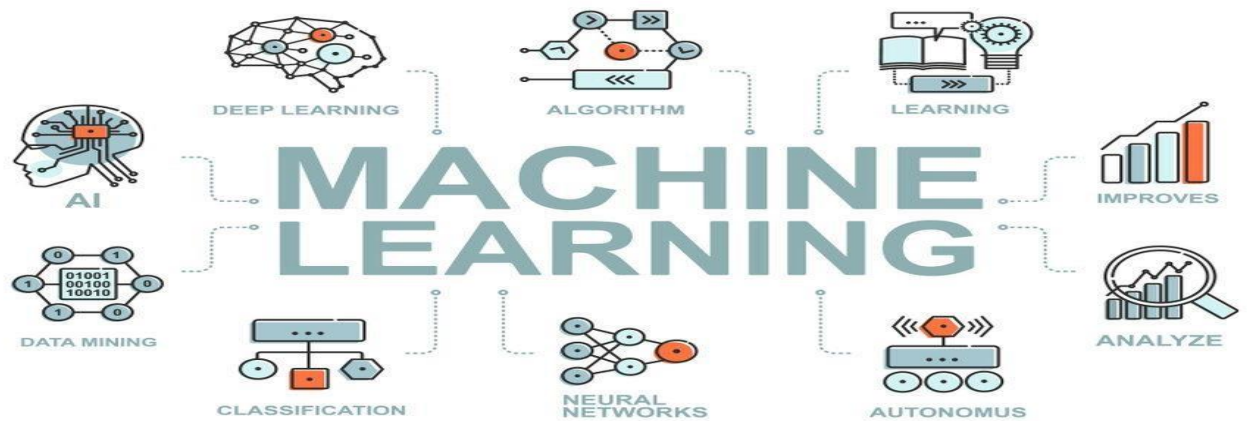


Fig 1.3 : Machine Learning

1.3.1 MACHINE LEARNING METHODS

Machine learning algorithms are often categorized as supervised or unsupervised. **Supervised Machine Learning Algorithms** can apply what has been learned in the past to new data using labelled examples to predict future events. In contrast, **Unsupervised Machine Learning Algorithms** are used when the information used to train is neither classified nor labelled. **Semi-supervised machine learning algorithms** fall somewhere in between supervised and unsupervised learning, since they use both labelled and unlabelled data for training – typically a small amount of labelled data and a large amount of unlabelled data. **Reinforcement machine learning algorithms** is a learning method that interacts with its environment by producing actions and discovers errors or rewards.

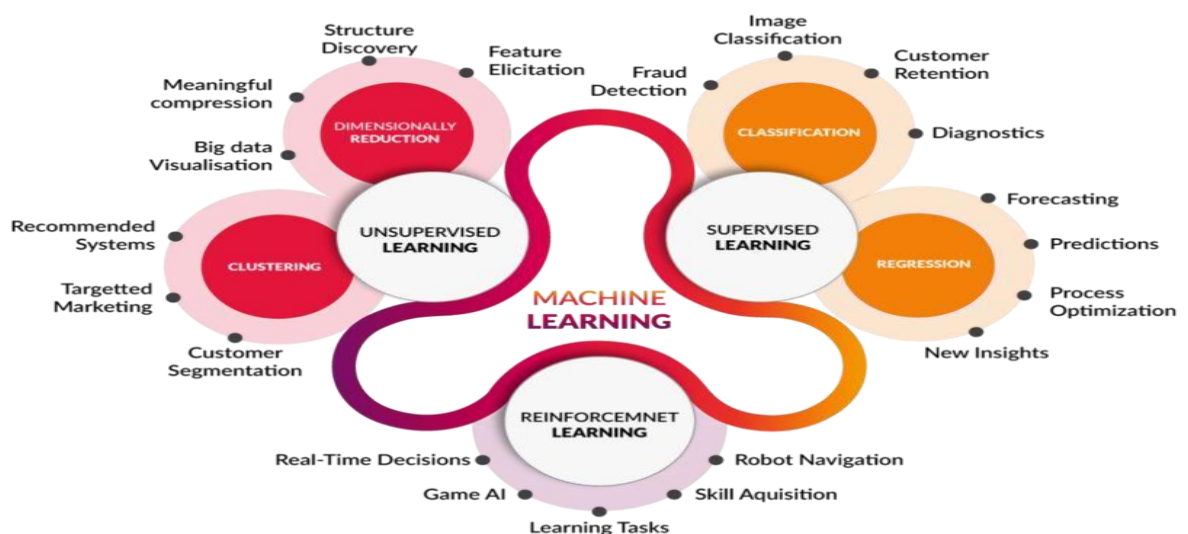


Fig 1.3.1 : Machine Learning Methods

LITERATURE SURVEY

2. LITERATURE SURVEY

2.1 REQUIREMENTS

2.1.1 INPUT DESIGN:

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy.

2.1.2 OUTPUT DESIGN:

- Designing computer output should proceed in an organized, well thought out manner.
- Select methods for presenting information.
- Create document, report, or other formats that contain information produced by the system.
- Convey information about past activities, current status or projections of the Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

2.2 EXISTING SYSTEM

There are two distinct categories of cyber-attack detection methods, namely signature based and anomaly based. The machine learning techniques are used in both of them. Recently machine learning-based algorithms have been used for developing signatures that will efficiently identify both the code and behaviour of the malicious code. The Network-based Signature Generation (NSG), Length-based Signature Generation (LSEG) and F-Sign are the examples of algorithms designed for automated and fast extraction of signatures of polymorphic worms. The LSEG algorithm targets those worms that use buffer overflow attack to infect victims, whereas the F-Sign extracts the signature on a basis of the code of a worm (such signature can be used to detect and stop the worm from spreading).

2.2.1 EXISTING WORKS

Within the ever-growing and quickly increasing field of cyber security, it is nearly impossible to quantify or justify the explanations why cyber security has such an outsized impact. Permitting malicious threats to run any place, at any time or in any context is a long way from being acceptable, and may cause forceful injury. It particularly applies to the Byzantine web of consumers and using the net and company information that cyber security groups are finding it hard to shield and contain. Cyber security may be a necessary thought for people and families alike, also for businesses, governments, and academic establishments that operate inside compass of the world network or net.

2.2.2 DRAWBACKS

- Errors, Needs Manual Calculations, Inconsistency And More Man Power.
- Slow, Tedious, Lack Of Security, Information Redundancy and Time Consuming.

2.3 PROPOSED SYSTEM

Machine Learning algorithms can be used to train and detect if there has been a cyber attack. As soon as the attack is detected, an email notification can be sent to the security engineers or users. Any classification algorithm can be used to categorize if it is a DoS/DDoS attack or not. One example of a classification algorithm is Support Vector Machine (SVM) which is a supervised learning method that analyses data and recognizes patterns. Since we cannot control when, where or how an attack may come our way, and absolute prevention against these cannot be guaranteed yet, our best shot for now is early detection which will help mitigate the risk of irreparable damage such incidents can cause. Organizations can use existing solutions or build their own to detect cyber attacks at a very early stage to minimize the impact. Any system that requires minimal human intervention would be ideal.

2.3.1 ADVANTAGES

The system is very simple in design and to implement. The system requires very low system resources and the system will work in almost all configurations. It has got following features;

- Minimize manual data entry and Better Service.
- Minimum time needed for the various processing.
- Greater efficiency With User Friendliness And Interactive.

SYSTEM REQUIREMENTS ANALYSIS

3.SYSTEM REQUIREMENTS ANALYSIS

3.1 REQUIREMENT SPECIFICATION

Functional Requirements:

- Graphical User interface with the User.

Operating Systems supported:

- Windows 7, Windows XP, Windows 8, Windows 10, Windows 11

Technologies and Languages used to Develop:

- Python

Debugger and Emulator:

- Any Browser (Particularly Chrome)

3.2 HARDWARE REQUIREMENTS

❖ **System** : Intel i3 2.2Ghz

❖ **Hard Disk** : 320 GB.

❖ **Ram** : 4 GB.

3.3 SOFTWARE REQUIREMENTS

❖ **Operating system** : Windows 7/8/10/11.

❖ **Coding Language** : Python.

❖ **Front-End** : HTML, CSS, JS.

❖ **IDE** : Visual Studio Code.

3.4 FEASIBILITY STUDY

The feasibility of the project is analysed 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

3.4.1 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.

3.4.2 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.

3.4.3 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.

3.5 ALGORITHMS

LOGISTIC REGRESSION:

This type of statistical model (also known as *logit model*) is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure. This is also commonly known as the log odds, or the natural logarithm of odds, and this logistic function is represented by the following formulas: $\text{Logit}(\pi) = 1/(1 + \exp(-\pi))$

$$\ln(\pi/(1-\pi)) = \text{Beta}_0 + \text{Beta}_1 * X_1 + \dots + \text{Beta}_k * X_k$$

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. The random forest algorithm is an extension of the bagging method as it utilizes both bagging and feature randomness to create an uncorrelated forest of decision trees.

SUPPORT VECTOR MACHINE:

“Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges.

$$f(x) = B_0 + \sum(a_i * (x, x_i))$$

This is an equation that involves calculating the inner products of a new input vector (x) with all support vectors in training data. The coefficients B_0 and a_i (for each input) must be estimated from the training data by the learning algorithm.

