AUDISANKARA COLLEGE OF ENGINEERING & TECHNOLOGY (AUTONOMOUS)

A Capstone Project Report on

CYBER CRIME PREDICTION IN THE DEPARTMENT OF TELECOMMUNICATIONS

Submitted in a partial fulfillment for the award of the degree

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by,

DUPPALA CHENNAKESAVA (21G21A0546)

DUVVURU MONISH (21G21A0548)

GANDAVARAM DEVA SUMANTH REDDY (21G21A0554)

GALAJU SIDDARDA ACHARI (21G21A0550)

Under the esteemed Guidance of

Mrs.A. BHARATHI MSc (CS), MTech.,
Assistant Professor,
Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AUDISANKARA COLLEGE OF ENGINEERING & TECHNOLOGY(A)
Accredited by NAAC with 'A+' Grade | Accredited by NBA Approved by AICTE | Affiliated to JNTUA
NH5 Bypass Road, Gudur – 524101, Tirupati (DT.), Andhra Pradesh

www.audisankara.ac.in
2025

AUDISANKARA COLLEGE OF ENGINEERING & TECHNOLOGY (AUTONOMOUS)

Accredited by NAAC with 'A+' Grade | Accredited by NBA Approved by AICTE | Affiliated to JNTUA
NH5 Bypass Road, Gudur – 524101, Tirupati (DT.), Andhra Pradesh

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Major Project report on entitled " CYBER CRIME PREDICTION IN THE DEPARTMENT OF TELECOMMUNICATIONS " is the Bonafide work done by the students DUPPALA CHENNAKESAVA (21G21A0546), DUVVURU MONISH (21G21A0548), GANDAVARAM DEVA SUMANTH REDDY (21G21A0554), GALAJU SIDDARDA ACHARI (21G21A0550) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering, from Jawaharlal Nehru Technological University Anantapur, Anantapuramu, during the year 2024-2025.

Supervisor	**Head of the Department**
Mrs. A. Bharathi MSc (CS), M.Tech.,	**Prof. D. V. Varaprasad, M.Tech, (Ph.D),**
Assistant Professor	Associate Professor & HOD
Department of CSE	Department of CSE
ASCET, GUDUR – TIRUPATI (DT).	**ASCET, GUDUR-TIRUPATI(DT).**

Submitted for the viva-voce Examination held on:

Internal Examiner **External Examiner**

AUDISANKARA COLLEGE OF ENGINEERING & TECHNOLOGY (AUTONOMOUS)

Accredited by NAAC with 'A+' Grade | Accredited by NBA Approved by AICTE | Affiliated to JNTUA

NH5 Bypass Road, Gudur – 524101, Tirupati (DT.), Andhra Pradesh

DECLARATION

We, DUPPALA CHENNAKESAVA (21G21A0546), DUVVURU MONISH (21G21A0548), GANDAVARAM DEVA SUMANTH REDDY (21G21A0554) and GALAJU SIDDARDA ACHARI (21G21A0550), hereby declare that the Major Project Work entitled — **CYBER CRIME PREDICTION IN THE DEPARTMENT OF TELECOMMUNICATIONS** has been carried out by us under the esteemed guidance of **Mrs. A. BHARATHI, MSc(CS), M.Tech, Assistant Professor, Department of CSE**. This Major Project Report is submitted in partial fulfillment of the requirements for the award of the **Bachelor's degree in Computer Science and Engineering**.

Date:

Place:

PROJECT ASSOCIATES

DUPPALA CHENNAKESAVA (21G21A0546)

DUVVURU MONISH (21G21A0548)

GANDAVARAM DEVA SUMANTH REDDY (21G21A0554)

GALAJU SIDDARDA ACHARI (21G21A0550)

ACKNOWLEDGEMENT

We express our deepest gratitude to **Dr. Vanki Penchalaiah**, Chairman of Audisankara College of Engineering & Technology, for providing us with an excellent academic environment and unwavering support throughout our project.

We extend our heartfelt thanks to **Smt. Vanki Anusha**, Vice Chairperson, for her encouragement and continuous motivation, which have been instrumental in our academic journey.

We sincerely appreciate the guidance and support of **Dr. Raja Murugadoss Jeyaraju**, Director(Engg.,& Principal), whose leadership and vision have inspired us to strive for excellence.

We extend our heartfelt thanks to **Mrs. A. Bharathi,** project guide, for her invaluable insights, encouragement, and unwavering support, all of which played a significant role in the successful completion of our project.

Our sincere gratitude goes to **Prof. D. V. Varaprasad,** M.Tech,Ph.D, Head of the Department, for valuable insights, motivation, and support, which greatly contributed to the successful completion of our project.

We are thankful to **Dr. A. Immanuel,** Institute Project Coordinator, for his expert guidance and coordination, which played a key role in shaping our project.

We also extend our appreciation to **Mr. A. Hemantha Kumar,** Department Project Coordinator, for providing valuable feedback and ensuring the smooth progress of our project.

We express our gratitude to our **department faculty** for their constant encouragement, technical guidance, and mentorship throughout this journey.

Our special thanks to the **non-teaching faculty** for their assistance and support in various aspects of our project work.

We would like to extend our heartfelt appreciation to our **parents and friends**, whose encouragement, patience, and moral support have been our driving force in completing this capstone project successfully.

Finally, we acknowledge the collaborative efforts of our **team members**, whose dedication and hard work made this project possible.

PROJECT ASSOCIATES

Duppala Chennakesava (21G21A0546)

Duvvuru Monish (21G21A0548)

Gandavaram deva Sumanth Reddy (21G21A0554)

Galaju Siddarda Achari (21G21A0550)

INDEX

CONTENTS **PG-NO**

ABSTRACT **6**

CHAPTER-1: INTRODUCTION **7-14**

1.1 OVERVIEW……………………………………………………………………………………..8

1.2 MOTIVATION ……………………………………………………………………………………9

1.3 PROBLEM STATEMENT……………………………………………………………………....9

1.4 APPLICATION…………………………………………………………………………………10

1.5 LITERATURE SURVEY …………………………………………………………………...12

CHAPTER- 2: EXISTING SYSTEM **15-29**

2.1 LOGISTIC REGRESSION.. 16

2.2 PSEUDO TABLE 17

2.3 PROPOSED SYSTEM………………………………………………………………………..19-22

2.3.1 DATA PREPROCESSING 20

2.3.2 STANDARD SCALER 21

2.3.3 DATASET SPLITTING 22

2.4 RANDOM FOREST ALGORITHM...22-24

2.4.1 IMPORTANT FEATURES OF RANDOM FOREST………………………………...23

2.4.2 ASSUMPTIONS FOR RANDOM FOREST………………………………………….23

2.4.3 TYPES OF ENSEMBLES……………………………………………………………..24

2.5 UML DIAGRAMS..25

CHAPTER- 3: SOFTWARE ENVIRONMENT **30-43**

3.1 PYTHON…………………………………………………………………………………………...31

3.2 PYTHON INSTALLATION………………………………………………………………………..38

CHAPTER- 4: REQUIREMENTS & SPECIFICATIONS **44-56**

4.1 SYSTEM REQUIREMENTS SPECIFICATIONS……………………………………………...…45

4.2 FUNCTIONAL REQUIREMENTS..46

4.3 SOURCECODE 50

CHAPTER- 5: RESULT AND DISCUSSION **57-65**

5.1 IMPLEMENTATION DESCRIPTION 58

5.2 DATASET DESCRIPTION 59

CHAPTER- 6: CONCLUSION AND FUTURE SCOPE **66-68**

6.1 CONCLUSION 67

6.2 FUTURE SCOPE 67

REFERENCES & CERTIFICATIONS **69-72**

ABSTRACT

In today's interconnected world, the widespread adoption of Internet of Things (IoT) devices has brought forth a host of conveniences and opportunities. However, this technological revolution has also opened the door to a new breed of cyber threats, with attackers exploiting vulnerabilities in IoT devices to compromise user privacy, disrupt critical services, and wreak havoc. Traditional security measures have proven inadequate to combat the evolving complexity of these cyber-attacks, necessitating a more advanced and adaptive approach. This urgency has given rise to the development of a Machine Learning Model for Cyber Attack Detection and Classification in IoT Environments (ML-IoT-CD). In addition, the need for a robust cybersecurity solution in IoT environments has become paramount due to the increasing reliance on these devices for critical applications. Existing intrusion detection systems and conventional security measures often lack the scalability and agility needed to keep pace with rapidly evolving attack techniques. As a result, there is a pressing demand for an intelligent, automated, and proactive cyber defense mechanism capable of real-time detection and classification of emerging cyber threats. The ML-IoT-CD model aims to fulfill this need by harnessing the power of machine learning algorithms to analyze vast amounts of data generated by IoT devices. By doing so, it can effectively distinguish between legitimate and malicious activities, thereby bolstering the security posture of IoT ecosystems.

CHAPTER-1
INTRODUCTION

INTRODUCTION

1.1 Overview

The general idea of the Internet of Things (IoT) is to allow for communication between human-to-thing or thing-to-thing(s). Things denote sensors or devices, whilst human or an object is an entity that can request or deliver a service [1]. The interconnection amongst the entities is always complex. IoT is broadly acceptable and implemented in various domains, such as healthcare, smart home, and agriculture. However, IoT has a resource constraint and heterogeneous environments, such as low computational power and memory. These constraints create problems in providing and implementing a security solution in IoT devices. These constraints further escalate the existing challenges for IoT environment. Therefore, various kinds of attacks are possible due to the vulnerability of IoT devices.

IoT-based botnet attack is one of the most popular, spreads faster and create more impact than other attacks. In recent years, several works have been conducted to detect and avoid this kind of attacks [2]–[3] by using novel approaches. Hence, a plethora of relevant of relevant models, methods, and etc. have been introduced over the past few years, with quite a reasonable number of studies reported in the research domain.

Many studies are trying to protect against these botnet attacks on the IoT environment. However, there are many gaps still existing to develop an effective detection mechanism. An intrusion detection system (IDS) is one of the efficient ways to deal with attacks. However, the traditional IDSs are often not able to be deployed for the IoT environments due to the resource constraint problem of these devices. The complex cryptographic mechanisms cannot be embedded in many IoT devices either for the same reason. There are mainly two kinds of IDSs: the anomaly and misuse approaches. The misuse-based, also called the signature-based, approach, is based on the attacks' signatures, and they can also be found in most public IDSs, specifically Suricata [4]. Formally, the attacker can easily circumvent the signature- based approaches, and these mechanisms cannot guarantee to detect the unknown attacks and the variances of known attacks. The anomaly-based systems are based on normal data and can support to identify the unknown attacks. However, the different nature of IoT devices is being faced with the difficulty of collecting common normal data. The machine learning-based detection can guarantee detection of not only the known attacks and their variances. Therefore, we proposed a machine learning- based botnet attack detection architecture. We also adopted a feature selection method to reduce the demand for processing resources for performing the detection system on resource constraint devices. The experiment results indicate that the detection accuracy of our proposed system is high enough to detect the botnet attacks. Moreover, it can support the extension for detecting the new attacks.

1.2. Motivation

Detecting and preventing attacks in IoT sensor data is a crucial and rewarding endeavor. IoT devices are increasingly integrated into critical infrastructure, such as power grids and healthcare systems. Ensuring their security is vital for public safety. The cyber-attacks can result in significant financial losses for individuals, businesses, and governments. Developing effective attack detection mechanisms can reduce these losses and stabilize economies.

Many IoT devices collect sensitive personal data. Detecting attacks helps protect user privacy and maintain trust in IoT technologies. Securing IoT devices fosters innovation and growth in the industry, making IoT technologies safer for deployment. IoT security expertise is in high demand, offering lucrative job opportunities in the dynamic field of cybersecurity. IoT security is a global challenge that requires collaboration and innovation to protect IoT ecosystems against cyber threats. Work in Cyber- attack detection can drive advancements in cybersecurity, machine learning, data analytics, and network security. Tackling complex problems leads to personal and intellectual growth, as you learn and adapt to new challenges. Ultimately, work in Cyber-attack detection contributes to a safer and more secure future, where IoT technology can be used without the fear of malicious attacks.

1.3 Problem Statement

The increasing proliferation of IoT devices has led to a pressing need for robust attack detection mechanisms to safeguard the integrity, confidentiality, and availability of data generated and transmitted by these devices. The problem at hand is to develop and implement efficient and effective Cyber-attack detection solutions that can identify and mitigate a wide range of cyber threats targeting IoT sensor data. The rapid growth of IoT devices across various domains, including critical infrastructure, healthcare, smart homes, and industrial applications, has created an extensive attack surface susceptible to cyber threats. IoT sensor data is vulnerable to various types of attacks, including but not limited to malware infections, DDoS (Distributed Denial of Service) attacks, data breaches, and physical tampering. These threats can lead to data compromise, service disruptions, and privacy breaches. Protecting the integrity and confidentiality of sensor data is crucial, as it often includes sensitive information. Unauthorized access or tampering with this data can have severe consequences.

IoT environments pose unique challenges for attack detection due to their diverse and resource-constrained nature. Traditional security measures may not be directly applicable. Many IoT devices have limited computational power, memory, and bandwidth, making it challenging to implement resource-intensive security solutions. The need for real-time or near-real-time attack detection is critical, as prompt responses are essential to prevent or mitigate the impact of attacks on IoT ecosystems. IoT deployments can range from a few devices to thousands or even millions. Solutions must be scalable to handle large-scale IoT deployments.

IoT devices often come from different manufacturers and use various communication protocols. Attack detection solutions should be able to operate in heterogeneous environments. Balancing security with privacy is a concern in IoT environments, as data collection can involve personal or sensitive information. Solutions must respect privacy regulations. The goal is to develop a holistic approach to Cyber-attack detection that covers a wide spectrum of potential threats and vulnerabilities.

1.4 Applications

Cyber-attack detection has a wide range of applications across various industries and domains due to the increasing use of IoT devices and the growing threat landscape. Here are some key applications:

— Critical Infrastructure Security: Protecting critical infrastructure such as power grids, water treatment facilities, and transportation systems from cyberattacks is paramount. Cyber-attack detection ensures the reliability and security of these systems.

— Healthcare: In the healthcare sector, IoT devices are used for patient monitoring, drug administration, and medical equipment management. Detecting attacks in this context safeguards patient data and ensures the accuracy of medical procedures.

— Smart Cities: IoT plays a pivotal role in creating smart cities with improved services and sustainability. Attack detection helps secure smart city infrastructure, including traffic management, waste management, and public safety systems.

— Industrial IoT (IIoT): IIoT applications in manufacturing, agriculture, and logistics rely on IoT sensors and devices. Detecting attacks in IIoT environments ensures the smooth operation of critical processes.

— Smart Homes: IoT devices are common in smart homes, controlling lighting, heating, security, and entertainment systems. Attack detection safeguards personal data and home automation systems.

— Automotive and Transportation: Connected vehicles and smart transportation systems use IoT for navigation, traffic management, and safety features. Detecting attacks is crucial for passenger safety and system reliability.

— Agriculture: Precision agriculture relies on IoT sensors for monitoring and optimizing crop growth and livestock management. Attack detection protects agricultural data and ensures efficient farming practices.

— Energy Management: IoT devices are used for energy monitoring, optimization, and grid management. Detecting attacks helps prevent disruptions to energy supply and enhances resource management.

— Environmental Monitoring: IoT sensors are employed for environmental monitoring, including air quality, water quality, and weather data collection. Attack detection ensures the accuracy and reliability of environmental data.

— Retail and Supply Chain: In retail, IoT devices are used for inventory management, supply chain tracking, and customer engagement. Attack detection safeguards business operations and customer information.

— Telecommunications: IoT is integrated into telecommunications networks for device management and service provisioning. Detecting attacks helps maintain the integrity of these networks.

— Financial Services: IoT devices are used in financial institutions for asset tracking, security, and customer service. Attack detection is crucial to protect sensitive financial data.

