A

Major Project Report

on

DETECTING UNBALANCED NETWORK TRAFFIC INSTRUTIONS WITH DEEP LEARNING

A Report submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology

by

K.Sai Rohith
(21EG105G27)

P.Akshitha
(21EG105G51)

T.Dhanush Tej
(21EG105G58)

Under the guidance

of

Dr.J.Balaraju

Assistant Professor

Department Of Computer Science and Engineering

ANURAG UNIVERSITY

Venkatapur (V), Ghatkesar (M), Medchal (D), T.S-500088 Year

2024-25

CERTIFICATE

This is to certify that the Report entitled "**DETECTING UNBALANCED NETWORK TRAFFIC INTRUSIONS WITH DEEP LEARNING**" is being submitted by **Mr K. Sai Rohith** bearing the Hall Ticket number **21EG105G27** and **Ms. P. Akshitha** bearing the Hall Ticket number **21EG105G51**and **Mr T. Dhanush Tej** bearing the Hall Ticket number **21EG105G58** in partial fulfillment for the award of Bachelor of Technology in Computer Science and Engineering to the Anurag University is a record of bonafide work carried out by them under my guidance and supervision for the academic year 2024 to 2025.

Signature of Supervisor	Signature of Dean
Name : Dr.J.Balaraju	**Dr.G.Vishnu Murthy**
Assistant Professor	Dean, CSE

ACKNOWLEDGMENT

We would like to express our sincere thanks and deep sense of gratitude to project supervisor **Dr.J.BALARAJU** for his/her constant encouragement and inspiring guidance without which this project could not have been completed. His/her critical reviews and constructive comments improved our grasp of the subject and steered to the fruitful completion of the work. His/her patience, guidance and encouragement made this project possible.

We would like to express our sincere gratitude to **Dr.Archana Mantri**, Vice Chancellor, Anurag University and **Dr.P.Bhaskara Reddy**, Registrar, Anurag University for their encouragement and support.

We would like to express our special thanks to **Dr. V. Vijaya Kumar**, Dean School of Engineering, Anurag University, for his encouragement and timely support in our B.Tech program.

We would like to acknowledge our sincere gratitude for the support extended by **Dr. G. Vishnu Murthy**, Dean, Department of Computer Science Engineering, Anurag University. We also express my deep sense of gratitude to **Dr.P V S Shiva Prasad**, academic coordinator. **Dr.G,Balram**, PQMC Chair, **Dr.J.Himabindu Priyanka** , Project Coordinator, Project Review and Quality & Monitoring Committee members, whose research expertise and commitment to the highest standards continuously motivated us during the crucial stage of our project work

K.SaiRohith

21EG105G27

P.Akshitha

21EG105G51

T.Dhanust Tej

21EG105G58

DECLARATION

We hereby declare that the Report entitled "**DETECTING UNBALANCED NETWORK TRAFFIC INTRUSIONS WITH DEEP LEARNING**" submitted for the award of Bachelor of Technology Degree is my original work. It has not been submitted to any other University or Institution for the award of any degree or diploma.

K.Sai Rohith

21EG105G27

P.Akshitha

21EG105G51

T.Dhanush Tej

21EG105G37

ABSTRACT

Detecting unbalanced network traffic intrusions is critical for building resilient Intrusion Detection Systems (IDS) capable of accurately identifying malicious activities in network traffic. The growing prevalence of cyber threats necessitates advanced techniques to address the challenges posed by class imbalance in network datasets. This study uses the CIC-IDS 2017 dataset, a widely recognized benchmark for network intrusion detection. The project explores multiple machine learning and deep learning algorithms, including XGBoost, LSTM, minVGG (1D CNN), AlexNet (1D CNN), CNN + BiLSTM, and a Voting Classifier combining Random Forest (RF) and Decision Tree (DT). A robust ensemble approach is applied, combining the predictions of multiple models to enhance overall detection accuracy. The results demonstrate that the Voting Classifier and CNN+BiLSTM achieve outstanding performance, reaching an accuracy of 100%. These ensemble techniques significantly improve the IDS's ability to detect and classify intrusions in highly imbalanced network traffic, ensuring enhanced detection reliability and resilience in the face of evolving cyber threats.

TABLE OF CONTENTS

1. INTRODUCTION

Digital communication underpins many essential infrastructures and services in our environment, making the security of network systems critically important. Cyber-attacks are becoming sophisticated and pervasive, presenting significant problems for organisations striving to safeguard sensitive data and sustain their operations. Network traffic incursions are a highly insidious cyber threat, capable of exploiting weaknesses and undermining the integrity of communication networks. These breaches in network traffic are very perilous. Conventional methods of intrusion detection mostly depend on rule-based systems or signature-based techniques, which are constrained in their capacity to adjust to the evolving landscape of contemporary cyber threats. Consequently, there is an increasing need for sophisticated and adaptable intrusion detection systems that can efficiently identify and assess developing threats. In response to this difficulty, the use of machine learning methods has emerged as a viable strategy to augment the efficacy of intrusion detection systems. Unbalanced network traffic occurs when some segments of a computer network have much more activity than others. It resembles a situation when some lanes on a roadway are much more congested than others. This disparity may lead to complications, as areas with high circulation may get inundated, whilst less frequented sections may receive little oversight.

1.1 Objective:

The objective of this study is to develop an effective Intrusion Detection System (IDS) capable of accurately detecting malicious activities in network traffic, addressing the challenge of class imbalance in network datasets. Using the CIC-IDS 2017 dataset, the project explores various algorithms including XGBoost, LSTM, minVGG (1D CNN), AlexNet (1D CNN), CNN + BiLSTM, and Voting Classifier. The goal is to enhance detection accuracy by leveraging ensemble techniques and deep learning models, improving IDS performance against evolving cyber threats.

1.2 Problem Statement:

- The rapid growth of cyber threats and the increasing volume of network traffic contribute to the challenge of detecting malicious activities, with unbalanced data further complicating the accurate identification of intrusions.
- Intrusion detection systems (IDS) struggle to maintain high performance in highly imbalanced datasets, where the majority of network traffic consists of benign data, making it difficult for traditional models to identify rare but critical intrusions.
- Organizations and individuals relying on secure network infrastructures, such as businesses and government entities, are affected by the inability of current IDS to effectively detect and prevent cyberattacks, leading to potential data breaches and financial losses.
- The lack of accurate IDS can result in compromised network security, loss of sensitive data, financial damage, and undermining public trust in the security of digital systems, creating vulnerabilities to evolving cyber threats.
- To address this, we will implement a combination of deep learning and ensemble techniques, including CNN+BiLSTM and Voting Classifier, to improve the accuracy of intrusion detection in imbalanced network traffic datasets.

1.3 SOFTWARE REQUIREMENTS

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

1.4 HARDWARE REQUIREMENTS

1) Operating System: Windows Only

2) Processor: i5 and above

3) Ram: 8GB and above

4) Hard Disk: 25 GB in local drive

2. FEASIBILITY STUDY

Feasibility study examines the viability or sustainability of an idea, project, or business. The study examines whether there are enough resources to implement it, and the concept has the potential to generate reasonable profits. In addition, it will demonstrate the benefits received in return for taking the risk of investing in the idea.

Types of Feasibility Study

There are several different kinds of feasibility studies. Understanding the types of feasibility studies and the technicalities of the concept is important for any business. They are elaborated below:

Technical Feasibility

Technical feasibility study checks for accessibility of technical resources in the organization. In case technological resources exist, the study team will conduct assessments to check whether the technical team can customize or update the existing technology to suit the new method of workings for the project by properly checking the health of the hardware and software. Many factors need to be taken into consideration here, like staffing requirements, transportation, and technological competency.

Financial Feasibility

Financial feasibility allows an organization to determine cost-benefit analysis. It gives details about the investment that has to go in to get the desired level of benefit (profit). Factors such as total cost and expenses are considered to arrive simultaneously. With this data, the companies know their present state of financial affairs and anticipate future monetary requirements and the sources from which the company can acquire them. Investors can largely benefit from the economic analysis done. Assessing the return on investment of a particular asset or acquisition can be a financial feasibility study example.

Market Feasibility

It assesses the industry type, the existing marketing characteristics and improvements to make it better, the growth evident and needed, competitive environment of the company's products and services. Preparations of sales projections can thus be a good market feasibility study example.

Organization Feasibility

Organization feasibility focuses on the organization's structure, including the legal system, management team's competency, etc. It checks whether the existing conditions will suffice to implement the business idea. Many factors need to be taken into consideration here, like staffing requirements, transportation, and technological competency. In addition, it will demonstrate the benefits received in return for taking the risk of investing in the idea.

3. LITERATURE SURVEY

3.1 A dependable hybrid machine learning model for network intrusion detection:

https://www.sciencedirect.com/science/article/abs/pii/S2214212622002496

ABSTRACT: Network intrusion detection systems (NIDSs) play an important role in computer network security. There are several detection mechanisms where anomaly-based automated detection outperforms others significantly. Amid the sophistication and growing number of attacks, dealing with large amounts of data is a recognized issue in the development of anomaly-based NIDS. However, do current models meet the needs of today's networks in terms of required accuracy and dependability? In this research, we propose a new hybrid model that combines machine learning and deep learning to increase detection rates while securing dependability. Our proposed method ensures efficient pre-processing by combining SMOTE for data balancing and XGBoost for feature selection. We compared our developed method to various machine learning and deep learning algorithms in order to find a more efficient algorithm to implement in the pipeline. Furthermore, we chose the most effective model for network intrusion based on a set of benchmarked performance analysis criteria.

3.2 CGAN-Based Collaborative Intrusion Detection for UAV Networks: A Blockchain-Empowered Distributed Federated Learning Approach:

https://ieeexplore.ieee.org/abstract/document/9863068

ABSTRACT: Numerous resource-constrained Internet of Things (IoT) devices make the edge IoT consisting of unmanned aerial vehicles (UAVs) vulnerable to network intrusion. Therefore, it is critical to design an effective intrusion detection system (IDS). However, the differences in local data sets among UAVs show small samples and uneven distribution, further reducing the detection accuracy of network intrusion. This article proposes a conditional generative adversarial net (CGAN)-based collaborative intrusion detection algorithm with blockchain-empowered distributed federated learning to solve the above problems. This study introduces long short-term memory (LSTM) into the CGAN training to improve the effect of generative networks. Based on the feature extraction ability of LSTM networks, the generated data with CGAN are used as augmented data and applied in the detection and classification of intrusion data. Distributed federated learning with differential privacy ensures data security and privacy and allows collaborative training of CGAN models using multiple distributed data sets. Blockchain stores and shares the training models to ensure security when the global model's aggregation and updating. The proposed method has good generalization ability, which can greatly improve the detection of intrusion data.

3.3 Sea turtle foraging algorithm with hybrid deep learning-based intrusion detection for the internet of drones environment:

https://www.sciencedirect.com/science/article/pii/S0045790623001283

ABSTRACT: The Internet of Drones (IoD) allows for coordinated control of airspace for Unmanned Aerial Vehicles (UAVs), also known as drones. The decreasing costs of processors, sensors, and wireless connectivity have made it possible to use UAVs in many variety of military to civilian applications. While most applications utilizing the drones in the IoD have been real-time related, users are now interested in obtaining real-time services from drones that are tailored to a specific fly zone. This study develops a Sea Turtle Foraging Algorithm with Hybrid Deep Learning-based Intrusion Detection (STFA-HDLID) as a algorithm that recognizes and categorizes intrusions in the IoD environment. For this purpose, it is necessary to implement data pre-processing to standardize the input data via min-max normalization. Additionally, the feature selection process is also based on the STFA. Finally, a Deep Belief Network (DBN) with a Sparrow Search Optimization (SSO) algorithm is used for classification. A comprehensive experimental analysis is performed on a benchmark dataset to demonstrate the performance of the STFA-HDLID, which achieves maximum accuracy of 99.51% and 98.85% on the TON_IoT and UNSW-NB15 datasets, respectively, outperforming other techniques.

3.4 An Intrusion Detection System for Drone Swarming Utilizing Timed Probabilistic Automata:

https://www.mdpi.com/2504-446X/7/4/248

ABSTRACT: Unmanned aerial vehicles (UAVs), commonly known as drones, have found extensive applications across diverse sectors, such as agriculture, delivery, surveillance, and military. In recent times, drone swarming has emerged as a novel field of research, which involves multiple drones working in collaboration towards a shared objective. This innovation holds immense potential in transforming the way we undertake tasks, including military operations, environmental monitoring, and search and rescue missions. However, the emergence of drone swarms also brings new security challenges, as they can be susceptible to hacking and intrusion. To address these concerns, we propose utilizing a timed probabilistic automata (TPA)-based intrusion detection system (IDS) to model the normal behavior of drone swarms and identify any deviations that may indicate an intrusion. This IDS system is particularly efficient and adaptable in detecting different types of attacks in drone swarming.

3.5 Crystal Structure Optimization with Deep-Autoencoder-Based Intrusion Detection for Secure Internet of Drones Environment:

https://www.mdpi.com/2504-446X/6/10/297

ABSTRACT: Drone developments, especially small-sized drones, usher in novel trends and possibilities in various domains. Drones offer navigational inter-location services with the involvement of the Internet of Things (IoT). On the other hand, drone networks are highly prone to privacy and security risks owing to their strategy flaws. In order to achieve the desired efficiency, it is essential to create a secure network. The purpose of the current study is to have an overview of the privacy and security problems that recently impacted the Internet of Drones (IoD). An Intrusion Detection System (IDS) is an effective approach to determine the presence of intrusions in the IoD environment. The current study focuses on the design of Crystal Structure Optimization with Deep-Autoencoder-based Intrusion Detection (CSODAE-ID) for a secure IoD environment. The aim of the presented CSODAE-ID model is to identify the occurrences of intrusions in IoD environment. In the proposed CSODAE-ID model, a new Modified Deer Hunting Optimization-based Feature Selection (MDHO-FS) technique is applied to choose the feature subsets. At the same time, the Autoencoder (AE) method is employed for the classification of intrusions in the IoD environment. The CSO algorithm, inspired by the formation of crystal structures based on the lattice points, is employed at last for the hyperparameter-tuning process. To validate the enhanced performance of the proposed CSODAE-ID model, multiple simulation analyses were performed and the outcomes were assessed under distinct aspects. The comparative study outcomes demonstrate the superiority of the proposed CSODAE-ID model over the existing techniques.

4. SYSTEM ANALYSIS

4.1 EXISTING SYSTEM:

The existing system explores various facets of cloud computing and cybersecurity, addressing key areas such as cloud service selection, service ranking prediction, and encryption algorithm performance. Notable frameworks and methodologies include dimensionality reduction strategies for detecting Distributed Denial of Service (DDoS) attacks within cloud environments and intrusion detection approaches for unmanned aerial vehicles (UAVs). This includes the use of deep belief networks optimized by Particle Swarm Optimization (PSO) for UAV intrusion detection and machine learning models for autonomous malicious event detection in drone networks. Additionally, the system incorporates a machine-learning-enabled intrusion detection mechanism for cellular-connected UAV networks. These studies collectively contribute to enhancing cloud service efficiency, improving cybersecurity measures, and adapting to the evolving landscape of digital threats.

4.1.1 DISADVANTAGES OF EXISTING SYSTEM:

1. The existing system focuses on cloud service selection and DDoS detection but lacks comprehensive integration of advanced machine learning and deep learning algorithms, which limits its potential in identifying complex network intrusions.
2. The system primarily uses traditional methods and optimization techniques like PSO for intrusion detection, which may not fully capture the evolving and sophisticated nature of modern cyberattacks, especially in drone or UAV networks.
3. The approach to cloud security is fragmented, relying on separate methodologies for different domains (e.g., UAVs and DDoS attacks), which

might reduce the overall cohesion and scalability for large-scale implementations.

4. Although machine learning models are used for UAV intrusion detection, there is insufficient emphasis on handling imbalanced datasets, which are critical for ensuring accurate detection in environments with highly skewed attack and normal data distributions.

4.2 Proposed System:

The proposed system aims to enhance Intrusion Detection System (IDS) performance by leveraging advanced machine learning and deep learning algorithms to accurately detect intrusions in imbalanced network traffic datasets. We will utilize the CIC-IDS 2017 dataset, a widely recognized benchmark, to train and evaluate the models. The system will incorporate various algorithms, including XGBoost, Long Short-Term Memory (LSTM), minVGG (1D CNN), and AlexNet (1D CNN), each designed to capture different features and patterns in network traffic data. Additionally, the system will integrate a hybrid approach, combining Convolutional Neural Networks (CNN) and Bidirectional Long Short-Term Memory (BiLSTM) networks to enhance feature extraction and context modeling. To further improve accuracy and robustness, an ensemble method, specifically a Voting Classifier combining Random Forest (RF) and Decision Tree (DT), will be employed.

4.2.1 Advantages of proposed system:

1. The proposed system leverages advanced machine learning and deep learning techniques, including XGBoost, LSTM, and CNN, to enhance intrusion detection by capturing intricate patterns in network traffic, improving detection accuracy.

2. By incorporating a hybrid approach combining CNN and BiLSTM, the system significantly improves feature extraction and contextual modeling, enabling more precise identification of network intrusions with better adaptability to complex attack scenarios.
3. The use of an ensemble method like Voting Classifier, combining RF and DT, improves the robustness and accuracy of intrusion detection, allowing the system to make more reliable decisions, even in the presence of noisy or imbalanced data.
4. The CIC-IDS 2017 dataset provides a widely recognized benchmark, ensuring that the proposed system's performance can be evaluated against established standards, which facilitates better comparison and validation of its effectiveness in real-world scenarios.

4.3 FUNCTIONAL REQUIREMENTS

1. Data Collection
2. Data Pre-processing
3. Training and Testing
4. Modelling
5. Predicting

4.4 NON FUNCTIONAL REQUIREMENTS

Scalability

The system must be able to handle increasing volumes of network traffic data without significant performance degradation, ensuring it can adapt to growing network sizes and evolving threat landscapes in future applications.