Step 1 : Receive an input post

Step 2 : Convert the post to a word embedding sequence $x_1: T$

Step 3 : Extract the input semantic key $K(i)$

Step 4 : Map it to the output semantic key $k_0 = S(k(i))$

Step 5 : Find the associated memory block matrix m

Step 6 : Encode the post using the encoder $P, c_1:T = E(x_1:T)$

Step 7 : Read the memory context vector from the external semantic memory $R = R(m, p)$

Step 8 : Append r to the original encoder context vectors $C = \{[c_1;r], [c_2;r], \dots, [c_t;r]\}$

Step 9 : Decode to generate the comment $Y = D(p, C)$

SYSTEM DESIGN

4.1 OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system to handle large volume of data. The goal of designing input is to make data.

2. It is achieved by creating user-friendly screens for the data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

4.2 MODULES

1. DATA PREPROCESSING

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.

2. MODEL BUILDING

Machine learning models are powerful tools used to efficiently and effectively perform vital tasks and solve complex problems. An exponential increase in data across the modern world means organisations from a range of sectors are ready to deploy machine learning models. These models have a huge range of uses, whether Machine learning in finance proactively monitoring bank transfers for signs of fraud, or machine learning in healthcare powering the next generation of diagnostic tools.

3.LOGISTIC REGRESSION:

This type of statistical model (also known as *logit model*) is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure. This is also commonly known as the log odds, or the natural logarithm of odds, and this logistic function is represented by the following formulas: $\text{Logit}(\pi) = 1/(1 + \exp(-\pi))$

$$\ln(\pi/(1-\pi)) = \text{Beta}_0 + \text{Beta}_1 * X_1 + \dots + B_k * K_k$$

4.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. The random forest algorithm is an extension of the bagging method as it utilizes both bagging and feature randomness to create an uncorrelated forest of decision trees.

5.SUPPORT VECTOR MACHINE:

“Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges.

$$f(x) = B_0 + \sum(a_i * (x, x_i))$$

This is an equation that involves calculating the inner products of a new input vector (x) with all support vectors in training data.

4.3 SYSTEM ARCHITECTURE

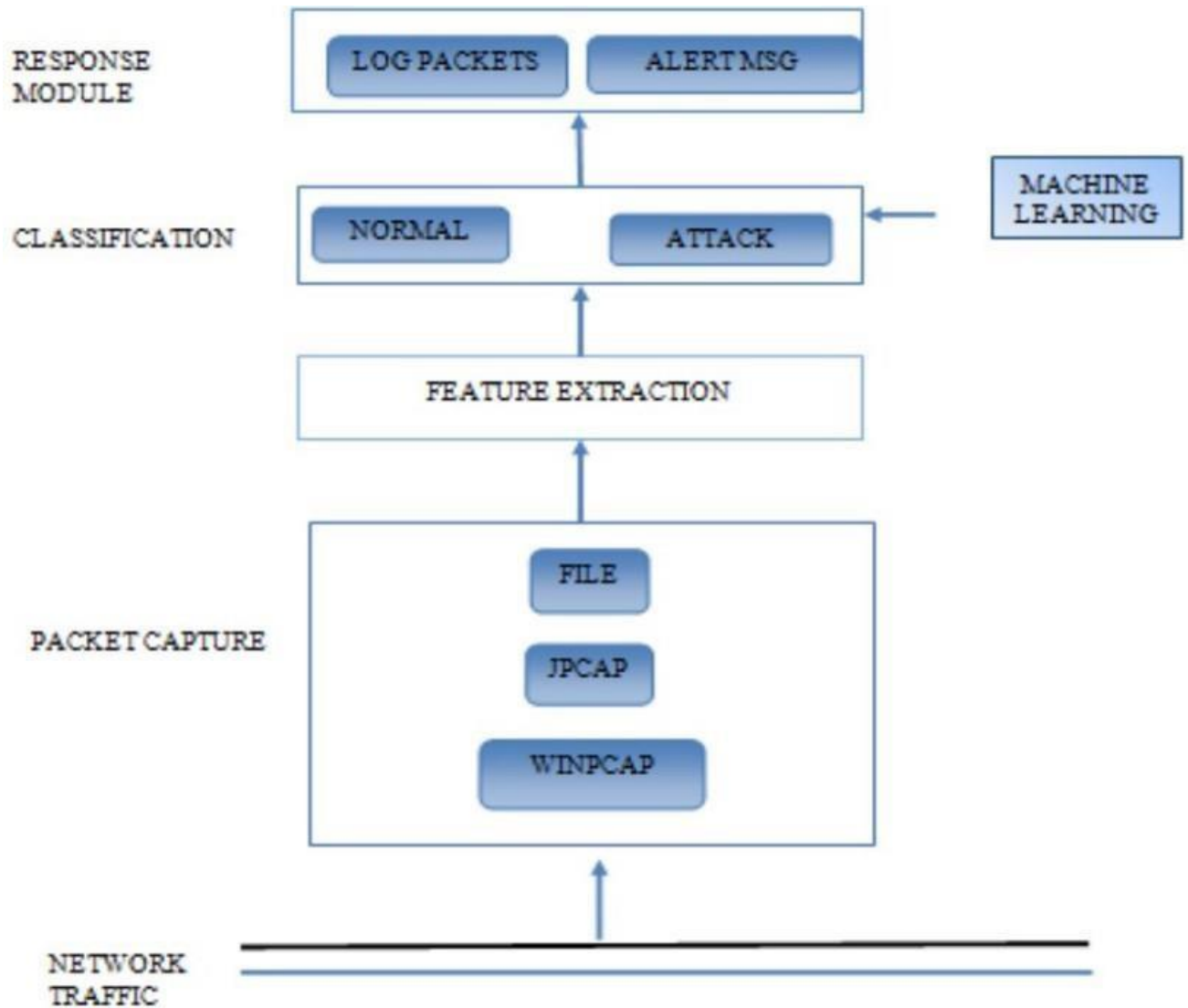


Fig 4.3 : System Architecture

4.3.1 NETWORK TRAFFIC

Network traffic refers to the amount of data moving across a network at a given point of time. Network data is mostly encapsulated in network packets, which provide the load in the network. Network traffic is the main component for network traffic measurement, network traffic control and simulation. The proper organization of network traffic helps in ensuring the quality of service in a given network.

Proper analysis of network traffic provides the organization with the following benefits:

Identifying network bottlenecks - There could be users or applications that consume high amounts of bandwidth, thus constituting a major part of the network traffic. Different solutions can be implemented to tackle these. Network security - Unusual amount of traffic in a network is a possible sign of an attack. Network traffic reports provide valuable insights into preventing such attacks. Network engineering - Knowing the usage levels of the network allows future requirements to be analysed.

4.3.2 PACKET CAPTURE

Packet Capture is a networking term for intercepting a data packet that is crossing a specific point in a data network. Once a packet is captured in real-time, it is stored for a period of time so that it can be analysed, and then either be downloaded, archived or discarded. Packets are captured and examined to help diagnose and solve network problems such as:

Identifying security threats:

Troubleshooting undesirable network behaviours.

Identifying network congestion.

Identifying data/packet loss.

Forensic network analysis.

4.3.3 CLASSIFICATION

Classification is another extensively used supervisory machine learning task. In cyber security, spam detection is successfully implemented by ML based classifiers which involves discriminating a given email message as spam or not. The spam filter models are able to separate spam messages from non-spam messages.

Machine learning techniques for classification include Logistic Regression, KNearest Neighbours, Support Vector Machine, Naïve Bayes, Decision Tree, Random Forest Classification. Upon the availability of large collection of past data with labels.

Deep Learning classification models involving Restricted Boltzmann Machines(RBM), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), or Long-Short Term Memory (LSTMs) cells for feature extraction followed by a densely connected neural network have become more efficient in solving complex tasks. Applicability of the above supervisory machine learning techniques is conditioned based on the availability of large collections of labeled data.

4.3.4 RESPONSE MODULE

Incident response is a term used to describe the process by which an organization handles a data breach or cyber attack, including the way the organization attempts to manage the consequences of the attack or breach (the “incident”). Ultimately, the goal is to effectively manage the incident so that the damage is limited and both recovery time and costs, as well as collateral damage such as brand reputation, are kept at a minimum and also sends an alert message to security analysts.

4.4 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 nonsoftware systems.