— Education: IoT devices are increasingly used in educational settings for remote learning and campus management. Attack detection safeguards educational data and ensures the availability of online resources.

— Entertainment: IoT devices are used in the entertainment industry for home automation, content delivery, and augmented reality experiences. Detecting attacks enhances the security of entertainment systems.

— Defense and Military: IoT is used in defense applications for surveillance, asset tracking, and communication. Detecting attacks is vital to maintain national security and protect sensitive military data.

1.5 Literature Survey

The overall detection performance achieves around 99% for the botnet attack detection using three different ML algorithms, including artificial neural network (ANN), J48 decision tree, and Naïve Bayes. The experiment result indicated that the proposed architecture can effectively detect botnet-based attacks, and also can be extended with corresponding sub-engines for new kinds of attacks.

Ali et al. [6] outlined the existing proposed contributions, datasets utilised, network forensic methods utilised and research focus of the primary selected studies. The demographic characteristics of primary studies were also outlined. The result of this review revealed that research in this domain is gaining momentum, particularly in the last 3 years (2018-2020). Nine key contributions were also identified, with Evaluation, System, and Model being the most conducted.

Irfan et al. [7] classified the incoming data in the IoT, contain a malware or not. In this research, this work under sample the dataset because the datasets contain imbalance class. After that, this work classified the sample using Random Forest. This work used Naive Bayes, K-Nearest Neighbor and Decision Tree too as a comparison. The dataset that has been used in this research are from UCI Machine Learning Depository's Website. The dataset showed the data traffic from the IoT Device in a normal condition and attacked by Mirai or Bashlite.

Shah et al. [8] presented a concept called 'login puzzle' to prevent capture of IoT devices in a large scale. Login puzzle is a variant of client puzzle, which presented a puzzle to the remote device during the login process to prevent unrestricted log-in attempts. Login puzzle is a set of multiple mini puzzles with a variable complexity, which the remote device is required to solve before logging into any IoT device. Every unsuccessful log-in attempt increases the complexity of solving the login puzzle for the next attempt. This paper introduced a novel mechanism to change the complexity of puzzle after every unsuccessful login attempt. If each IoT device had used login puzzle, Mirai attack would have required almost two months to acquire devices, while it acquired them in 20 h.

Tzagkarakis et al. [9] presented an IoT botnet attack detection method based on a sparsity representation framework using a reconstruction error thresholding rule for identifying malicious network traffic at the IoT edge coming from compromised IoT devices. The botnet attack detection is performed based on small-sized benign IoT network traffic data, and thus we have no prior knowledge about malicious IoT traffic data. We present our results on a real IoT-based network dataset and show the efficacy of proposed technique against a reconstruction error-based autoencoder approach.

Meidan et al. [10] proposed a novel network-based anomaly detection method for the IoT called

N-BaIoT that extracts behavior snapshots of the network and uses deep autoencoders to detect anomalous network traffic from compromised IoT devices. To evaluate the method, this work infected nine commercial IoT devices in our lab with two widely known IoT-based botnets, Mirai and BASHLITE. The evaluation results demonstrated the proposed methods ability to detect the attacks accurately and instantly as they were being launched from the compromised IoT devices that were part of a botnet.

Popoola et al. [11] proposed the federated DL (FDL) method for zero-day botnet attack detection to avoid data privacy leakage in IoT-edge devices. In this method, an optimal deep neural network (DNN) architecture is employed for network traffic classification. A model parameter server remotely coordinates the independent training of the DNN models in multiple IoT-edge devices, while the federated averaging (FedAvg) algorithm is used to aggregate local model updates. A global DNN model is produced after several communication rounds between the model parameter server and the IoT-edge devices. The zero-day botnet attack scenarios in IoT-edge devices are simulated with the Bot-IoT and N- BaIoT data sets.

Hussain et al. [12] produced a generic scanning and DDoS attack dataset by generating 33 types of scans and 60 types of DDoS attacks. In addition, this work partially integrated the scan and DDoS attack samples from three publicly available datasets for maximum attack coverage to better train the machine learning algorithms. Afterwards, this work proposed a two-fold machine learning approach to prevent and detect IoT botnet attacks. In the first fold, this work trained a state-of-the-art deep learning model, i.e., ResNet-18 to detect the scanning activity in the premature attack stage to prevent IoT botnet attacks. While, in the second fold, this work trained another ResNet-18 model for DDoS attack identification to detect IoT botnet attacks.

Abu et al. [13] proposed an ensemble learning model for botnet attack detection in IoT networks called ELBA-IoT that profiles behavior features of IoT networks and uses ensemble learning to identify anomalous network traffic from compromised IoT devices. In addition, this IoT-based botnet detection approach characterizes the evaluation of three different machine learning techniques that belong to decision tree techniques (AdaBoosted, RUSBoosted, and bagged). To evaluate ELBA-IoT, we used the N-BaIoT-2021 dataset, which comprises records of both normal IoT network traffic and botnet attack traffic of infected IoT devices.

Alharbi et al. [14] proposed Gaussian distribution used in the population initialization. Furthermore, the local search mechanism was followed by the Gaussian density function and local-global best function to achieve better exploration during each generation. Enhanced BA was further employed for neural network hyperparameter tuning and weight optimization to classify ten different botnet attacks with an additional one benign target class. The proposed LGBA-NN algorithm was tested on an N-BaIoT data set with extensive real traffic data with benign and malicious target classes. The performance of LGBA-NN was compared with several recent advanced approaches such as weight optimization using Particle Swarm Optimization (PSO-NN) and BA-NN.

Ahmed et al. [15] proposed a model for detecting botnets using deep learning to identify zero-day botnet attacks in real time. The proposed model is trained and evaluated on a CTU-13 dataset with multiple neural network designs and hidden layers. Results demonstrated that the deep-learning artificial neural network model can accurately and efficiently identify botnets.

CHAPTER-2

EXISTING SYSTEM

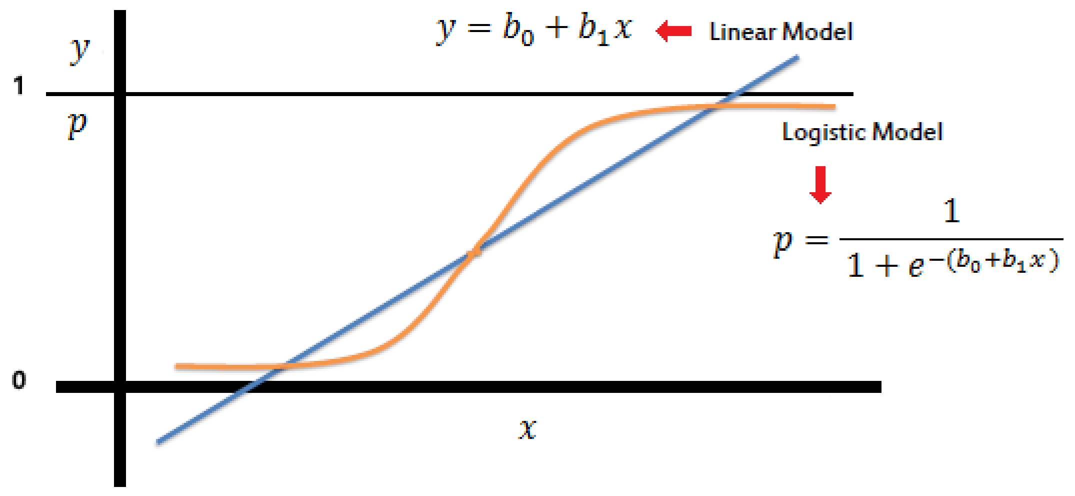
EXISTING SYSTEM

2.1 Logistic Regression

Logistic regression predicts the probability of an outcome that can only have two values (i.e., a dichotomy). The prediction is based on the use of one or several predictors (numerical and categorical). A linear regression is not appropriate for predicting the value of a binary variable for two reasons:

- A linear regression will predict values outside the acceptable range (e.g., predicting probabilities
- outside the range 0 to 1)
- Since the dichotomous experiments can only have one of two possible values for each experiment, the residuals will not be normally distributed about the predicted line.

On the other hand, a logistic regression produces a logistic curve, which is limited to values between 0 and 1. Logistic regression is similar to a linear regression, but the curve is constructed using the natural logarithm of the "odds" of the target variable, rather than the probability. Moreover, the predictors do not have to be normally distributed or have equal variance in each group.



In the logistic regression the constant (b0) moves the curve left and right and the slope (b1) defines the steepness of the curve. By simple transformation, the logistic regression equation can be written in terms of an odds ratio.

$$\frac{p}{1-p} = \exp\left(b_0 + b_1 x\right)$$

Finally, taking the natural log of both sides, we can write the equation in terms of log-odds (logit) which is a linear function of the predictors. The coefficient (*b₁*) is the amount the logit (log-odds) changes with a one-unit change in *x*.

$$ln\left(\frac{p}{1-p}\right) = b_0 + b_1 x$$

As mentioned before, logistic regression can handle any number of numerical and/or categorical variables.

$$p = \frac{1}{1+e^{-(b_0+b_1x_1+b_2x_2+\cdots+b_px_p)}}$$

There are several analogies between linear regression and logistic regression. Just as ordinary least square regression is the method used to estimate coefficients for the best fit line in linear regression, logistic regression uses maximum likelihood estimation (MLE) to obtain the model coefficients that relate predictors to the target. After this initial function is estimated, the process is repeated until LL (Log Likelihood) does not change significantly.

$$\boldsymbol{\beta}^1 = \boldsymbol{\beta}^0 + [\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X}]^{-1}.\boldsymbol{X}^T(\boldsymbol{y}-\boldsymbol{\mu})$$

$\boldsymbol{\beta}$ is a vector of the logistic regression coefficients.

$\boldsymbol{W}$ is a square matrix of order N with elements $n_i\pi_i(1-\pi_i)$ on the diagonal and zeros everywhere else.

$\boldsymbol{\mu}$ is a vector of length N with elements $\mu_i = n_i\pi_i$.

A pseudo R^2 value is also available to indicate the adequacy of the regression model. Likelihood ratio test is a test of the significance of the difference between the likelihood ratio for the baseline model minus the likelihood ratio for a reduced model. This difference is called "model chi-square “. Wald test is used to test the statistical significance of each coefficient (*b*) in the model (i.e., predictors contribution).

2.2 Pseudo Table

There are several measures intended to mimic the R2 analysis to evaluate the goodness-of-fit of logistic models, but they cannot be interpreted as one would interpret an R2 and different pseudo R2 can arrive at very different values. Here we discuss three pseudo R2measures.

Pseudo R^2	Equation	Description
Efron's	$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - p_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$	*'p'* is the logistic model predicted probability. The model residuals are squared, summed, and divided by the total variability in the dependent variable.
McFadden's	$R^2 = 1 - \frac{LL_{full\ model}}{LL_{intercept}}$	The ratio of the log-likelihoods suggests the level of improvement over the intercept model offered by the full model.
Count	$R^2 = \frac{\#\ Corrects}{Total\ Count}$	The number of records correctly predicted, given a cutoff point of .5 divided by the total count of cases. This is equal to the accuracy of a classification model.

Likelihood Ratio Test

The likelihood ratio test provides the means for comparing the likelihood of the data under one model (e.g., full model) against the likelihood of the data under another, more restricted model (e.g., intercept model).

$$LL = \sum_{i=1}^{n} y_i ln(p_i) + (1 - y_i) ln(1 - p_i)$$

where '*p'* is the logistic model predicted probability. The next step is to calculate the difference between these two log-likelihoods.

$$2(LL_1 - LL_2)$$

The difference between two likelihoods is multiplied by a factor of 2 in order to be assessed for statistical significance using standard significance levels (Chi^2 test). The degrees of freedom for the test will equal the difference in the number of parameters being estimated under the models (e.g., full and intercept).

Drawbacks of Logistic Regression

- If the number of observations is lesser than the number of features, Logistic Regression should not be used, otherwise, it may lead to overfitting.
- The major limitation of Logistic Regression is the assumption of linearity between the dependent variable and the independent variables.
- Logistic Regression requires average or no multicollinearity between independent variables.
- In Linear Regression independent and dependent variables are related linearly. But Logistic Regression needs that independent variables are linearly related to the log odds (log(p/(1-p)).

2.3 Proposed System

Detecting cyberattacks using a combination of data preprocessing techniques, such as Standard Scaling, and a RFC classifier is a common approach in cybersecurity. Figure 4.1 shows a cyber-attack attack detection system model.

Step 1: Preprocessing: Gather a dataset of network traffic or system logs, where each data point is labeled as either a normal activity or a cyber-attack. Preprocess the data to make it suitable for training a RFC classifier. This may include handling missing values, encoding categorical variables, and scaling numerical features.

Step 2: Standard Scaling: Extract relevant features from the dataset. Common features for cyber-attack detection may include network traffic statistics, log event patterns, and more. Feature selection or dimensionality reduction techniques can be applied if the dataset has many features. Use Standard Scaling to standardize the numerical features in the dataset. Standard Scaling is preferred over standard scaling when dealing with data that may have outliers. Standard Scaling scales the features in a way that is less affected by extreme values, making it a suitable choice for cybersecurity datasets.

Step 3: Split the Data: Divide your dataset into training, validation, and test sets. A common split might be 80% for training, and 20% for testing.

Step 4: RFC Classifier: Design and build an RFC classifier.

Step 5: Training: Train the RFC classifier using the training dataset. During training, monitor performance on the validation set to avoid overfitting and adjust hyperparameters accordingly.

Step 6: Evaluation: Evaluate the trained model on the test dataset to assess its performance. Common evaluation metrics for cyber-attack detection include accuracy, precision, recall, F1-score, and confusion matrix.

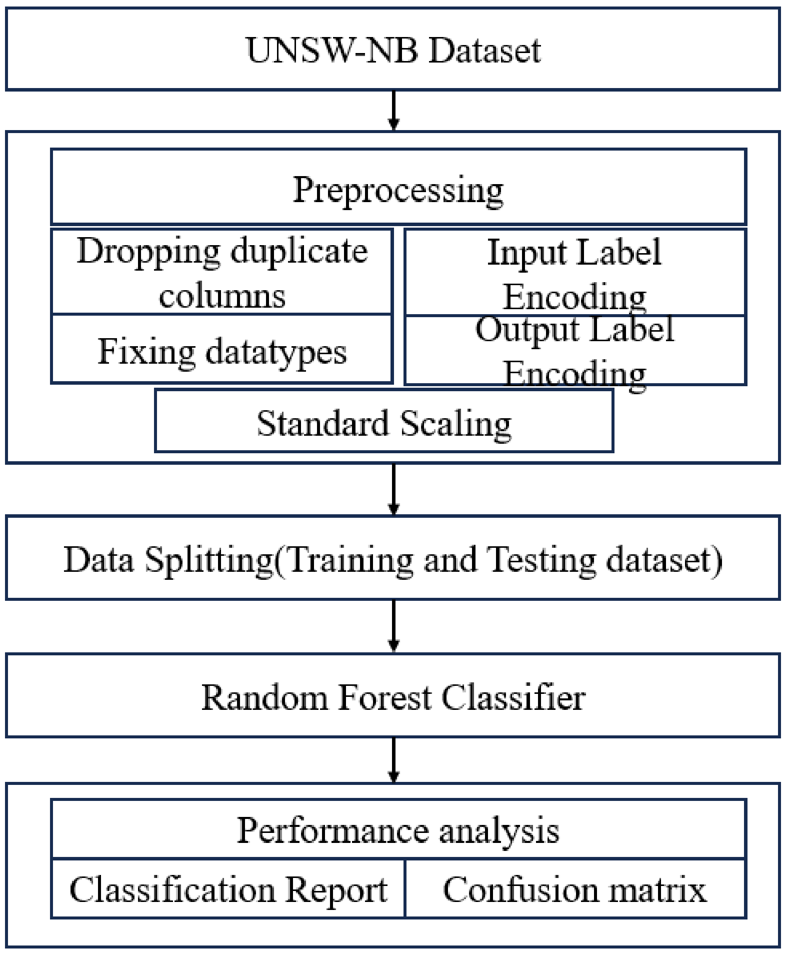


Fig. 4.1: Block diagram of proposed system

2.3.1 Data Preprocessing

Data pre-processing 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 pre-processing task.