Reliability

The system should operate consistently under various conditions, ensuring that intrusion detection processes run smoothly with minimal downtime, even during peak network traffic periods, to maintain uninterrupted protection.

Security

The system must ensure that sensitive data, including network traffic and detection results, are securely stored and transmitted, adhering to encryption standards and protecting against unauthorized access or tampering throughout the intrusion detection process.

Usability

The user interface should be intuitive and easy to navigate for security personnel, enabling them to quickly interpret alerts, manage system configurations, and make informed decisions without requiring extensive technical expertise.

Maintainability

The system should be designed with modular components that can be easily updated, modified, or replaced without disrupting overall functionality, allowing for efficient bug fixes, feature enhancements, and integration of new techniques as needed.

5. SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE:

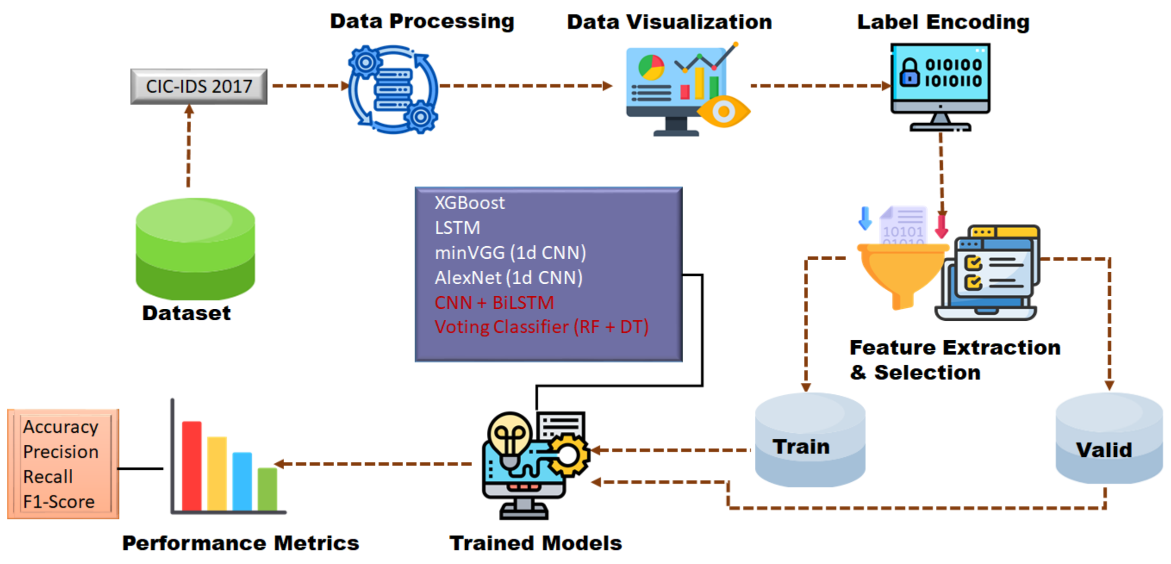


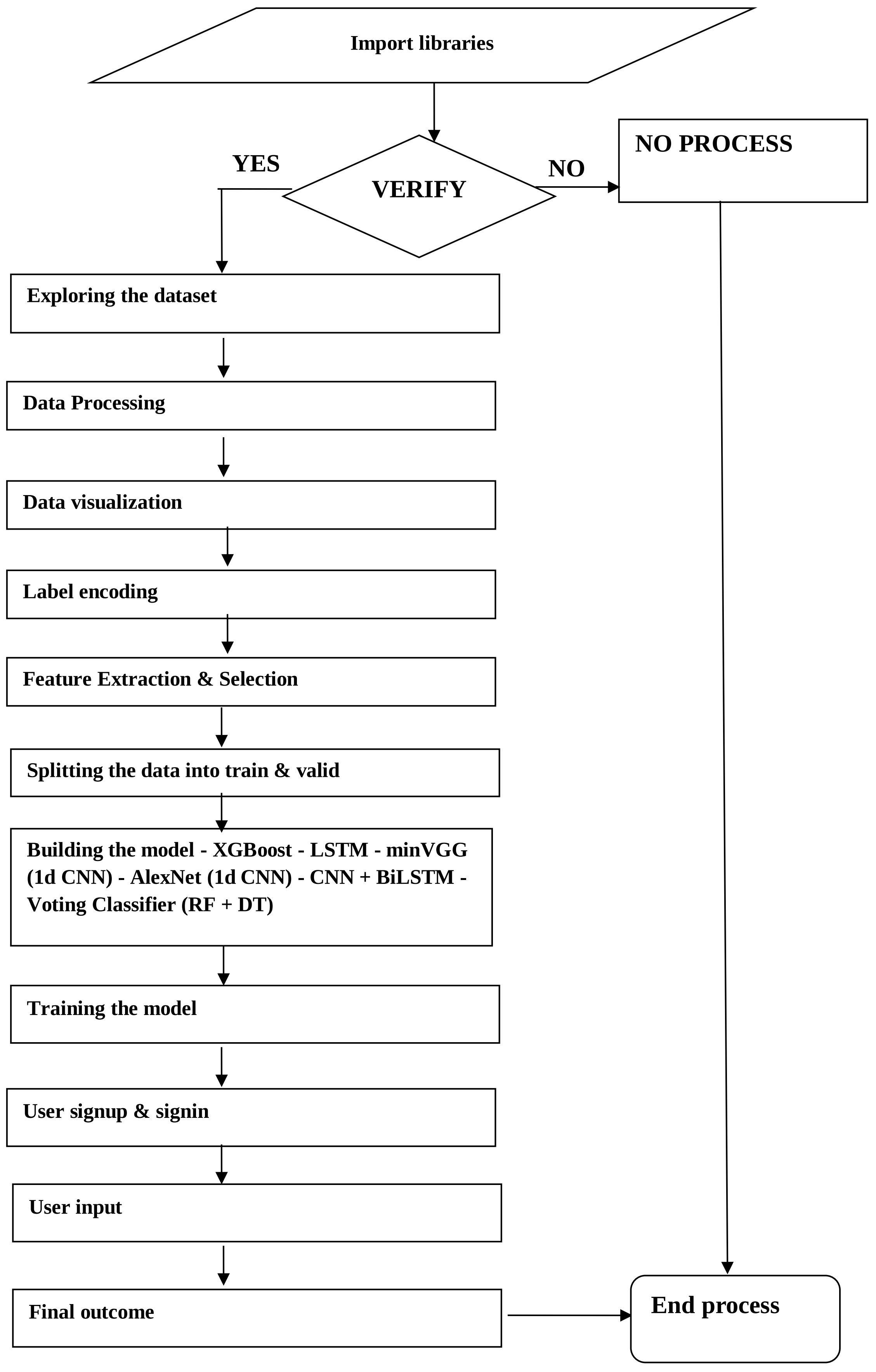
Fig.5.1.1 System architecture

DATA FLOW DIAGRAM:

A Data Flow Diagram (DFD) is a visual representation that illustrates the flow of data within a system, showing how data is processed, stored, and transferred between various components. In the context of an Intrusion Detection System (IDS) project, a DFD can depict how network traffic data flows through the system, from data input to detection and response. It shows how incoming network traffic is processed, how malicious activity is identified, and how alerts or actions are triggered. The DFD helps in understanding the interaction between different system modules such as data preprocessing, feature extraction, intrusion detection, and decision-making, ensuring clarity and smooth operation of the system. It provides an overview of the system's structure and how different functions communicate with one another.

Goals of DFD

- The DFD helps visualize the flow of network traffic data through the IDS, ensuring clarity in how data is collected, processed, and analyzed to identify potential intrusions in the system.
- It aids in identifying key system components, such as data preprocessing, feature extraction, and detection processes, ensuring that each part of the system interacts seamlessly for efficient intrusion detection.
- By mapping data movement, the DFD enables the identification of bottlenecks or vulnerabilities within the IDS, helping optimize performance and enhance the system’s overall detection capabilities.
- The DFD simplifies communication between different parts of the IDS, providing an accessible overview of the system’s architecture for troubleshooting, improvement, and future system upgrades to address emerging threats.


Import libraries
VERIFY
YES
NO
NO PROCESS
Exploring the dataset
Data Processing
Data visualization
Label encoding
Feature Extraction & Selection
Splitting the data into train & valid
Building the model - XGBoost - LSTM - minVGG (1d CNN) - AlexNet (1d CNN) - CNN + BiLSTM - Voting Classifier (RF + DT)
Training the model
User signup & signin
User input
Final outcome
End process

5.2 UML DIAGRAMS

Unified Modeling Language (UML) is a standardized visual language used to model the design and structure of a system. In the context of an Intrusion Detection System (IDS) project, UML helps represent the system's components and their interactions. It provides various diagrams to model different aspects of the system, such as use case diagrams to depict user interactions, class diagrams for defining system structure, and sequence diagrams to visualize data flow and system processes over time. UML assists in mapping out the IDS's architecture, from data collection to intrusion detection, ensuring that all system elements work together cohesively. By using UML, the project's design is clearly communicated, helping in system analysis, design, and future enhancements.

Goals of UML

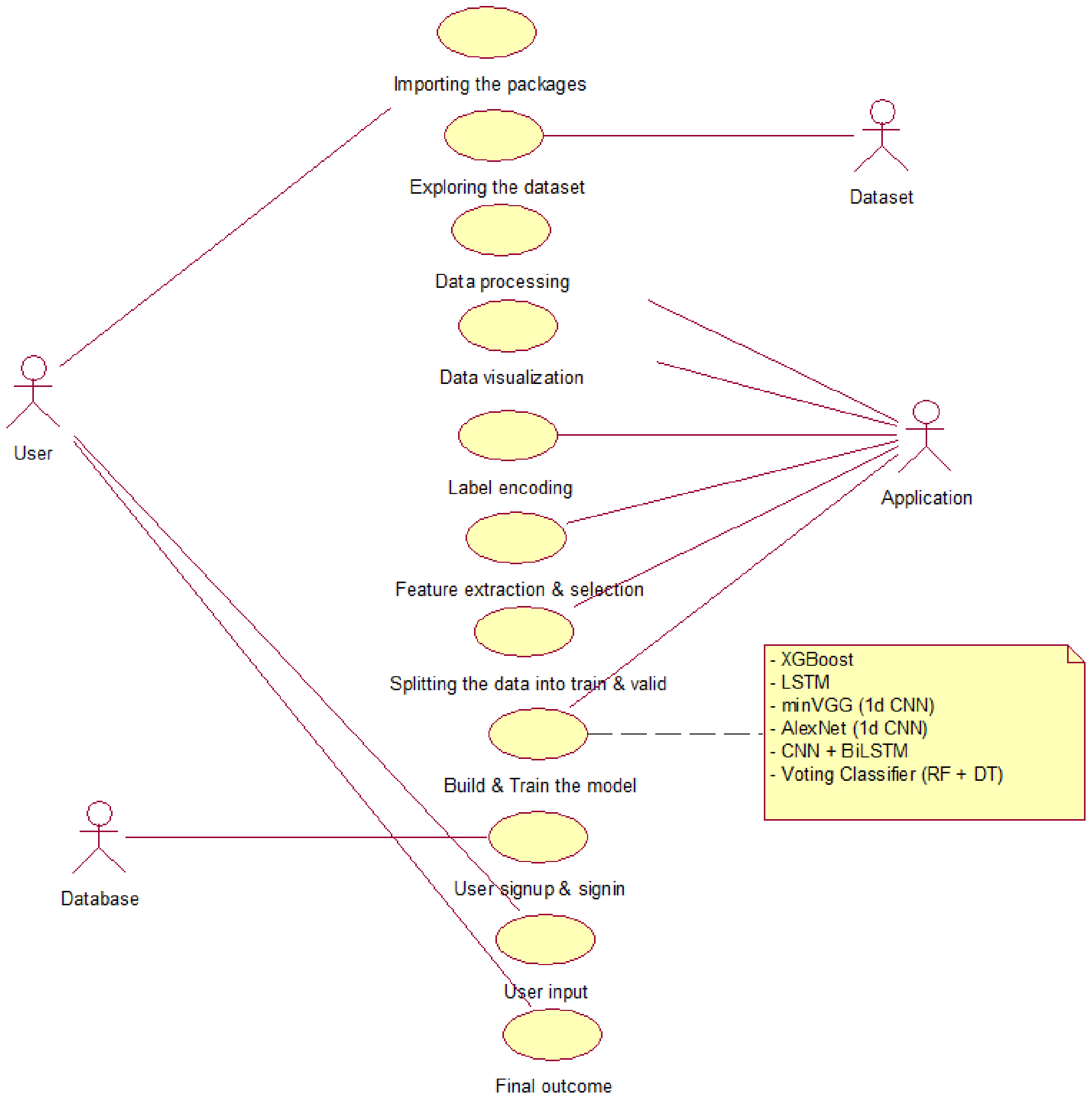
UML helps in visually representing the structure and components of the Intrusion Detection System (IDS), ensuring a clear understanding of how various system elements, such as detection modules and data sources, interact and function together.

It provides a standardized way to model system behavior, allowing for accurate representation of how the IDS processes network traffic, identifies intrusions, and responds to potential threats, enhancing system design and implementation.

UML diagrams serve as a communication tool, enabling seamless interaction between different team members, clarifying the system's functionalities and roles to ensure alignment during development and integration phases.

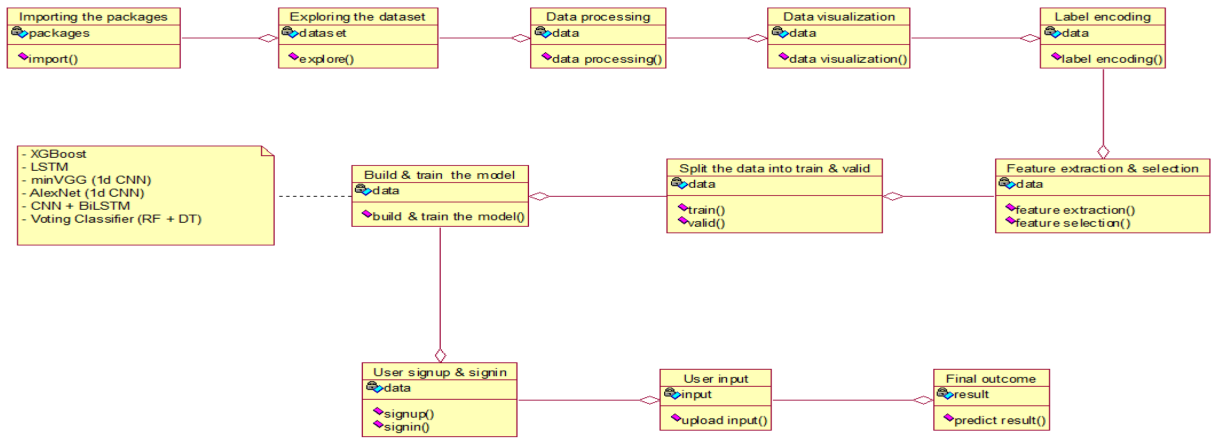
Case Diagram

A Use Case Diagram in the project represents interactions between actors and the system. Actors, such as users and the IDS, are linked to use cases like user signup, data processing, and model training.



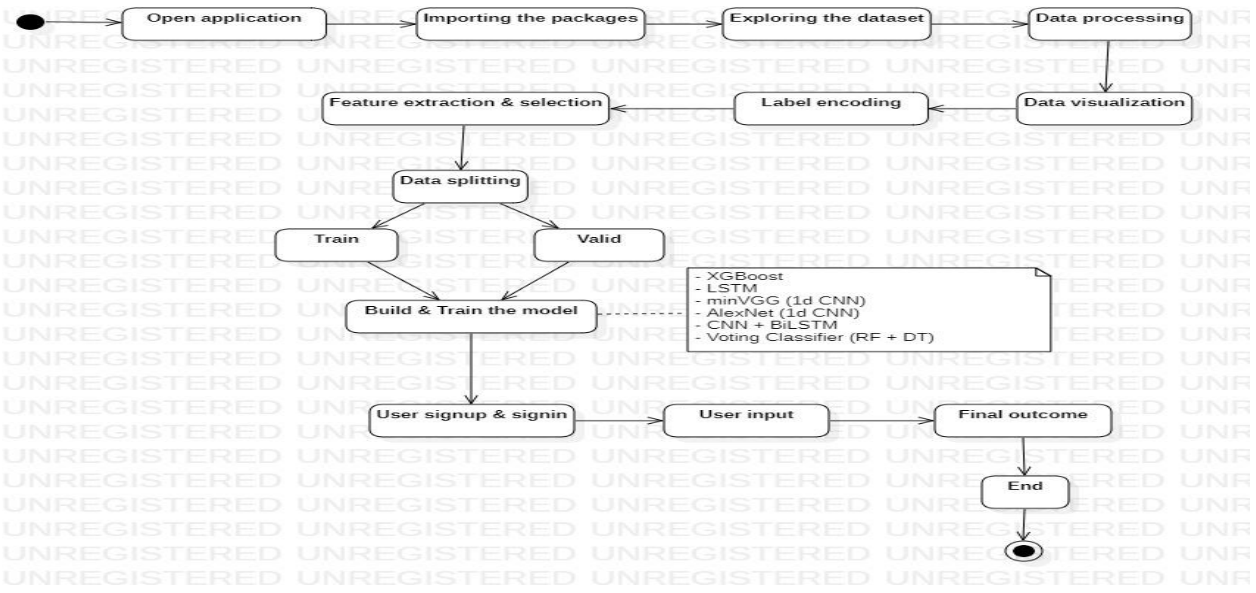
Class Diagram

A Class Diagram models the system's structure by defining classes such as User and Model. Attributes represent properties like username or dataset, while operations describe actions like training or signing in.



