A Project Report on

Detecting and Mitigating Botnet Attacks in Software Defined Networks

in partial fulfillment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

submitted by

MUSKUDI NANI BABU
20B91A05J4

Under the esteemed guidance of

Dr V CHANDRA SEKHAR
Head of the Department (CSE)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRKR ENGINEERING COLLEGE (A)

ChinnaAmiram, Bhimavaram, West Godavari Dist., A.P.

[2023 – 2024]

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRKR ENGINEERING COLLEGE (A)

ChinnaAmiram, Bhimavaram, West Godavari Dist., A.P.

[2023 – 2024]

BONAFIDE CERTIFICATE

This is to certify that the project work **"DETECTING AND MITIGATING BOTNET ATTACKS IN SOFTWARE-DEFINED NETWORKS"** is the bonafide work of **M NANI BABU bearing 20B91A05J4** who carried out the project work under my supervision in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

SUPERVISOR	**HEAD OF THE DEPARTMENT**
(Dr V Chandra Sekhar)	(Dr V Chandra Sekhar)
Professor	Professor

SELF DECLARATION

We hereby declare that the project work entitled “Detecting and Mitigating Botnet Attacks in Software Defined Networks” is a genuine work carried out by us in B.Tech., (Computer Science and Engineering) at SRKR Engineering College(A), Bhimavaram and has not been submitted either in part or full for the award of any other degree or diploma in any other institute or University.

MUSKUDI NANI BABU **20B91A05J4**

ABSTRACT

Software-defined networking (SDN) presents a dynamic architecture facilitating the management and communication of extensive networks with ease. By centralizing control, SDNs become susceptible to botnet attacks, exploiting controller resources and manipulating network traffic. Despite revolutionizing network management with its programmable and centralized interfaces, SDN introduces security challenges, notably single points of failure. The vulnerabilities in SDN infrastructure are exploited by botnets and distributed denial of service (DDoS) attacks, necessitating robust security measures. Leveraging deep learning (DL) and Machine Learning (ML) has emerged as a promising avenue for swift threat detection and mitigation. This project evaluates DL methods' efficacy in detecting botnet-driven DDoS attacks within SDN environments. Utilizing a Kaggle dataset, we assess DL and ML models, employing feature weighting and tuning techniques to enhance detection accuracy. The study's objective is to identify efficient DL and ML methods for botnet-based DDoS detection using readily available data.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

The fast extension of the web has uncovered restrictions in conventional systems, often requiring fixing to address rising issues. In any case, this approach regularly leads to bloated networks and debilitated control capabilities. The presentation of software-defined networking (SDN) has revolutionized organize engineering by decoupling the information and control planes. SDN's centralized control design, encouraged by SDN controllers, empowers efficient management of systems through openflow switches and open south API interfacing. SDN facilitates energetic decision-making and customization to meet advancing requests. However, this adaptability comes with a trade-off, uncovering systems to a modern range of security challenges, eminently the helplessness of single focuses of disappointment. In this setting, malicious actors misuse SDN shortcomings through strategies like botnets and Dispersed Dissent of Service (DDoS) assaults, posturing noteworthy dangers to arrange astuteness and accessibility. Botnet attacks can seize the controller and dispatch malevolent exercises. So, DL-based strategies and dedicated bot administration are ideal arrangements for SDN to bargain with botnet activity. To counter these dangers, the integration of profound learning (DL) in security applications has gained traction, promising quick risk location and mitigation.

SOFTWARE DEFINED NETWORKS:

Software-Defined Organizing (SDN) is an inventive organizing engineering that separates the control capacities from the sending capacities in a organize. This partition permits for greater mechanization and programmability inside the arrange, empowering arrange administrators to react rapidly to changes in commerce necessities. SDN makes a centralized brain for the arrange that can communicate and command the rest of the organize, enhancing flexibility, productivity, and security in arrange management.

The SDN can be understood from the Fig 1.1

1. Data Plane
2. Control Plane

The data plane and control plane are two fundamental components of network architecture that perform different functions. The control plane is responsible for determining how the network should behave, while the data plane is responsible for implementing that behavior on the network devices.

Data plane: The Data plane, too known as the client plane, sending plane, carrier plane, or carrier plane, is the portion of a organize that carries client activity. It is mindful for empowering information exchange to and from clients, dealing with different discussions through different conventions, and overseeing discussions with inaccessible peers. The information plane manages application behavior and executes everything from service-level assentions and arrangements to retries and keepalives.

Control plane: The control plane decides the organize topology, controls all exercises related to activity directing and parcel sending, and makes steering tables that give way points of interest for the information bundles. The directing table is utilized by the control plane to decide how and to which ports parcels ought to be sent. Once the control plane decides how and to which ports bundles ought to be sent, the information plane alludes to the rationale and really advances the packets.

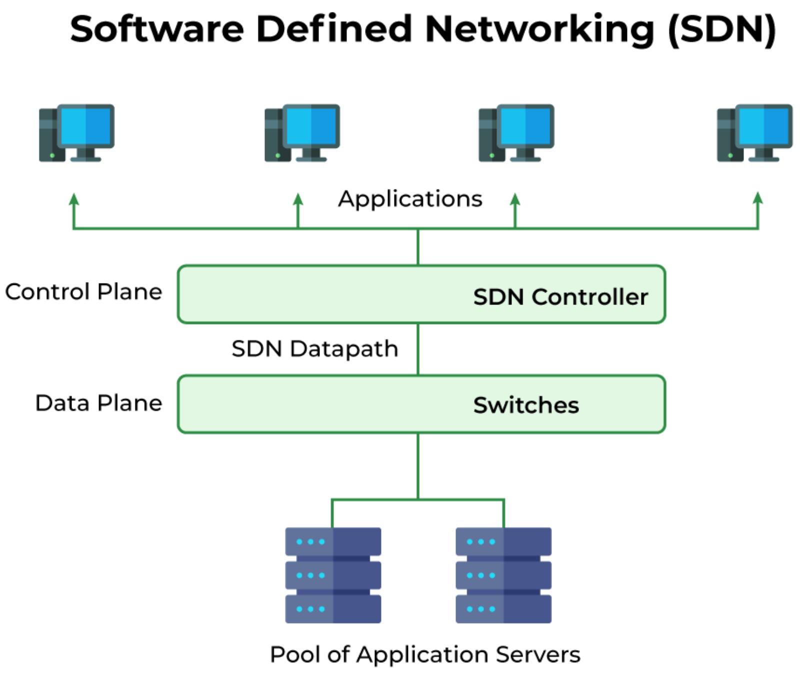


Fig 1.1. SOFTWARE DEFINED NETWORK

In traditional networking, all three planes in the network topology are implemented in the firmware of routers and switches. In contrast, software-defined networking (SDN) decouples the data and control planes and implements the data plane in software instead, which enables programmatic access to the network, making network administration much more flexible.