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

One-Hot Encoding: Categorical variables are one-hot encoded to convert them into a numerical format suitable for machine learning models. The code uses the pd.get_dummies() function to create binary columns for each category within categorical variables. This transformation allows machine learning algorithms to work with categorical data effectively.

2.3.2 Standard Scaler

Standard Scaler is applied to scale numeric features, ensuring that they have a mean of 0 and a standard deviation of 1. The 'Standard Scaler' from scikit-learn is used to standardize specific numeric features. Standardization is a common preprocessing step to bring features to a similar scale, which can improve the performance of some machine learning algorithms. This transformation is important for several reasons:

Equal Scaling: Standard Scaler scales each feature to have the same scale. This is crucial for algorithms that are sensitive to the scale of features, such as gradient-based optimization algorithms (e.g., in neural networks) and distance-based algorithms (e.g., k-means clustering).

Mean Centering: By subtracting the mean from each data point, Standard Scaler centers the data around zero. This can help algorithms converge faster during training and improve their performance.

Normalization: Scaling by the standard deviation normalizes the data, ensuring that features have comparable variances. This can prevent certain features from dominating others in the modeling process.

Interpretability: Standardized data is more interpretable because it puts all features on a common scale, making it easier to compare the relative importance of feature.

The key advantages of using the Standard Scaler are:

Robustness to Outliers: Outliers have a limited impact on the scaling process because the scale is determined by the IQR, which is less sensitive to extreme values than the standard deviation.

Preservation of Data Distribution: The Standard Scaler retains the shape of the original data distribution, unlike some other scaling methods that transform the data into a Gaussian (normal) distribution.

Useful for Skewed Data: It is effective for features with skewed or non-normally distributed data.

2.3.3 Dataset Splitting

In machine learning data pre-processing, we divide our dataset into a training set and test set. This is one of the crucial steps of data pre-processing as by doing this, we can enhance the performance of our machine learning model. Suppose if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models. If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So, we always try to make a machine learning model which performs well with the training set

Training Set: A subset of dataset to train the machine learning model, and we already know the output. **Test set**: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

2.4 Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

RPF Steps

Step 1: In Random Forest n number of random records are taken from the data set having k number of records.

Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.

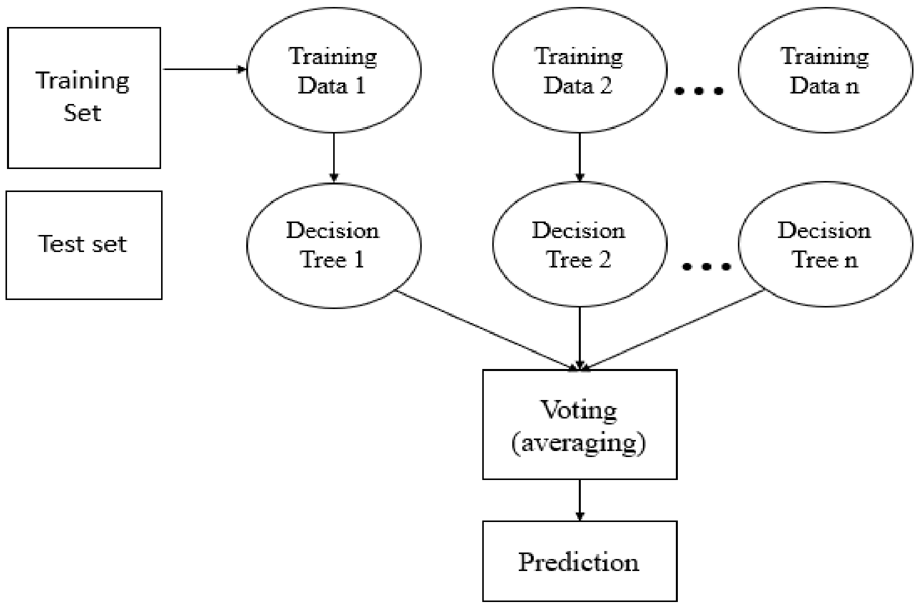


Fig. 3.1: Random Forest algorithm.

2.4.1 Important Features of Random Forest

- **Diversity-** Not all attributes/variables/features are considered while making an individual tree, each tree is different.
- **Immune to the curse of dimensionality-** Since each tree does not consider all the features, the feature space is reduced.
- **Parallelization-**Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests.
- **Train-Test split-** In a random forest we don't have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.
- **Stability-** Stability arises because the result is based on majority voting/ averaging.

2.4.2 Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random Forest classifier:

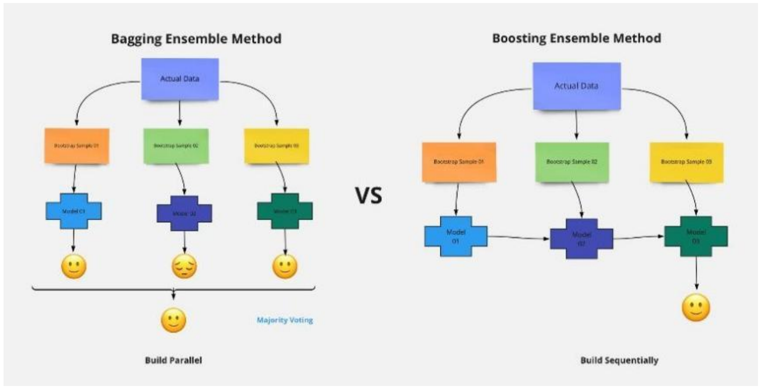
- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Below are some points that explain why we should use the Random Forest algorithm

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

2.4.3 Types of Ensembles

Before understanding the working of the random forest, we must look into the ensemble technique. Ensemble simply means combining multiple models. Thus, a collection of models is used to make predictions rather than an individual model. Ensemble uses two types of methods:



Bagging

It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, Random Forest. Bagging, also known as Bootstrap Aggregation is the ensemble technique used by random forest. Bagging chooses a random sample from the data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement known as row sampling. This step of row sampling with replacement is called bootstrap. Now each model is trained independently which generates results. The final output is based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting is known as aggregation.

Boosting

It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, ADA BOOST, XG BOOST.

2.5 Uml Diagrams

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

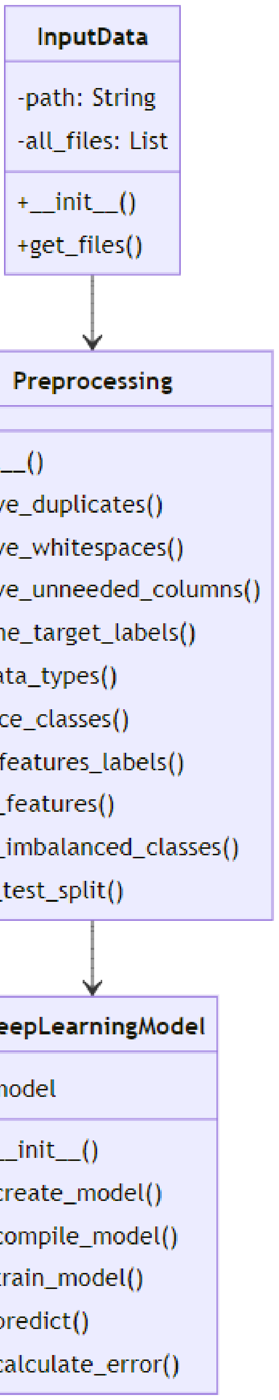
The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects- oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

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

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

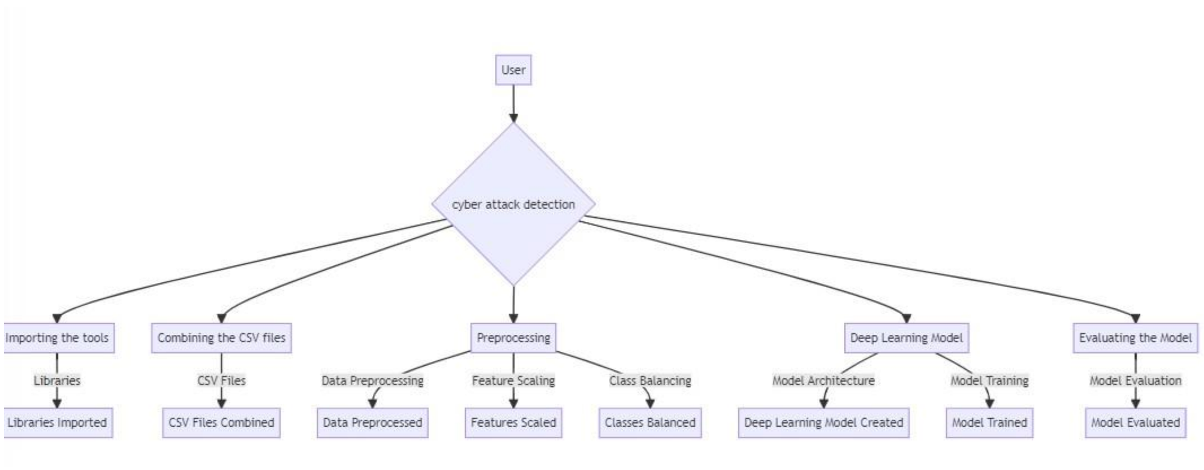
Class diagram

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



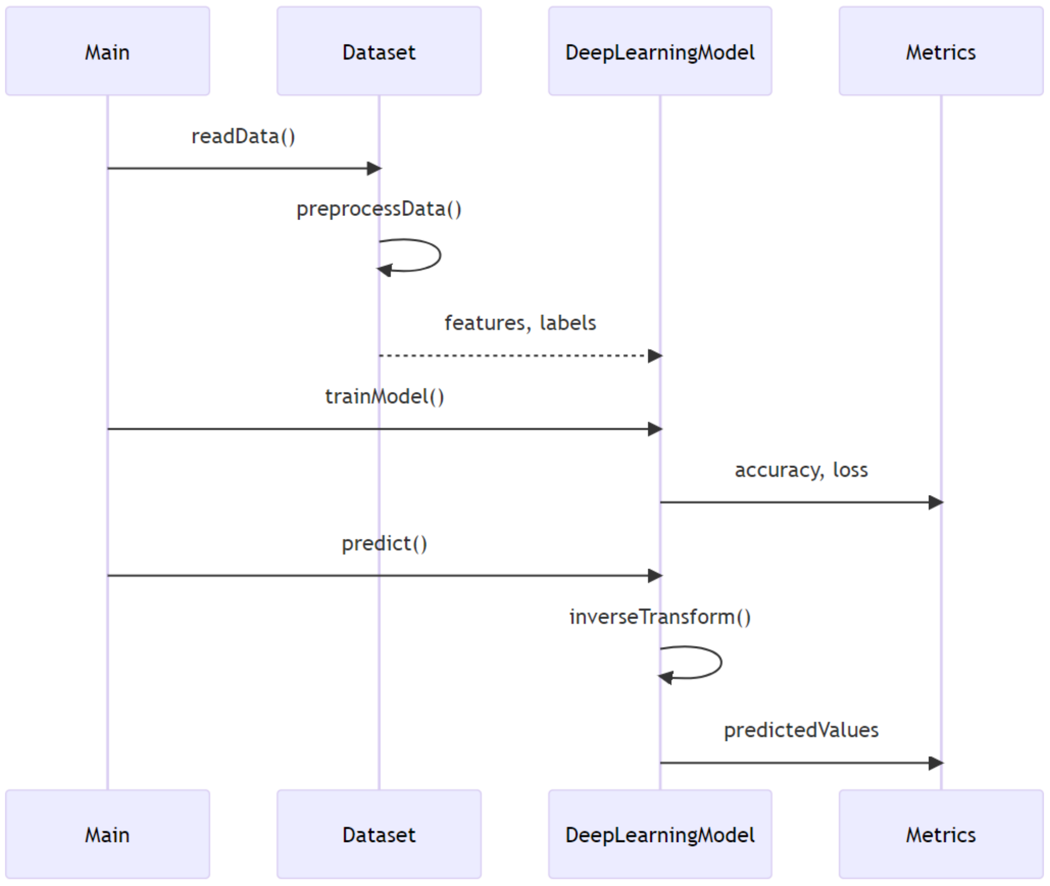
Use case Diagram

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

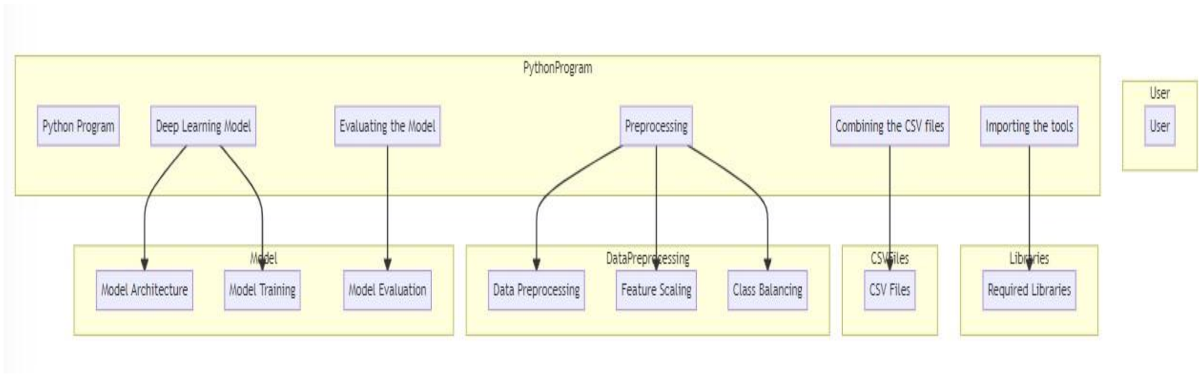


Sequence Diagram

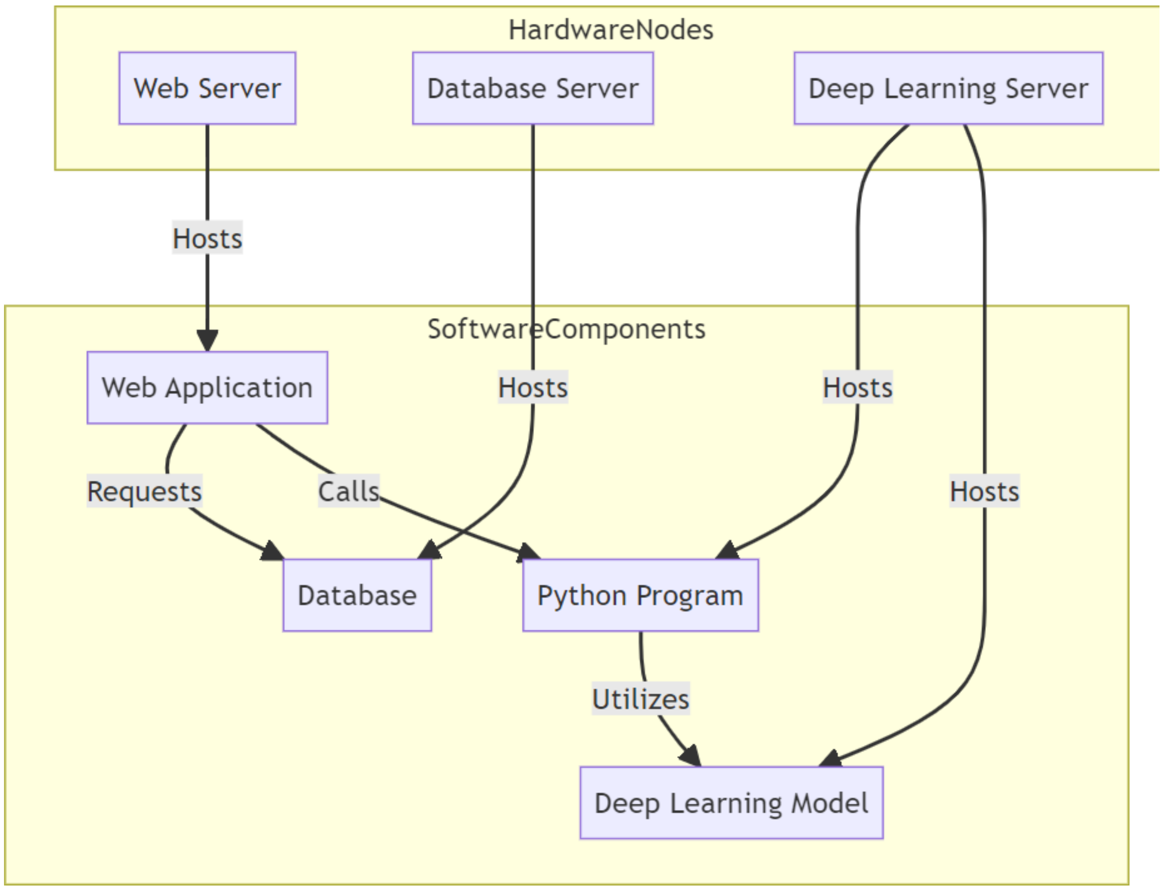
A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows, as parallel vertical lines ("lifelines"), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.