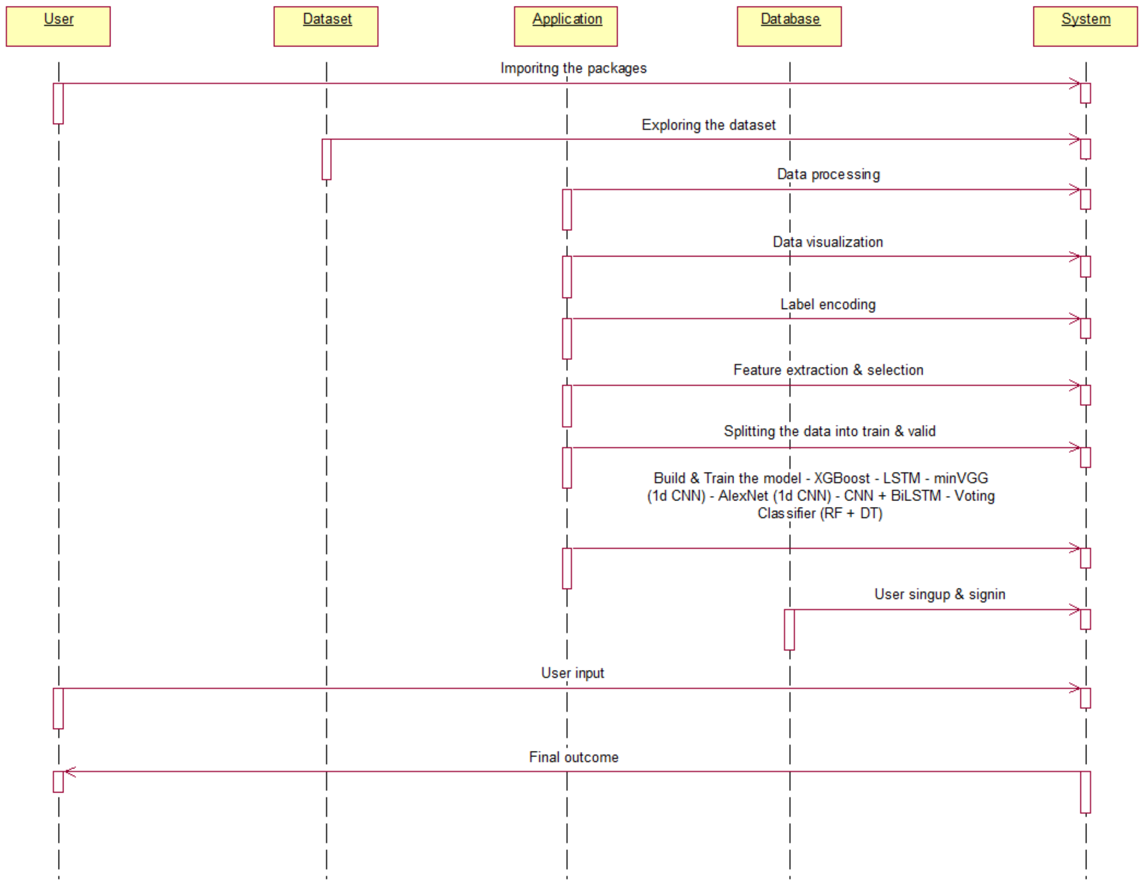
Activity Diagram

The Activity Diagram captures system workflows. Actions like data processing, model training, and user input are linked by decisions such as validation checks.



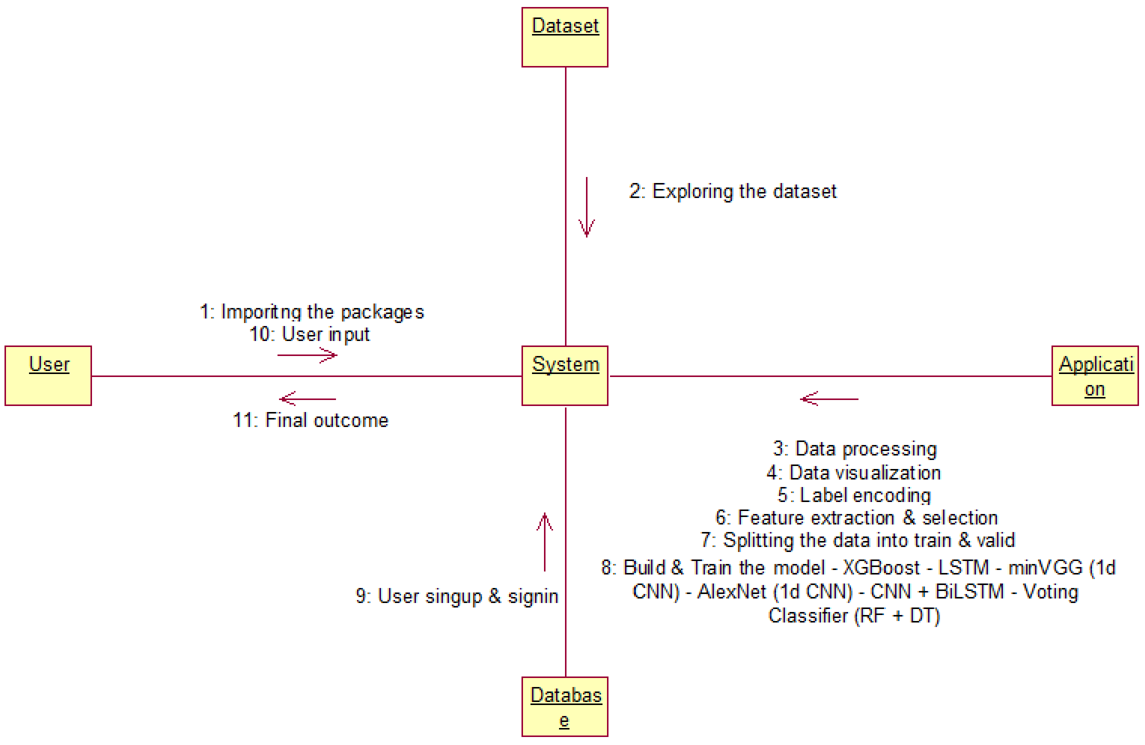
Sequence Diagram

The Sequence Diagram outlines the sequence of messages between objects. Objects such as user and system interact with messages to execute tasks, including user signup or model training. The flow of these messages determines task execution order.


User
Dataset
Application
Database
System
Imporitng the packages
Exploring the dataset
Data processing
Data visualization
Label encoding
Feature extraction & selection
Splitting the data into train & valid
Build & Train the model - XGBoost - LSTM - minVGG (1d CNN) - AlexNet (1d CNN) - CNN + BiLSTM - Voting Classifier (RF + DT)
User singup & signin
User input
Final outcome

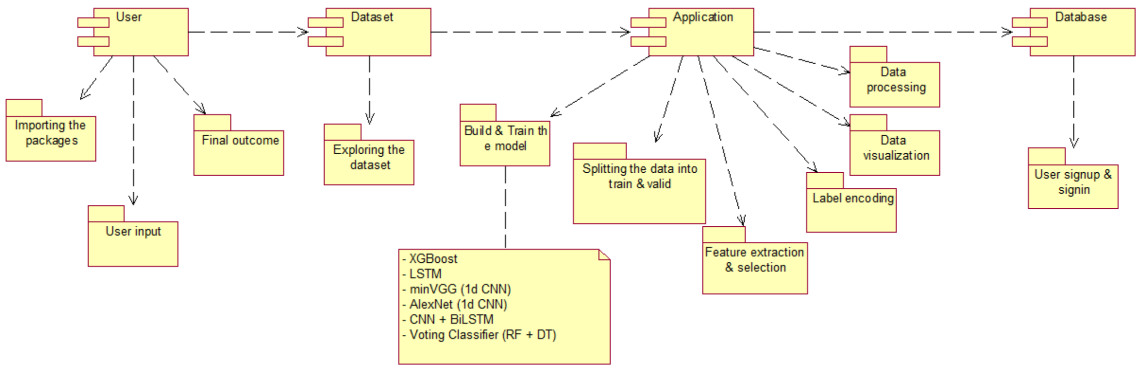
Collaboration Diagram

A Collaboration Diagram focuses on object interactions. Objects like User, Model, and IDS communicate through link messages to carry out operations like data processing or prediction. These links depict how objects work together to achieve system goals.


Dataset
2: Exploring the dataset
1: Imporitng the packages
10: User input
User
System
Applicati on
11: Final outcome
3: Data processing
4: Data visualization
5: Label encoding
6: Feature extraction & selection
7: Splitting the data into train & valid
8: Build & Train the model - XGBoost - LSTM - minVGG (1d CNN) - AlexNet (1d CNN) - CNN + BiLSTM - Voting Classifier (RF + DT)
9: User singup & signin
Databas e

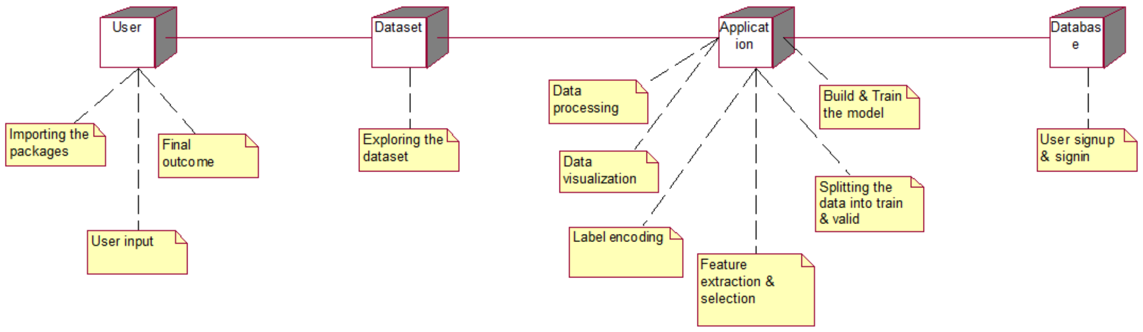
Component Diagram

A Component Diagram illustrates system components like User Interface, Data Processing Module, and Machine Learning Module. Packages group related components, while dependencies show how different components interact, ensuring the smooth functioning of the IDS.



Deployment Diagram

A Deployment Diagram shows physical deployment, where nodes represent servers or devices hosting components like the IDS. Connectors illustrate communication between these nodes, and anchor notes clarify specific configurations, ensuring the system’s proper setup and execution.



6. IMPLEMENTATION

MODULES:

- **Data loading:** using this module we are going to import the dataset.
- **Data Processing:** Data processing involves removing duplicate data and performing drop cleaning to handle missing values or irrelevant entries. These steps ensure the dataset is clean and ready for analysis, improving the quality and accuracy of model training.
- **Data Visualization:** Data visualization helps in understanding the patterns and distributions within the dataset. Techniques like histograms, scatter plots, and box plots are used to represent data trends, correlations, and anomalies, aiding in insightful decision-making during model development.
- **Label Encoding:** Label encoding converts categorical string data into numerical format, making it suitable for machine learning models. It replaces each unique category with an integer, ensuring that the model can process the data effectively without losing information.
- **Feature Extraction:** Feature extraction selects the target variable (y) and input features (X) from the dataset. This process isolates relevant features, such as traffic characteristics, to be used in the model, ensuring the algorithm has appropriate inputs for detection.
- **Feature Selection:** Feature selection identifies and selects the most significant features that contribute to predicting network intrusions. By removing irrelevant or redundant features, it optimizes model performance, reducing complexity and improving the accuracy of intrusion detection.

- **Splitting data into train & valid:** using this module data will be divided into train & valid
- **Model generation:** Model building - XGBoost - LSTM - minVGG (1d CNN) - AlexNet (1d CNN) - CNN + BiLSTM - Voting Classifier (RF + DT). Performance evaluation metrics for each algorithm is calculated.
- **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

Extension:

We expanded upon the original study by integrating ensemble techniques like Voting Classifier and CNN+BiLSTM, significantly improving attack prediction accuracy. Additionally, we created a Flask-based front end for user testing, which includes user authentication to enhance security.

Advantages:

1. Voting Classifier combines multiple model predictions, enhancing accuracy and robustness in detecting cyber attacks through diverse input sources.
2. CNN+BiLSTM architecture captures spatial and temporal features, improving the model's ability to analyze complex attack patterns effectively.
3. The Flask interface provides an easy-to-navigate platform for users, facilitating real-time testing and evaluation of the model’s performance.
4. User authentication ensures secure access to the system, protecting sensitive data and maintaining the integrity of the testing environment.

Algorithms:

XGBoost:

XGBoost is a gradient boosting algorithm known for its efficiency in classification tasks. In the project, it is used to predict network intrusions by building an ensemble of weak decision trees, improving model accuracy and handling imbalanced data, making it effective for intrusion detection in network traffic.

LSTM:

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) that excels in processing sequential data. In the project, LSTM is applied to network traffic data to recognize patterns and sequences indicative of malicious activity, ensuring accurate intrusion detection over time.

minVGG (1d CNN):

minVGG, a 1D Convolutional Neural Network (CNN), processes sequential data to identify patterns. In the project, it is utilized to detect network intrusions by extracting features from the traffic data, focusing on local patterns within the input sequences for improved detection accuracy.

AlexNet (1d CNN):

AlexNet, a deep CNN, is adapted to 1D input for intrusion detection. In this project, it is used to classify network traffic data by extracting complex features through multiple convolutional layers, aiding in distinguishing between normal and malicious network activities.

CNN + BiLSTM:

The combination of CNN and Bidirectional Long Short-Term Memory (BiLSTM) networks enhances intrusion detection by leveraging CNN's feature extraction capability and BiLSTM's ability to capture both past and future dependencies in the sequential traffic data, improving model robustness.

Voting Classifier (RF + DT):

The Voting Classifier is an ensemble method that combines the predictions of Random Forest (RF) and Decision Tree (DT) classifiers. In the project, it improves the accuracy and reliability of the intrusion detection system by aggregating the strengths of both models, addressing class imbalance effectively.