Decoupling allows dynamic access and administration, enabling a network administrator to shape traffic from a centralized control console without having to touch individual switches. It also makes it easier to troubleshoot the network architecture, create a more scalable and flexible infrastructure that can more easily keep up with evolving business needs, and implement unified, end-to-end network management strategies for multiple tools.

SDN ARCHITECTURE:

SDN (Software-Defined Organizing) is a organize structural show that empowers automatic administration, control, and optimization of arrange assets. It decouples arrange setup and activity designing from the fundamental equipment foundation, guaranteeing all encompassing and steady control of the arrange utilizing open APIs. The SDN design gives a few key characteristics to address the concerns of cutting edge trade IT, counting decoupled engineering planes, organize programmability, centralized control plane, and open standards-based design. These characteristics empower organizations to have a more adaptable, adaptable, and reasonable arrange foundation that can back energetic computing situations and meet the requests of present day commerce IT.

The OpenFlow convention is a foundational component for building SDN arrangements, giving a standard interface for isolating the organize control and information planes. When executed through open guidelines, SDN disentangles arrange plan and operation since informational are given by SDN controllers instep of numerous, vendor-specific gadgets and conventions. SDN is a noteworthy building alter over conventional organizing framework. In any case, it does not require organizations to disturb their existing arrange and supplant it with totally unused equipment and program. Instep, organizations can start with particular SDN utilize cases such as optimizing the arrange, distinguishing account affinities and get to control limits, or organizing particular workloads as they embrace the innovation at scale. SDN as it were reshapes the design to empower centralized control with different levels of mechanization over the network.

A typical SDN architecture consists of three layers as shown in fig 1.2.

Application Layer: This is the highest layer where applications communicate inside the SDN arrange. It speaks to the interface where commerce applications arrange and connected with the arrange. The application layer is capable for giving the rationale to oversee information stream and make arrange applications in programming dialects like Python.

Control layer: This layer is frequently alluded to as the brain of the SDN arrange. It coordinates activity inside the organize and is the central point of control that directs how information is directed and overseen over the arrange. The control layer oversees information stream and acts as a interpreter between the application layer over and the foundation layer underneath, indicating the way of information through the network.

Infrastructure layer: This layer comprises of arrange gadgets that contain the information plane and work on OpenFlow convention or utilize OpenFlow API to communicate. It incorporates physical switches, switches, and other organize gadgets that forward activity based on rules advertised by a controller. The foundation layer is mindful for collecting organize statuses like organize topology, activity measurements, arrange utilization, and sending them to the control layer.

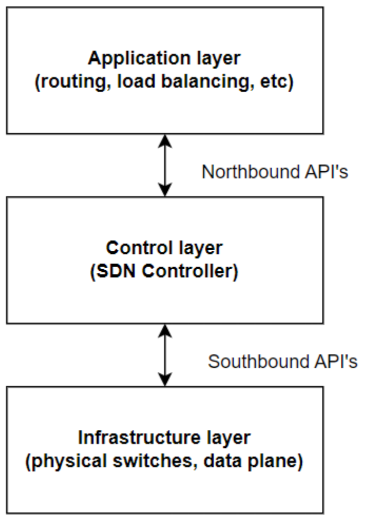


Fig 1.2. SDN Architecture

BOTNET ATTACKS:

Botnet assaults are a frame of cyberattack where a bunch of internet-connected gadgets is contaminated by malware, empowering programmers to control them and unleash a string of assaults. Botnets are frequently utilized to carry out Disseminated Refusal of Benefit (DDoS) assaults, send spam, take touchy data, or dispatch other pernicious exercises as appeared in Fig 1.3

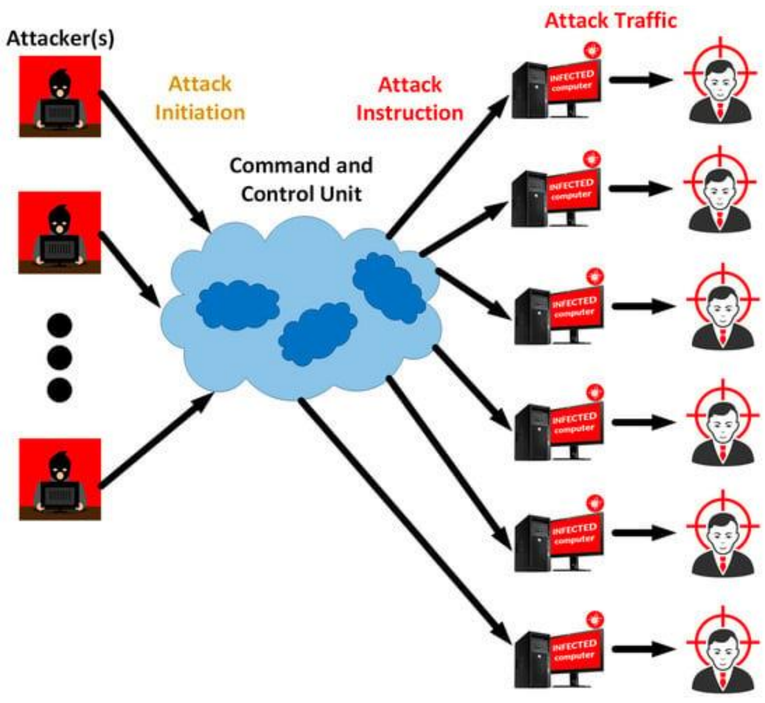


Fig 1.3. Network Architecture of a typical botnet attack

The prepare of a botnet assault ordinarily includes three stages. In the to begin with organize, programmers pick up unauthorized get to to gadgets by infusing Trojan infections or abusing vulnerabilities in computer program or websites. Once they effectively pick up get to, they control these gadgets with computer program that permits them to carry out assaults. In the moment organize, the programmer organizes all the tainted machines into a arrange, frequently looking for to contaminate and control thousands, tens of thousands, or indeed millions of computers. This organize, or botnet, can at that point be utilized to dispatch assaults against particular targets.

In the third organize, the programmer takes control of each computer in the botnet and employments it to carry out assaults. Botnets can be utilized for different noxious exercises, such as sending spam emails, taking touchy data, propelling DDoS assaults, or carrying out brute drive assaults to figure login data or encryption keys. Botnets can too be utilized to mine cryptocurrencies or capture cryptocurrency transactions.

Preventing botnet assaults requires modern location instruments and proactive measures. A few best hones for avoiding botnet assaults incorporate keeping program frameworks up to date, checking the arrange for odd action, utilizing solid and interesting passwords, actualizing multi-factor verification, and teaching representatives around the dangers of phishing assaults and other social designing strategies. Moreover, organize division and the utilize of interruption location and avoidance frameworks can offer assistance to restrain the spread of botnet malware and distinguish and react to botnet assaults in a convenient manner.