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

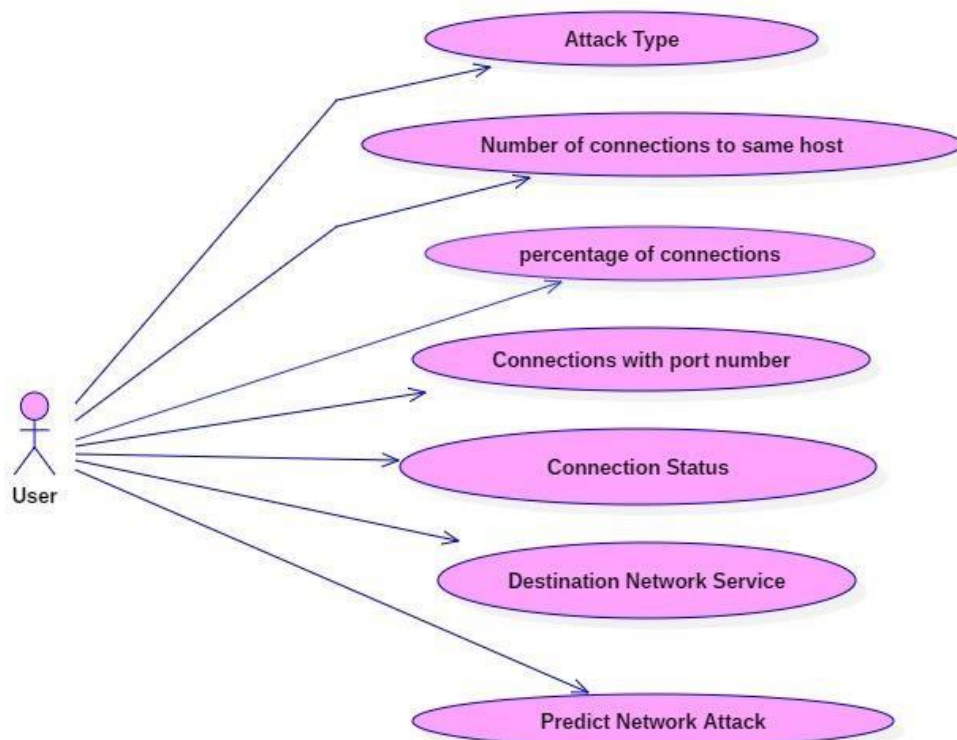
The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

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.

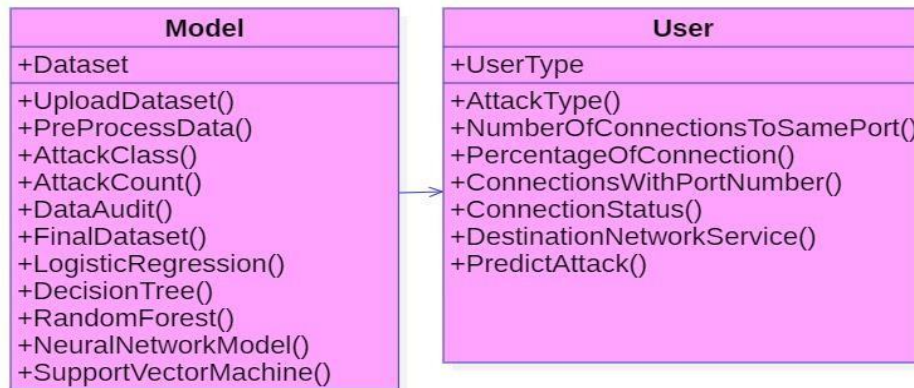
4.4.1 USECASE DIAGRAM

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.



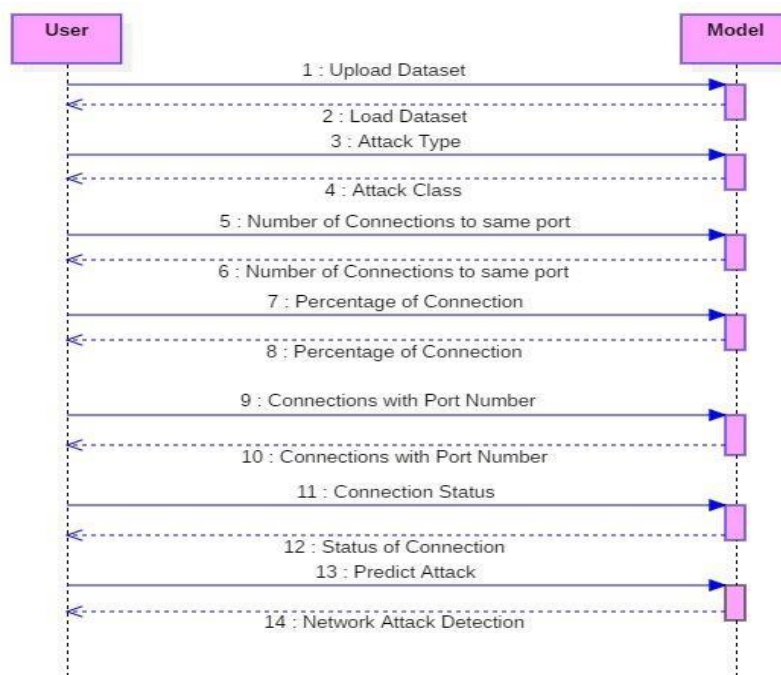
4.4.2 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



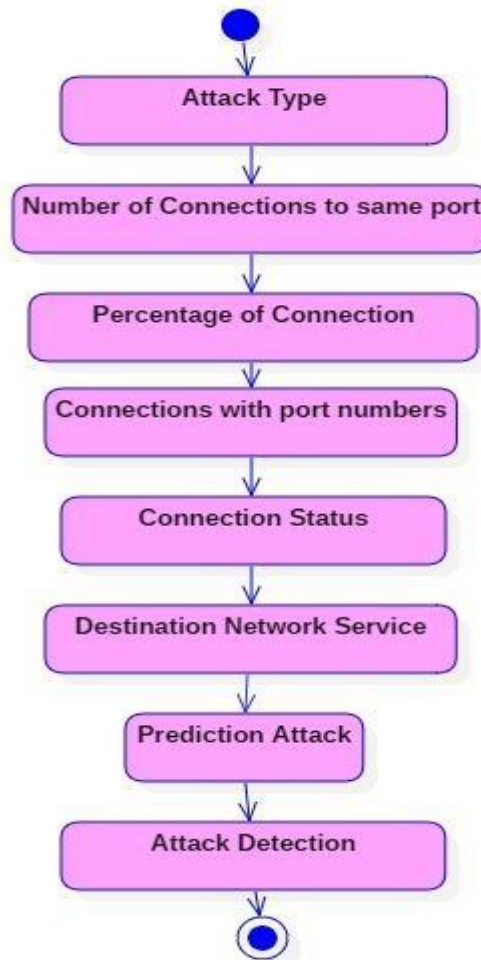
4.4.3 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



4.4.4 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



SYSTEM IMPLEMENTATION

5. SYSTEM IMPLEMENTATION

5.1 IMPLEMENTATION DETAILS

The system is implemented by using ANACONDA software , Anaconda is the world's most popular data science platform and the foundation of modern machine learning. We pioneered the use of Python for data science, champion its vibrant community, and continue to steward open-source projects that make tomorrow's innovations possible. Our enterprise-grade solutions enable corporate, research, and academic institutions around the world to harness the power of opensource for competitive advantage, ground breaking research, and a better world. Providing powerful open source tools in a centralized, collaborative, and version controlled environment Offering a package repository Giving the ability to monitor data science activity via auditing, versioning, and logging Automating model training and deployment on scalable, container-based infrastructure and some monitoring levels have been tested.

5.2 MONITORING LEVELS

1.System Level:

Cumulative and per user CPU usage.

Usage of real and virtual memory.

Amount of swap space currently available.

Amount of free memory I/O and disk usage.

2.User Level:

Type of user and user privileges.

Duration of the connection.

Login/Logout period and location.

Access of resources and directories.

Type of software/programs use, Key stroke pattern (use in future).

Average number of packets sent and received.

3.Process Level:

The number of processes and their types Relationship among processes.

4.Packet Level:

Average number of packets sent and received.

5.3 SAMPLE CODE

```

import numpy as np
from flask import Flask, request, jsonify, render_template
import joblib

app = Flask(__name__)
model = joblib.load('model.pkl')

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict():

    int_features = [float(x) for x in request.form.values()]

    if int_features[0]==0:
        f_features=[0,0,0]+int_features[1:]
    elif int_features[0]==1:
        f_features=[1,0,0]+int_features[1:]
    elif int_features[0]==2:
        f_features=[0,1,0]+int_features[1:]
    else:
        f_features=[0,0,1]+int_features[1:]

    if f_features[6]==0:
        fn_features=f_features[:6]+[0,0]+f_features[7:]
    elif f_features[6]==1:
        fn_features=f_features[:6]+[1,0]+f_features[7:]
    else:
        fn_features=f_features[:6]+[0,1]+f_features[7:]

    final_features = [np.array(fn_features)]

```



```

predict = model.predict(final_features)

if predict==0:
    output='Normal'
elif predict==1:
    output='DOS'
elif predict==2:
    output='PROBE'
elif predict==3:
    output='R2L'
else:
    output='U2R'

return render_template('index.html', output=output)

@app.route('/results',methods=['POST'])
def results():

    data = request.get_json(force=True)
    predict = model.predict([np.array(list(data.values()))])

    if predict==0:
        output='Normal'
    elif predict==1:
        output='DOS'
    elif predict==2:
        output='PROBE'
    elif predict==3:
        output='R2L'
    else:
        output='U2R'

    return jsonify(output)

if __name__ == "__main__":
    app.run()

```

5.4 TECHNOLOGY USED

What is Python?

Python is a High level, structured, open-source programming language that can be used for a wide variety of programming tasks.

Python within itself is an interpreted programming language that is automatically compiled into bytecode before execution.

It is also a dynamically typed language that includes (but does not require one to use) object-oriented features.

NASA has used Python for its software systems and has adopted it as the standard scripting language for its Integrated Planning System.

Python is also extensively used by Google to implement many components of its Web Crawler and Search Engine & Yahoo! for managing its discussion groups.

History of Python

Python was created by Guido Van Rossum.

The design began in the late 1980s and was first released in February 1991.

Why the name Python?