6.2 SAMPLE CODE:

```
import numpy as np

import pandas as pd

import warnings

import matplotlib.pyplot as plt

import seaborn as sns

import tensorflow as tf

from tensorflow.keras import regularizers

import xgboost as xgb

from sklearn.decomposition import PCA

from sklearn import tree

from sklearn.naive_bayes import GaussianNB

from sklearn.linear_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import RobustScaler

from sklearn.ensemble import RandomForestClassifier,
RandomForestRegressor

from sklearn.model_selection import train_test_split

from sklearn import svm

from sklearn import metrics

pd.set_option('display.max_columns',None)

warnings.filterwarnings('ignore')

%matplotlib inline

# Read Train and Test dataset

data_train = pd.read_csv("../input/nslkdd/KDDTrain+.txt")

# Check data

data_train.head()

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_at
tempted','num_root','num_file_creations'

,'num_shells','num_access_files','num_outbound_cmds','is_host_login','
is_guest_login','count','srv_count','serror_rate'

,'srv_serror_rate','rerror_rate','srv_rerror_rate','same_srv_rate','di
ff_srv_rate','srv_diff_host_rate','dst_host_count','dst_host_srv_count
'

,'dst_host_same_srv_rate','dst_host_diff_srv_rate','dst_host_same_src_
port_rate','dst_host_srv_diff_host_rate','dst_host_serror_rate'

,'dst_host_srv_serror_rate','dst_host_rerror_rate','dst_host_srv_rerro
r_rate','outcome','level'])

# Assign name for columns

data_train.columns = columns

data_train.head()

data_train.info()
data_train.describe().style.background_gradient(cmap='Blues').set_prop
erties(**{'font-family':'Segoe UI'})

data_train.loc[data_train['outcome'] == "normal", "outcome"] =
'normal'

data_train.loc[data_train['outcome'] != 'normal', "outcome"] =
'attack'

def pie_plot(df, cols_list, rows, cols):

    fig, axes = plt.subplots(rows, cols)

    for ax, col in zip(axes.ravel(), cols_list):

        df[col].value_counts().plot(ax=ax, kind='pie', figsize=(15,
15), fontsize=10, autopct='%1.0f%%')

        ax.set_title(str(col), fontsize = 12)

    plt.show()

pie_plot(data_train, ['protocol_type', 'outcome'], 1, 2)

def Scaling(df_num, cols):

    std_scaler = RobustScaler()

    std_scaler_temp = std_scaler.fit_transform(df_num)

    std_df = pd.DataFrame(std_scaler_temp, columns =cols)

    return std_df

cat_cols = ['is_host_login','protocol_type','service','flag','land',
'logged_in','is_guest_login', 'level', 'outcome']

def preprocess(dataframe):

    df_num = dataframe.drop(cat_cols, axis=1)

    num_cols = df_num.columns

    scaled_df = Scaling(df_num, num_cols)


    dataframe.drop(labels=num_cols, axis="columns", inplace=True)

    dataframe[num_cols] = scaled_df[num_cols]


    dataframe.loc[dataframe['outcome'] == "normal", "outcome"] = 0
    dataframe.loc[dataframe['outcome'] != 0, "outcome"] = 1


    dataframe = pd.get_dummies(dataframe, columns = ['protocol_type', 
'service', 'flag'])

    return dataframe

scaled_train = preprocess(data_train)

x = scaled_train.drop(['outcome', 'level'] , axis = 1).values

y = scaled_train['outcome'].values

y_reg = scaled_train['level'].values


pca = PCA(n_components=20)

pca = pca.fit(x)

x_reduced = pca.transform(x)

print("Number of original features is {} and of reduced features is 
{}".format(x.shape[1], x_reduced.shape[1]))


y = y.astype('int')

x_train, x_test, y_train, y_test = train_test_split(x, y, 
test_size=0.2, random_state=42)

x_train_reduced, x_test_reduced, y_train_reduced, y_test_reduced = 
train_test_split(x_reduced, y, test_size=0.2, random_state=42)

x_train_reg, x_test_reg, y_train_reg, y_test_reg = train_test_split(x, 
y_reg, test_size=0.2, random_state=42)

kernal_evals = dict()

def evaluate_classification(model, name, X_train, X_test, y_train, 
y_test):

    train_accuracy = metrics.accuracy_score(y_train, 
model.predict(X_train))

    test_accuracy = metrics.accuracy_score(y_test, 
model.predict(X_test))
```

```
    train_precision = metrics.precision_score(y_train, 
model.predict(X_train))
    test_precision = metrics.precision_score(y_test, 
model.predict(X_test))


    train_recall = metrics.recall_score(y_train, 
model.predict(X_train))
    test_recall = metrics.recall_score(y_test, model.predict(X_test))


    kernal_evals[str(name)] = [train_accuracy, test_accuracy, 
train_precision, test_precision, train_recall, test_recall]
    print("Training Accuracy " + str(name) + " {}  Test Accuracy 
".format(train_accuracy*100) + str(name) + " 
{}".format(test_accuracy*100))
    print("Training Precesion " + str(name) + " {}  Test Precesion 
".format(train_precision*100) + str(name) + " 
{}".format(test_precision*100))
    print("Training Recall " + str(name) + " {}  Test Recall 
".format(train_recall*100) + str(name) + " 
{}".format(test_recall*100))


    actual = y_test
    predicted = model.predict(X_test)
    confusion_matrix = metrics.confusion_matrix(actual, predicted)
    cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = 
confusion_matrix, display_labels = ['normal', 'attack'])


    fig, ax = plt.subplots(figsize=(10,10))
    ax.grid(False)
    cm_display.plot(ax=ax)
lr = LogisticRegression().fit(x_train, y_train)
```

```
evaluate_classification(lr, "Logistic Regression", x_train, x_test,
y_train, y_test)

knn = KNeighborsClassifier(n_neighbors=20).fit(x_train, y_train)

evaluate_classification(knn, "KNeighborsClassifier", x_train, x_test,
y_train, y_test)

gnb = GaussianNB().fit(x_train, y_train)

evaluate_classification(gnb, "GaussianNB", x_train, x_test, y_train,
y_test)

lin_svc = svm.LinearSVC().fit(x_train, y_train)

evaluate_classification(lin_svc, "Linear SVC(LBasedImpl)", x_train,
x_test, y_train, y_test)

dt = DecisionTreeClassifier(max_depth=3).fit(x_train, y_train)

tdt = DecisionTreeClassifier().fit(x_train, y_train)

evaluate_classification(tdt, "DecisionTreeClassifier", x_train,
x_test, y_train, y_test)

def f_importances(coef, names, top=-1):

    imp = coef

    imp, names = zip(*sorted(list(zip(imp, names))))


    # Show all features

    if top == -1:

        top = len(names)


    plt.figure(figsize=(10,10))

    plt.barh(range(top), imp[::-1][0:top], align='center')

    plt.yticks(range(top), names[::-1][0:top])

    plt.title('feature importances for Decision Tree')

    plt.show()


features_names = data_train.drop(['outcome', 'level'] , axis = 1)
```

```
f_importances(abs(tdt.feature_importances_), features_names, top=18)

fig = plt.figure(figsize=(15,12))

tree.plot_tree(dt , filled=True)

rf = RandomForestClassifier().fit(x_train, y_train)

evaluate_classification(rf, "RandomForestClassifier", x_train, x_test, 
y_train, y_test)

f_importances(abs(rf.feature_importances_), features_names, top=18)

xg_r = xgb.XGBRegressor(objective ='reg:linear',n_estimators = 
20).fit(x_train_reg, y_train_reg)

name = "XGBOOST"

train_error = metrics.mean_squared_error(y_train_reg, 
xg_r.predict(x_train_reg), squared=False)

test_error = metrics.mean_squared_error(y_test_reg, 
xg_r.predict(x_test_reg), squared=False)

print("Training Error " + str(name) + " {}  Test error 
".format(train_error) + str(name) + " {}".format(test_error))

y_pred = xg_r.predict(x_test_reg)

df = pd.DataFrame({"Y_test": y_test_reg , "Y_pred" : y_pred})

plt.figure(figsize=(16,8))

plt.plot(df[:80])

plt.legend(['Actual' , 'Predicted'])

rrf = RandomForestClassifier().fit(x_train_reduced, y_train_reduced)

evaluate_classification(rrf, "PCA RandomForest", x_train_reduced, 
x_test_reduced, y_train_reduced, y_test_reduced)

model = tf.keras.Sequential([

    tf.keras.layers.Dense(units=64, activation='relu', 
input_shape=(x_train.shape[1:]),

                          kernel_regularizer=regularizers.L1L2(l1=1e-
5, l2=1e-4),

                          bias_regularizer=regularizers.L2(1e-4),

                          activity_regularizer=regularizers.L2(1e-5)),
    tf.keras.layers.Dropout(0.4),

    tf.keras.layers.Dense(units=128, activation='relu',

                          kernel_regularizer=regularizers.L1L2(l1=1e-
5, l2=1e-4),

                          bias_regularizer=regularizers.L2(1e-4),

                          activity_regularizer=regularizers.L2(1e-5)),

    tf.keras.layers.Dropout(0.4),

    tf.keras.layers.Dense(units=512, activation='relu',

                          kernel_regularizer=regularizers.L1L2(l1=1e-
5, l2=1e-4),

                          bias_regularizer=regularizers.L2(1e-4),

                          activity_regularizer=regularizers.L2(1e-5)),

    tf.keras.layers.Dropout(0.4),

    tf.keras.layers.Dense(units=128, activation='relu',

                          kernel_regularizer=regularizers.L1L2(l1=1e-
5, l2=1e-4),

                          bias_regularizer=regularizers.L2(1e-4),

                          activity_regularizer=regularizers.L2(1e-5)),

    tf.keras.layers.Dropout(0.4),

    tf.keras.layers.Dense(units=1, activation='sigmoid'),

])

model.compile(optimizer='adam',
loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
metrics=['accuracy'])

model.summary()

from keras.utils.vis_utils import plot_model

plot_model(model, to_file='model_plot.png', show_shapes=True,
show_layer_names=True)

history = model.fit(x_train, y_train, validation_data=(x_test,
y_test), epochs=10, verbose=1)
```

```
plt.plot(history.history['loss'], label='loss')

plt.plot(history.history['val_loss'], label='val_loss')

plt.xlabel('Epoch')

plt.ylabel('SCCE Loss')

plt.legend()

plt.grid(True)

plt.plot(history.history['accuracy'], label='accuracy')

plt.plot(history.history['val_accuracy'], label='val_accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend()

plt.grid(True)

keys = [key for key in kernal_evals.keys()]

values = [value for value in kernal_evals.values()]

fig, ax = plt.subplots(figsize=(20, 6))

ax.bar(np.arange(len(keys)) - 0.2, [value[0] for value in values],
color='darkred', width=0.25, align='center')

ax.bar(np.arange(len(keys)) + 0.2, [value[1] for value in values],
color='y', width=0.25, align='center')

ax.legend(["Training Accuracy", "Test Accuracy"])

ax.set_xticklabels(keys)

ax.set_xticks(np.arange(len(keys)))

plt.ylabel("Accuracy")

plt.show()

keys = [key for key in kernal_evals.keys()]

values = [value for value in kernal_evals.values()]

fig, ax = plt.subplots(figsize=(20, 6))

ax.bar(np.arange(len(keys)) - 0.2, [value[2] for value in values],
color='g', width=0.25, align='center')
```

```
ax.bar(np.arange(len(keys)) + 0.2, [value[3] for value in values],
color='b', width=0.25, align='center')

ax.legend(["Training Precesion", "Test Presision"])

ax.set_xticklabels(keys)

ax.set_xticks(np.arange(len(keys)))

plt.ylabel("Precesion")

plt.show()

keys = [key for key in kernal_evals.keys()]

values = [value for value in kernal_evals.values()]

fig, ax = plt.subplots(figsize=(20, 6))

ax.bar(np.arange(len(keys)) - 0.2, [value[2] for value in values],
color='g', width=0.25, align='center')

ax.bar(np.arange(len(keys)) + 0.2, [value[3] for value in values],
color='b', width=0.25, align='center')

ax.legend(["Training Recall", "Test Recall"])

ax.set_xticklabels(keys)

ax.set_xticks(np.arange(len(keys)))

plt.ylabel("Recall")

plt.show()
```

7. SOFTWARE ENVIRONMENT

Cyber Security:

Cyber security refers to the body of technologies, processes, and practices designed to protect networks, devices, programs, and data from attack, damage, or unauthorized access. Cyber security may also be referred to as information technology security.

Types of Cyber Threats

The most common types of cyber threats include:

- Hacking
- Social Engineering
- Physical Security Attacks
- Viruses and Malware (malicious software)
- Ransomware attacks

Challenges of Cyber Security

For effective cyber security, an organization needs to coordinate its efforts throughout its entire information system. Elements of cyber encompass all of the following:

Network security: The process of protecting the network from unwanted users, attacks and intrusions.

Application security: Apps require constant updates and testing to ensure these programs are secure from attacks.

Endpoint security: Remote access is a necessary part of business, but can also be a weak point for data. Endpoint security is the process of protecting remote access to a company's network.

Data security: Inside of networks and applications is data. Protecting company and customer information is a separate layer of security.

Identity management: Essentially, this is a process of understanding the access every individual has in an organization.

Database and infrastructure security: Everything in a network involves databases and physical equipment. Protecting these devices is equally important.

Cloud security: Many files are in digital environments or "the cloud". Protecting data in a 100% online environment presents a large amount of challenges.

Security for mobile devices: Cell phones and tablets involve virtually every type of security challenge in and of themselves.

Disaster recovery/business continuity planning: In the event of a security breach, natural disaster or other event data must be protected and business must go on. For this, you'll need a plan. End-user education: Users may be employees accessing the network or customers logging on to a company app. Educating good habits (password changes and having a strong password, 2-factor authentication, etc.) is an important part of cybersecurity.

The most difficult challenge in cyber security is the ever-evolving nature of security risks themselves. Traditionally, organizations and the government have focused most of their cyber security resources on perimeter security to protect only their most crucial system components and defend against known threats. Today, this approach is insufficient, as the threats advance and change more quickly than organizations can keep up with. As a result, advisory organizations promote more proactive and adaptive approaches to cyber security. Similarly, the National Institute of Standards and Technology (NIST) issued guidelines in its risk assessment framework that recommend a shift toward continuous monitoring and real-time assessments, a data-focused approach to security as opposed to the traditional perimeter-based model.

Types of cybersecurity

Cybersecurity encompasses various specialized areas that collectively protect different aspects of digital environments. Let’s explore the types of cybersecurity that address specific vulnerabilities and threats.

Application security

Application security protects applications both in cloud environments and on-premises. Applications may need patching to meet current security standards. Apps may need code alterations to neutralize exploit kits. Access management tools also add an extra level of protection to application security, admitting only authorized users.

Network security

Network security protects the whole network from external threats. Robust network security involves complete awareness of all connected devices and endpoints. Network architecture should include protections like segmentation and encryption. Employing network security tools is crucial for a secure user experience.

Cloud security

Cloud security protects SaaS, IaaS, and PaaS services. These services operate at arm's length from on-premises networks. They are generally maintained by third-party vendors. Companies using cloud security services must encrypt data residing on cloud servers. They must protect data in transit between on-premises networks and the cloud. Effective cloud security measures ensure secure use of cloud applications.

Information security

Information security protects high-value data. This sensitive data could be on company networks, remote work devices, or third-party storage services. Regulations like HIPAA or the GDPR framework may inform information security strategies. The goal is to lock down confidential data without compromising user access.

Human security

Human security involves training company staff, contractors, and – in some cases – customers to strengthen cybersecurity. Employees must know how to avoid social engineering attacks such as phishing. Remote devices must be used safely, with appropriate tools to guard against cyber attacks. Mobile cybersecurity measures may also cover encryption and secure email usage.

Disaster recovery

Disaster recovery seeks to restore critical applications and network resources in the event of a cyber attack. How does cybersecurity work in disaster recovery? It includes plans to detect and neutralize network intruders. Companies need plans to quarantine threats. They must set out ways to assess the damage and restore a secure working environment.

Critical infrastructure security

Critical infrastructure security protects network assets deemed critical by industry regulations and legislation. It does not apply to all companies. For example, military contractors must protect assets relevant to national security. Healthcare companies must have cyber security plans to guard confidential data.

Social engineering

Social engineering exploits human psychology to manipulate individuals into divulging confidential information or performing actions that compromise security. Tactics include phishing, pretexting, baiting, and tailgating. Training and awareness are key defenses against these manipulative attacks, which rely on trickery rather than technical vulnerabilities.

The function of cybersecurity

Cyber security has five core functions. These functions act together to protect business assets and include:

Identify – Companies must first identify network assets and their vulnerabilities. Organizations must develop the internal skills required to understand cybersecurity risks. This function includes risk assessment processes and creating a risk management strategy. Company structures must also change to reflect the central role of cyber security.

Protect – This cybersecurity function deals with the implementation of protective security solutions. These solutions may include antivirus software and anti-malware scanning. Identity and Access Management systems determine who can access critical assets. Information is protected by encryption and data security tools. Staff training is also central to network protection.

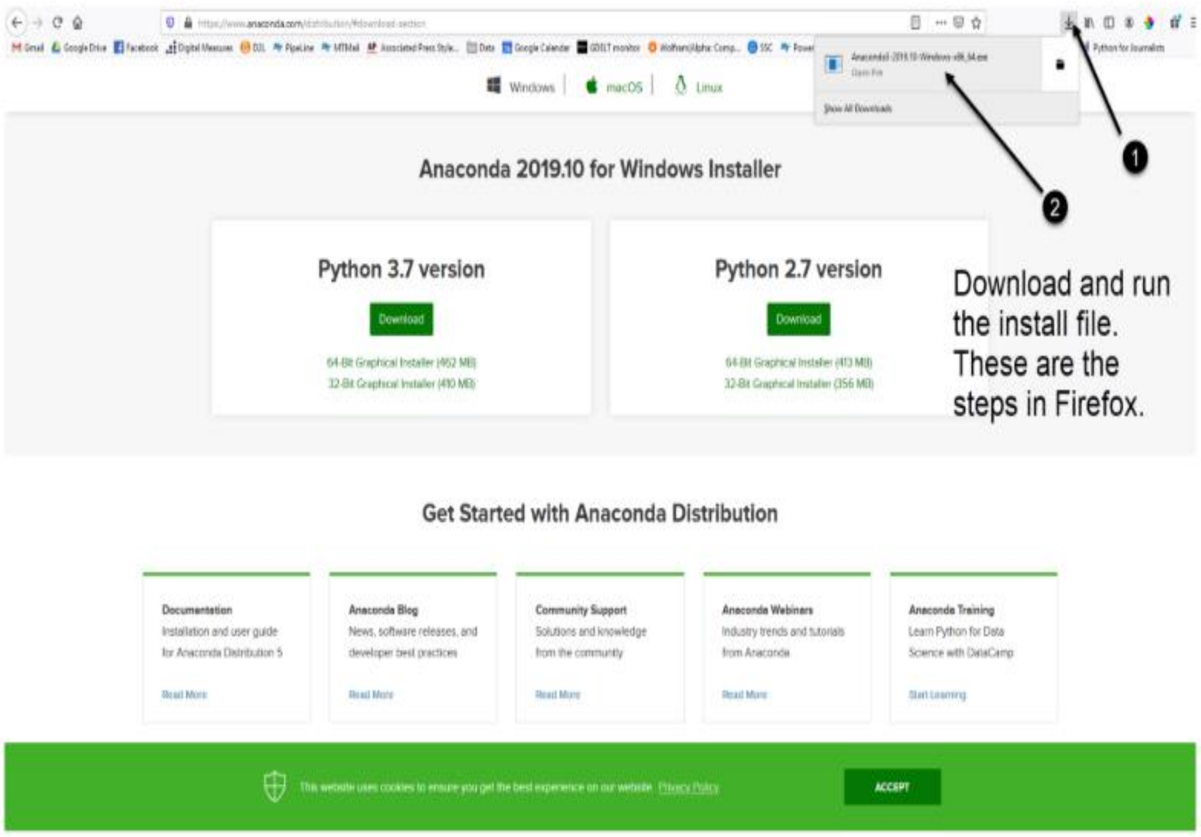
Detect – This cyber security function involves detecting and identifying malicious activity. Systems must detect cyber attacks as quickly as possible. They must provide information regarding the nature and extent of the threat, providing the means to respond. Continuous threat detection and regular security audits combine to provide maximum awareness.

Respond – Companies must have the tools to respond when cyber attacks occur. Security solutions should not only isolate malicious entities but also minimize the harm they cause. IT teams must assess the damage and initiate recovery procedures. All security stakeholders within the organization should communicate while responding. There should also be processes determining when threats are no longer active. Finally, responses feed into institutional learning. Constant learning helps to improve threat management in future incidents.

Recovery – This function of cyber security restores business functionality after cyber attacks. Companies should operate resilience plans documenting how to restore core applications and protect data. Network teams should have the resources to repair or replace damaged assets. Regularly test recovery processes to ensure smooth operation.