TYPES OF BOTNET ATTACKS:

Botnet attacks in SDN can be sophisticated DDoS attacks launched using botnet technology, typically used in SDN and IoT-based networks. The following are various attacks, such as:

- **Distributed Denial of Service (DDoS):** A DDoS attack occurs when botnets flood a targeted application or server with requests, causing it to crash.This can be done through various techniques such as synchronization code or SYN floods in a TCP connection, UDP floods, and DNS amplification, with the objective of depleting the target's bandwidth and preventing valid requests from being processed.
- **Sniffing and keylogging botnet attacks:** These attacks involve the use of botnets to capture sensitive information, such as usernames, passwords, and other confidential data, by recording keystrokes or collecting photos and sending them to the bot-master.
- **Spamming Botnets:** These botnets are used to send out massive volumes of spam emails, often containing malicious links or attachments, to deceive recipients into revealing sensitive information or downloading malware.
- **Phishing Botnets:** Phishing botnets are designed to launch phishing campaigns, where attackers impersonate legitimate entities to trick individuals into providing confidential information such as passwords, credit card details, or personal data.
- **Cryptojacking Botnets:** These botnets are used to mine cryptocurrencies by infecting devices and using their computational power to mine digital currencies without the users' consent or knowledge.

- **Brute Force Botnets:** In a brute force attack, botnets attempt to guess passwords by systematically trying all possible combinations until the correct one is found, allowing unauthorized access to systems or accounts.
- **Device Bricking Botnets:** Botnets can be used to launch device bricking attacks, where malware is deployed to render devices inoperable by deleting their contents, making them useless.
- **Network Probing Botnets:** These botnets are focused on scanning the internet to identify vulnerable systems that can be exploited for further attacks or to recruit them into the botnet.
- **Information Stealing Botnets:** Botnets designed for information stealing collect personal data from infected machines, such as keystrokes, screenshots, or login credentials, which are then sent back to the attacker for malicious purposes

LITERATURE REVIEW

The summary of various studies in the related to the Detection of botnet type of attacks in software defined networks are listed below with references mentioned at the end of the report.

[1]. Detecting and Mitigating Botnet Attacks in Software-Defined Networks Using Deep Learning Techniques:

The effectiveness of Convolutional Neural Networks (CNNs) in detecting and mitigating botnet activity was investigated in this study. By training the model on an extensive dataset of network traffic features, it could discern intricate patterns distinguishing legitimate network behavior from malicious botnet communications. The results highlight that CNN architectures attained notable accuracy in identifying botnet attacks within the SDN environment.

[2]. DDoS attacks detection using machine learning and deep learning techniques analysis and comparison:

This study conducted a comprehensive analysis and comparison of machine learning and deep learning techniques for detecting Distributed Denial-of-Service (DDoS) attacks. Our investigation highlighted the significant advantages of both approaches in identifying malicious traffic patterns. Machine learning algorithms, such as Support Vector Machines (SVMs) and K-Nearest Neighbors (KNN), proved effective in DDoS detection by leveraging historical traffic data to detect anomalies. Deep learning models, particularly Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), demonstrated superior performance in capturing intricate relationships within network traffic data, resulting in even more precise DDoS attack detection.

[3]. Early Detection of DDoS Attacks in Software Defined Networks Controller:

A solution tailored for SDN is the focus of this research, leveraging its specifications, strengths, and limitations. The SDN specification, which mandates the forwarding of new packets to the controller, was capitalized on. By utilizing the controller's comprehensive network view, entropy statistics collection was incorporated. The criticality of maintaining continuous controller connectivity was recognized, and a solution to detect threats at their inception was devised. While entropy has been utilized in DDoS detection in traditional networks, our solution represents the first application of this approach in SDN, distinguishing it as a novel contribution to the field.

[4]. Mitigation of Distributed Denial of Service (DDoS) Attacks over Software Defined Networks (SDN) using Machine Learning and Deep Learning Techniques:

The utilization of Machine Learning (ML) and Deep Learning (DL) to combat DDoS attacks on Software Defined Networks (SDN) is explored in this research paper. SDN's centralized control plane is susceptible to such attacks, prompting the concept of training ML/DL models to recognize malicious traffic patterns. Leveraging SDN's programmability enables the implementation of real-time countermeasures informed by these insights, potentially enhancing the speed and accuracy of DDoS mitigation efforts.

[5]. Performance of Botnet Detection by Neural Networks in Software-Defined Networks:

This study by Letteri et al. concludes that Artificial Neural Networks (ANNs) offer a promising and effective approach for detecting botnets within Software-Defined Networks (SDNs). By analyzing OpenFlow messages, their simplified ANN technique achieved high accuracy in botnet identification, potentially surpassing conventional botnet detection methods. This suggests that ANNs can leverage the unique data available in SDNs for robust botnet mitigation.

[6]. SDN-Based Engineering for Transport and Application Layer DDoS Assault Discovery by Utilizing Machine and Profound Learning:

Yungaicela-Naula et al. present an SDN-based design tending to the particular challenge of recognizing Transport and Application Layer DDoS assaults, infamous for their complexity. Their system coordinating Machine Learning and Profound Learning for organize activity examination, with the objective of revealing covered up designs characteristic of DDoS endeavors. Leveraging SDN's programmability empowers the usage of real-time moderation methodologies educated by these experiences, possibly upgrading the adequacy of defense against these complicated attacks.

[7]. Machine Learning Based Botnet Detection in Software Defined Networks:

A Machine Learning (ML) based approach for botnet detection within Software-Defined Networks (SDNs) is proposed by Tariq and Baig. This method involves analyzing network flows and extracting features such as packet size and flow duration to identify botnet patterns. The paper utilizes the C4.5 decision tree, a supervised machine learning algorithm for botnet classification. The model is likely trained to recognize these features and potentially learn from them to identify both known and unknown botnets. This approach capitalizes on the unique data available in SDNs for botnet detection, offering a potential alternative to traditional methods limited by data access.

[8]. Detection and Mitigation of Malicious DDoS Floods in Software Dened Networks:

The research had its prime focus on applying an authentic and lightweight solution for detecting various DDoS attacks at their early stages. In this research, resilience was built into the SDN controller, which was responsible for detecting and mitigating malicious DDoS activity based on variations in network traffic. A high detection rate of 98.75% was achieved with an average delay of 14.08 seconds under varied attack patterns when attacks were performed on a single victim inside the network. Similarly, to extend the adverse effects of DDoS attacks, multi-destination attacks were conducted, where an overall accuracy between 96.25% was achieved with an average delay of 20.9 seconds for mitigation, despite a slight decline in performance.

[9]. Detection And Mitigation Of Distributed Denial Of Service Attacks On Network Architecture Software Defined Networking Using The Naive Bayes Algorithm:

A DDoS attack detection system for SDN is proposed in the research, utilizing the Naive Bayes algorithm. Network traffic features are analyzed by this system, which employs a trained classifier to detect DDoS attacks. While the Naive Bayes approach provides efficient training with moderate data requirements, its effectiveness depends on the accuracy of the training data and the assumption of feature independence, which may not always be ideal for real-world scenarios.