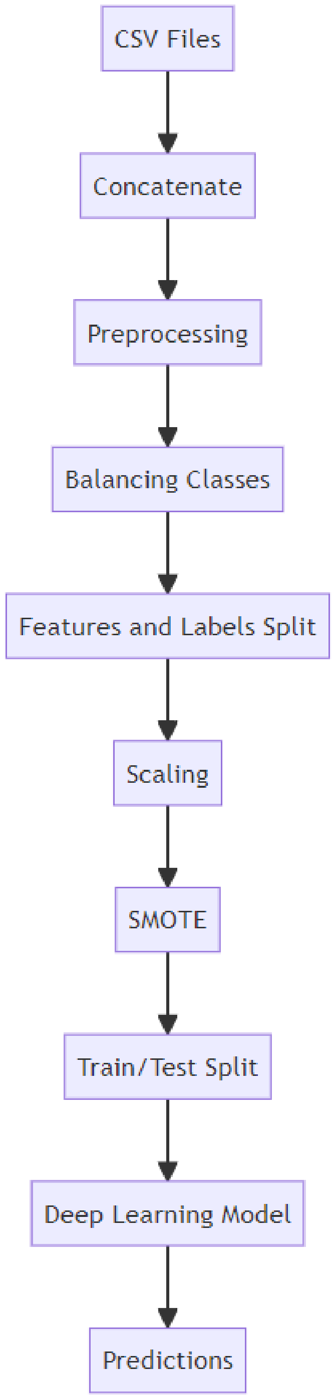
Component Diagram



Deployment diagram: The deployment diagram visualizes the physical hardware on which the software will be deployed.



DFD diagram:


CSV Files
Concatenate
Preprocessing
Balancing Classes
Features and Labels Split
Scaling
SMOTE
Train/Test Split
Deep Learning Model
Predictions

CHAPTER-3

SOFTWARE ENVIRONMENT

SOFTWARE ENVIRONMENT

3.1 Python

Below are some facts about Python.

- Python is currently the most widely used multi-purpose, high-level programming language.
- Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
- Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
- Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard libraries which can be used for the following

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

Advantages of Python

Let's see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet of Things. This is a way to connect the language with the real world.

6. Simple and Easy

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. These further aids the readability of the code.

8. Object-Oriented

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

9. Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn’t the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.
Any doubts till now in the advantages of Python? Mention in the comment section.

Advantages of Python Over Other Languages

1. **Less Coding**

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don’t have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. **Affordable**

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.
The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. **Python is for Everyone**

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we’ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let’s now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

History of Python

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So, I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

Python Development Steps

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt. sources in February 1991. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it. Some changes in Python 7.3:

- Print is now a function.
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e., int. long is int as well.
- The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behavior.
- Text Vs. Data Instead of Unicode Vs. 8-bit

Purpose

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Modules Used in Project

TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

NumPy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.
It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

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

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

Pandas

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

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.
For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

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

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

3.2 Python Installation

Install Python Step-by-Step in Windows and Mac

Python a versatile programming language doesn’t come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

How to Install Python on Windows and Mac

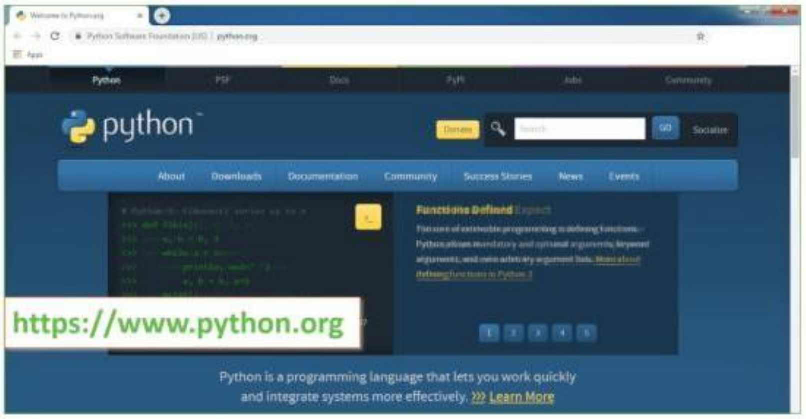
There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e., operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So, the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheat sheet here. The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

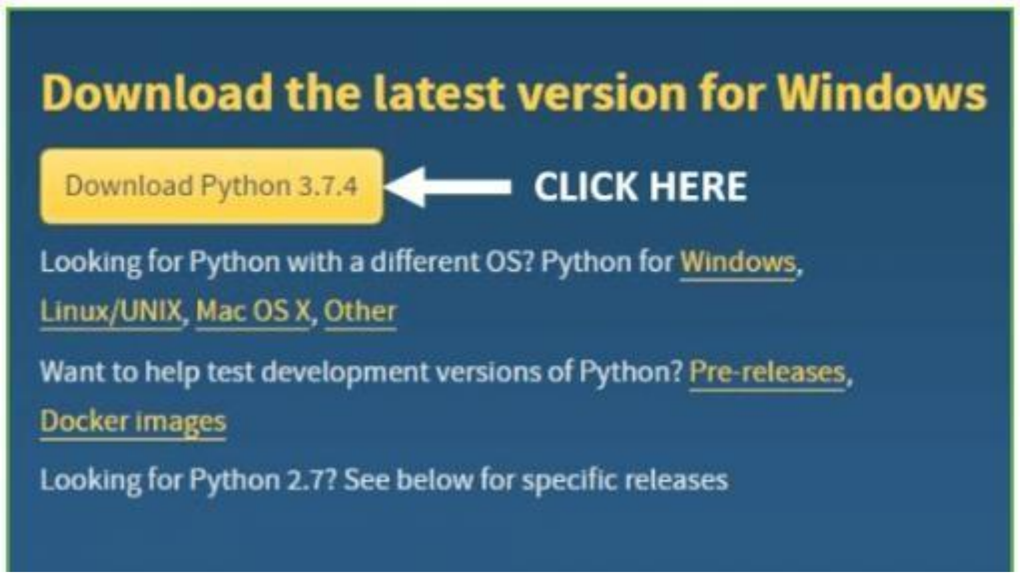
Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.org


python
About
Downloads
Documentation
Community
Success Stories
News
Events
https://www.python.org
Python is a programming language that lets you work quickly
and integrate systems more effectively. >>> Learn More

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.


Download the latest version for Windows

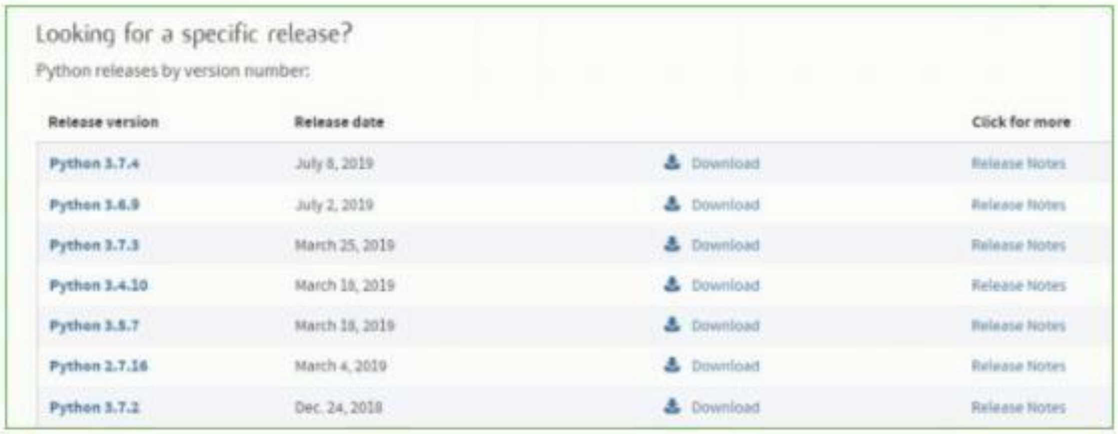
Download Python 3.7.4 ← CLICK HERE

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases, Docker images

Looking for Python 2.7? See below for specific releases

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

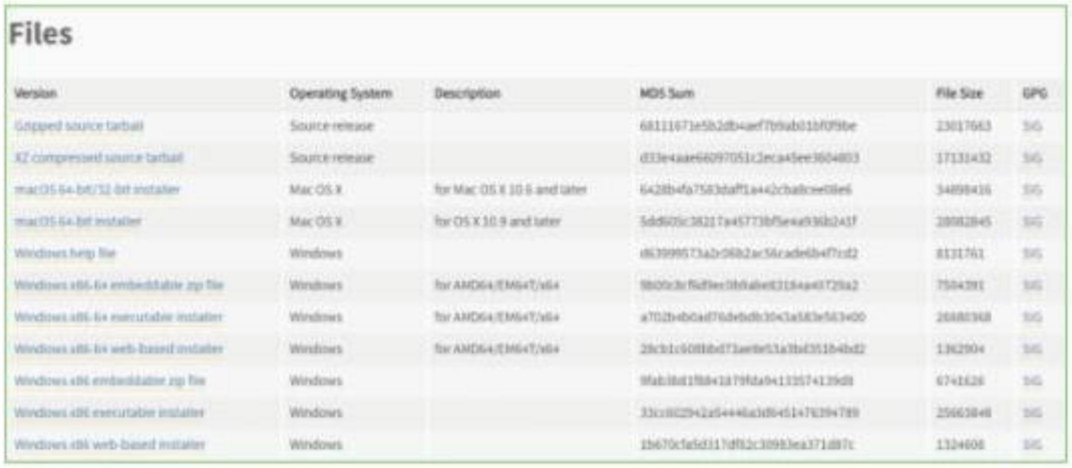


Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Step 4: Scroll down the page until you find the Files option.
Step 5: Here you see a different version of python along with the operating system.



Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		68111671e5b2db4aef7b9ab01bf0f9be	23017663	SIG
XZ compressed source tarball	Source release		d33e4aae66097051c2eca45ee3604803	17131432	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa7583daff1a442cba8cee08e6	34898416	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a45773bf5e4a936b241f	28082845	SIG
Windows help file	Windows		d63999573a2c06b2ac56cade6b4f7cd2	8131761	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9b00c8cf6d9ec0b9abe83184a40729a2	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76debdb3043a583e563400	26680368	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28cb1c608bbd73ae8e53a3bd351b4bd2	1362904	SIG
Windows x86 embeddable zip file	Windows		9fab3b81f8841879fda94133574139d8	6741626	SIG
Windows x86 executable installer	Windows		33cc602942a54446a3d6451476394789	25663848	SIG
Windows x86 web-based installer	Windows		1b670cfa5d317df82c30983ea371d87c	1324608	SIG

- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.

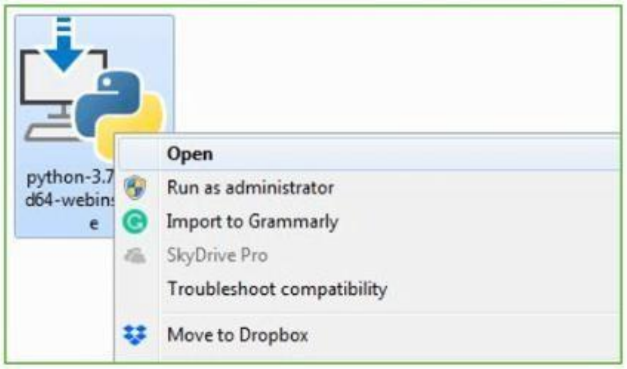
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e., Installation

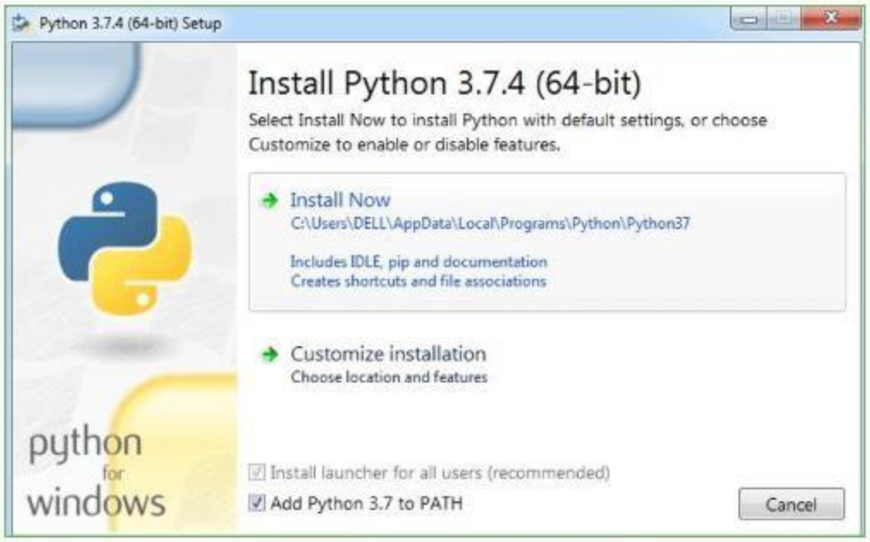
Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process



Step 2: Before you click on Install Now, make sure to put a tick on Add Python 3.7 to PATH.


Python 3.7.4 (64-bit) Setup

Install Python 3.7.4 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

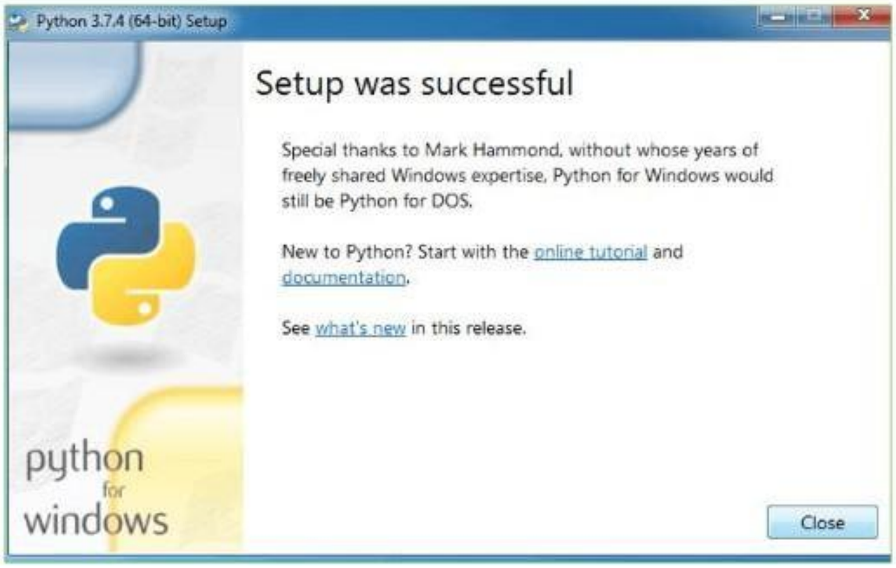
Install Now
C:\Users\DELL\AppData\Local\Programs\Python\Python37

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

[✓] Install launcher for all users (recommended)
[✓] Add Python 3.7 to PATH

Step 3: Click on Install NOW After the installation is successful. Click on Close.