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late 70s. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

Python Version History

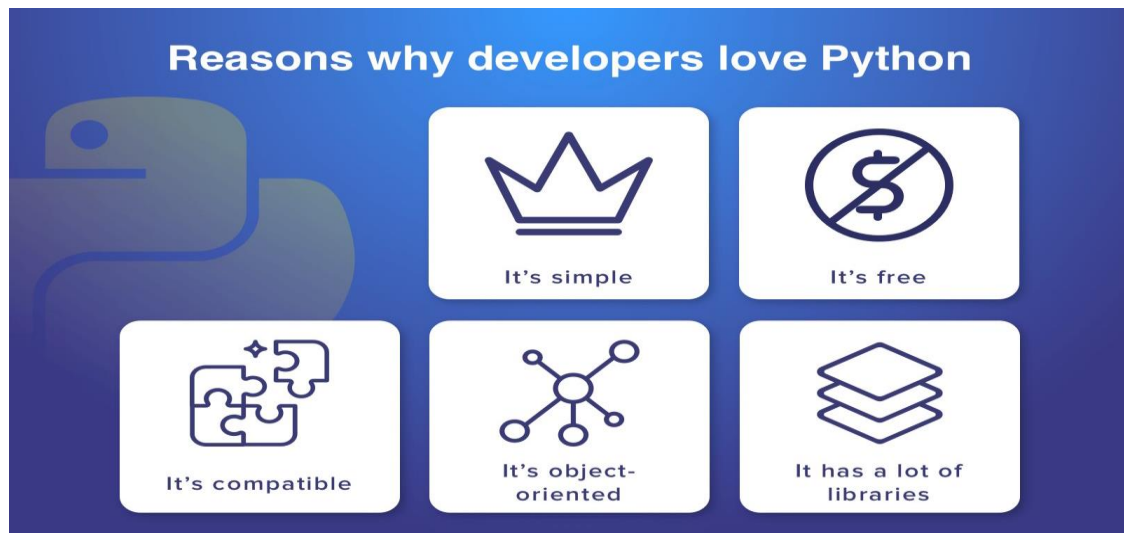
Implementation started - December 1989

Internal releases – 1990

Version No.	Date of Released
0.9	February 20, 1991

1.0	January, 1994
2.0	October 16, 2000
3.0	December 3, 2008
3.1	June 27, 2009
3.2	February 20, 2011
3.3	September 29, 2012
3.4	March 16, 2014
3.5	September 13, 2015
3.6	December 23, 2016
3.7	June 27, 2018

Features of Python Programming



Python Version History

A simple language which is easier to learn

- Python has a very simple and elegant syntax.
- It's much easier to read and write Python programs compared to other languages like: C++, Java, C#.
- Python makes programming fun and allows you to focus on the solution rather than syntax.
- If you are a newbie, it's a great choice to start your journey with Python.

Free and open-source

- You can freely use and distribute Python, even for commercial use.
- Not only you can use and distribute software's written in it, you can even make changes to the Python's source code.
- Python has a large community constantly improving it in each iteration.

Portability

- You can move Python programs from one platform to another and run it without any changes.
- It runs seamlessly on almost all platforms including Windows, Mac OS and Linux.

Extensible and Embeddable

- Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code.
- This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

A high-level, interpreted language

- Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on.
- Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.

Large standard libraries to solve common tasks

- Python has several standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself.
- For example: Need to connect MySQL database on a Web server? You can use MySQL dB library using import MySQL db.
- Standard libraries in Python are well tested and used by hundreds of people. So, you can be sure that it won't break your application.

Object-oriented

- Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.
- With OOP, you can divide these complex problems into smaller sets by creating objects.

Reasons to Choose Python as First Language**Simple Elegant Syntax**

- Programming in Python is fun. It's easier to understand and write Python code. **Why?** The syntax feels natural. Take this source code for an example:

```
a = 2 b =
3 sum = a
+ b
print(sum)
```

- Even if you have never programmed before, you can easily guess that this program adds two numbers and prints it.

Not overly strict

- You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement.
- Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

Expressiveness of the language

- Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of Tic-tac-toe game with a graphical interface and a smart computer opponent in less than 500 lines of code. This is just an example. You will be amazed how much you can do with Python once you learn the basics.

Great Community and Support

- Python has a large supporting community. There are numerous active forums online which can be handy if you are stuck. Some of them are:
- Google Forum for Python
- Python Questions - Stack Overflow



Python vs PHP

From the development point of view, PHP is a web-oriented language.

Choosing between **Python or PHP for web applications** pay attention to these characteristics:

Python vs PHP		
	 python™	 php
Popularity	Very popular	Very popular
Frameworks	A lot of frameworks	A few frameworks
Learning	Easy to learn	Harder to learn

Python vs Java

	 python™	 Java
Learning	Easy to learn	Harder to learn
Cross-platform apps	×	✓
Compatibility with operating systems	✓	✓
Network based apps	×	✓

Python vs C#

	 python™	
Simplicity	✓	×
Script writing	In any environment	Only in IDE
Libraries	A lot of libraries	Few libraries
Performance	Low	High

Difference between Ruby and Python

In terms of the first language, **Ruby and Python** are the most popular ones. Ruby is extremely popular technology for building websites. Among the most famous are Twitter (the early version), Basecamp, Github, Airbnb, Slideshare and Groupon.

Python vs Ruby		
	 python™	
Approach to a problem	One solution	A lot of solutions
Community	Large	Large
Syntax	Very simple	More complex

Installing and Running Python in Windows

1. Go to Download Python page on the official site and click **Download Python 3.7** (You may see different version name).
2. When the download is completed, double-click the file and follow the instructions to install it.

When Python is installed, a program called IDLE is also installed along with it. It provides graphical user interface to work with Python.

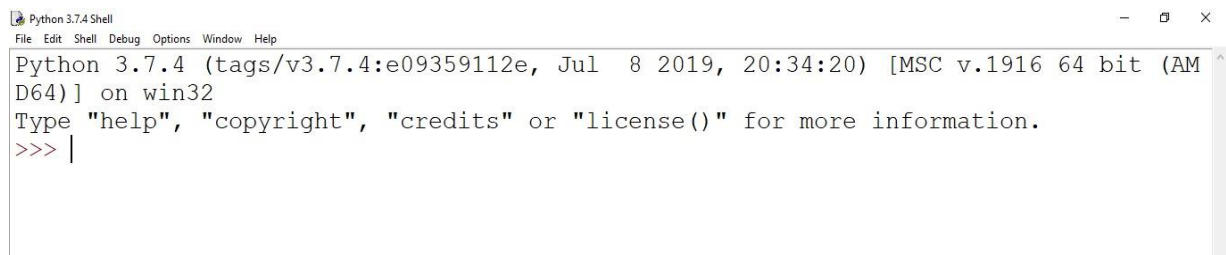
3. Open IDLE, copy the following code below and press enter.
4. `print("Hello, World!")`
5. To create a file in IDLE, go to **File > New Window** (Shortcut: **Ctrl+N**).
6. Write Python code (you can copy the code below for now) and save (Shortcut: **Ctrl+S**) with **.py** file extension like: `hello.py` or `your-first-program.py`
7. `print("Hello, World!")`

8. Go to **Run > Run module** (Shortcut: **F5**) and you can see the output. Congratulations, you've successfully run your first Python program.

PYTHON HAS TWO BASIC MODES:

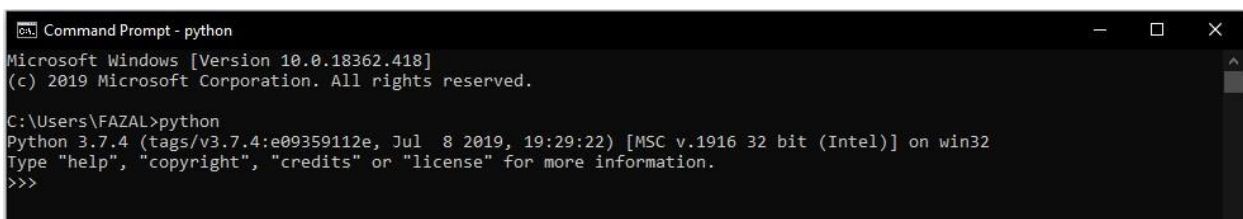
Interactive mode: is a command line shell which gives immediate output for each statement, while running previously statements in **active** memory.

This mode is also referred as REPL (Read Evaluate Print Loop)



```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

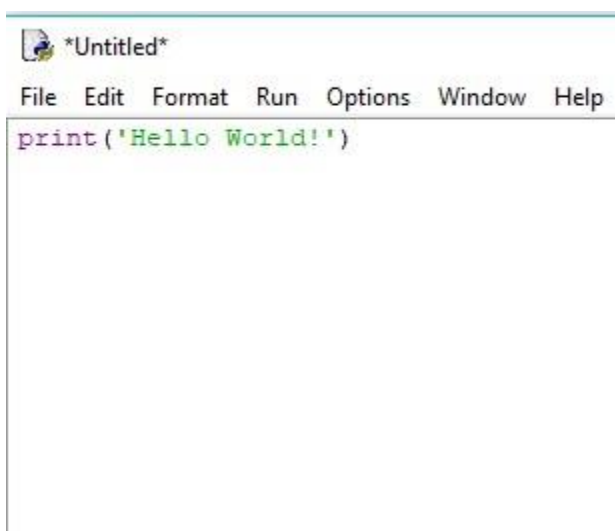
We can start an interactive session from Command Prompt Directly.