[10]. A Method of DDoS Attack Detection and Mitigation for the Comprehensive Coordinated Protection of SDN Controllers:

The CC-Guard framework, proposed in this research paper, offers a comprehensive approach to protecting SDN controllers from DDoS attacks. It achieves this through a combination of modules that trigger a swift response, ensure controller functionality, accurately detect anomalies, and collaboratively mitigate threats. This framework promises real-time defense with high accuracy and efficient resource utilization.

[11]. Deep learning approaches for detecting DDoS attacks: a systematic review:

A systematic review on Deep Learning (DL) approaches for DDoS attack detection was conducted by Mittal et al. (2022). Their analysis underscores the effectiveness of DL in addressing complex attack patterns and its capability to operate with limited data. The review offers a valuable categorization of existing research, outlining strengths and weaknesses of different methodologies, datasets utilized, and performance metrics. Additionally, it identifies opportunities for future research, such as investigating new DL architectures and enhancing model interpretability, thereby laying the groundwork for even more advanced DDoS detection techniques.

PROBLEM STATEMENT

Software-Defined Networking (SDN) presents inherent security challenges due to its vulnerability to malicious attacks, notably botnets and Distributed Denial of Service (DDoS) attacks. Traditional security measures struggle to keep pace with these dynamic threats, necessitating innovative solutions. SDN's centralized control plane becomes a prime target for attackers aiming to disrupt network operations. Inadequacy of traditional conventional security approaches may fail to detect and mitigate sophisticated attacks targeting SDN environments. Timely identification and response are crucial to minimize the impact of botnet-based DDOS attacks on network performance and availability.

METHODOLOGY

1. **Data collection:** Due to the unavailability of open-source data pertaining to cyber attacks, we curated a custom dataset by amalgamating diverse datasets sourced from reputable repositories like Zenodo. Employing rigorous preprocessing techniques, we transformed the raw data into a structured format, culminating in the creation of a comprehensive CSV file christened 'Storm.'
2. **Data Preprocessing:** At this stage the collected dataset undergoes few preprocessing steps for the purpose of cleaning of data. Which involves handling the missing values, duplicated values, Encoding techniques etc.

   We have used One Hot Encoding Techniques in our project. One-hot encoding is a technique used in machine learning to convert categorical variables into a binary format. It involves creating binary columns for each category, where only one column is hot (1) indicating the presence of that category, while others remain cold (0). This encoding is useful for algorithms that require numerical input, such as neural networks, as it preserves categorical information without imposing ordinality.
3. **Feature Engineering and Data analysis:** In this step the features relevant for the botnet attack detection are extracted from the preprocessed data. Here we have performed some visualization on the features of the dataset to see the insights present in the data. We have plotted some Bar graphs, Pie charts, Histograms, Violin chart, line chart. These plots are used to see the attacks by various packets, port numbers and so on.
4. **Model Training and Evaluation:** For this project we are using the deep learning techniques like ANN, LSTM and along with these we are also using few ML models for training to get the accurate predictions. The models will be trained on a portion of the dataset and tested on a separate unseen portion to assess their generalizability.
5. **Model selection:** Based on the evaluation results, the best performing model will be selected. For model selection we consider the performance metrices like Accuracy, Precision, Recall, F1 score. The best model for our project is the Hybrid Model which is the combination of Random Forest and KNN.