Python Anaconda Installation

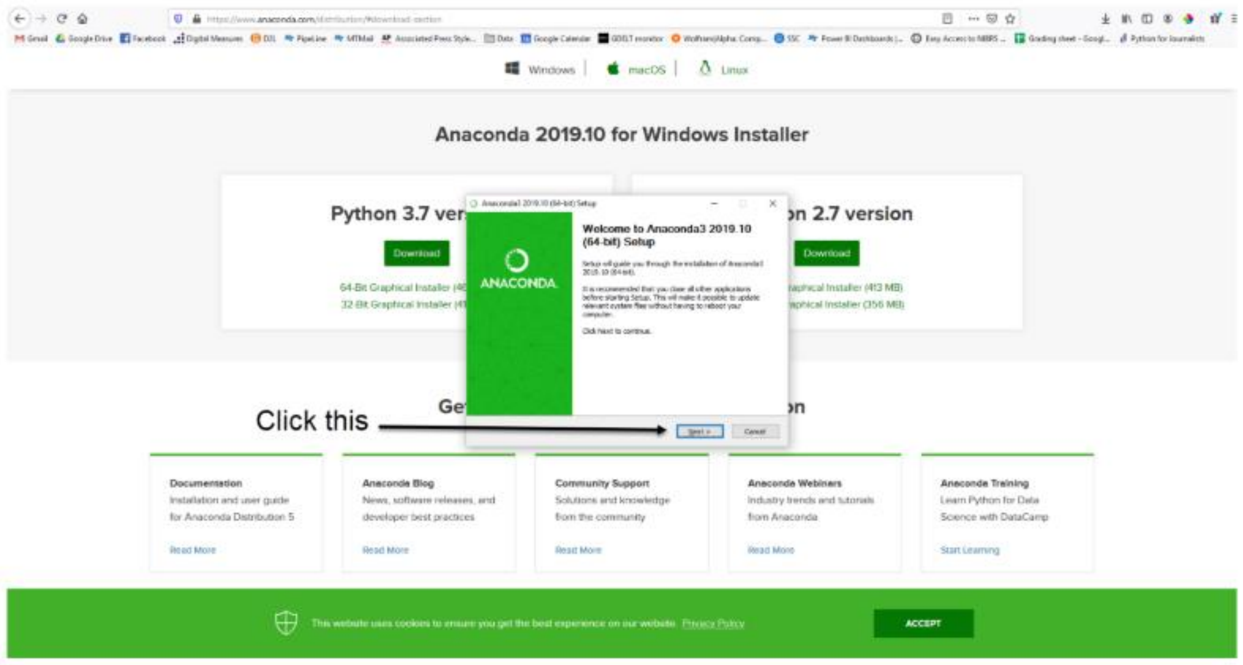
1. Go to this link and download Anaconda for Windows, Mac, or Linux: – Download anaconda


Windows
macOS
Linux
Anaconda 2019.10 for Windows Installer
Python 3.7 version
Download
64-Bit Graphical Installer (462 MB)
32-Bit Graphical Installer (410 MB)
Python 2.7 version
Download
64-Bit Graphical Installer (413 MB)
32-Bit Graphical Installer (356 MB)
1
2
Show All Downloads
Get Started with Anaconda Distribution
Documentation
Anaconda Blog
Community Support
Anaconda Webinars
Anaconda Training
ACCEPT

Download and run the install file. These are the steps in Firefox.

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.



Welcome to Anaconda3 2019.10 (64-bit) Setup

Setup will guide you through the installation of Anaconda3 2019.10 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Installation and user guide for Anaconda Distribution 5

News, software releases, and developer best practices

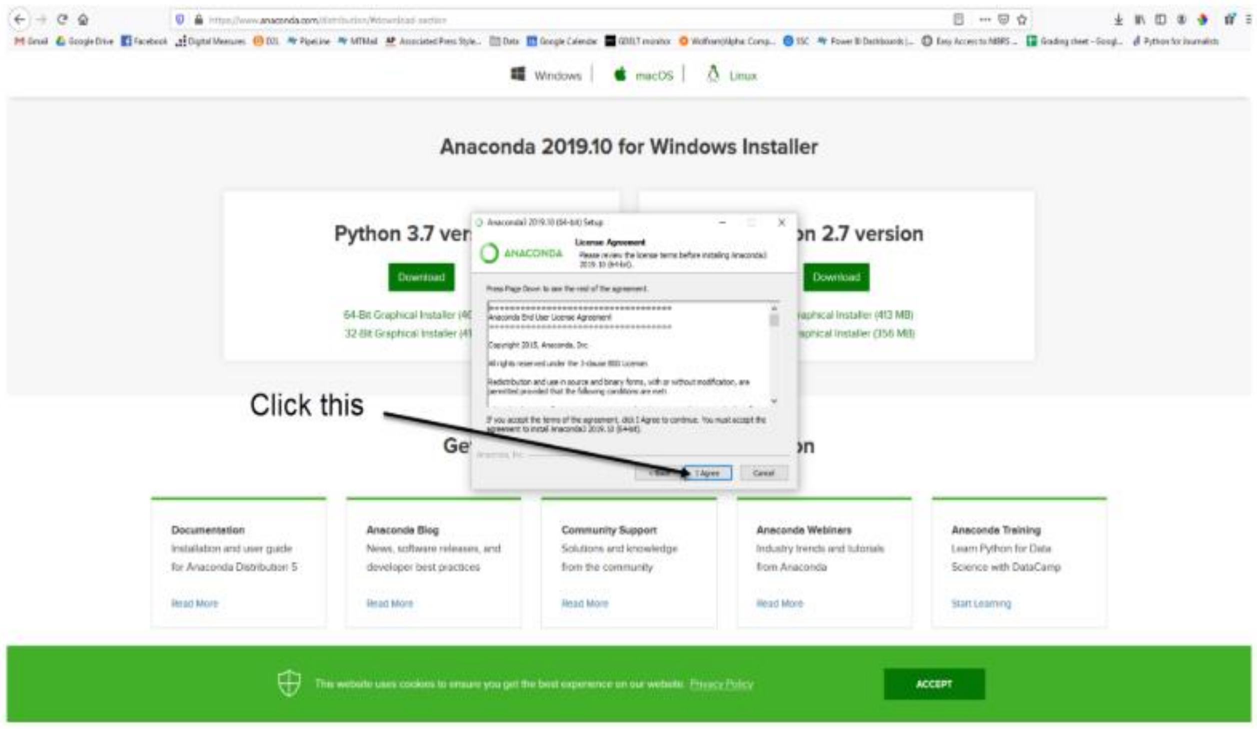
Solutions and knowledge from the community

Industry trends and tutorials from Anaconda

Learn Python for Data Science with DataCamp

This website uses cookies to ensure you get the best experience on our website. Privacy Policy

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



License Agreement

Please review the license terms before installing Anaconda3 2019.10 (64-bit).

Press Page Down to see the rest of the agreement.

```
***********************************
Anaconda End User License Agreement
***********************************

Copyright 2015, Anaconda, Inc.

All rights reserved under the 3-clause BSD License:

Redistribution and use in source and binary forms, with or without modification, are
permitted provided that the following conditions are met:
```

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Anaconda3 2019.10 (64-bit).

Anaconda, Inc.

Installation and user guide for Anaconda Distribution 5

News, software releases, and developer best practices

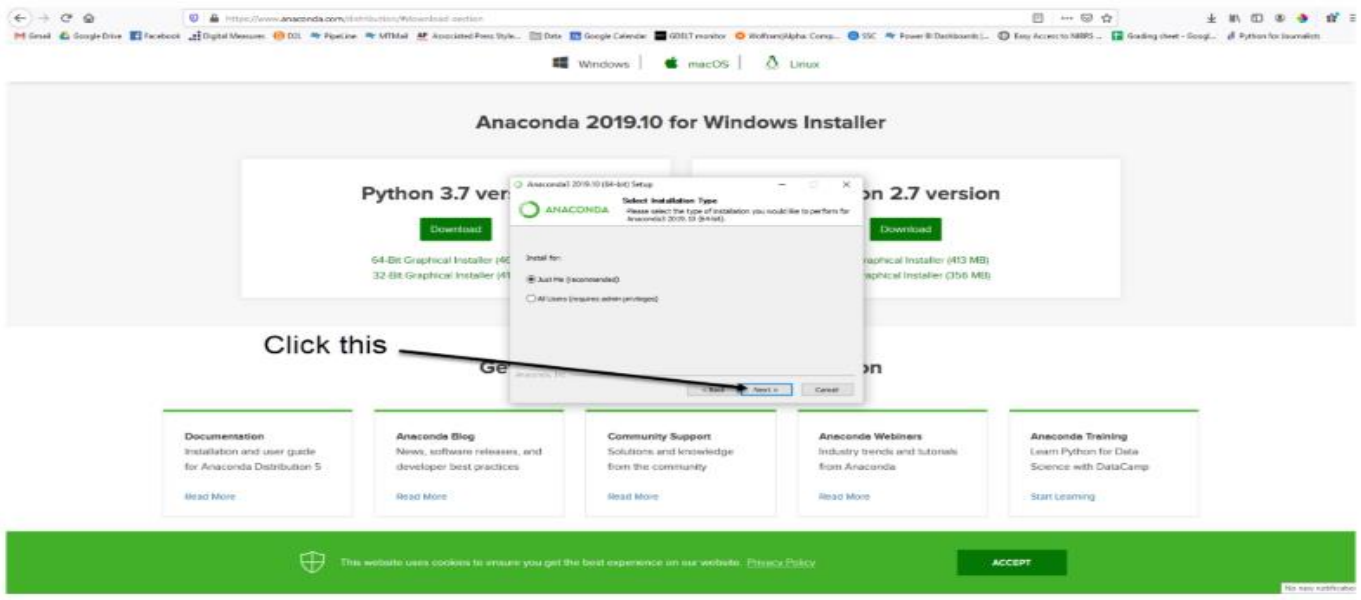
Solutions and knowledge from the community

Industry trends and tutorials from Anaconda

Learn Python for Data Science with DataCamp

This website uses cookies to ensure you get the best experience on our website. Privacy Policy

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.



Anaconda3 2019.10 (64-bit) Setup

Select Installation Type

Please select the type of installation you would like to perform for Anaconda3 2019.10 (64-bit).

Install for:

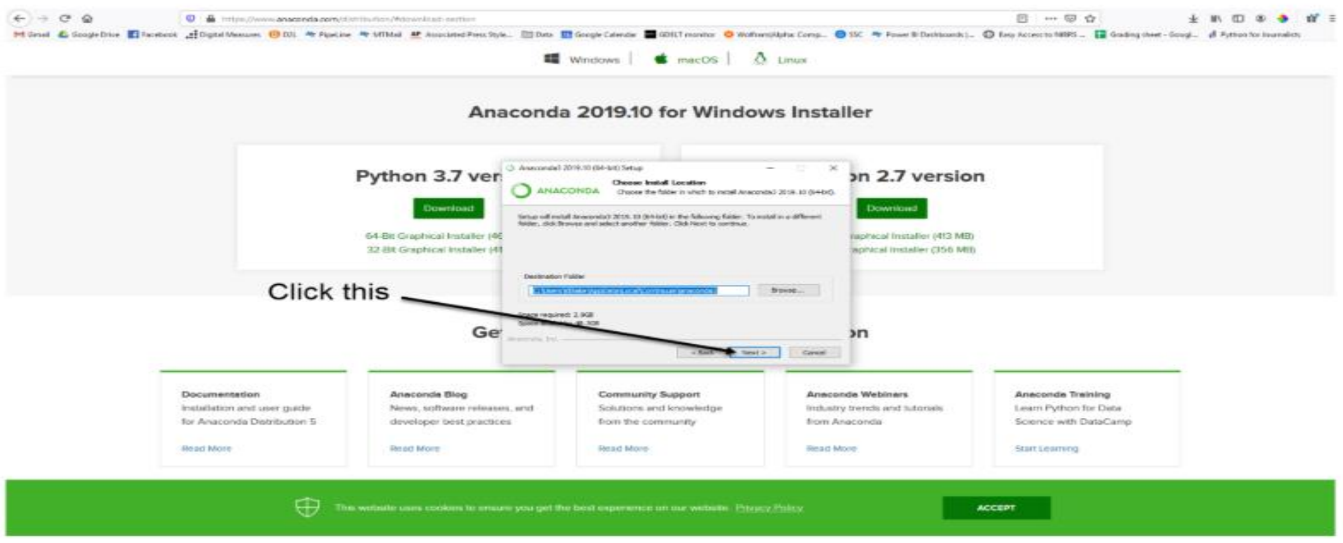
[✓ Just Me (recommended)]

All Users (requires admin privileges)

< Back Next > Cancel

This website uses cookies to ensure you get the best experience on our website. Privacy Policy

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



Anaconda3 2019.10 (64-bit) Setup

Choose Install Location

Choose the folder in which to install Anaconda3 2019.10 (64-bit).

Setup will install Anaconda3 2019.10 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder

[illegible]

Browse...

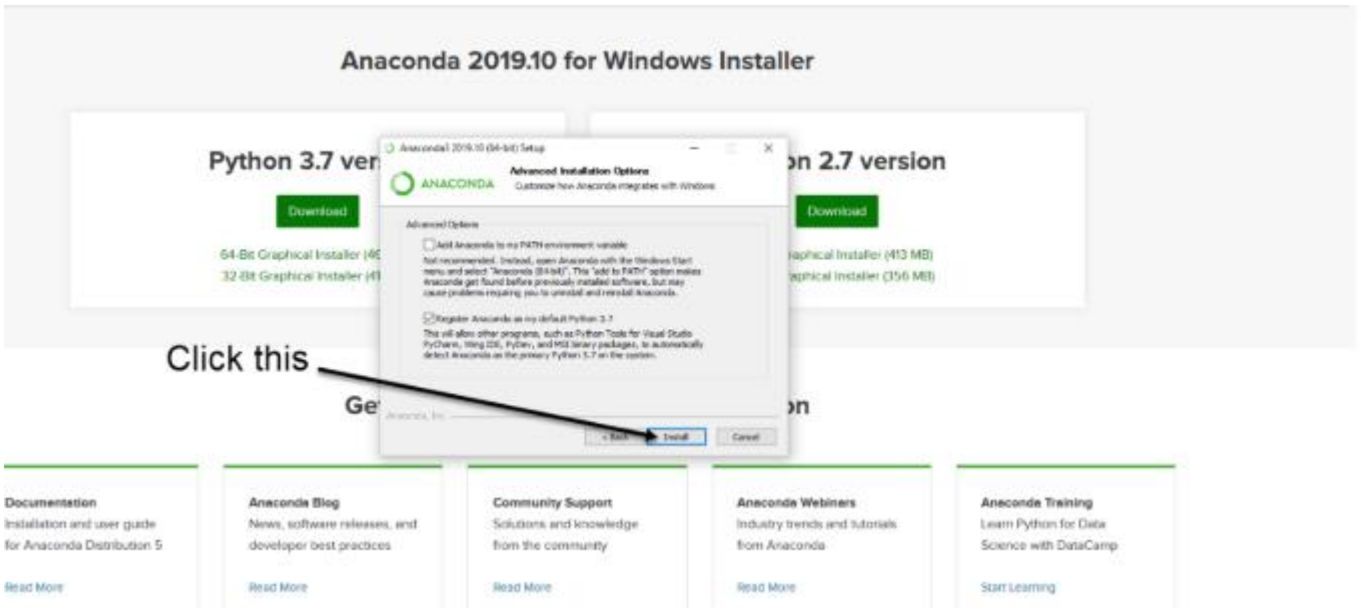
Space required: 2.9GB

Space available: [illegible]

< Back Next > Cancel

This website uses cookies to ensure you get the best experience on our website. Privacy Policy

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'.


Anaconda 2019.10 for Windows Installer
Python 3.7 ver
Download
64-Bit Graphical Installer (46
32-Bit Graphical Installer (41
on 2.7 version
Download
aphical Installer (413 MB)
aphical Installer (356 MB)
Click this
Ge
on
Documentation
Installation and user guide for Anaconda Distribution 5
Read More
Anaconda Blog
News, software releases, and developer best practices
Read More
Community Support
Solutions and knowledge from the community
Read More
Anaconda Webinars
Industry trends and tutorials from Anaconda
Read More
Anaconda Training
Learn Python for Data Science with DataCamp
Start Learning

Anaconda3 2019.10 (64-bit) Setup

Advanced Installation Options

Customize how Anaconda integrates with Windows

Advanced Options

[ ] Add Anaconda to my PATH environment variable

Not recommended. Instead, open Anaconda with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

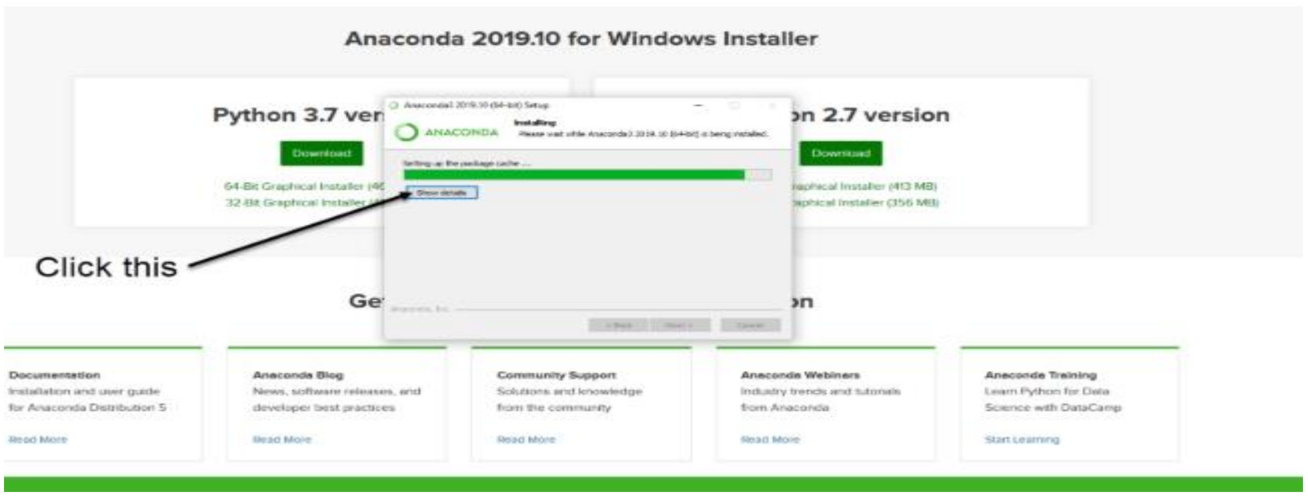
[✓] Register Anaconda as my default Python 3.7

This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.7 on the system.