```
Command Prompt - python
Microsoft Windows [Version 10.0.18362.418]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\FAZAL>python
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

Normal mode: is where the scripted python file (.py) run in the Python interpreter.



```
*Untitled*
File Edit Format Run Options Window Help
print('Hello World!')
```


Python Quick start

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

```
C:\Users\Your Name>python helloworld.py
```

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

```
helloworld.py
```

```
print("Hello, World!")
```

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

```
C:\Users\Your Name>python helloworld.py
```

The output should read:

```
Hello, World!
```

Congratulations, you have written and executed your first Python program.

Why Django?

Django is a Web framework written in Python.

A Web framework is a software that supports the development of dynamic Web sites, applications, and services.

It provides a set of tools and functionalities that solves many common problems associated with Web development, such as security features, database access, sessions, template processing, URL routing, internationalization, localization, and much more.

Using a Web framework, such as Django, enables us to develop secure and reliable Web applications very quickly in a standardized way.

The development of Django is supported by the [Django Software Foundation](#), and it's sponsored by companies like JetBrains and Instagram. Who's Using Django?

It's good to know who is using Django out there, so to have an idea what you can do with it. Among the biggest Web sites using Django we have: [Instagram](#), [Disqus](#), [Mozilla](#), [Bitbucket](#), [Last.fm](#), [National Geographic](#).

Installation

The first thing we need to do is install some programs on our machine so to be able to start playing with Django. The basic setup consists of installing

- **Python**
- **Virtualenv**
- **Django**

Using virtual environments is not mandatory, but it's highly recommended.

Installing Virtualenv we are going to use **pip**, a tool to manage and install Python packages, to install **virtualenv**.

In the Command Prompt, execute the command below:

```
pip install virtualenv
```

From now on, everything we install, including Django itself, will be installed inside a Virtual Environment.

```
mkdir myproject  
cd myproject
```

This folder is the higher level directory that will store all the files and things related to our Django project, including its virtual environment.

let's start by creating our very first virtual environment and installing Django.

Inside the **myproj** folder:

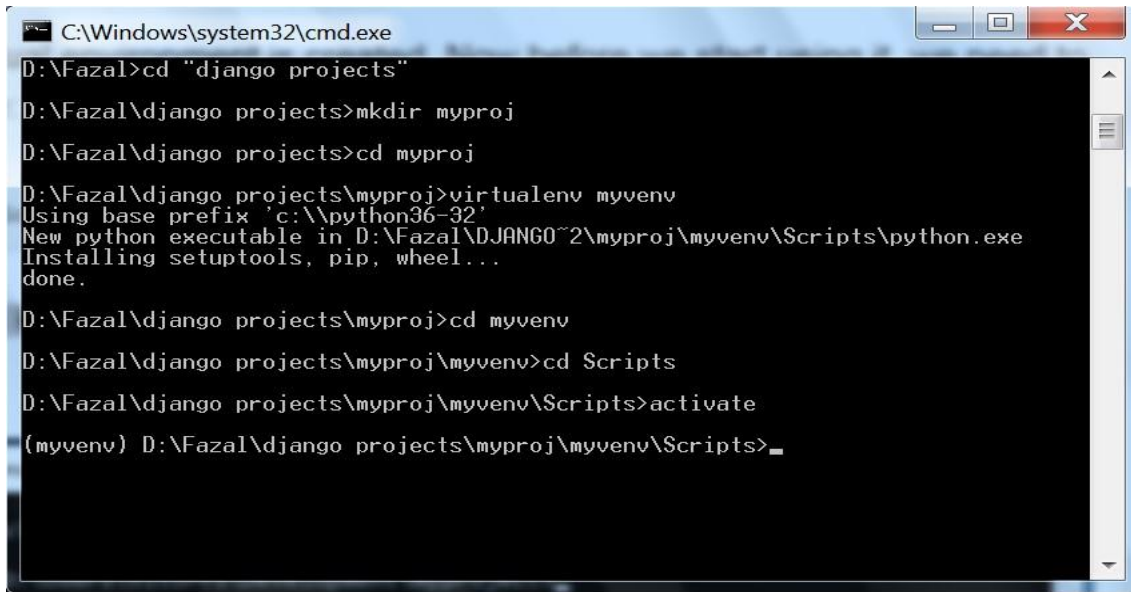
```
virtualenv myenv
```

Our virtual environment is created.

Now before we start using it, we need to activate:

```
myenv\Scripts\activate
```

You will know it worked if you see **(venv)** in front of the command line, like this:



```
C:\Windows\system32\cmd.exe
D:\Fazal>cd "django projects"
D:\Fazal\django projects>mkdir myproj
D:\Fazal\django projects>cd myproj
D:\Fazal\django projects\myproj>virtualenv myenv
Using base prefix 'c:\python36-32'
New python executable in D:\Fazal\DJANGO~2\myproj\myenv\Scripts\python.exe
Installing setuptools, pip, wheel...
done.
D:\Fazal\django projects\myproj>cd myenv
D:\Fazal\django projects\myproj\myenv>cd Scripts
D:\Fazal\django projects\myproj\myenv\Scripts>activate
(myenv) D:\Fazal\django projects\myproj\myenv\Scripts>_
```

to deactivate the **venv** run the command below:

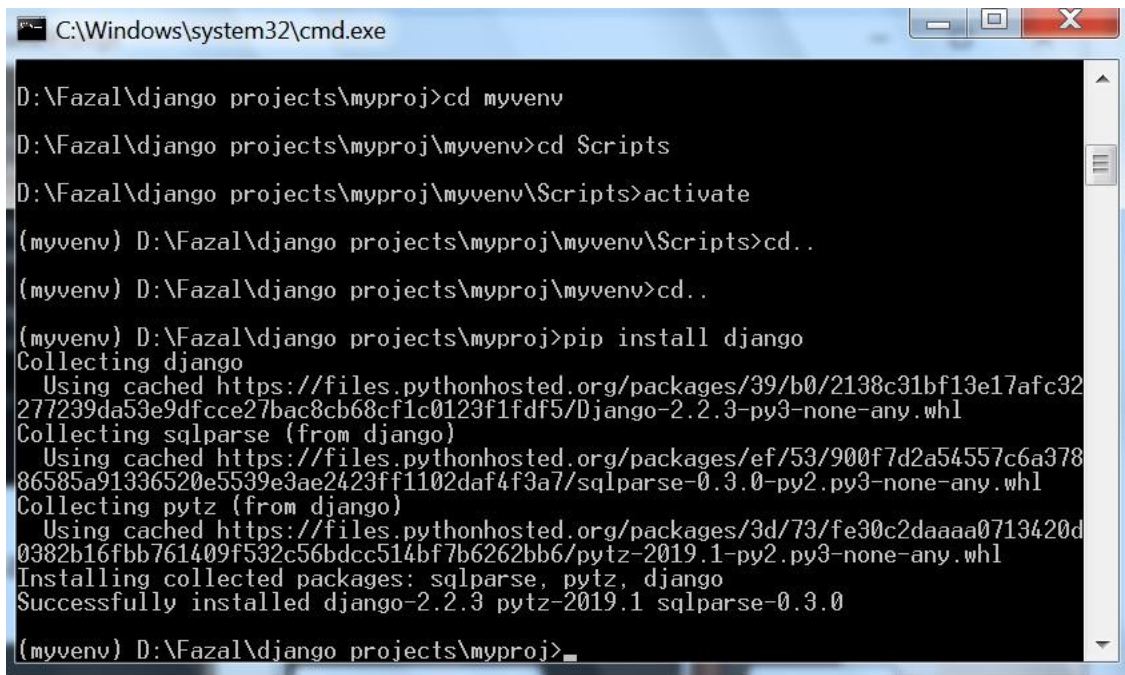
```
venv\Scripts\deactivate.bat
```

But let's keep it activated for the next steps.

Installing Django

Now that we have the **venv** activated, run the following command to install Django:

```
pip install django
```



```

C:\Windows\system32\cmd.exe

D:\Fazal\django projects\myproj>cd myvenv
D:\Fazal\django projects\myproj\myvenv>cd Scripts
D:\Fazal\django projects\myproj\myvenv\Scripts>activate
(myvenv) D:\Fazal\django projects\myproj\myvenv\Scripts>cd..
(myvenv) D:\Fazal\django projects\myproj\myvenv>cd..
(myvenv) D:\Fazal\django projects\myproj>pip install django
Collecting django
  Using cached https://files.pythonhosted.org/packages/39/b0/2138c31bf13e17afc32
277239da53e9dfcce27bac8cb68cf1c0123f1fdf5/Django-2.2.3-py3-none-any.whl
Collecting sqlparse (from django)
  Using cached https://files.pythonhosted.org/packages/ef/53/900f7d2a54557c6a378
86585a91336520e5539e3ae2423ff1102daf4f3a7/sqlparse-0.3.0-py2.py3-none-any.whl
Collecting pytz (from django)
  Using cached https://files.pythonhosted.org/packages/3d/73/fe30c2daaaa0713420d
0382b16fbb761409f532c56bdcc514bf7b6262bb6/pytz-2019.1-py2.py3-none-any.whl
Installing collected packages: sqlparse, pytz, django
Successfully installed django-2.2.3 pytz-2019.1 sqlparse-0.3.0
(myvenv) D:\Fazal\django projects\myproj>_

```

Starting a New Project

To start a new Django project, run the command below:

```
django-admin startproject myproject
```

The command-line utility **django-admin** is automatically installed with Django.