4.1 SYSTEM ARCHITECTURE

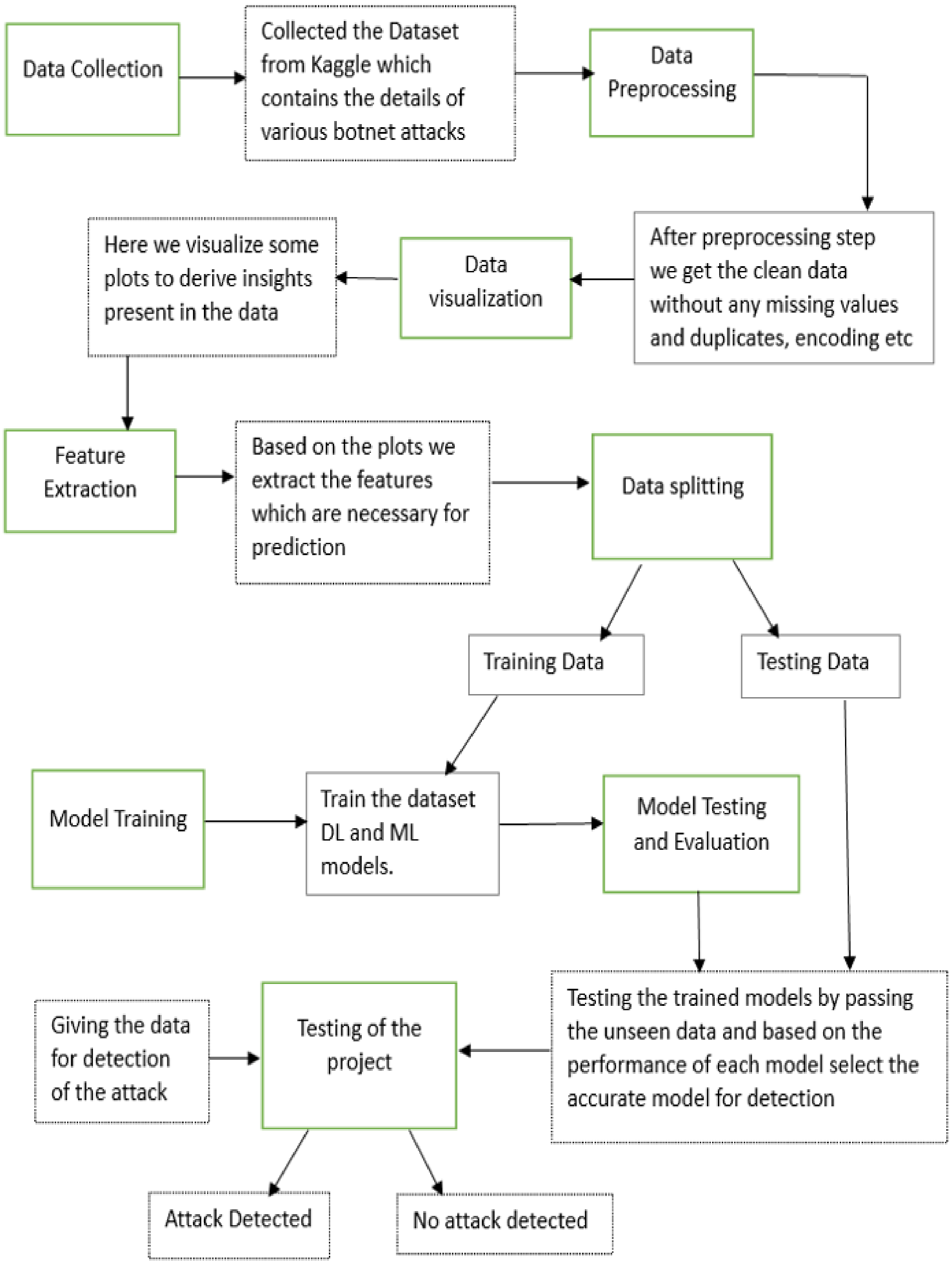

Data Collection
Collected the Dataset from Kaggle which contains the details of various botnet attacks
Data Preprocessing
After preprocessing step we get the clean data without any missing values and duplicates, encoding etc
Data visualization
Here we visualize some plots to derive insights present in the data
Feature Extraction
Based on the plots we extract the features which are necessary for prediction
Data splitting
Training Data
Testing Data
Model Training
Train the dataset DL and ML models.
Model Testing and Evaluation
Testing the trained models by passing the unseen data and based on the performance of each model select the accurate model for detection
Giving the data for detection of the attack
Testing of the project
Attack Detected
No attack detected

Fig 4.1 System Architecture

4.2 MODELS USED:

HYBRID MODEL (RANDOM FOREST AND KNN):

Base Models:

Random Forest Classifier: Ensemble learning method that fits a number of decision tree classifiers on various sub-samples of the dataset and averages predictions to improve accuracy and control over-fitting.

K-Nearest Neighbours Classifier: A lazy learning algorithm where predictions are made based on the majority class of its k nearest neighbours in the feature space.

Training:

The base models (Random Forest and KNN) are trained independently on the same training dataset. Each model learns from the input features (X_train) and their corresponding labels (y_train).

Hybrid Prediction:

For a given input data point (X), predictions are obtained from both the Random Forest (rf_preds) and KNN (knn_preds) models. The predictions from both models are combined using a majority voting scheme. In this scheme, if the sum of the predictions from both models is greater than or equal to 1, the combined prediction is considered as class 1; otherwise, it is class 0. The combined predictions form the output of the hybrid model for the given input data.

Evaluation:

The performance of the hybrid model is evaluated using the testing dataset (X_test and y_test). Predictions are obtained from the hybrid model for the testing dataset. The accuracy of the hybrid model is calculated by comparing the predicted labels (y_pred) with the true labels (y_test) using the accuracy_score metric.

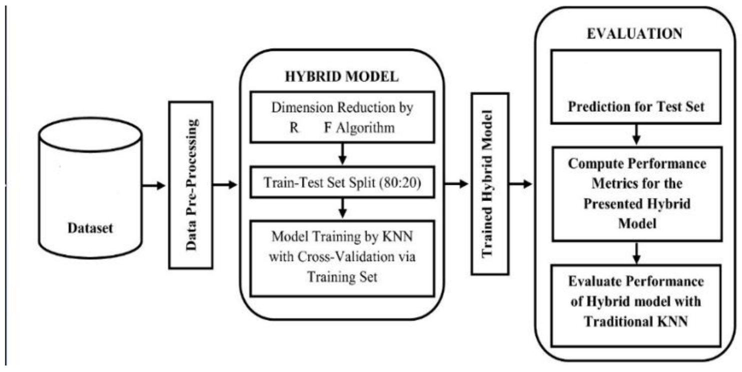

Dataset
Data Pre-Processing
HYBRID MODEL
Dimension Reduction by R F Algorithm
Train-Test Set Split (80:20)
Model Training by KNN with Cross-Validation via Training Set
Trained Hybrid Model
EVALUATION
Prediction for Test Set
Compute Performance Metrics for the Presented Hybrid Model
Evaluate Performance of Hybrid model with Traditional KNN

Fig 4.1 Hybrid model

ARTIFICIAL NEURAL NETWORK(ANN):

ANN or Counterfeit Neural Organize, is a sort of machine learning demonstrate that is propelled by the structure and work of the human brain. It is composed of interconnected hubs, or neurons, that prepare and transmit data. ANN models comprise of three primary layers as appeared in fig.4.2

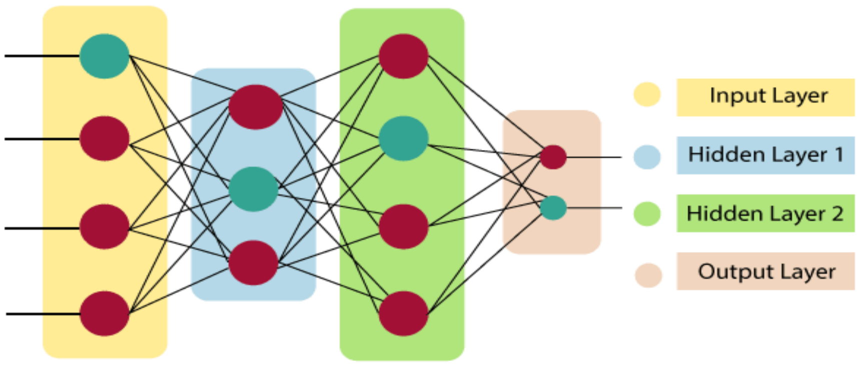


Fig 4.2. ANN Architecture

Input Layer:

The input layer gets input values from the informative properties for each perception in the dataset. The number of input hubs in the input layer is ordinarily break even with to the number of informative variables. The input layer presents the designs to the organize and communicates with one or more covered up layers. Nodes in the input layer are inactive and do not alter the information; they get a single esteem on their input and copy it to their numerous outputs.

Hidden Layer:

Hidden layers apply changes to the input values inside the network. Each covered up hub gets inputs from other covered up hubs or input hubs associated to it. The real handling in the neural organize happens through a framework of weighted associations in the covered up layer. The values entering a covered up hub are increased by weights, and the weighted inputs are at that point included to create a single number.

Output Layer:

The yield layer gets associations from covered up layers or the input layer. It returns an yield esteem that compares to the forecast of the reaction variable. In classification issues, there is as a rule as it were one yield hub in the yield layer. The dynamic hubs in the yield layer combine and change the information to deliver the last yield values.

LONG SHORT TERM MEMORY(LSTM):

LSTM, or Long Short-Term Memory, is a sort of repetitive neural organize (RNN) design that is able of learning long-term conditions in information. It was created to address the vanishing angle issue that is regularly experienced in conventional RNNs, which makes it troublesome for the arrange to learn long-term conditions. LSTM systems are composed of memory cells that store data over time and three sorts of doors that control the stream of data into and out of the memory cells. These entryways are the input door, the yield door, and the disregard entryway. The input entryway decides how much unused data is included to the memory cell, the yield door decides how much data is yield from the memory cell, and the disregard entryway decides how much data is held or disposed of from the memory cell. LSTM systems have been utilized in a assortment of applications, counting characteristic dialect preparing, discourse acknowledgment, and time arrangement forecast. They have been appeared to be especially compelling in assignments that require the organize to keep in mind data over long periods of time, such as dialect interpretation and estimation examination.