Python 3.7.4 (64-bit) Setup

Setup was successful

Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the online tutorial and documentation.

See what's new in this release.

With these above three steps on python installation, you have successfully and correctly installed

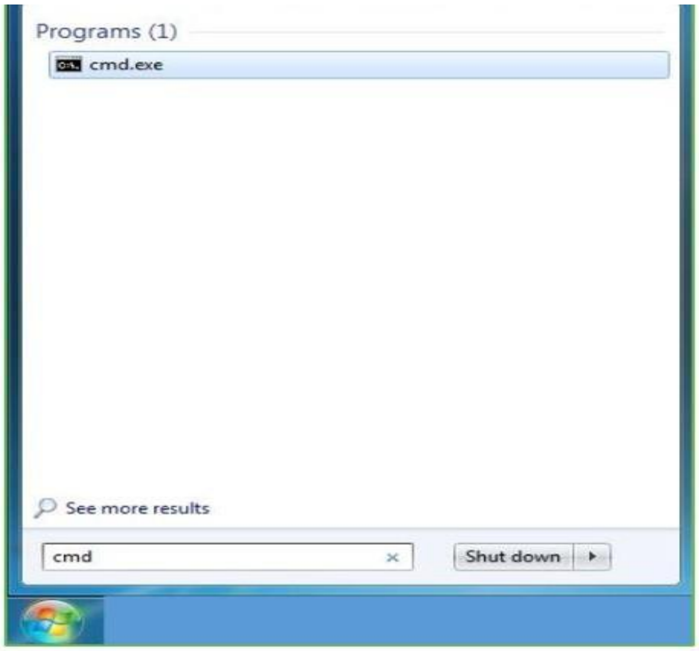
Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

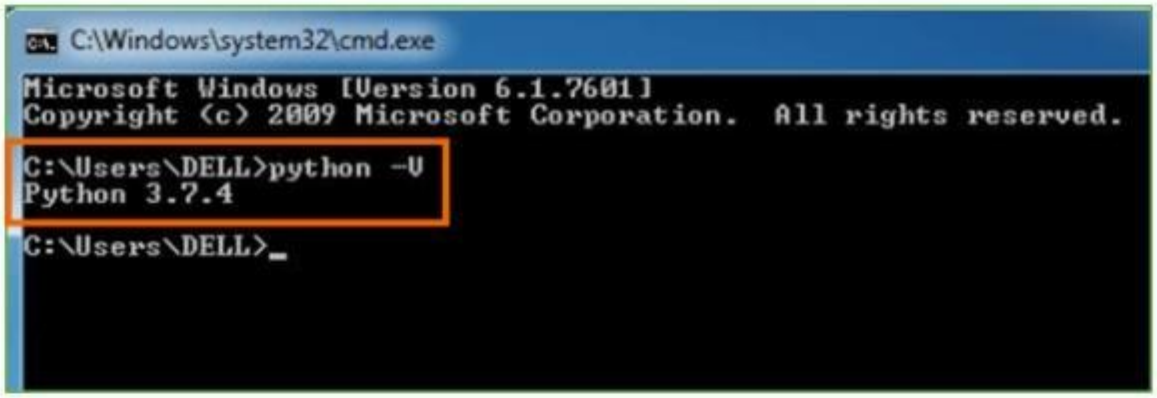
Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python –V and press Enter.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\DELL>python -V
Python 3.7.4

C:\Users\DELL>_
```

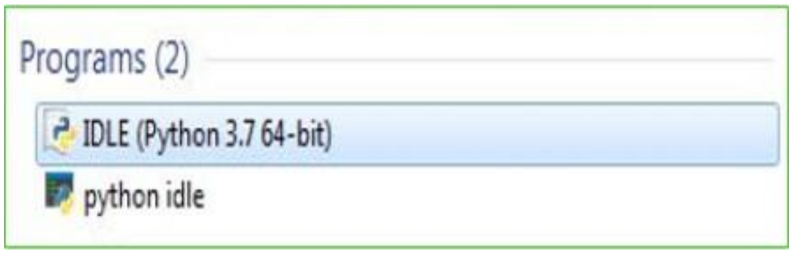
Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

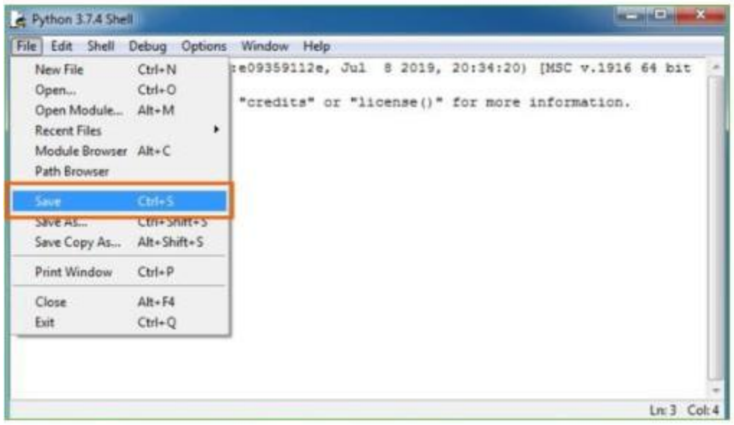
Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.



Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save

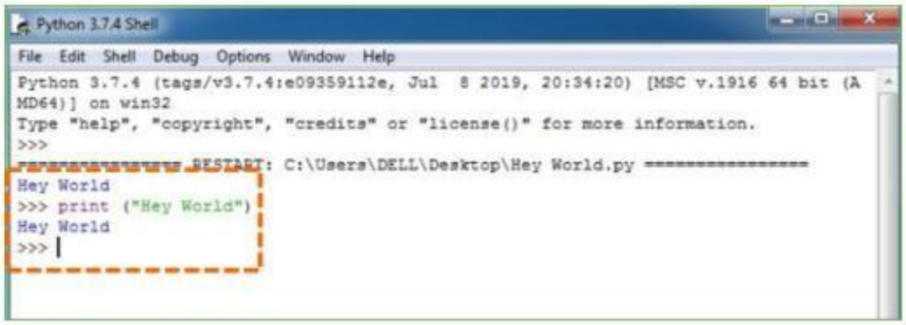


```
:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit

"credits" or "license()" for more information.
```

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g., enter print (“Hey World”) and Press Enter.



```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit (A
MD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================ RESTART: C:\Users\DELL\Desktop\Hey World.py ================
Hey World
>>> print ("Hey World")
Hey World
>>> |
```

You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system. Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won’t work.

CHAPTER-4

REQUIREMENTS & SPECIFICATIONS

4.1 System Requirements Specifications

Software Requirements

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

- Python IDLE 3.7 version (or)
- Anaconda 3.7 (or)
- Jupiter (or)
- Google colab

Product Perspective and Features

The application is developed to operate seamlessly in both standard desktop environments and cloud-based platforms. It incorporates core functionalities such as data visualization, user input/output handling, and efficient file processing. These features ensure that users can interact with the system effectively while also gaining insights from data through graphical representations.

Hardware Requirements

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

Operating system	:	Windows, Linux
Processor	:	minimum intel i3
Ram	:	minimum 4 GB
Hard disk	:	minimum 250GB

4.2 Functional Requirements

Output Design

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

- External Outputs, whose destination is outside the organization
- Internal Outputs whose destination is within organization and they are the
- User’s main interface with the computer.
- Operational outputs whose use is purely within the computer department.
- Interface outputs, which involve the user in communicating directly.

Output Definition

The outputs should be defined in terms of the following points:

- Type of the output
- Content of the output
- Format of the output
- Location of the output
- Frequency of the output
- Volume of the output
- Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

Input Design

Input design is a part of overall system design. The main objective during the input design is as given below:

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

Input Stages

The main input stages can be listed as below:

- Data recording
- Data transcription
- Data conversion
- Data verification
- Data control
- Data transmission
- Data validation
- Data correction

Input Types

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

- External inputs, which are prime inputs for the system.
- Internal inputs, which are user communications with the system.
- Operational, which are computer department's communications to the system?
- Interactive, which are inputs entered during a dialogue.

Input Media

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

- Type of input
- Flexibility of format
- Speed
- Accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Storage and handling requirements
- Security
- Easy to use
- Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive.

Input data is to be the directly keyed in by the user, the keyboard can suitable to the input device.

Error Avoidance

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

Error Detection

Even though every effort is made to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

Data Validation

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

User Interface Design

It is essential to consult the system users and discuss their needs while designing the user interface:

User Interface Systems Can Be Broadly Classified As:

- User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
- Computer initiated interfaces

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

User Initiated Interfaces

User initiated interfaces fall into two approximate classes:

- Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
- Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form.
- The forms-oriented interface is chosen because it is the best choice.

Computer-Initiated Interfaces

The following computer – initiated interfaces were used:

- The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
- Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

Error Message Design

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

Performance Requirements

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be able to interface with the existing system
- The system should be accurate
- The system should be better than the existing system
- The existing system is completely dependent on the user to perform all the duties.

4.3 Source code

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder


# In[2]:


dataset=pd.read_csv(r"C:\Users\Varsha Reddy\OneDrive\Desktop\cyber attack\UNSW_NB.csv")


# In[3]:


dataset


# In[4]:


labels=set(dataset['attack_cat'])
labels


# In[5]:


dataset.isnull().sum()
```

```
# In[6]:



lab=LabelEncoder()



# In[7]:


dataset['attack_cat']=lab.fit_transform(dataset['attack_cat'])
dataset['service']=lab.fit_transform(dataset['service'])
dataset['proto']=lab.fit_transform(dataset['proto'])
dataset['state']=lab.fit_transform(dataset['state'])



# In[8]:



dataset



# In[9]:



x=dataset.iloc[:,0:43]
x



# In[10]:



y=dataset.iloc[:,43]
y
```

```
# In[ ]:



# In[11]:



from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
print("x shape is",x.shape)
print("y shape is",y.shape)
print("x_test shape is",x_test.shape)
print("x_train shape is",x_train.shape)
print("y_test shape is",y_test.shape)
print("y_train shape is",y_train.shape)



# In[ ]:





# In[12]:



df=pd.DataFrame(dataset)



# In[13]:



print("Original DataFrame:\n",df,"\n")
```

```
# In[14]:


result=df.applymap(np.isreal)


# In[15]:


print("Result:\n",result)


# In[16]:


import seaborn as sns
import matplotlib.pyplot as plt
plot=sns.countplot(data=dataset,x=("attack_cat"))
value_counts=dataset["attack_cat"].value_counts()
for i,count in enumerate(value_counts):
    plot.text(x=i,y=count+1,s=str(count),ha="center")
plot.set_xlabel("attack_cat")
plot.set_title("Count Plot of Cyber Attacks")
plt.show()


# In[17]:



from sklearn.metrics import accuracy_score

from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
y_pred = dt.predict(x_test)
print("Original y_test values are ",y_test)
print("predicted y_test values are",y_pred)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
print(f'Accuracy: {accuracy:.2f}')
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1 Score: {f1:.2f}')
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8,6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues' , xticklabels=labels, yticklabels=labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix of DTC')
plt.show()
class_report = classification_report(y_test, y_pred)
print('Classification Report of DTC:')
print(class_report)


# In[18]:
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
y_pred = rf.predict(x_test)
print("Original y_test values are ",y_test)
print("predicted y_test values are",y_pred)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
print(f'Accuracy: {accuracy:.2f}')
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1 Score: {f1:.2f}')
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8,6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',xticklabels=labels, yticklabels=labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix of RFC')
plt.show()
class_report = classification_report(y_test, y_pred)
print('Classification Report of RFC:')
print(class_report)


# In[22]:
```