Anaconda, Inc.

< Back | Install | Cancel

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


Anaconda 2019.10 for Windows Installer
Python 3.7 ver
Download
64-Bit Graphical Installer (46
32-Bit Graphical Installer (41
on 2.7 version
Download
aphical Installer (413 MB)
aphical Installer (356 MB)
Click this
Ge
on
Documentation
Installation and user guide for Anaconda Distribution 5
Read More
Anaconda Blog
News, software releases, and developer best practices
Read More
Community Support
Solutions and knowledge from the community
Read More
Anaconda Webinars
Industry trends and tutorials from Anaconda
Read More
Anaconda Training
Learn Python for Data Science with DataCamp
Start Learning

Anaconda3 2019.10 (64-bit) Setup

Installing

Please wait while Anaconda3 2019.10 (64-bit) is being installed.

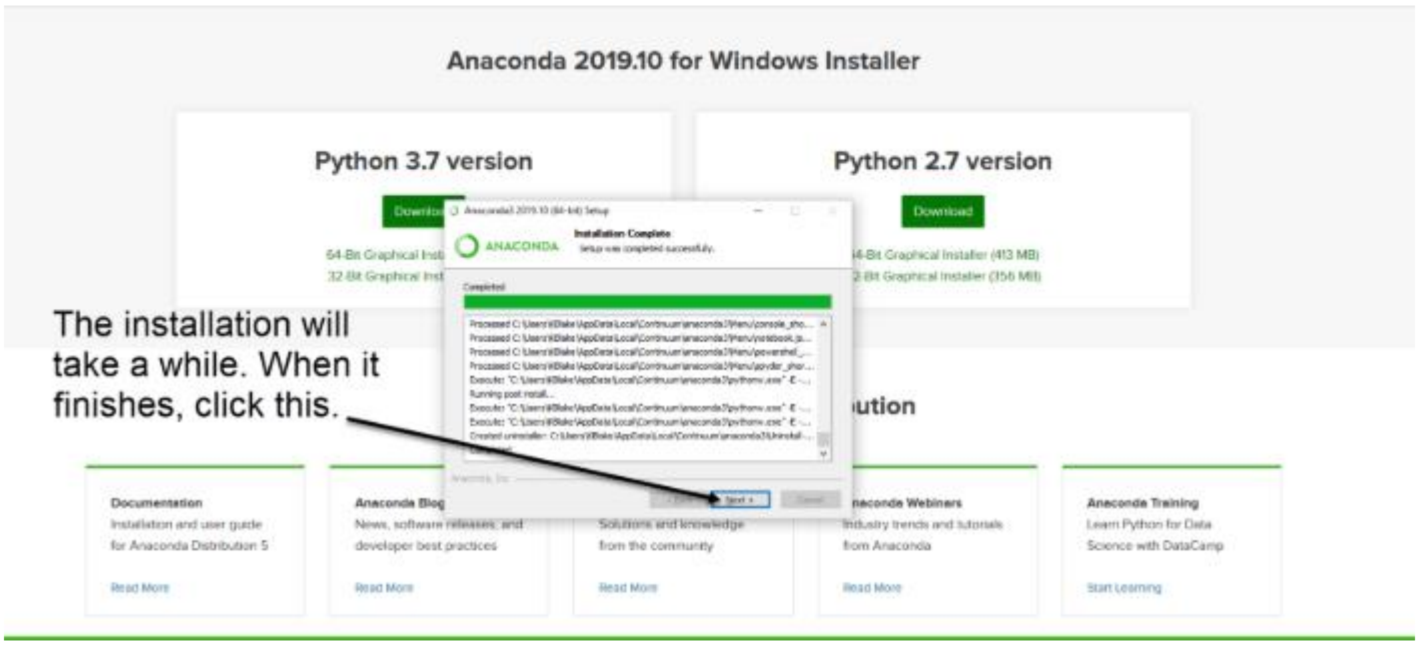
Setting up the package cache ...

Show details

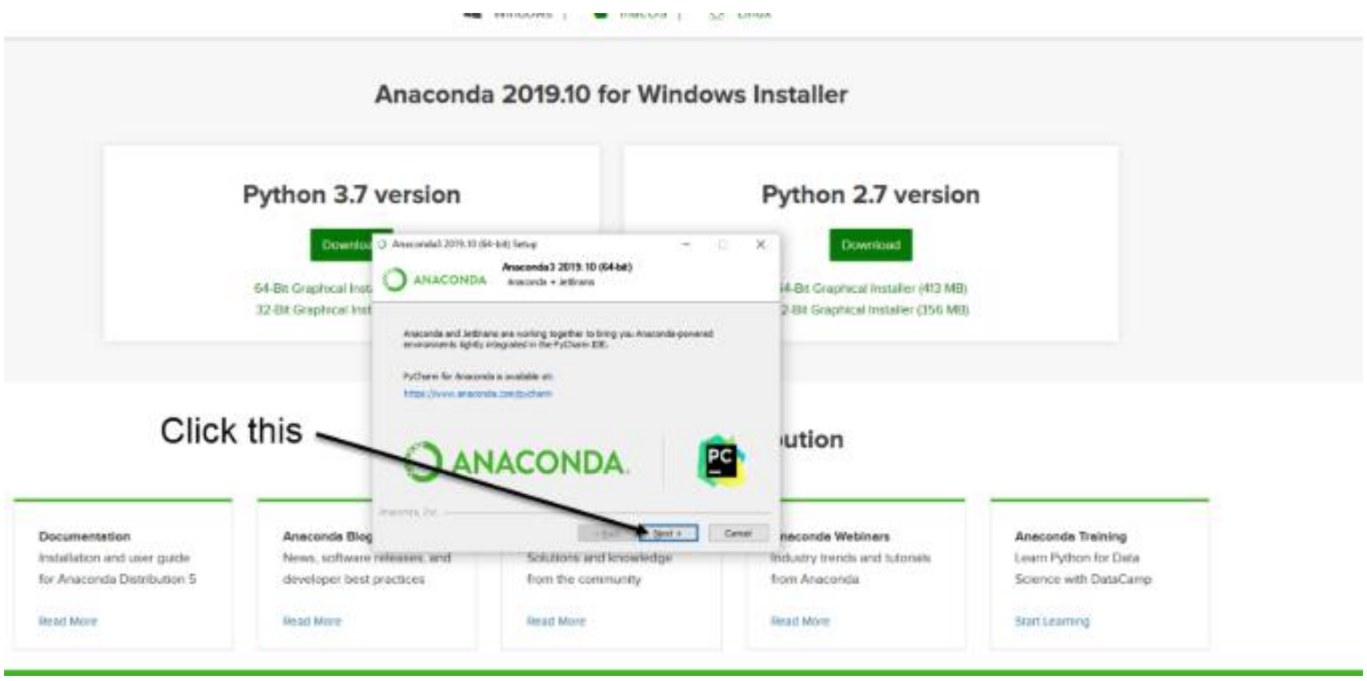
Anaconda, Inc.

< Back | Next > | Cancel

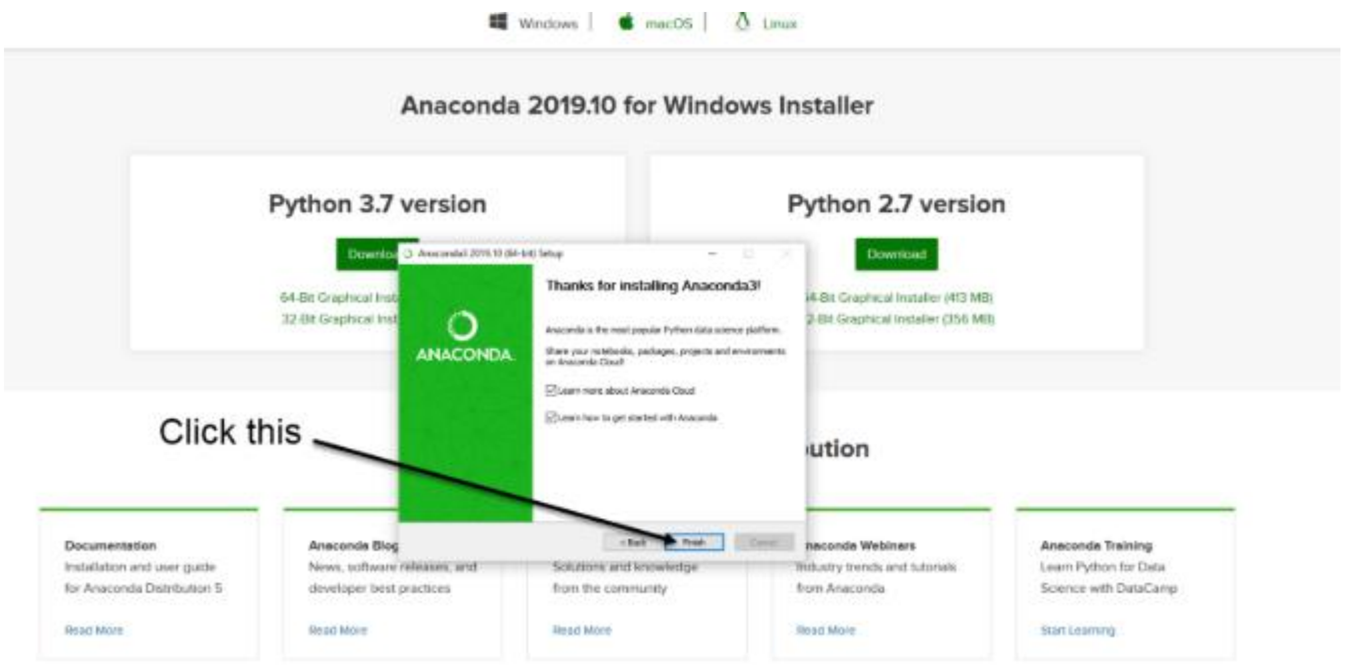
8. The installation is complete. Click Next.


Anaconda 2019.10 for Windows Installer
Python 3.7 version
Python 2.7 version
Installation Complete
Next >
The installation will take a while. When it finishes, click this.

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.



Windows | macOS | Linux

Anaconda 2019.10 for Windows Installer

Python 3.7 version

64-Bit Graphical Inst
32-Bit Graphical Inst

Python 2.7 version

64-Bit Graphical Installer (413 MB)
32-Bit Graphical Installer (356 MB)

Anaconda3 2019.10 (64-bit) Setup

Thanks for installing Anaconda3!

Anaconda is the most popular Python data science platform.

Share your notebooks, packages, projects and environments on Anaconda Cloud!

☑ Learn more about Anaconda Cloud

☑ Learn how to get started with Anaconda

Click this

Documentation
Installation and user guide for Anaconda Distribution 5
Read More

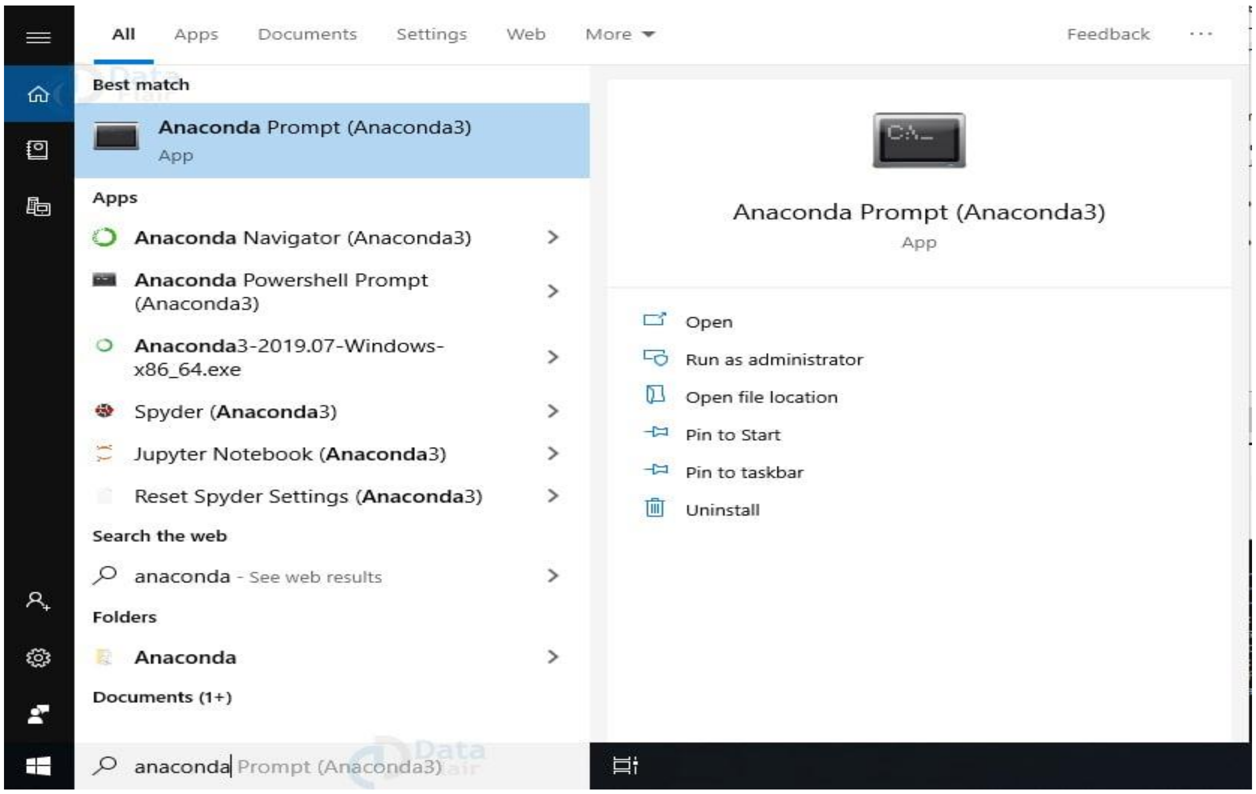
Anaconda Blog
News, software releases, and developer best practices
Read More

Solutions and knowledge from the community
Read More

Anaconda Webinars
Industry trends and tutorials from Anaconda
Read More

Anaconda Training
Learn Python for Data Science with DataCamp
Start Learning

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



All Apps Documents Settings Web More Feedback

Best match

Anaconda Prompt (Anaconda3)
App

Apps

Anaconda Navigator (Anaconda3)
Anaconda Powershell Prompt (Anaconda3)
Anaconda3-2019.07-Windows-x86_64.exe
Spyder (Anaconda3)
Jupyter Notebook (Anaconda3)
Reset Spyder Settings (Anaconda3)

Search the web

anaconda - See web results

Folders

Anaconda

Documents (1+)

anaconda Prompt (Anaconda3)

Anaconda Prompt (Anaconda3)
App

Open
Run as administrator
Open file location
Pin to Start
Pin to taskbar
Uninstall

LIBRARIES/PACKGES :-

TensorFlow: This library was developed by Google in collaboration with the Brain Team. It is an open-source library used for high-level computations. It is also used in machine learning and deep learning algorithms. It contains a large number of tensor operations. Researchers also use this Python library to solve complex computations in Mathematics and Physics.

Matplotlib: This library is responsible for plotting numerical data. And that’s why it is used in data analysis. It is also an open-source library and plots high-defined figures like pie charts, histograms, scatterplots, graphs, etc.

Pandas: Pandas are an important library for data scientists. It is an open-source machine learning library that provides flexible high-level data structures and a variety of analysis tools. It eases data analysis, data manipulation, and cleaning of data. Pandas support operations like Sorting, Re-indexing, Iteration, Concatenation, Conversion of data, Visualizations, Aggregations, etc.

Numpy: The name “Numpy” stands for “Numerical Python”. It is the commonly used library. It is a popular machine learning library that supports large matrices and multi-dimensional data. It consists of in-built mathematical functions for easy computations. Even libraries like TensorFlow use Numpy internally to perform several operations on tensors. Array Interface is one of the key features of this library.

Scikit-learn: It is a famous Python library to work with complex data. Scikit-learn is an open-source library that supports machine learning. It supports variously supervised and unsupervised algorithms like linear regression, classification, clustering, etc. This library works in association with Numpy and SciPy.

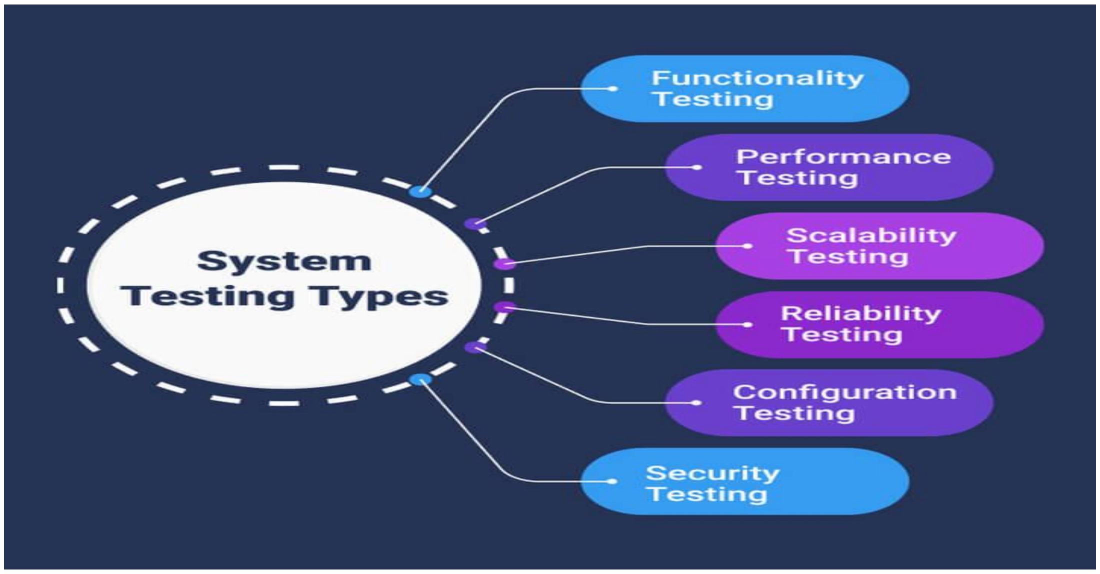
8. SYSTEM TESTING

System testing is a form of software testing that assesses the overall behavior and efficiency of a software solution that is fully developed and integrated. It checks if the system conforms to the defined criteria and if it is ready to be deployed to the final users. System testing is usually performed by a professional testing team that is independent of the development team.