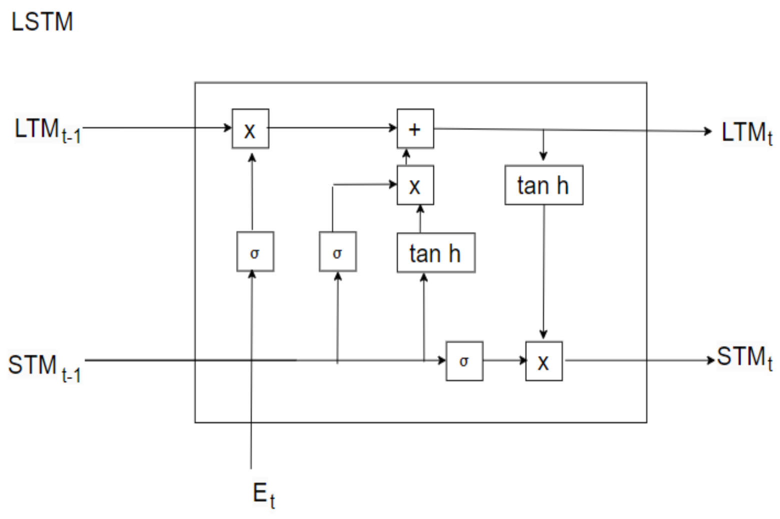


Fig 4.3. Simplified LSTM Architecture

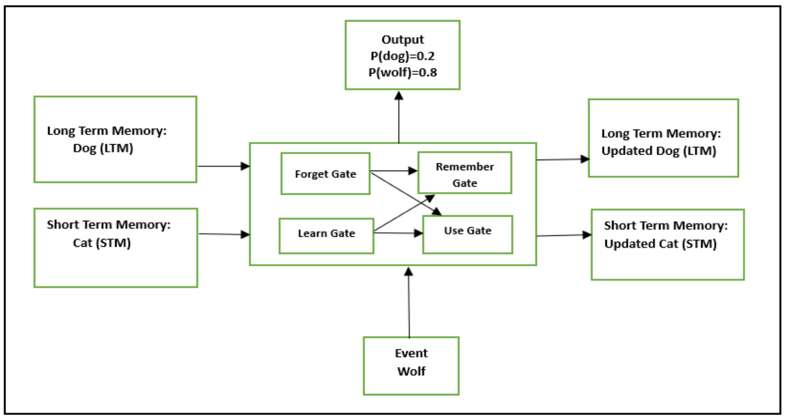

Output
P(dog)=0.2
P(wolf)=0.8
Long Term Memory: Dog (LTM)
Short Term Memory: Cat (STM)
Forget Gate
Remember Gate
Learn Gate
Use Gate
Long Term Memory: Updated Dog (LTM)
Short Term Memory: Updated Cat (STM)
Event Wolf

Fig 4.4. Mathematical Architecture of LSTM

SUPPORT VECTOR MACHINE

Support Vector Machine (SVM) is a administered machine learning calculation utilized for both classification and relapse assignments. It is a effective calculation for relapse, classification, and indeed exception discovery assignments. SVMs can handle high-dimensional information and nonlinear issues effectively.

The Fig 4.5 appears primary objective of the SVM calculation which is to discover the ideal hyperplane in an N-dimensional space that can partitioned the classes with the most extreme edge. The hyperplane tries to maximize the edge between the closest focuses of diverse classes. The measurement of the hyperplane depends on the number of highlights. If the number of input highlights is two, at that point the hyperplane is fair a line. If the number of input highlights is three, at that point the hyperplane gets to be a 2-D plane.

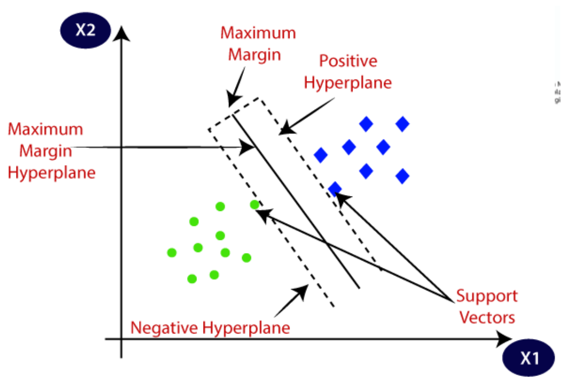


Fig 4.5. Support Vector Machine

SVMs are broadly utilized in machine learning as they can handle both straight and nonlinear classification assignments. When the information is not straightly distinct, part capacities are utilized to change the information into a higher-dimensional space to empower straight partition. The choice of part work, such as straight parts, polynomial parts, spiral premise work (RBF) bits, or sigmoid bits, depends on information characteristics and the particular utilize case.

LOGISTIC REGRESSION

Logistic Regression is a factual show utilized for foreseeing double results or probabilities of an occasion happening based on one or more indicators or autonomous factors. It is a sort of classification calculation utilized in machine learning to anticipate the probability of an occasion having a place to a specific category or class.

The show conveys a twofold or dichotomous result restricted to two conceivable results, such as yes/no, 0/1, or true/false. Calculated relapse analyzes the relationship between one or more free factors and classifies information into discrete categories. It is broadly utilized in prescient modeling, where the show gauges the numerical likelihood of whether an occurrence has a place to a particular category or not.

The calculated relapse condition is based on the calculated work, which is an S-shaped bend that maps any real-valued number into a likelihood esteem between 0 and 1. The calculated work is characterized as:

p = 1 / (1 + e^(-z))

where p is the likelihood of the occasion happening, and z is the direct combination of the free factors and their coefficients.

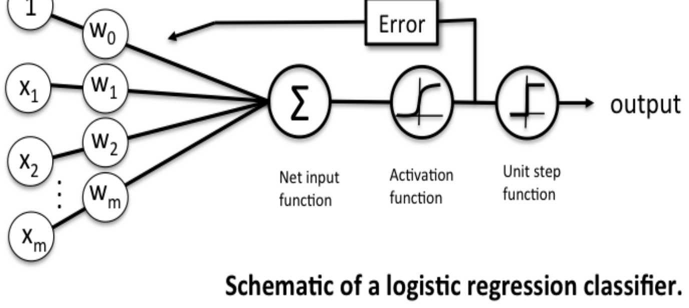


Fig 4.6. Logistic Regression

DECISION TREE CLASSIFIER:

A decision tree classifier is a machine learning algorithm used for classification tasks. It builds a tree structure to represent decisions and their possible consequences, including chance event outcomes, resource costs, and utility.

The decision tree classifier uses a set of rules to predict the class of a given input. Each internal node in the tree represents a feature or attribute of the input, each branch represents a decision rule based on that feature, and each leaf node represents a class label. The decision tree classifier recursively splits the input space into subspaces based on the feature values until all instances in a subspace belong to the same class. The decision tree classifier can handle both categorical and numerical data and can be used for both classification and regression tasks.