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
LR=LogisticRegression()
LR.fit(x_train,y_train)
y_pred = LR.predict(x_test)
print("Original y_test values are ",y_test)
print("predicted y_test values are",y_pred)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
print(f'Accuracy: {accuracy:.2f}')
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1 Score: {f1:.2f}')
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8,6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',xticklabels=labels, yticklabels=labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix of LR')
plt.show()
class_report = classification_report(y_test, y_pred)
print('Classification Report of LR:')
print(class_report)
```

CHAPTER-5
RESULT AND DISCUSSION

RESULT AND DISCUSSION

5.1 Implementation description

This Python code is a machine learning pipeline for a classification task using a deep learning model.

- Importing Libraries:

The necessary libraries and modules are imported. These include libraries for data manipulation (Pandas), visualization (Matplotlib, Seaborn), regular expressions (re), date handling (datetime), and machine learning tools (NumPy, Scikit-learn, TensorFlow).

- Combining CSV Files:

The code reads multiple CSV files using the glob module that match a specified pattern (files in the cicids2017 directory). It then concatenates them into a single Data Frame named dataset.

- Data Preprocessing:

Duplicates are dropped from the dataset, and the index is reset.

Column names are cleaned by removing spaces.

A specific column called "FwdHeaderLength.1" is dropped.

Features (input data) are selected, and a target variable y is extracted.

- Label Cleaning:

Labels in the dataset are cleaned by removing special characters and spaces, and replacing spaces with underscores.

- Data Type and Value Fixing:

The data types of columns (except the label column) are converted to float64. Any occurrences of infinity or negative infinity are replaced with NaN, and rows with NaN values are dropped.

- Class Imbalance Handling:

The code identifies and addresses class imbalance in the target variable by removing a specific subset of data.

- Feature and Target Label Splitting:

The features (input data) are stored in features, and the target labels are stored in labels.

- Data Scaling:

The features are standardized using StandardScaler.

- Data Splitting:

The data is split into training and testing sets using the train_test_split function.

- RFC and ML Model: The model is compiled with the Adam optimizer and sparse categorical cross- entropy loss function. It is then trained on the training data for 10 epochs. Early stopping and model checkpointing callbacks are used to monitor and save the best model based on validation loss.
- Performance Metrics:

If the model was trained, it calculates and prints accuracy, classification report, and plots a confusion matrix.

- Please note that the code contains some placeholders like your model, which should be replaced with meaningful names. Additionally, there are some commented out sections (#CLEANING, #Standard scalar) that might be part of the code but are currently not explained in the provided context.

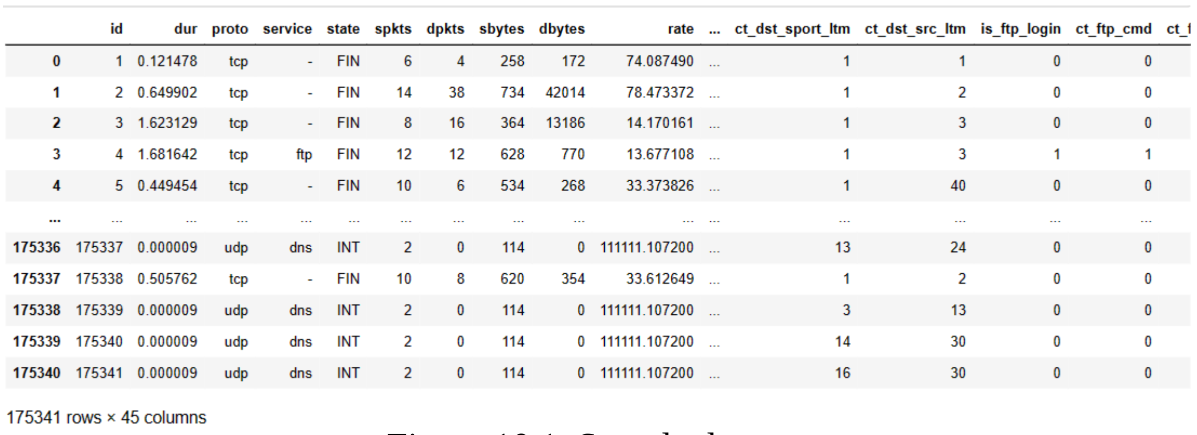
5.2 Dataset Description

The UNSW-NB15 dataset is a network traffic dataset that is commonly used for cyber-attack detection system evaluation. Here's a brief description of each column in the dataset:

- **id**: A unique identifier for each record in the dataset.
- **dur**: Duration of the connection in seconds. Represents the time elapsed for the connection.
- **proto**: The transport layer protocol used in the connection, such as TCP, UDP, ICMP, etc.
- **service**: The network service on the destination, indicating the type of service being accessed (e.g., HTTP, FTP).
- **state**: The connection state, indicating the status of the connection (e.g., FIN, CON, INT).
- **spkts**: The count of packets sent from the source to the destination.
- **dpkts**: The count of packets sent from the destination to the source.
- **sbytes**: The number of source-to-destination bytes in the connection.
- **dbytes**: The number of destination-to-source bytes in the connection.
- **rate**: Data transfer rate, representing the average number of bits transferred per second.
- **sttl**: Source time to live, indicating the remaining time to live of the source in the connection.
- **dttl**: Destination time to live, indicating the remaining time to live of the destination in the connection.
- **sload**: Source bits per second, representing the data load on the source side.
- **dload**: Destination bits per second, representing the data load on the destination side.
- **sloss**: Source packets retransmitted or lost during the connection.
- **dloss**: Destination packets retransmitted or lost during the connection.
- **sinpkt**: Source inter-packet arrival time, indicating the time between consecutive packets from the source.

- **dinpkt**: Destination inter-packet arrival time, indicating the time between consecutive packets from the destination.
- **sjit**: Source jitter, representing the variability in inter-packet arrival times on the source side.
- **djit**: Destination jitter, representing the variability in inter-packet arrival times on the destination side.
- **swin**: Source TCP window size, indicating the size of the TCP window on the source side.
- **stcpb**: Source TCP base sequence number, indicating the initial sequence number of the connection on the source side.
- **dtcpb**: Destination TCP base sequence number, indicating the initial sequence number of the connection on the destination side.
- **dwin**: Destination TCP window size, indicating the size of the TCP window on the destination side.
- **tcprtt**: TCP connection setup round-trip time, representing the time taken for the TCP connection setup.
- **synack**: Time taken for the TCP connection setup (SYN-ACK phase).
- **ackdat**: Time taken for the TCP connection setup (ACK-DAT phase).
- **smean**: Mean of the source payload data size, representing the average size of data sent from the source.
- **dmean**: Mean of the destination payload data size, representing the average size of data received at the destination.
- **trans_depth**: Connection transaction depth, indicating the depth of the transaction in the connection.
- **response_body_len**: Length of the response body, indicating the size of the response payload.
- **ct_srv_src**: Number of connections to the same service as the current connection in the past two seconds.
- **ct_state_ttl**: Number of connections with the same source TTL (Time to Live) value.

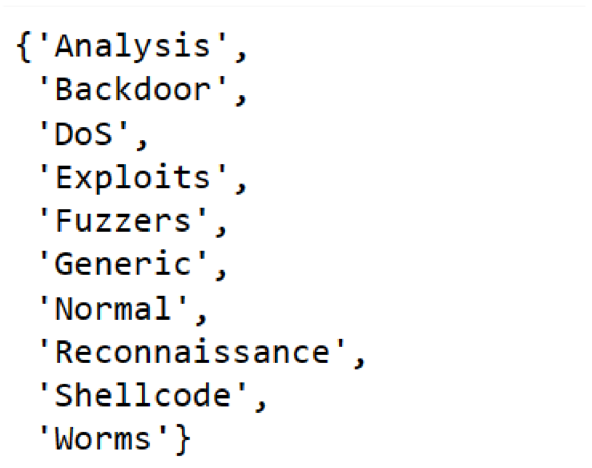
- **ct_dst_ltm**: Number of connections with the same destination IP address in the past two seconds.
- **ct_src_dport_ltm**: Number of connections with the same source port to the same destination port in the past two seconds.
- **ct_dst_sport_ltm**: Number of connections with the same destination port to the same source port in the past two seconds.
- **ct_dst_src_ltm**: Number of connections with the same source and destination IP addresses in the past two seconds.
- **is_ftp_login**: Binary indicator of whether the FTP login was successful or not.
- **ct_ftp_cmd**: Number of FTP commands carried in the connection.
- **ct_flw_http_mthd**: Number of HTTP methods carried in the connection.
- **ct_src_ltm**: Number of connections with the same source IP address in the past two seconds.
- **ct_srv_dst**: Number of connections with the same source and destination service in the past two seconds.
- **is_sm_ips_ports**: Binary indicator of whether the source and destination ports are the same.
- **attack_cat**: The category of the attack, such as DoS (Denial of Service), Probe, R2L (Unauthorized access from a remote machine), U2R (Unauthorized access to privileged local resources).
- **label**: Binary label indicating normal (0) or malicious (1) activity in the network connection.



	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp_cmd	ct_f
0	1	0.121478	tcp	-	FIN	6	4	258	172	74.087490	...	1	1	0	0	
1	2	0.649902	tcp	-	FIN	14	38	734	42014	78.473372	...	1	2	0	0	
2	3	1.623129	tcp	-	FIN	8	16	364	13186	14.170161	...	1	3	0	0	
3	4	1.681642	tcp	ftp	FIN	12	12	628	770	13.677108	...	1	3	1	1	
4	5	0.449454	tcp	-	FIN	10	6	534	268	33.373826	...	1	40	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
175336	175337	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	13	24	0	0	
175337	175338	0.505762	tcp	-	FIN	10	8	620	354	33.612649	...	1	2	0	0	
175338	175339	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	3	13	0	0	
175339	175340	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	14	30	0	0	
175340	175341	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	16	30	0	0	

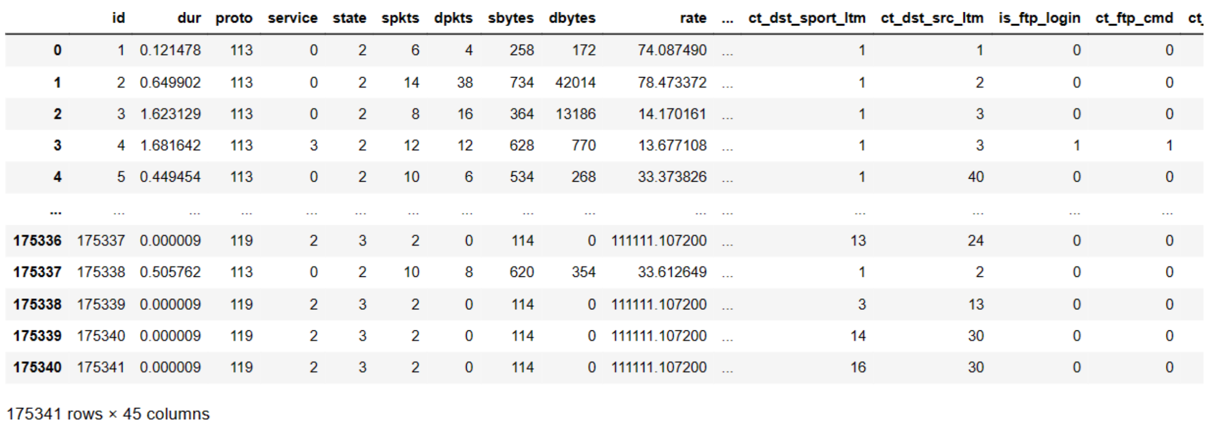
175341 rows × 45 columns

Figure 10.1. Sample dataset.



```
{'Analysis',
 'Backdoor',
 'DoS',
 'Exploits',
 'Fuzzers',
 'Generic',
 'Normal',
 'Reconnaissance',
 'Shellcode',
 'Worms'}
```

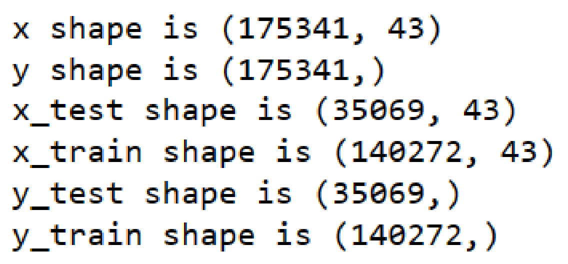
Figure 10.2. Attack classes in dataset.



	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp_cmd	ct_
0	1	0.121478	113	0	2	6	4	258	172	74.087490	...	1	1	0	0	
1	2	0.649902	113	0	2	14	38	734	42014	78.473372	...	1	2	0	0	
2	3	1.623129	113	0	2	8	16	364	13186	14.170161	...	1	3	0	0	
3	4	1.681642	113	3	2	12	12	628	770	13.677108	...	1	3	1	1	
4	5	0.449454	113	0	2	10	6	534	268	33.373826	...	1	40	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
175336	175337	0.000009	119	2	3	2	0	114	0	111111.107200	...	13	24	0	0	
175337	175338	0.505762	113	0	2	10	8	620	354	33.612649	...	1	2	0	0	
175338	175339	0.000009	119	2	3	2	0	114	0	111111.107200	...	3	13	0	0	
175339	175340	0.000009	119	2	3	2	0	114	0	111111.107200	...	14	30	0	0	
175340	175341	0.000009	119	2	3	2	0	114	0	111111.107200	...	16	30	0	0	

175341 rows × 45 columns

Figure 10.3. Dataset after label encoding.



```
x shape is (175341, 43)
y shape is (175341,)
x_test shape is (35069, 43)
x_train shape is (140272, 43)
y_test shape is (35069,)
y_train shape is (140272,)
```

Figure 10.4. Train and Test sizes in dataset.

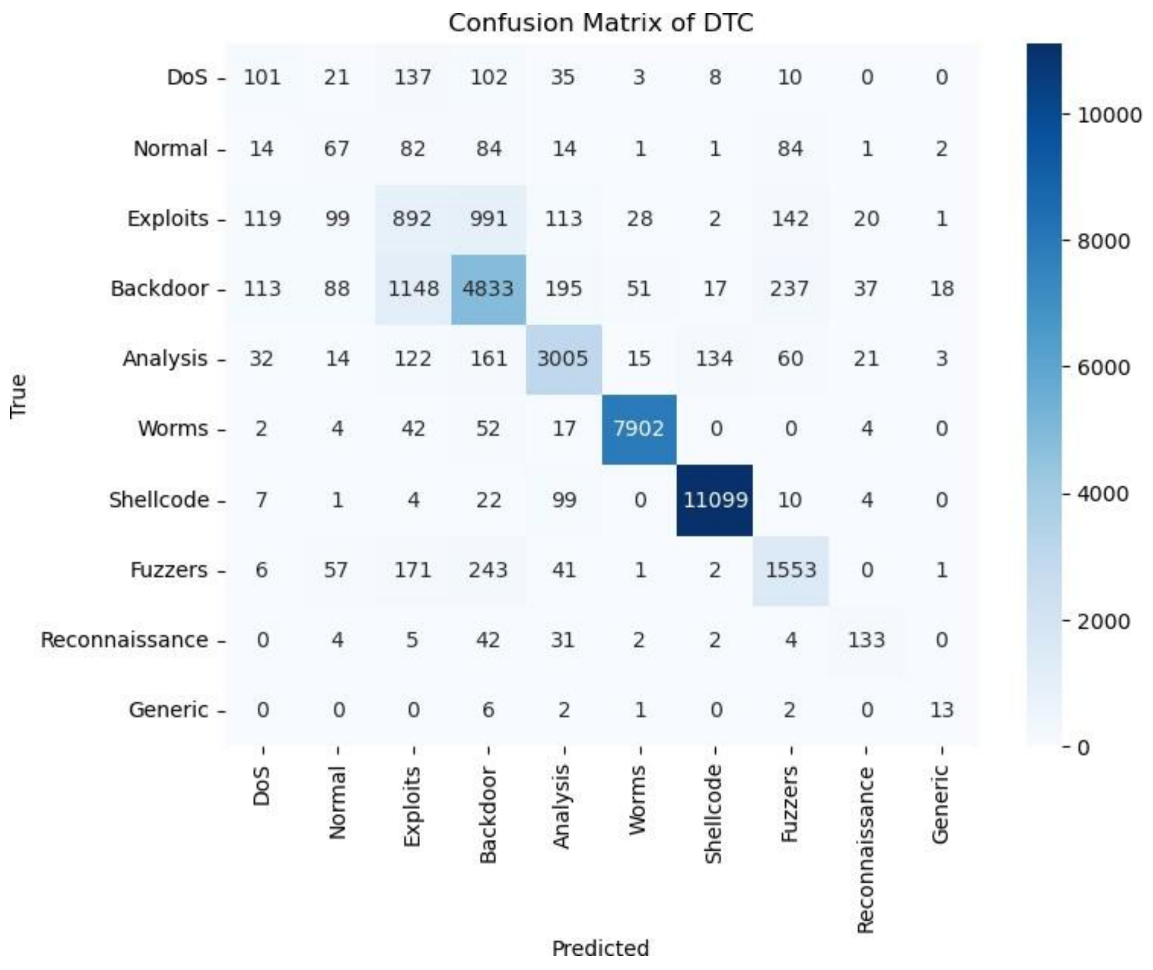
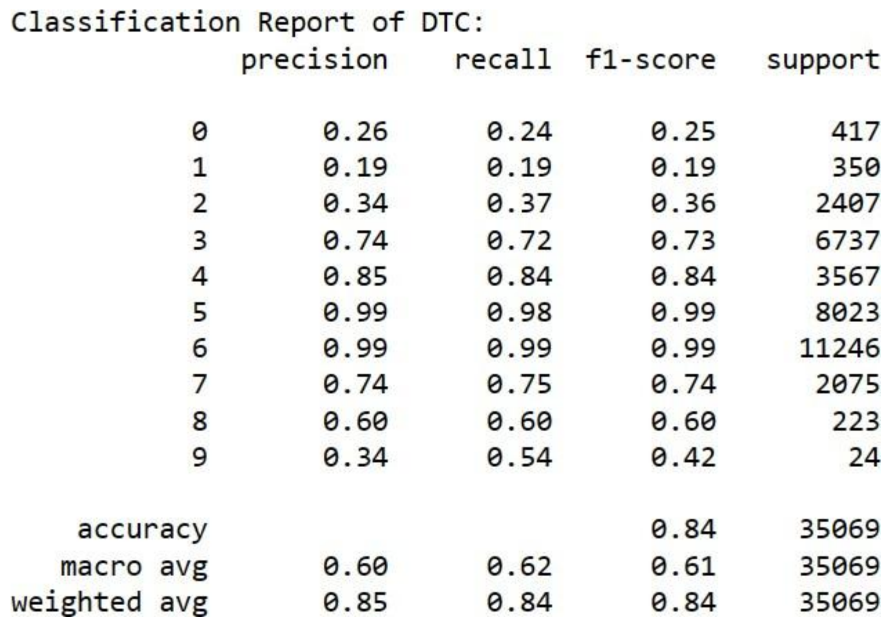


Figure 10.5. Existing DTC confusion matrix.