After we run the command above, it will generate the base folder structure for a Django project.

Our initial project structure is composed of five files:

- **manage.py**: a shortcut to use the **django-admin** command-line utility. It's used to run management commands related to our project.

We will use it to run the development server, run tests, create migrations and much more.

- **__init__.py**: this empty file tells Python that this folder is a Python package.
- **settings.py**: this file contains all the project's configuration.
- **urls.py**: this file is responsible for mapping the routes and paths in our project.

For example, if you want to show something in the URL , you `/about/` have to map it here first.

- **wsgi.py**: this file is a simple gateway interface used for deployment.

You don't have to bother about it. Just let it be for now.

Django comes with a simple web server installed.

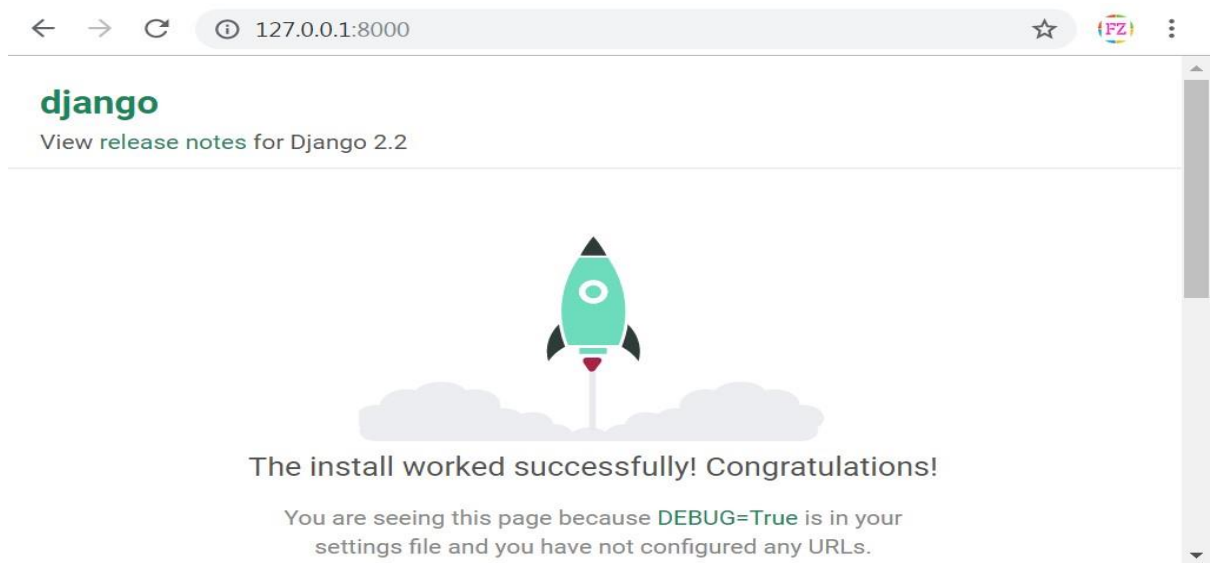
It's very convenient during the development, so we don't have to install anything else to run the project locally.

We can test it by executing the command:

```
python manage.py runserver
```

For now, you can ignore the migration errors; we will get to that later.

Now open the following URL in a Web browser: **http://127.0.0.1:8000** and you should see the following page:



Hit CTRL + BREAK to stop the development server.

Django Apps

In the Django philosophy we have two important concepts:

- **app**: is a Web application that does something.

An app usually is composed of a set of models (database tables), views, templates, tests.

- **project**: is a collection of configurations and apps.

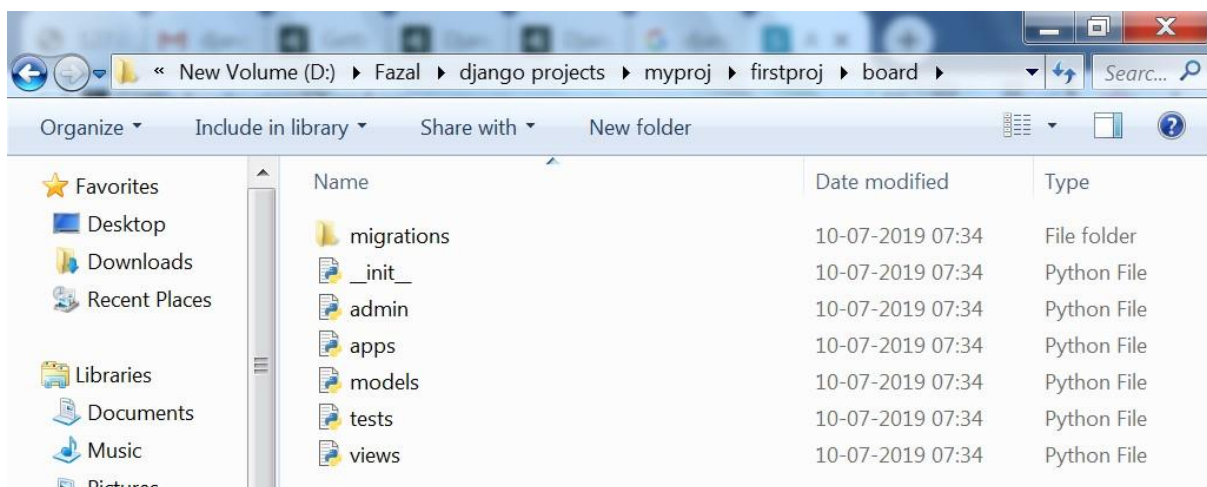
One project can be composed of multiple apps, or a single app.

It's important to note that you can't run a Django **app** without a **project**. Simple websites like a blog can be written entirely inside a single app, which could be named **blog** or **weblog** for example.

let's create a simple Web Forum or Discussion Board. To create our first app, go to the directory where the **manage.py** file is and executes the following command:

```
django-admin startapp boards
```

Notice that we used the command **startapp** this time.



So, let's first explore what each file does:

- **migrations/**: here Django store some files to keep track of the changes you create in the **models.py** file, so to keep the database and the **models.py** synchronized.
- **admin.py**: this is a configuration file for a built-in Django app called **Django Admin**.
- **apps.py**: this is a configuration file of the app itself.
- **models.py**: here is where we define the entities of our Web application. The models are translated automatically by Django into database tables.
- **tests.py**: this file is used to write unit tests for the app.
- **views.py**: this is the file where we handle the request/response cycle of our Web application.

Now that we created our first app, let's configure our project to *use* it.

To do that, open the **settings.py** and try to find the **INSTALLED_APPS** variable:

settings.py

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

As you can see, Django already come with 6 built-in apps installed. They offer common functionalities that most Web applications need, like authentication, sessions, static files management (images, javascripts, css, etc.) and so on.

Hello, World!

Let's write our first **view**. We will explore it in great detail in the next tutorial. But for now, let's just experiment how it looks like to create a new page with Django.

Open the **views.py** file inside the **boards** app, and add the following code: **views.py**

```
from django.http import HttpResponse
```

```
def home(request):
```

```
    return HttpResponse('Hello, World!')
```

Views are Python functions that receive an `HttpRequest` object and returns an `HttpResponse` object.

Receive a *request* as a parameter and returns a *response* as a result. That's the flow you have to keep in mind!

So, here we defined a simple view called **home** which simply returns a message saying **Hello, World!**.

Now we have to tell Django *when* to serve this view. It's done inside the **urls.py** file: **urls.py**

```
from django.conf.urls import url
```

```
from django.contrib import admin
```

```
from boards import views
```

```
urlpatterns = [
    url(r'^$', views.home, name='home'),
    url(r'^admin/', admin.site.urls),
]
```

If you compare the snippet above with your **urls.py** file, you will notice I added the following new line: `url(r'^$', views.home, name='home')` and imported the **views** module from our app **boards** using `from boards import views.`

As I mentioned before, we will explore those concepts in great detail later on.

But for now, Django works with **regex** to match the requested URL. For our **home** view, I'm using the `^$` regex, which will match an empty path, which is the homepage (this url: **http://127.0.0.1:8000**).

If I wanted to match the URL **http://127.0.0.1:8000/homepage/**, my url would be:

```
url(r'^homepage/$', views.home,
    name='home')
```

Let's see what happen:

```
python manage.py runserver
```

In a Web browser, open the `http://127.0.0.1:8000` URL.