The decision tree classifier algorithm uses a measure of impurity, such as entropy or Gini impurity, to determine the best feature to split on at each node. The goal is to maximize the homogeneity of the resulting subspaces, i.e., to minimize the impurity. The decision tree classifier can be pruned to avoid overfitting and to improve generalization performance. Pruning involves removing branches that do not contribute significantly to the accuracy of the classifier. The decision tree classifier can also be used in ensemble methods, such as random forests, to improve accuracy and reduce variance.

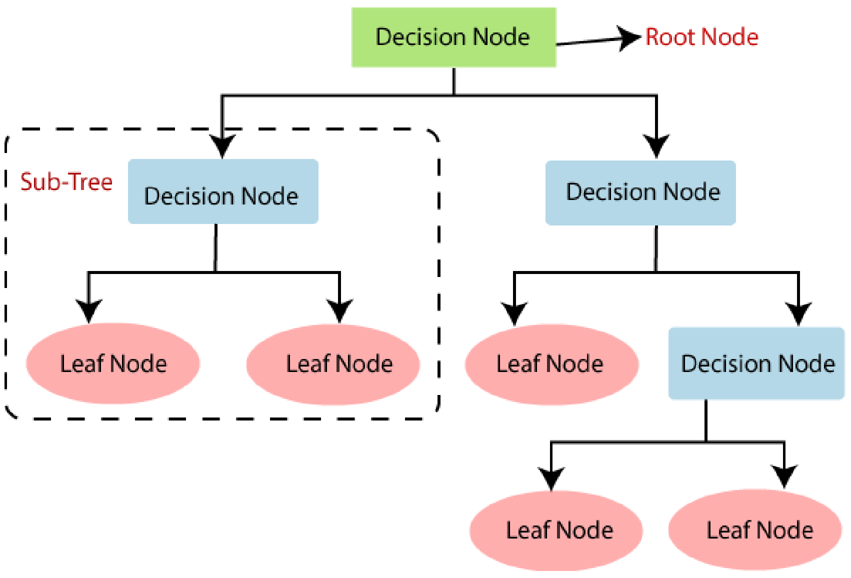


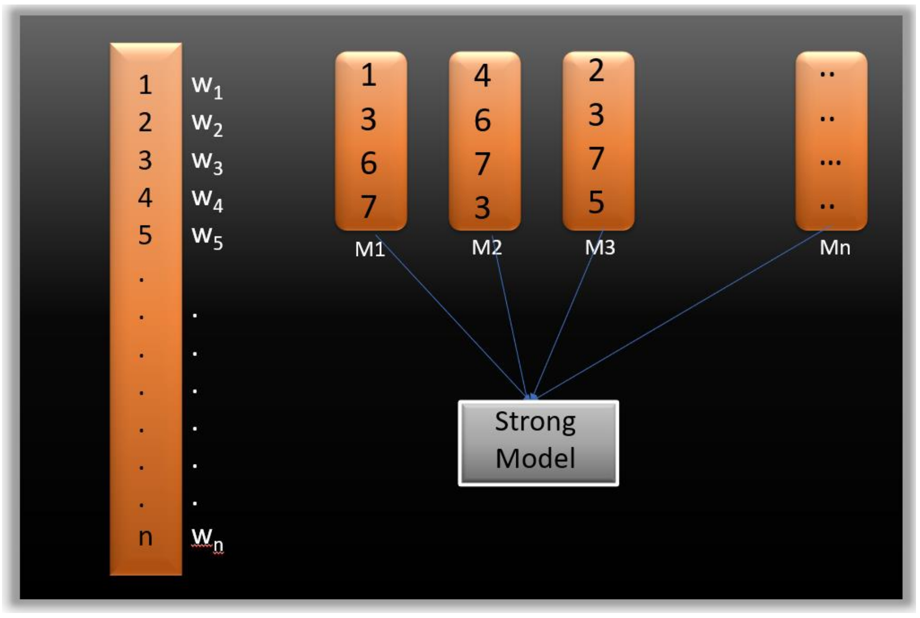
Fig 4.7. Decision Tree Classifier

GRADIENT BOOSTING CLASSIFIER:

Gradient Boosting is a machine learning calculation utilized for relapse and classification issues. It is a sort of boosting calculation that builds an gathering of frail forecast models, ordinarily choice trees, in a stage-wise mold. The calculation generalizes other boosting strategies by permitting optimization of an self-assertive differentiable misfortune work. Slope Boosting more often than not outflanks Irregular Timberland in terms of forecast accuracy.

As appeared in Fig 4.8 the calculation works by iteratively preparing modern models that center on precisely anticipating the cases where the past models perform ineffectively. Each progressive show endeavors to rectify for the deficiencies of the combined boosted outfit of all past models. The target results for each case are set based on the angle of the blunder with regard to the expectation, permitting the calculation to take a step in the heading that minimizes forecast blunder in the space of conceivable forecasts for each preparing case.

Gradient Boosting is broadly utilized in different applications, counting prescient modeling, information mining, and machine learning. It is known for its tall expectation exactness and capacity to handle complex information sets with numerous highlights. In any case, it can be touchy to clamor and exceptions and may require cautious tuning of its parameters to accomplish ideal performance.



1 2 3 4 5 . . . . n
w_1 w_2 w_3 w_4 w_5 . . . . . . w_n
1 3 6 7
4 6 7 3
2 3 7 5
.. ..
M1
M2
M3
Mn
Strong Model

Fig 4.8. Gradient Boosting Classifier

4.3 SELECTION OF OPTIMAL MODEL:

The best suitable model from the above models is Hybrid model which is the combination of Random Forest and KNN. The following is the working of Random Forest and KNN.

Working of Random Forest:

The Random Forest algorithm works by creating an ensemble of multiple decision trees to achieve more accurate predictions. Here is a breakdown of how the Random Forest algorithm operates based on the provided sources:

Ensemble of Decision Trees: Random Forest is an ensemble learning method that combines multiple decision trees to improve prediction accuracy. Every decision tree is constructed utilizing a subset of the training data, and the ultimate prediction is derived from the aggregation of predictions made by all the trees.

Random Sampling: The algorithm randomly selects subsets of the training data and features to build individual decision trees. This randomness helps in creating diverse trees that collectively provide robust predictions.

Voting Mechanism: In the classification task, each decision tree in the Random Forest predicts the class for a given input. In the ensemble model, the final prediction is established through a majority vote among all the decision trees. In regression tasks specifically, the ultimate prediction frequently amounts to the average of predictions derived from individual trees.

Bagging Approach: Random Forest uses a bagging approach, which involves creating different training subsets from the sample training data with replacement. The final output is based on majority voting, where each tree contributes to the final prediction.