```
Classification Report of DTC:
              precision    recall  f1-score   support

           0       0.26      0.24      0.25       417
           1       0.19      0.19      0.19       350
           2       0.34      0.37      0.36      2407
           3       0.74      0.72      0.73      6737
           4       0.85      0.84      0.84      3567
           5       0.99      0.98      0.99      8023
           6       0.99      0.99      0.99     11246
           7       0.74      0.75      0.74      2075
           8       0.60      0.60      0.60       223
           9       0.34      0.54      0.42        24

    accuracy                           0.84     35069
   macro avg       0.60      0.62      0.61     35069
weighted avg       0.85      0.84      0.84     35069
```

Figure 10.6. Existing DTC classification report.

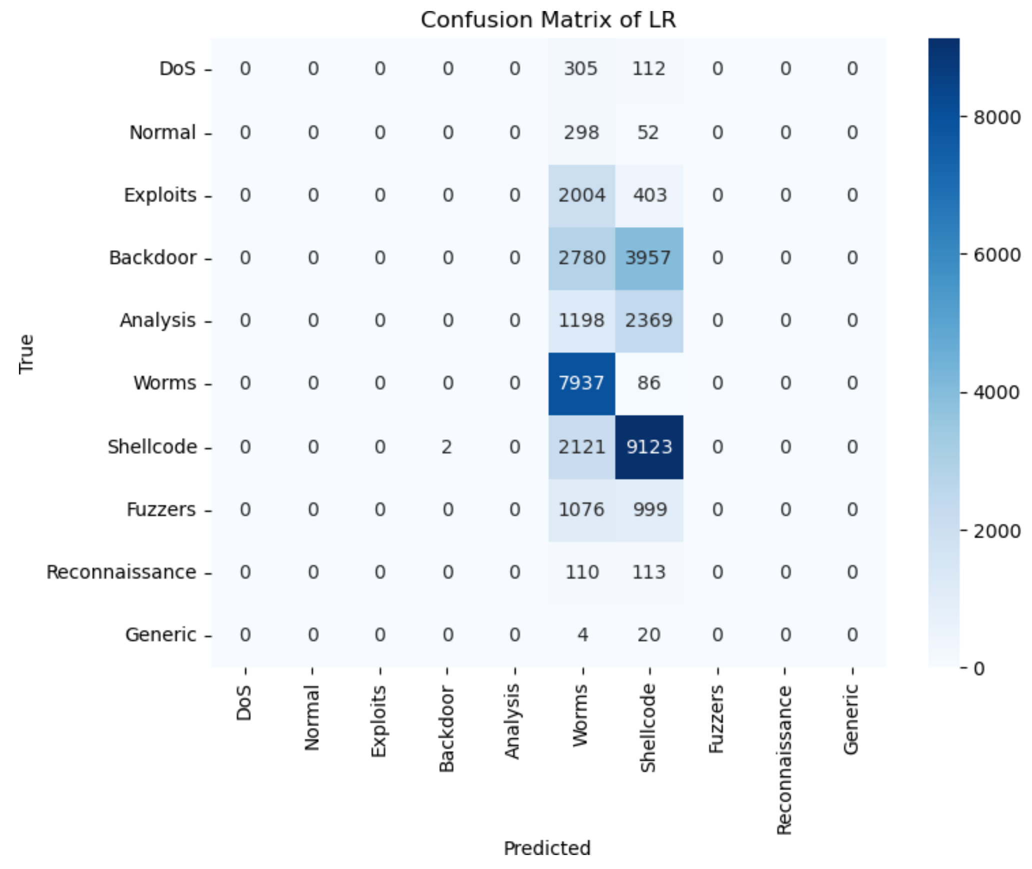
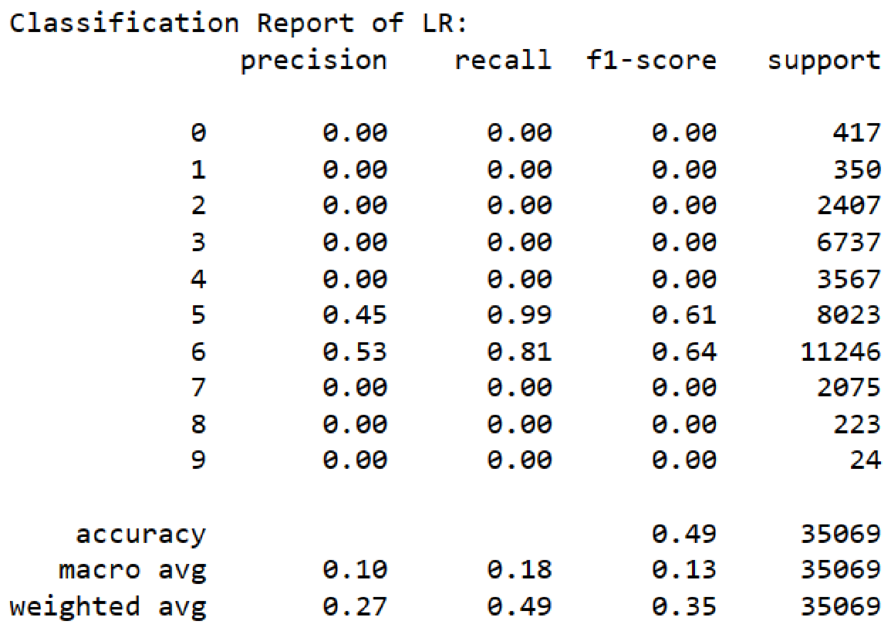


Figure 10.7. Existing LRC confusion matrix.



```
Classification Report of LR:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       417
           1       0.00      0.00      0.00       350
           2       0.00      0.00      0.00      2407
           3       0.00      0.00      0.00      6737
           4       0.00      0.00      0.00      3567
           5       0.45      0.99      0.61      8023
           6       0.53      0.81      0.64     11246
           7       0.00      0.00      0.00      2075
           8       0.00      0.00      0.00       223
           9       0.00      0.00      0.00        24

    accuracy                           0.49     35069
   macro avg       0.10      0.18      0.13     35069
weighted avg       0.27      0.49      0.35     35069
```

Figure 10.8. Existing LRC classification report.

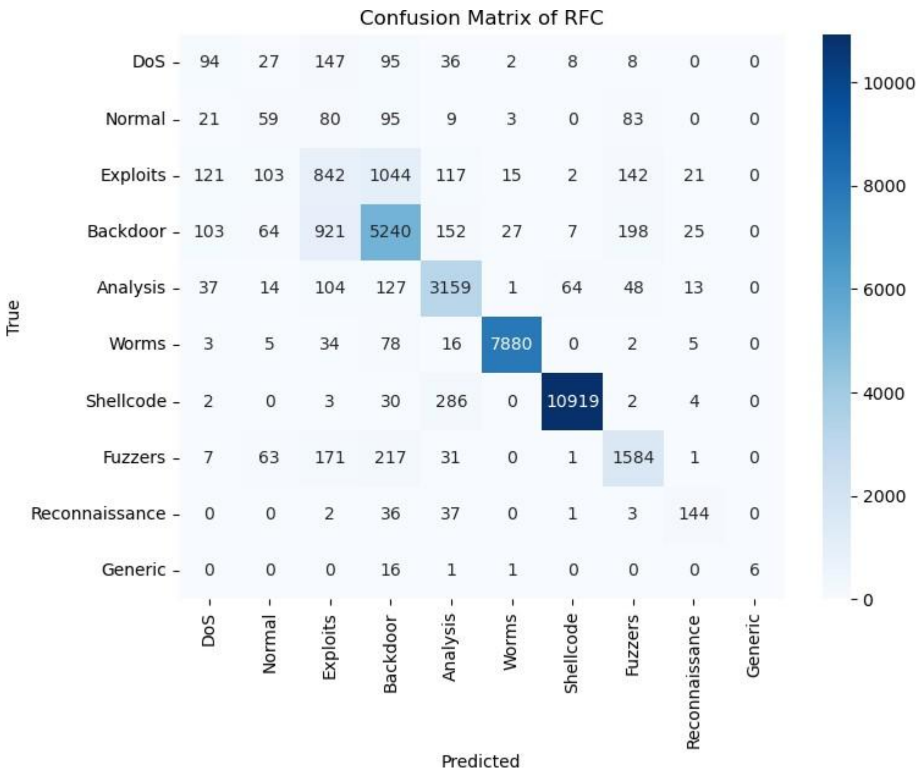
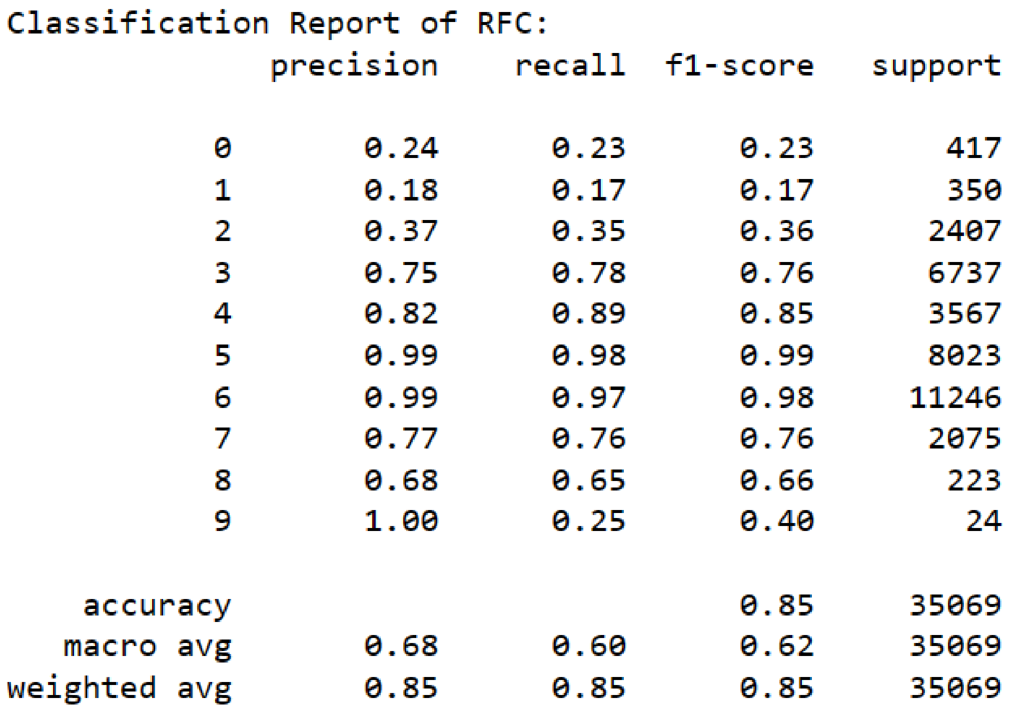


Figure 10.8. Proposed RFC confusion matrix.


```
Classification Report of RFC:
              precision    recall  f1-score   support

           0       0.24      0.23      0.23       417
           1       0.18      0.17      0.17       350
           2       0.37      0.35      0.36      2407
           3       0.75      0.78      0.76      6737
           4       0.82      0.89      0.85      3567
           5       0.99      0.98      0.99      8023
           6       0.99      0.97      0.98     11246
           7       0.77      0.76      0.76      2075
           8       0.68      0.65      0.66       223
           9       1.00      0.25      0.40        24