SCREENSHOTS

6.SCREENSHOTS

Packages : Numpy,Matplotlib,SKlearn, Pandas, Flask Tool : Python 3.7

```
: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

```
: import itertools
import seaborn as sns
import pandas_profiling
import statsmodels.formula.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from patsy import dmatrices
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.
import pandas.util.testing as tm
```

```
: from sklearn import datasets
from sklearn.feature_selection import RFE
import sklearn.metrics as metrics
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2, f_classif, mutual_info_classif
```

```
: train=pd.read_csv('/content/drive/My Drive/kdd/NSL_Dataset/Train.txt',sep=',')
test=pd.read_csv('/content/drive/My Drive/kdd/NSL_Dataset/Test.txt',sep=',')
```

Data preprocessing

```
n [6]: columns=["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","error_rate","srv_error_rate",
"error_rate","srv_error_rate","same_srv_rate","diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_srv",
"dst_host_diff_srv_rate","dst_host_same_src_port_rate",
"dst_host_srv_diff_host_rate","dst_host_error_rate","dst_host_srv_error_rate",
"dst_host_error_rate","dst_host_srv_error_rate","attack","last_flag"]
```

```
n [7]: train.columns=columns
test.columns=columns
```

```
n [8]: train.head()
```

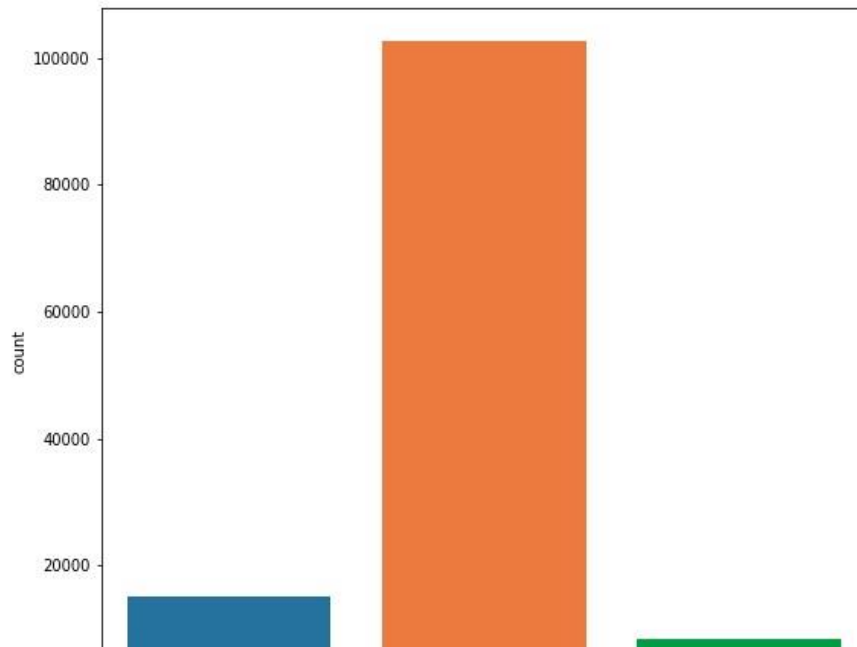
```
ut[8]:
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	root_s
0	0	udp	other	SF	146	0	0	0	0	0	0	0	0	0
1	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0
2	0	tcp	http	SF	232	8153	0	0	0	0	0	1	0	0
3	0	tcp	http	SF	199	420	0	0	0	0	0	1	0	0
4	0	tcp	private	REJ	0	0	0	0	0	0	0	0	0	0

```
n [9]: test.head()
```

Data EDA

```
: # Protocol type distribution
plt.figure(figsize=(9,8))
sns.countplot(x="protocol_type", data=train)
plt.show()
```



Model Building

```
train_X=train_new[cols]
train_y=train_new['attack_class']
test_X=test_new[cols]
test_y=test_new['attack_class']
```

ML Deploy

Logistic Regression

```
# Building Models
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(random_state=0,solver='lbfgs',multi_class='multinomial')
logreg.fit( train_X, train_y)
logreg.predict(train_X) #by default, it use cut-off as 0.5
```

```
list( zip( cols, logreg.coef_[0] ) )
```

```
logreg.intercept_
```

```
logreg.score(train_X,train_y)
```

Decision Trees

```
train_X.shape
```

```
param_grid = {'max_depth': np.arange(2, 12),
              'max_features': np.arange(10,15)}
```

```
train_y.shape
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier, export_graphviz, export
tree = GridSearchCV(DecisionTreeClassifier(), param_grid, cv = 10, verbose=1, n_jobs=-1)
tree.fit( train_X, train_y )
```

```
tree.best_score_
```

```
tree.best_estimator_
tree.best_params_
```

```
train_pred = tree.predict(train_X)
```

```
print(metrics.classification_report(train_y, train_pred))
```

```
test_pred = tree.predict(test_X)
```

Random Forest

```
: from sklearn.ensemble import RandomForestClassifier
   pargrid_rf = {'n_estimators': [50,60,70,80,90,100],
                'max_features': [2,3,4,5,6,7]}
```

```
: from sklearn.model_selection import GridSearchCV
   gscv_rf = GridSearchCV(estimator=RandomForestClassifier(),
                          param_grid=pargrid_rf,
                          cv=10,
                          verbose=True, n_jobs=-1)

   gscv_results = gscv_rf.fit(train_X, train_y)
```

```
: gscv_results.best_params_
```

```
: gscv_rf.best_score_
```

```
: radm_clf = RandomForestClassifier(oob_score=True, n_estimators=80, max_features=5, n_jobs=-1)
   radm_clf.fit( train_X, train_y )
```

```
: radm_test_pred = pd.DataFrame( { 'actual': test_y,
                                   'predicted': radm_clf.predict( test_X ) } )
```

Support Vector Machine (SVM)

```
: from sklearn.svm import LinearSVC
   svm_clf = LinearSVC(random_state=0, tol=1e-5)
   svm_clf.fit(train_X, train_y)
```

```
: print(svm_clf.coef_)
   print(svm_clf.intercept_)
   print(svm_clf.predict(train_X))
```

```
: from sklearn.svm import SVC
   from sklearn.pipeline import make_pipeline

   model = SVC(kernel='rbf', class_weight='balanced', gamma='scale')
```

```
: model.fit(train_X, train_y)
```

```
: from sklearn.model_selection import GridSearchCV
   param_grid = {'C': [1, 10],
                 'gamma': [0.0001, 0.001]}
   grid = GridSearchCV(model, param_grid)

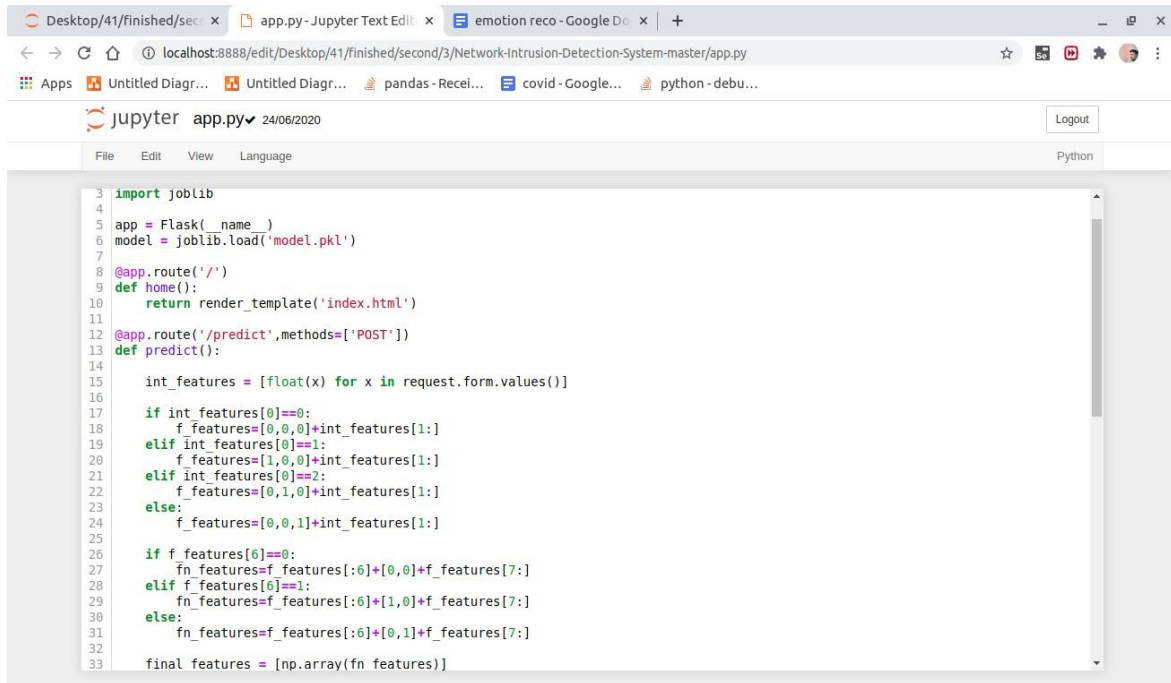
   grid.fit(train_X, train_y)
```

```
: print(grid.best_params_)
```

From the score accuracy we concluding the DT & RF give better accuracy.