System testing is different from unit testing and integration testing, which are done at lower levels of the software development life cycle. Unit testing focuses on testing individual components or modules of the software, while integration testing focuses on testing the interactions and interfaces between the components or modules. System testing, on the other hand, focuses on testing the entire system as a whole, as well as its compatibility with the intended environment and other systems.

8.1 Different Kinds of System Testing in Software Testing

There are many types of system testing, each with its own objectives, techniques, and tools. Some of the common types of system testing are:



Performance Testing

Performance testing is done to measure the speed, scalability, stability, and reliability of the software product or application. Performance testing helps to determine how the software product or application responds to various levels of load, stress, and concurrency, and how it handles errors and failures. Performance testing also helps to identify and eliminate any performance bottlenecks or issues in the software product or application.

Load Testing

Load testing is a subset of performance testing that is done to determine the behavior of the system or software product under extreme load. Load testing helps to simulate the real-world scenarios and conditions that the software product or application will face when it is deployed and used by the end-users. Load testing also helps to determine the maximum capacity and throughput of the software product or application, and to ensure that it can handle the expected peak load without compromising the quality or performance.

Security Testing

Security testing is done to ensure that the software product or application is secure and safe from any unauthorized access, manipulation, or attack. Security testing helps to verify that the software product or application follows the best practices and standards of security, and that it protects the data and information of the users and the system. Security testing also helps to identify and fix any vulnerabilities or weaknesses in the software product or application that could compromise its security or integrity.

Usability Testing

Usability testing is done to ensure that the software product or application is user-friendly and easy to use. Usability testing helps to evaluate the user interface, navigation, design, and functionality of the software product or application from the perspective of the end-users.

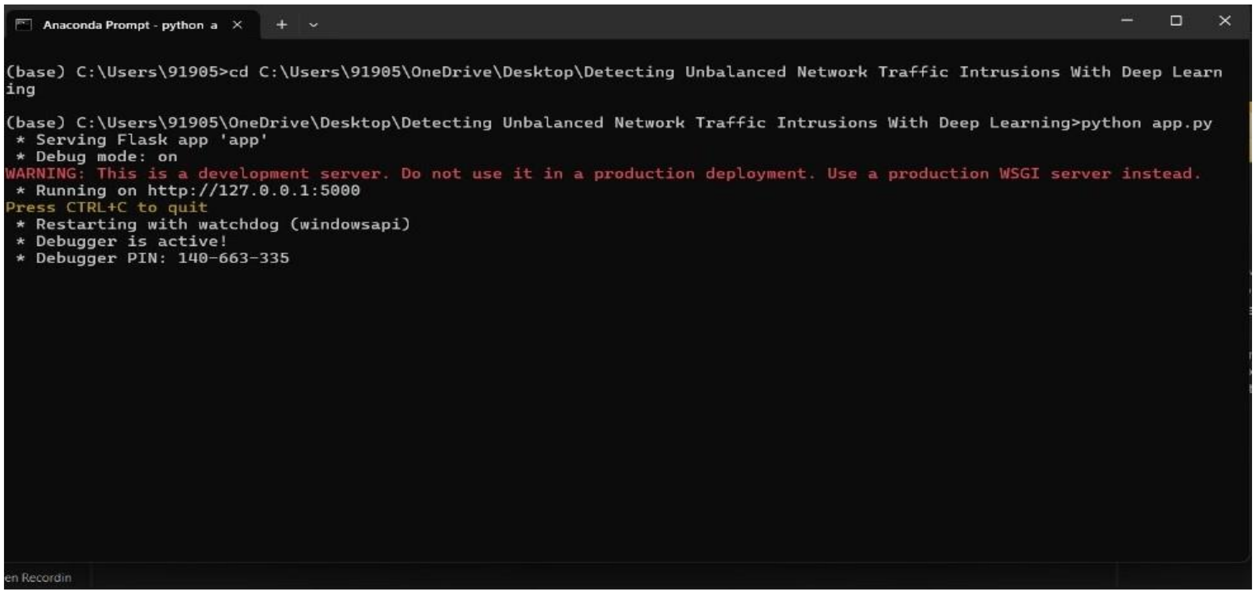
Compatibility Testing

Compatibility testing is done to ensure that the software product or application is compatible with the intended environment and other systems. Compatibility testing helps to verify that the software product or application works well with different operating systems, browsers, devices, hardware, software, and networks.

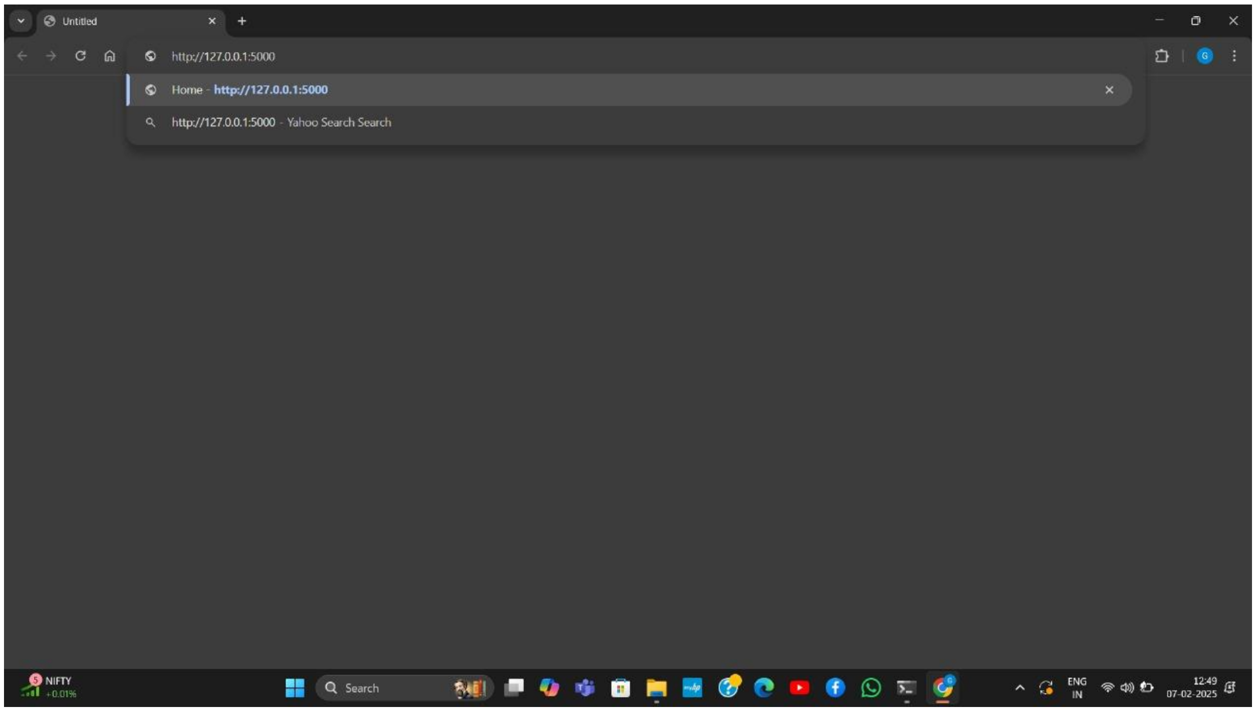
8.2 TEST CASES:

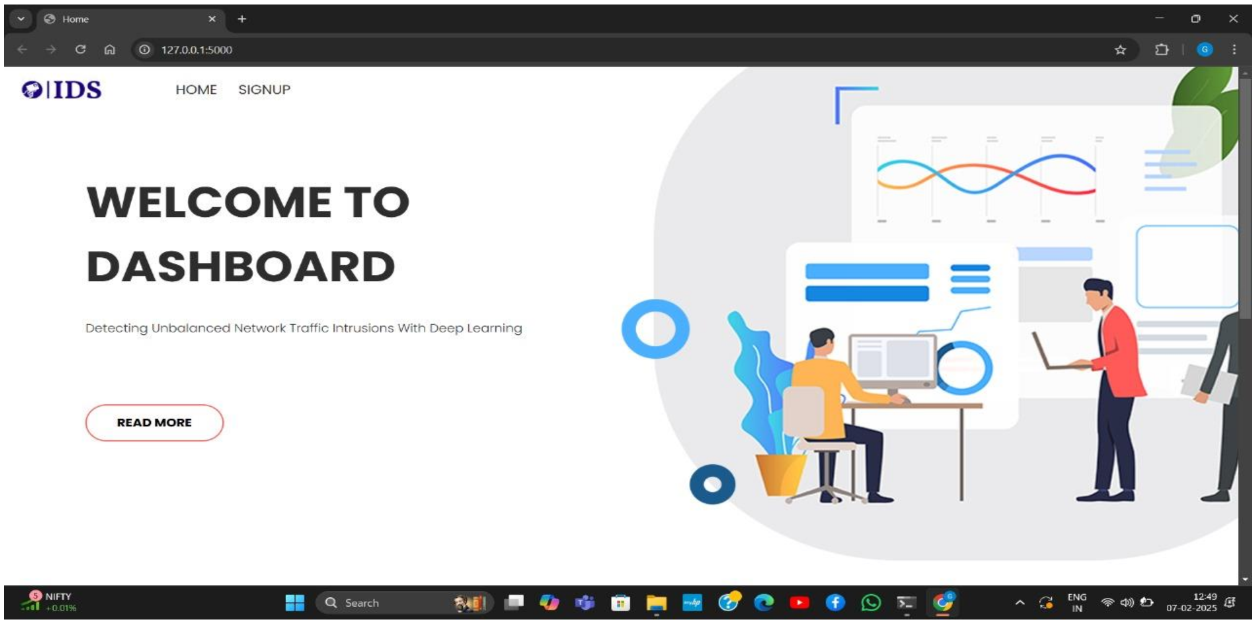
S.NO	INPUT	If available	If not available
1	User signup	User get registered into the application	There is no process
2	User signin	User get login into the application	There is no process
3	Enter input for prediction	Prediction result displayed	There is no process

9. SCREENSHOTS



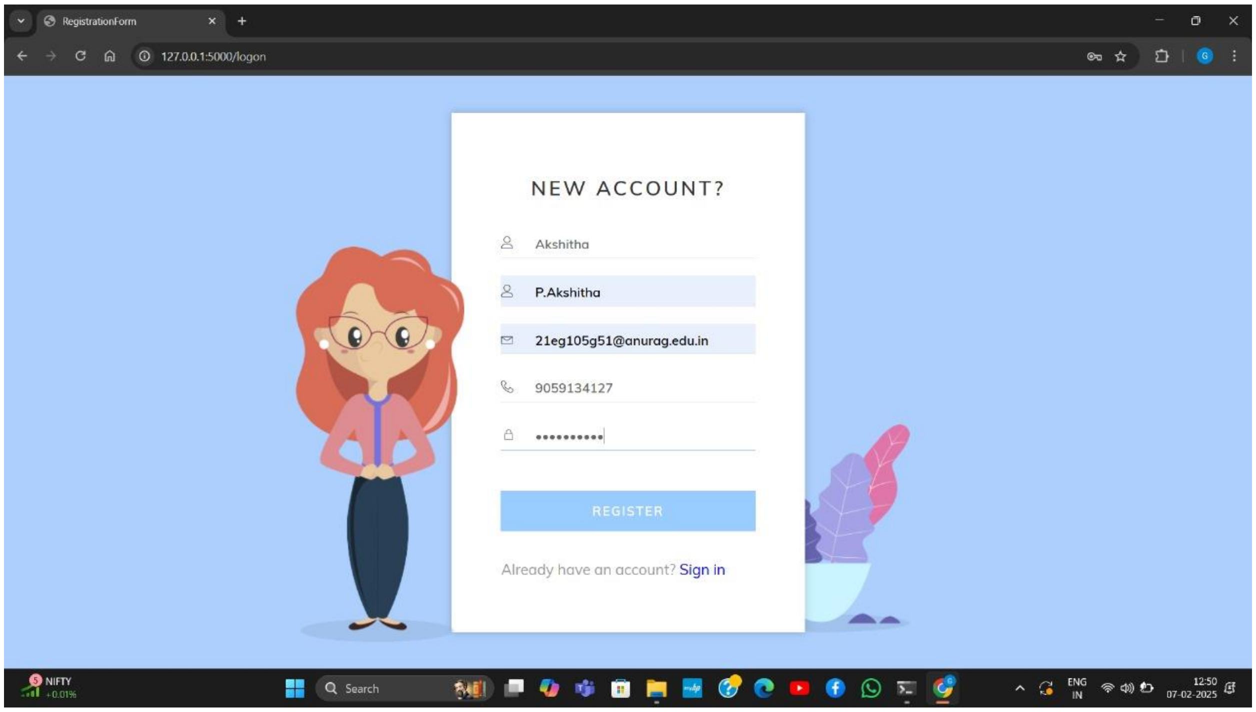
```
(base) C:\Users\91905>cd C:\Users\91905\OneDrive\Desktop\Detecting Unbalanced Network Traffic Intrusions With Deep Learn
ing

(base) C:\Users\91905\OneDrive\Desktop\Detecting Unbalanced Network Traffic Intrusions With Deep Learning>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 140-663-335
```


Untitled
http://127.0.0.1:5000
Home - http://127.0.0.1:5000
http://127.0.0.1:5000 - Yahoo Search Search
NIFTY
+0.01%
Search
ENG
IN
12:49
07-02-2025



WELCOME TO DASHBOARD

Detecting Unbalanced Network Traffic Intrusions With Deep Learning



NEW ACCOUNT?

Akshitha

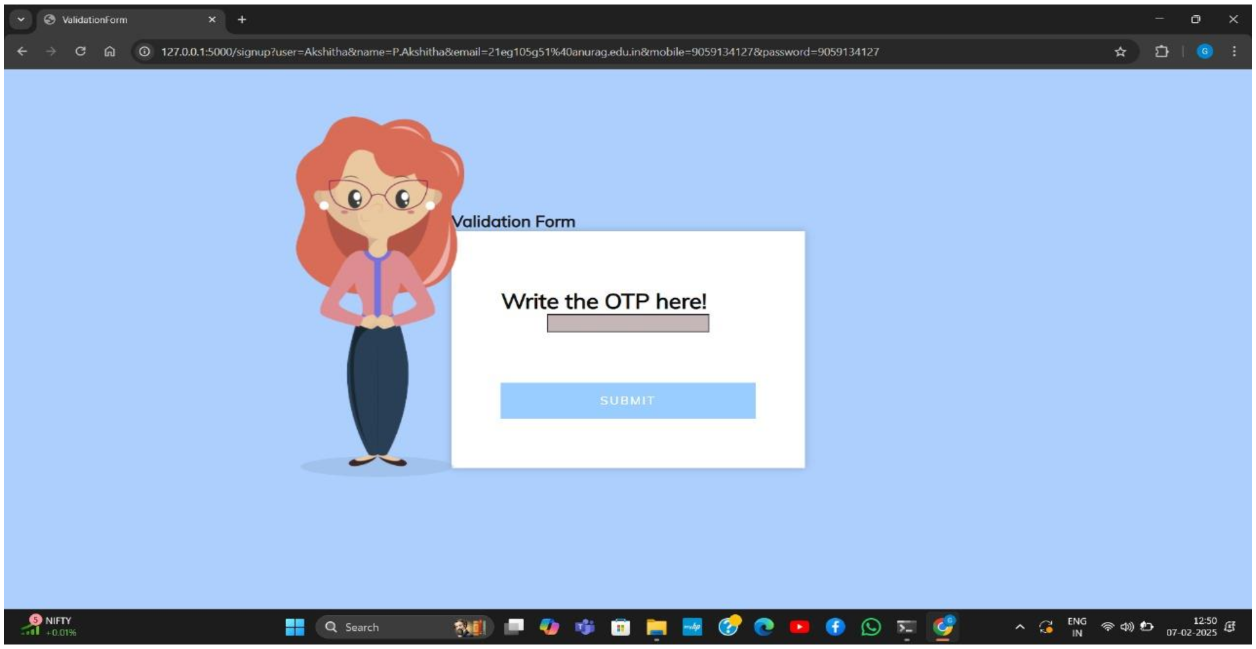
P.Akshitha

21eg105g51@anurag.edu.in

9059134127

••••••••••

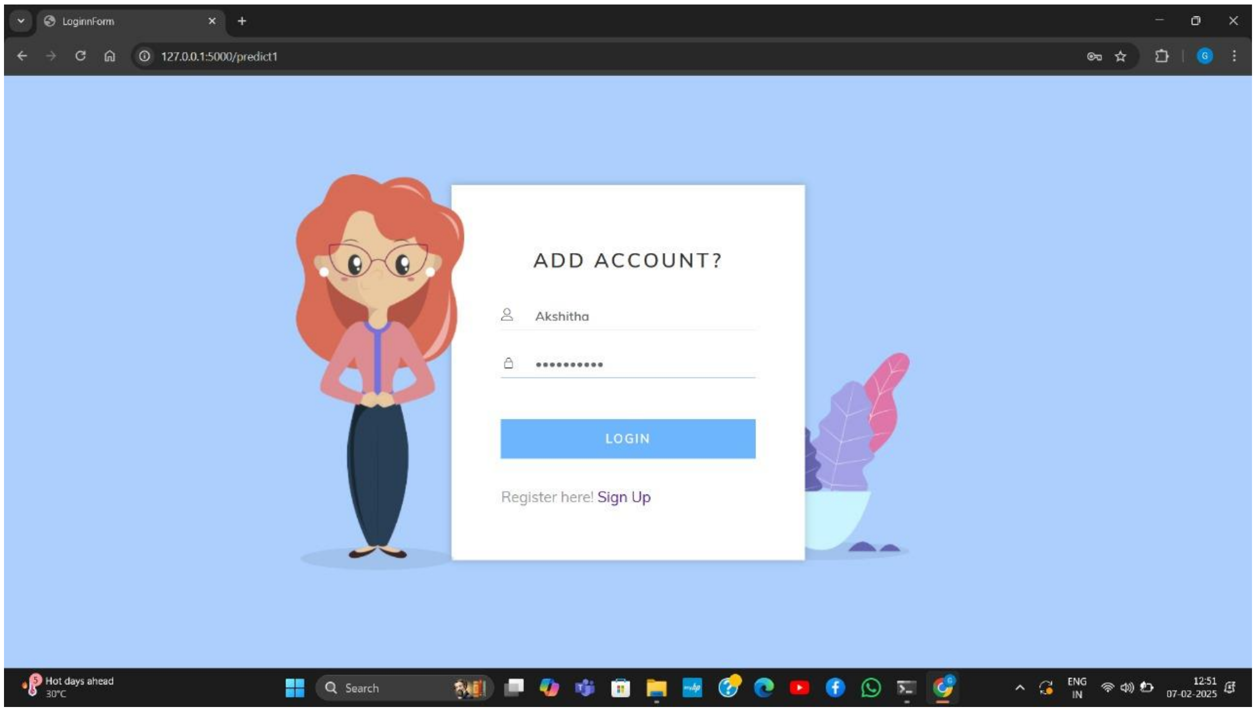
Already have an account? Sign in



127.0.0.1:5000/signup?user=Akshitha&name=P.Akshitha&email=21eg105g51%40anurag.edu.in&mobile=9059134127&password=9059134127

Validation Form

Write the OTP here!