    accuracy                           0.85     35069
   macro avg       0.68      0.60      0.62     35069
weighted avg       0.85      0.85      0.85     35069
```

CHAPTER-6

CONCLUSION AND FUTURE SCOPE

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

The approach of using standard scaling and a RFC classifier for cyber-attack detection is a promising one. It leverages advanced machine learning techniques to identify malicious activities in network traffic or system logs. By preprocessing the data effectively and training a RFC model, it is possible to achieve accurate and timely detection of cyber threats. However, it's important to note that the effectiveness of such a system depends on various factors, including the quality and diversity of the training data, the design of the RFC architecture, and the continuous monitoring and updating of the model.

6.2 Future scope

The field of cyber-attack detection is dynamic, and there are several avenues for future research and development:

— **Adversarial Attack Detection:** As cyber attackers become more sophisticated, there's a growing need to develop models that are robust against adversarial attacks. Future work may focus on improving the resilience of DLCNN-based models to evasion and poisoning attacks.

— **Explainable AI (XAI):** Enhancing the interpretability and explainability of DLCNN models for cyber-attack detection is crucial. Researchers can work on developing methods to make these models more transparent and understandable, which is essential for building trust and making informed decisions.

— **Streaming Data and Real-Time Detection:** With the increasing speed and volume of data generated in network environments, there's a need for real-time and streaming data processing techniques. Future research may involve developing DLCNN models and preprocessing methods that can handle data on the fly.

— **Anomaly Detection and Zero-Day Attacks:** Cyber attackers frequently employ novel techniques (zero-day attacks) that have not been seen before. Future work can focus on improving the ability of DLCNN models to detect anomalies and previously unseen attack patterns.

— **Integration with Security Orchestration Tools:** Integrating cyber-attack detection systems with security orchestration tools can enhance incident response capabilities. Future scope may involve developing seamless integrations with security information and event management (SIEM) systems and other cybersecurity solutions.

- **IoT and Edge Computing Security:** With the proliferation of Internet of Things (IoT) devices and edge computing, there's a growing need for cybersecurity solutions that can operate at the edge. Future research can explore DLCNN-based models tailored for edge devices and IoT security.

- **Data Privacy and Compliance:** As data privacy regulations continue to evolve, future work should consider the ethical and legal implications of cyber-attack detection systems, ensuring compliance with regulations like GDPR and CCPA.

- **Hybrid Models:** Combining DLCNN-based approaches with traditional rule-based or signature-based methods can lead to more robust and accurate detection systems. Future research may explore hybrid models that leverage the strengths of both approaches.

REFERENCES

[1] S. Dange and M. Chatterjee, "Iot botnet: The largest threat to the iot network" in Data Communication and Networks, Cham, Switzerland:Springer, pp. 137-157, 2020.

[2] J. Ceron, K. Steding-Jessen, C. Hoepers, L. Granville and C. Margi, "Improving IoT botnet investigation using an adaptive network layer", Sensors, vol. 19, no. 3, pp. 727, Feb. 2019.

[3] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, et al., "N-baiot-network-based detection of iot botnet attacks using deep autoencoders", IEEE Pervas. Comput., vol. 17, no. 3, pp. 12-22, 2018.

[4] Shah, S.A.R.; Issac, B. Performance comparison of intrusion detection systems and application of machine learning to Snort system. Futur. Gener. Comput. Syst. 2018, 80, 157–170.

[5] Soe YN, Feng Y, Santosa PI, Hartanto R, Sakurai K. Machine Learning-Based IoT-Botnet Attack Detection with Sequential Architecture. Sensors. 2020; 20(16):4372. https://doi.org/10.3390/s20164372

[6] I. Ali et al., "Systematic Literature Review on IoT-Based Botnet Attack," in IEEE Access, vol. 8, pp. 212220-212232, 2020, doi: 10.1109/ACCESS.2020.3039985.

[7] Irfan, I. M. Wildani and I. N. Yulita, "Classifying botnet attack on Internet of Things device using random forest", IOP Conf. Ser. Earth Environ. Sci., vol. 248, Apr. 2019.

[8] Shah, T., Venkatesan, S. (2019). A Method to Secure IoT Devices Against Botnet Attacks. In: Issarny, V., Palanisamy, B., Zhang, LJ. (eds) Internet of Things – ICIOT 2019. ICIOT 2019. Lecture Notes in Computer Science(), vol 11519. Springer, Cham. https://doi.org/10.1007/978-3-030-23357-0_3

[9] C. Tzagkarakis, N. Petroulakis and S. Ioannidis, "Botnet Attack Detection at the IoT Edge Based on Sparse Representation," 2019 Global IoT Summit (GIoTS), Aarhus, Denmark, 2019, pp. 1-6, doi: 10.1109/GIOTS.2019.8766388.

[10] Y. Meidan et al., "N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders," in IEEE Pervasive Computing, vol. 17, no. 3, pp. 12-22, Jul.-Sep. 2018, doi: 10.1109/MPRV.2018.03367731.

[11] S. I. Popoola, R. Ande, B. Adebisi, G. Gui, M. Hammoudeh and O. Jogunola, "Federated Deep Learning for Zero-Day Botnet Attack Detection in IoT-Edge Devices," in IEEE Internet of Things Journal, vol. 9, no. 5, pp. 3930-3944, 1 March1, 2022, doi: 10.1109/JIOT.2021.3100755.

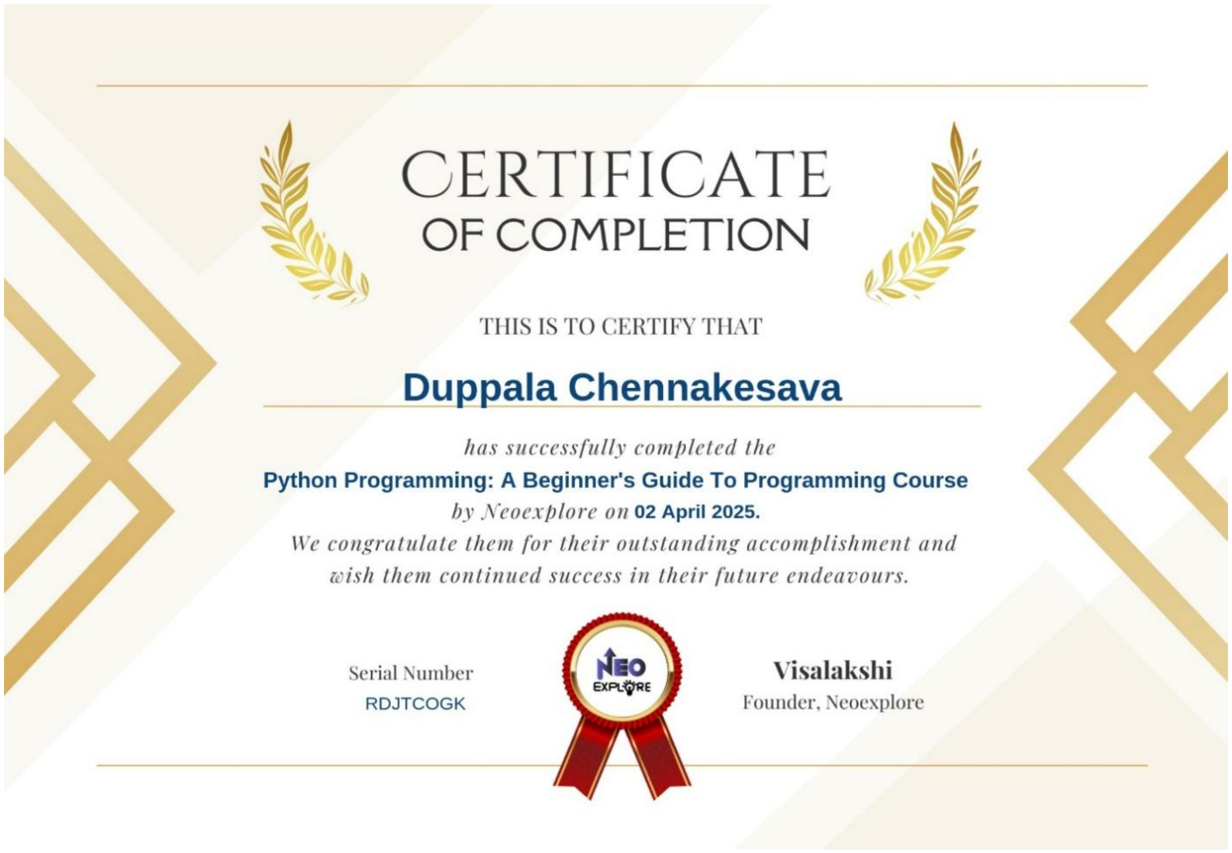
[12] F. Hussain et al., "A Two-Fold Machine Learning Approach to Prevent and Detect IoT Botnet Attacks," in IEEE Access, vol. 9, pp. 163412-163430, 2021, doi: 10.1109/ACCESS.2021.3131014.

[13] Abu Al-Haija Q, Al-Dala'ien M. ELBA-IoT: An Ensemble Learning Model for Botnet Attack Detection in IoT Networks. Journal of Sensor and Actuator Networks. 2022; 11(1):18. https://doi.org/10.3390/jsan11010018

[14] Alharbi A, Alosaimi W, Alyami H, Rauf HT, Damaševičius R. Botnet Attack Detection Using Local Global Best Bat Algorithm for Industrial Internet of Things. Electronics. 2021; 10(11):1341. https://doi.org/10.3390/electronics10111341

[15] Ahmed, A.A., Jabbar, W.A., Sadiq, A.S. et al. Deep learning-based classification model for botnet attack detection. J Ambient Intell Human Comput 13, 3457–3466 (2022). https://doi.org/10.1007/s12652-020-01848-9

CERTIFICATIONS



CERTIFICATE
OF COMPLETION

THIS IS TO CERTIFY THAT

Duppala Chennakesava

has successfully completed the

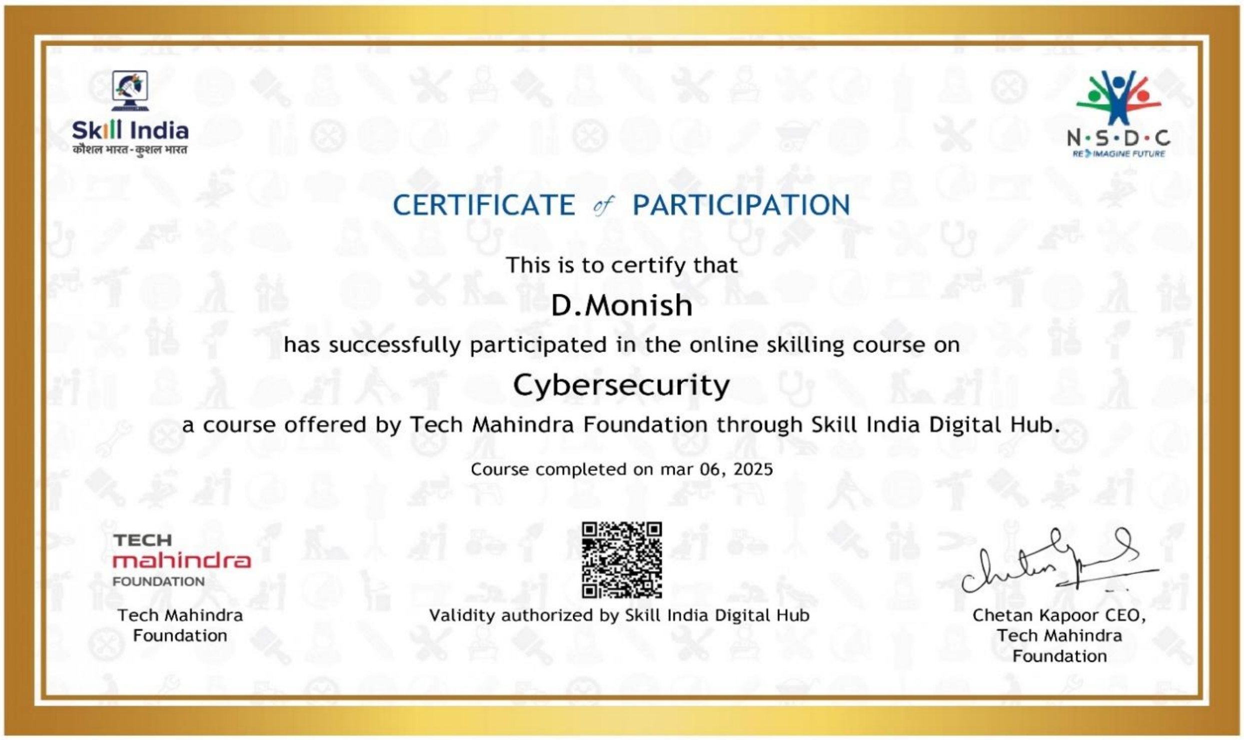
Python Programming: A Beginner's Guide To Programming Course

by Neoexplore on **02 April 2025.**

We congratulate them for their outstanding accomplishment and wish them continued success in their future endeavours.

Serial Number
RDJTCOGK

Visalakshi
Founder, Neoexplore



Skill India
कौशल भारत-कुशल भारत

N·S·D·C
RE IMAGINE FUTURE

CERTIFICATE *of* PARTICIPATION

This is to certify that

D.Monish

has successfully participated in the online skilling course on

Cybersecurity

a course offered by Tech Mahindra Foundation through Skill India Digital Hub.

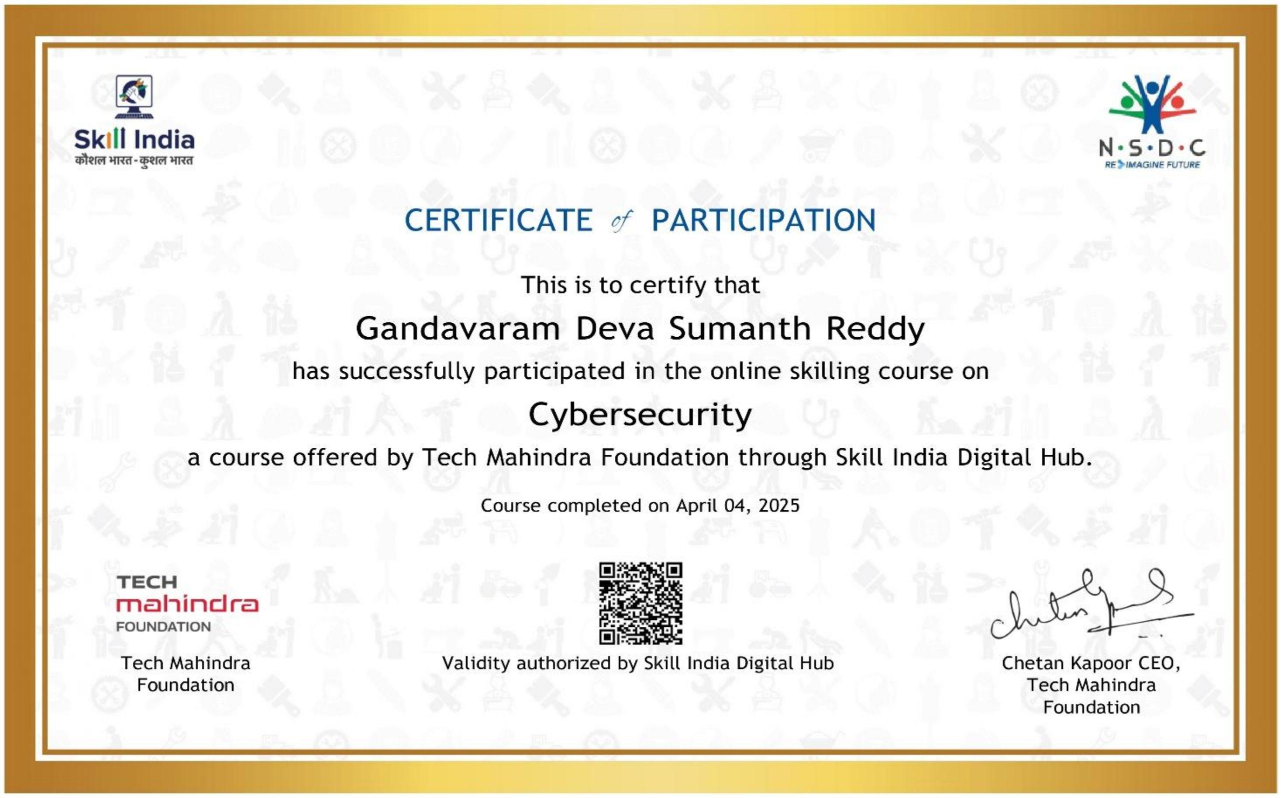
Course completed on mar 06, 2025

Tech Mahindra Foundation

Validity authorized by Skill India Digital Hub

Chetan Kapoor CEO,
Tech Mahindra
Foundation

Certified on: 06/03/2025 09:38



Skill India
कौशल भारत-कुशल भारत

N·S·D·C
RE IMAGINE FUTURE

CERTIFICATE of PARTICIPATION

This is to certify that

Gandavaram Deva Sumanth Reddy

has successfully participated in the online skilling course on

Cybersecurity

a course offered by Tech Mahindra Foundation through Skill India Digital Hub.

Course completed on April 04, 2025

TECH mahindra FOUNDATION

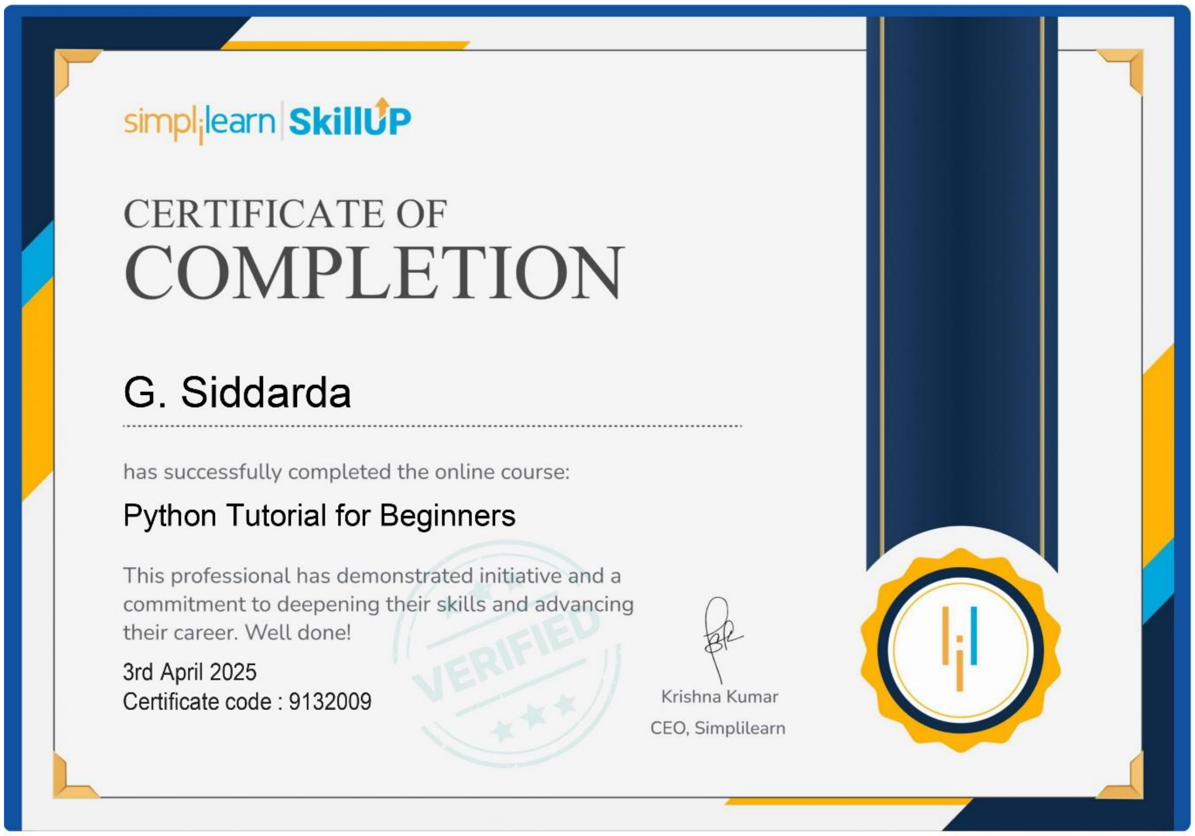
Tech Mahindra Foundation

Validity authorized by Skill India Digital Hub

[illegible]

Chetan Kapoor CEO, Tech Mahindra Foundation

Certified on: 04/04/2025 10:38



simplilearn | SkillUP

CERTIFICATE OF
COMPLETION

G. Siddarda

has successfully completed the online course:

Python Tutorial for Beginners

This professional has demonstrated initiative and a commitment to deepening their skills and advancing their career. Well done!

3rd April 2025
Certificate code : 9132009

VERIFIED

[illegible]

Krishna Kumar
CEO, Simplilearn