Building pickle file for predicting the user input:

Application

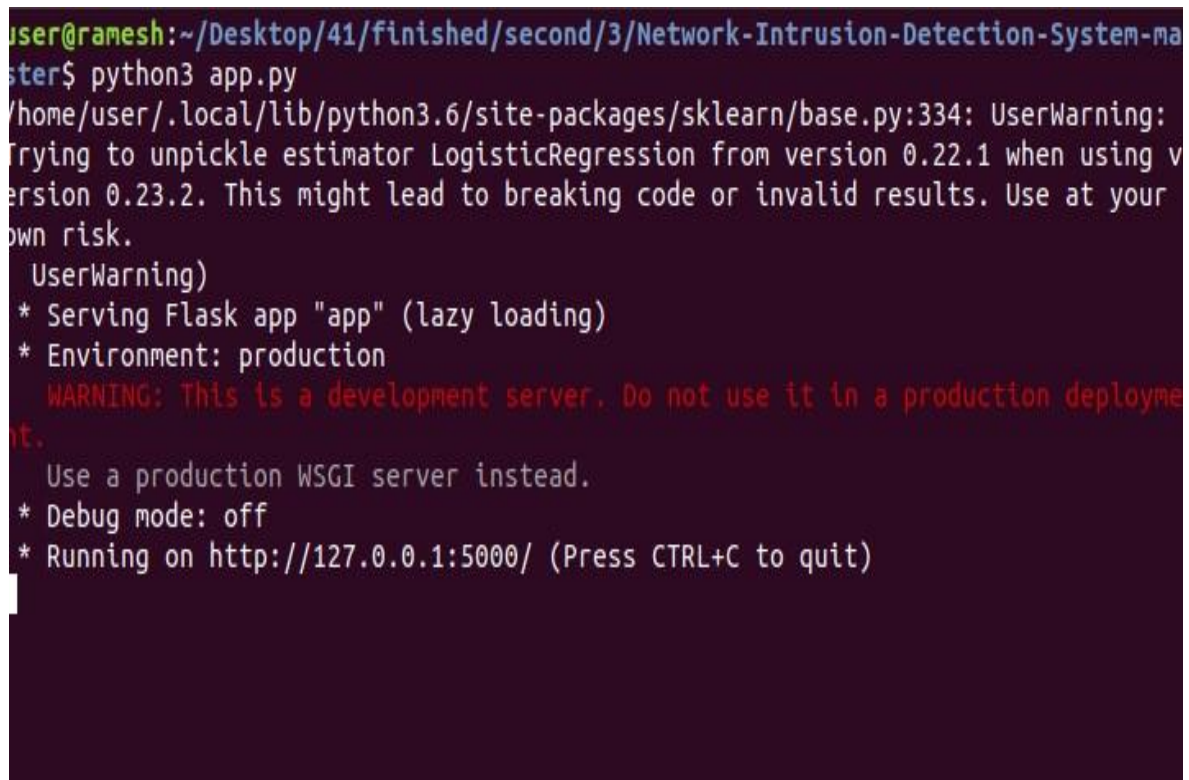


```

3 import joblib
4
5 app = Flask(__name__)
6 model = joblib.load('model.pkl')
7
8 @app.route('/')
9 def home():
10     return render_template('index.html')
11
12 @app.route('/predict', methods=['POST'])
13 def predict():
14
15     int_features = [float(x) for x in request.form.values()]
16
17     if int_features[0]==0:
18         f_features=[0,0,0]+int_features[1:]
19     elif int_features[0]==1:
20         f_features=[1,0,0]+int_features[1:]
21     elif int_features[0]==2:
22         f_features=[0,1,0]+int_features[1:]
23     else:
24         f_features=[0,0,1]+int_features[1:]
25
26     if f_features[6]==0:
27         fn_features=f_features[:6]+[0,0]+f_features[7:]
28     elif f_features[6]==1:
29         fn_features=f_features[:6]+[1,0]+f_features[7:]
30     else:
31         fn_features=f_features[:6]+[0,1]+f_features[7:]
32
33     final_features = [np.array(fn_features)]

```

Localhost - in cmd python app.py



```

user@ramesh:~/Desktop/41/finished/second/3/Network-Intrusion-Detection-System-master$ python3 app.py
/home/user/.local/lib/python3.6/site-packages/sklearn/base.py:334: UserWarning:
Trying to unpickle estimator LogisticRegression from version 0.22.1 when using v
ersion 0.23.2. This might lead to breaking code or invalid results. Use at your
own risk.
  UserWarning)
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

```

Network Intrusion Detection System

Attack:

Other

Number of connections to the same destination host as the current connection in the past two seconds :

count

The percentage of connections that were to different services, among the connections aggregated in dst_host_count :

dst_host_diff_srv_rate

The percentage of connections that were to the same source port, among the connections aggregated in dst_host_srv_count :

dst_host_same_src_port_rate

The percentage of connections that were to the same service, among the connections aggregated in dst_host_count :

dst_host_same_srv_rate

Number of connections having the same port number :

dst_host_srv_count

Enter the input

Status of the connection - Normal or Error :

Other

Last Flag :

3

1 if successfully logged in; 0 otherwise :

5

The percentage of connections that were to the same service, among the connections aggregated in count :

5

The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count :

2

Destination network service used http or not :

No

Predict

Predict attack

Predict

Attack Class should be **DOS**

TESTING

7.1 TESTCASES

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test.

Each test type addresses a specific testing requirement.

TYPES OF TESTS:

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such

as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

SUMMARY

8. SUMMARY

Most techniques used in today's IDS are not able to deal with the dynamic and complex nature of cyber-attacks on computer networks. Hence, efficient adaptive methods like various techniques of machine learning can result in higher detection rates, lower false alarm rates and reasonable computation and communication costs. We reviewed several influential algorithms for intrusion detection based on various machine learning techniques. Characteristics of ML techniques makes it possible to design IDS that have high detection rates and low false positive rates while the system quickly adapts itself to changing malicious behaviors. IDS using many Machine Learning Techniques like Random Forest, Decision tree and logistic regression to perform better in various metrics. The IDS should provide the most effective solutions based on the requirements. One thing is sure, any company failing to adopt these techniques now or in the immediate future risk compromising data or worse servers.

8.1 SUMMARY OF PREVIOUS WORK

Researcher	Research & work carried out	Remark
Hamed Alqahtani, Iqbal H.Sarker, AsraKalim, Minhaz Hossain, sheikhikhlq and Sohrab July 2020	Analysed Varies Popular Machine Learning Technique Bayesian Network (BN), Naive Bayes (NB), Random Forest (RF), Decision Tree (DT), Random Tree (RT), Decision Table (DTb), and Artificial Neural Network (ANN), for providing intelligent services in the domain of cybersecurity,	Machine Learning Algorithms
Mahdi Zamani 8 Dec 2013	Anomaly detection systems rely on	Anomaly Detection Technique
Lakshmanarao, M.Shashi JANUARY 2020	DS to deal with large network traffic volumes, highly uneven data distribution, the difficulty to realize decision boundaries between normal	Huge Network Traffic

	Andabnormal behavior	
R.Devakunchari, Sourabh,Prakhar Malik May 2019	Proposed a true time intrusion detection system supported time signatures	True time Intrusion Detection System
ManjeetRege,Raymond Blanch K. Mbah 2018	improving security solutions that help in identifying and decisively dealing withsecure knowledge in crafting and launching bigger and more sophisticated attacks	Improving Security Solutions

Fig 8.1 : Summary Of Previous Work

CONCLUSION

9. CONCLUSION

In this article, the method for application layer attack detection based on machine learning was proposed. The model consists of patterns that are obtained using graph-based segmentation technique and dynamic programming. The regular expressions are used for modelling the genuine behaviour of the applications and detecting cyber attacks.

We also presented the results that prove the efficiency of the proposed algorithm that can be effectively used for application layer attack detection.

The experiments show that the proposed approach can achieve 94.46% of detection ratio while having <4.5% of false positives.

REFERENCES

10. REFERENCES

- [1] CERT Polska Annual Report 2012. http://www.cert.pl/PDF/Report_CP_2012.pdf
- [2] Cisco Annual Report 2013.
http://www.cisco.com/web/about/ac49/ac20/ac19/ar2013/docs/2013_Annual_Report.pdf
- [3] LESG. <http://www.cs.northwestern.edu/~ychen/Papers/LESG-ICNP07.pdf>
- [4] D. Kong, J. Gong, S. Zhu, P. Liu and H. Xi. SAS: semantics aware signature generation for polymorphic worm detection. *International Journal of Information Security*, 50, 1–19, 2011.
- [5] M. Sharma and D. Toshniwal. Pre-clustering algorithm for anomaly detection and clustering that uses variable size buckets. *Recent Advances in Information Technology*, 515–519, 2012.
- [6] Kruegel, C., Toth, T., Kirda, E.: Service specific anomaly detection for network intrusion detection. In: *Proc. of ACM Symposium on Applied Computing*, pp. 201–208 (2002).
- [7] Choraś, M., Kozik, R., Puchalski, D., Hołubowicz, W.: Correlation Approach for SQL Injection Attacks Detection. In: Herrero, Á., et al. (eds.) *Int. Joint Conf. CISIS'12-ICEUTE'12-SOCO'12. AISC*, vol. 189, pp. 177–185. Springer, Heidelberg (2013).
- [8] Adaniya, M.H.A.C., Lima, M.F., Rodrigues, J.J.P.C., Abrao, T., Proenca, M.L.: Anomaly detection using DSNS and Firefly Harmonic Clustering Algorithm. In: *2012 IEEE International Conference on Communications (ICC)*, pp. 1183–1187 (2012).
- [9] Mazel, J., Casas, P., Labit, Y., Owezarski, P.: Sub-Space clustering, InterClustering Results Association and anomaly correlation for unsupervised network anomaly detection. In: *2011 7th International Conference on Network and Service Management (CNSM)*, pp. 1–8 (2011).
- [10] Yang, C., Deng, F., Yang, H.: An Unsupervised Anomaly Detection Approach using Subtractive Clustering and Hidden Markov Model. In: *Second International Conference on Communications and Networking in China, CHINACOM 2007*, pp. 313–316 (2007).