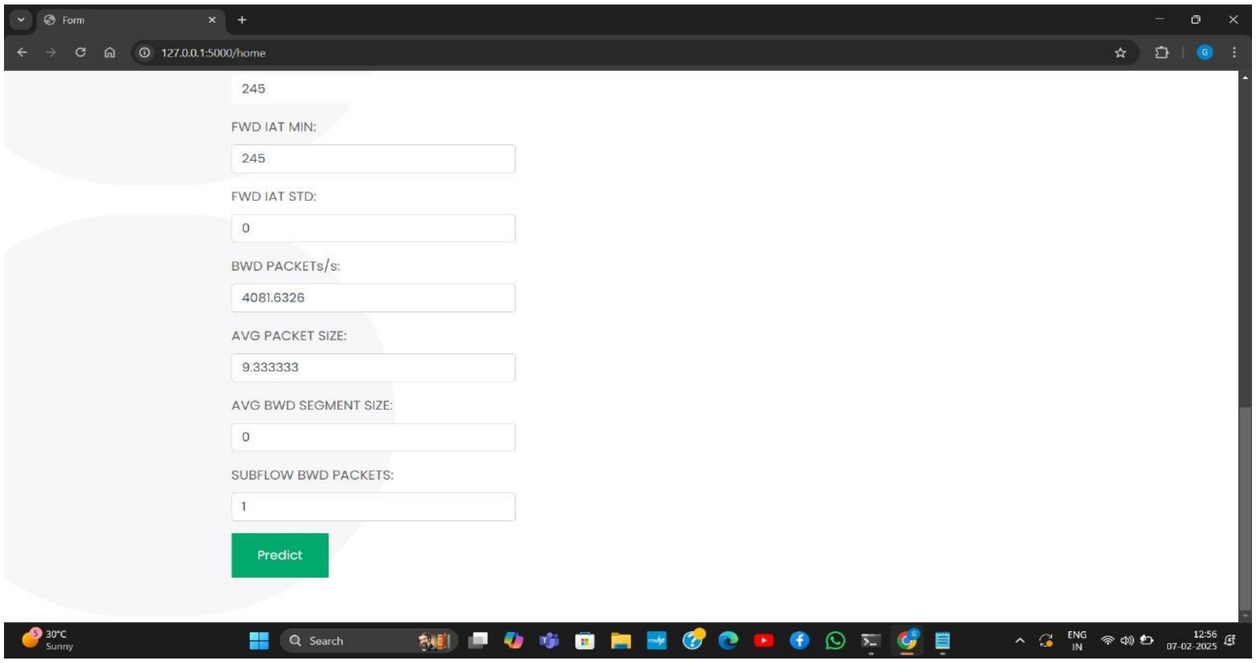
127.0.0.1:5000/predict1

ADD ACCOUNT?

Akshitha

••••••••••

Register here! Sign Up



245

FWD IAT MIN:

245

FWD IAT STD:

0

BWD PACKETs/s:

4081.6326

AVG PACKET SIZE:

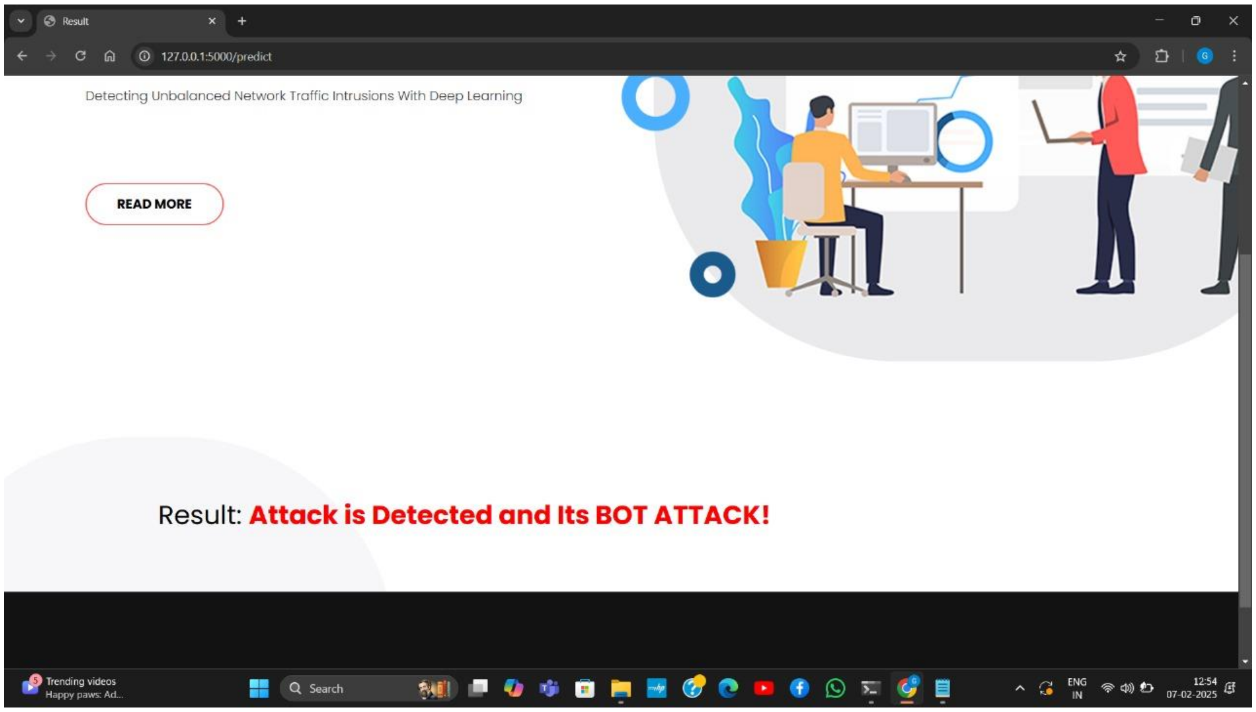
9.333333

AVG BWD SEGMENT SIZE:

0

SUBFLOW BWD PACKETS:

1



Detecting Unbalanced Network Traffic Intrusions With Deep Learning

Result: **Attack is Detected and Its BOT ATTACK!**

10. CONCLUSION

In conclusion, the proposed Intrusion Detection System (IDS) successfully addresses the challenges posed by imbalanced network traffic datasets using advanced machine learning and deep learning techniques. Among the various algorithms tested, the Voting Classifier and CNN+BiLSTM models stand out as the most effective in detecting network intrusions. The Voting Classifier, which combines Random Forest (RF) and Decision Tree (DT), demonstrates exceptional robustness and accuracy, effectively classifying both benign and malicious traffic, even in the presence of imbalanced data. Additionally, the CNN+BiLSTM model, leveraging the power of convolutional neural networks for feature extraction and bidirectional LSTM for contextual understanding, significantly enhances intrusion detection accuracy. These high-performance algorithms not only improve the detection of intrusions but also ensure the system's resilience against evolving cyber threats. By achieving 100% accuracy in identifying malicious network traffic, the proposed IDS contributes to creating a more secure network environment, safeguarding sensitive information and minimizing the risks associated with cyberattacks.

In the future, the project can be further enhanced by exploring additional techniques such as reinforcement learning for adaptive intrusion detection, hybrid deep learning models for better feature extraction, and attention mechanisms to improve the model's focus on critical network traffic patterns. Incorporating advanced anomaly detection techniques and autoencoder networks could also enhance the system’s ability to detect unknown intrusions. Additionally, the system's scalability and efficiency could be improved by implementing distributed learning approaches.

11. REFERENCES

[1] M. A. Talukder, K. F. Hasan, M. M. Islam, M. A. Uddin, A. Akhter, M. A. Yousuf, F. Alharbi, and M. A. Moni, ‘‘A dependable hybrid machine learning model for network intrusion detection,’’ J. Inf. Secur. Appl., vol. 72, Feb. 2023, Art. no. 103405.

[2] X. He, Q. Chen, L. Tang, W. Wang, and T. Liu, ‘‘CGAN-based collaborative intrusion detection for UAV networks: A blockchain-empowered distributed federated learning approach,’’ IEEE Internet Things J., vol. 10, no. 1, pp. 120–132, Jan. 2023.

[3] J. Escorcia-Gutierrez, M. Gamarra, E. Leal, N. Madera, C. Soto, R. F. Mansour, M. Alharbi, A. Alkhayyat, and D. Gupta, ‘‘Sea turtle foraging algorithm with hybrid deep learning-based intrusion detection for the Internet of Drones environment,’’ Comput. Electr. Eng., vol. 108, May 2023, Art. no. 108704.

[4] V. Subbarayalu and M. A. Vensuslaus, ‘‘An intrusion detection system for drone swarming utilizing timed probabilistic automata,’’ Drones, vol. 7, no. 4, p. 248, Apr. 2023.

[5] K. A. Alissa, S. S. Alotaibi, F. S. Alrayes, M. Aljebreen, S. Alazwari, H. Alshahrani, M. A. Elfaki, M. Othman, and A. Motwakel, ‘‘Crystal structure optimization with deep-autoencoder-based intrusion detection for secure Internet of Drones environment,’’ Drones, vol. 6, no. 10, p. 297, Oct. 2022.

[6] S. Lipsa and R. K. Dash, ‘‘A novel dimensionality reduction strategy based on linear regression with a fine-pruned decision tree classifier for detecting DDoS attacks in cloud computing environments,’’ in Proc. 1st Int. Symp. Artif. Intell., 2022, pp. 15–25.

[7] X. Tan, S. Su, Z. Zuo, X. Guo, and X. Sun, ‘‘Intrusion detection of UAVs based on the deep belief network optimized by PSO,’’ Sensors, vol. 19, no. 24, p. 5529, Dec. 2019.

[8] N. Moustafa and A. Jolfaei, ‘‘Autonomous detection of malicious events using machine learning models in drone networks,’’ in Proc. 2nd ACM MobiCom Workshop Drone Assist. Wireless Commun. 5G Beyond, Sep. 2020, pp. 61–66.

[9] R. Shrestha, A. Omidkar, S. A. Roudi, R. Abbas, and S. Kim, ‘‘Machine-Learning-Enabled intrusion detection system for cellular connected UAV networks,’’ Electronics, vol. 10, no. 13, p. 1549, Jun. 2021.

[10] O. Bouhamed, O. Bouachir, M. Aloqaily, and I. A. Ridhawi, ‘‘Lightweight IDS for UAV networks: A periodic deep reinforcement learning-based approach,’’ in Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM), May 2021, pp. 1032–1037.

[11] L. Wang, Y. Chen, P. Wang, and Z. Yan, ‘‘Security threats and counter-measures of unmanned aerial vehicle communications,’’ IEEE Commun. Standards Mag., vol. 5, no. 4, pp. 41–47, Dec. 2021.

[12] P. Sun, P. Liu, Q. Li, C. Liu, X. Lu, R. Hao, and J. Chen, ‘‘DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system,’’ Secur. Commun. Netw., vol. 2020, pp. 1–11, Aug. 2020, doi: 10.1155/2020/8890306.

[13] H. Bangui and B. Buhnova, ‘‘Recent advances in machine-learning driven intrusion detection in transportation: Survey,’’ Proc. Comput. Sci., vol. 184, pp. 877–886, Aug. 2021.

[14] Z. Chen, N. Lyu, K. Chen, Y. Zhang, and, W. Gao, ‘‘UAV network intrusion detection method based on spatio-temporal graph convolutional network,’’ J. Beijing Univ. Aeronaut. Astronaut., vol. 47, no. 5, pp. 1068–1076, 2021.

[15] E. Basan, M. Lapina, N. Mudruk, and E. Abramov, ‘‘Intelligent intrusion detection system for a group of UAVs,’’ in Proc. 12th Int. Conf. Adv. Swarm Intell., 2021, pp. 230–240.

[16] V. Praveena, A. Vijayaraj, P. Chinnasamy, I. Ali, R. Alroobaea, S. Y. Alyahyan, and M. A. Raza, ‘‘Optimal deep reinforcement learning for intrusion detection in UAVs,’’ Comput., Mater. Continua, vol. 70, no. 2, pp. 2639–2653, 2022.

[17] J. Whelan, A. Almehmadi, and K. El-Khatib, ‘‘Artificial intelligence for intrusion detection systems in unmanned aerial vehicles,’’ Comput. Electr. Eng., vol. 99, Apr. 2022, Art. no. 107784.

[18] Q. Abu Al-Haija and A. Al Badawi, ‘‘High-performance intrusion detection system for networked UAVs via deep learning,’’ Neural Comput. Appl., vol. 34, no. 13, pp. 10885–10900, Jul. 2022.

[19] R. Fotohi, M. Abdan, and S. Ghasemi, ‘‘A self-adaptive intrusion detection system for securing UAV-to-UAV communications based on the human immune system in UAV networks,’’ J. Grid Comput., vol. 20, no. 3, p. 22, Sep. 2022.

[20] R. R. Kumar, M. Shameem, and C. Kumar, ‘‘A computational framework for ranking prediction of cloud services under fuzzy environment,’’ Enterprise Inf. Syst., vol. 16, no. 1, pp. 167–187, Jan. 2022, doi: 10.1080/17517575.2021.1889037.

[21] E. Basan, A. Basan, A. Nekrasov, C. Fidge, E. Abramov, and A. Basyuk, ‘‘A data normalization technique for detecting cyber attacks on UAVs,’’ Drones, vol. 6, no. 9, p. 245, Sep. 2022.

[22] (2030). Cyber Security Market Share, Forecast | Growth Analysis. Accessed: Apr. 23, 2023. [Online]. Available: https://www.fortunebusinessinsights.com/industry-reports/cyber-security-market-101165Benamor

[23] M. A. Akbar, M. Shameem, S. Mahmood, A. Alsanad, and A. Gumaei, ‘‘Prioritization based taxonomy of cloud-based outsource software development challenges: Fuzzy AHP analysis,’’ Appl. Soft Comput., vol. 95, Oct. 2020, Art. no. 106557, doi: 10.1016/j.asoc.2020.106557.

[24] M. Bakro, S. K. Bisoy, A. K. Patel, and M. A. Naal, ‘‘Performance analysis of cloud computing encryption algorithms,’’ in Advances in Intelligent Computing and Communication, vol. 202. Cham, Switzerland: Springer, 2021, pp. 357–367.

[25] J. Whelan, T. Sangarapillai, O. Minawi, A. Almehmadi, and K. El- Khatib, ‘‘UAV attack dataset,’’ IEEE Dataport, vol. 167, no. 1, pp. 1561–1573, Jan. 2020.

[26] S. Choudhary and N. Kesswani, ‘‘Analysis of KDD-cup’99, NSL-KDD and UNSW-NB15 datasets using deep learning in IoT,’’ Proc. Comput. Sci., vol. 167, pp. 1561–1573, Jan. 2020.

[27] Z. Ma, G. Wu, P. N. Suganthan, A. Song, and Q. Luo, ‘‘Performance assessment and exhaustive listing of 500+ nature-inspired metaheuristic algorithms,’’ Swarm Evol. Comput., vol. 77, Mar. 2023, Art. no. 101248, doi: 10.1016/j.swevo.2023.101248.

[28] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, ‘‘Harris hawks optimization: Algorithm and applications,’’ Future Gener. Comput. Syst., vol. 97, pp. 849–872, Aug. 2019, doi: 10.1016/j.future.2019.02.028.

[29] S. Mirjalili, S. M. Mirjalili, and A. Lewis, ‘‘Grey wolf optimizer,’’ Adv. Eng. Softw., vol. 69, pp. 46–61, Mar. 2014, doi: 10.1016/j.advengsoft.2013.12.007.

[30] D. Molina, F. Moreno-García, and F. Herrera, ‘‘Analysis among winners of different IEEE CEC competitions on real-parameters optimization: Is there always improvement?’’ in Proc. IEEE Congr. Evol. Comput. (CEC), Donostia, Spain, Jun. 2017, pp. 805–812, doi: 10.1109/CEC.2017.7969392.