Random Forest Algorithm Steps:

Initialization:

- Choose the number of trees (n_estimators) to include in the forest.
- Determine the maximum depth of each tree to control overfitting.
- Select the number of features to consider at each split (max_features).

For each tree in the forest:

- Randomly sample the training data with replacement (bootstrap sampling).
- Randomly select a subset of features for splitting at each node.

- Construct a decision tree utilizing the provided data and features by selecting the optimal split at each node according to a criterion such as Gini impurity or entropy for classification, or mean squared error for regression.
- Repeat the splitting process until a stopping criterion is met (e.g., maximum tree depth reached).

Prediction:

For a new input sample:

- Let each decision tree in the forest make a prediction.
- For classification tasks, employ majority voting to determine the ultimate prediction.
- For regression tasks, average the predictions from all decision trees for the final prediction.

Ensemble Learning:

- Combine predictions from all trees to improve accuracy and generalization.
- Handle overfitting by aggregating predictions from diverse trees.

Evaluation:

- Evaluate the Random Forest model's performance using metrics such as accuracy, precision, recall, F1 score, or mean squared error, depending on the nature of the task.

Feature Importance:

- Calculate feature importance based on how much each feature contributes to the model's predictions.
- Identify key features that have the most impact on the model's performance.

Hyperparameter Tuning:

- Fine-tune hyperparameters like tree depth, number of trees, and max_features to optimize model performance.
- Use techniques like grid search or random search to find the best hyperparameter values.

Deployment:

- Deploy the trained Random Forest model to make predictions on new, unseen data.
- Monitor model performance and retrain periodically to maintain accuracy andrelevance.

Working of KNN:

The K-Nearest Neighbors (KNN) algorithm operates by identifying the K closest neighbors to a specific data point using a distance metric, such as the Euclidean distance. Subsequently, the class or value of the data point is established through either majority voting or averaging the attributes of the K neighbors. This methodology enables the algorithm to adjust to diverse patterns and formulate predictions grounded on the localized structure of the data.

Here are the key steps in the working of the KNN algorithm based on the provided sources:

Selecting Neighbors:

The algorithm identifies the K nearest neighbors to a given data point based on a chosen distance metric like Euclidean, Manhattan, or Minkowski distance.

Voting Mechanism:

- For classification tasks, the class of the data point is determined by a majority vote among its K nearest neighbors.
- For regression tasks, the value of the data point is often the average of the values of its K nearest neighbors.

Distance Metrics:

Different distance metrics, such as Euclidean, Manhattan, and Minkowski distances, are used to evaluate the similarity between data points.

Prediction Process:

The algorithm predicts the class or value of a new data point by considering the labels or values of its K nearest neighbors.

Local Structure Adaptation:

KNN functions on the principle of similarity, forecasting the label or value of a new data point by considering the labels or values of its K nearest neighbors.

Parameter Selection:

The value of K is crucial in the KNN algorithm and should be chosen based on the characteristics of the input data, such as the presence of outliers or noise.

Classification vs. Regression:

- In classification tasks, KNN uses a voting mechanism to assign a class label to a data point.
- In regression tasks, the K-nearest neighbors (KNN) technique determines the predicted value of a data point by averaging the values of its K nearest neighbors.

IMPLEMENTATION

5.1 SYSTEM IMPLEMENTATION

5.1.1- DATASET

Due to the unavailability of open-source data pertaining to cyber attacks, we curated a custom dataset by amalgamating diverse datasets sourced from reputable repositories like Zenodo. Employing rigorous preprocessing techniques, we transformed the raw data into a structured format, culminating in the creation of a comprehensive CSV file christened 'Storm.' This dataset serves as a valuable resource for cybersecurity research, facilitating the exploration of various attack vectors, detection methodologies, and mitigation strategies.

Dataset Link:
https://drive.google.com/file/d/1uJuz_WkTRw6632GRiXLL0_DwgcceC8nE/view?usp=sharing

5.1.2 SOFTWARE AND HARDWARE REQUIREMENTS

The software requirements specify the use of all required software products like data management systems. The required software product specifies the numbers and version. Each interface specifies the purpose of the interfacing software as related to this software product.

• Operating system - Windows 7/10

• Framework - Tensorflow Framework, Keras, Streamlit

• Workbench – Visual Studio Code

• Programming Language – Python

The hardware requirement specifies each interface of the software elements and the hardware elements of the system. These hardware requirements include configuration characteristics.

• System - Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz

• Hard Disk - 100GB

• RAM - 8 GB

5.1.3- SOFTWARE INSTALLATIONS:

1. PYTHON INSTALLATION:

- Visit the official Python website (www.python.org) and navigate to the Download for Windows section.
- Choose the desired version of Python to download. For example, if you have a 64-bit operating system, select the "Windows x86-64 executable installer".
- Run the installer and select the "Add Python to PATH" option before clicking "Install Now".
- Wait for the installation to complete, then open the command prompt and enter "python --version" to verify the installation.
- If the installation is successful, you can start coding in Python using IDLE or your preferred code editor.

2. CODE EDITER INSTALLATION: VISUAL STUDIO

Download Visual Studio:

- Visit the official Visual Studio website or use the provided download links.
- Choose the version and edition of Visual Studio you want to install (e.g., Visual Studio 2019 or Visual Studio 2022).

Run the Installer:

- Open the downloaded executable file (.exe) to start the installation process.
- Follow the on-screen instructions to proceed with the installation.

Select Features:

- During the installation, you may be prompted to select specific features or workloads based on your requirements.
- Choose the components you need for your development environment.

Customize Installation:

- Visual Studio allows you to customize the installation by selecting the desired components, workloads, and themes.
- You can also choose the installation location and other preferences.

Complete Installation:

- Wait for the installation process to complete. The duration may vary based on your internet speed and the selected components.
- Once the installation is finished, you may need to reboot your computer to finalize the setup.

Verify Installation:

- Open Visual Studio after the installation is complete.
- Choose a theme, set up your preferences, and start using Visual Studio for your development projects.

Optional Steps:

- For advanced users, there are options to create an offline installation layout or customize the installation further using command-line parameters.

5.1.4 -LIBRARIES NEEDED:

TensorFlow System:

TensorFlow is a free and open-source program library for machine learning and manufactured insights (AI) that can be utilized over a extend of assignments but has a specific center on preparing and deduction of profound neural systems. It was created by the Google Brain group and was to begin with discharged to the open in 2015. TensorFlow is accessible beneath the Apache permit, which permits for free utilize and modification.

TensorFlow permits engineers to make dataflow charts, which are structures that portray how information moves through a chart, or a arrangement of handling hubs. Each hub in the chart speaks to a scientific operation, and each association or edge between hubs is a multidimensional information cluster, or tensor. TensorFlow applications can be run on a assortment of targets, counting nearby machines, clusters in the cloud, iOS and Android gadgets, CPUs or GPUs, and indeed Google's custom TensorFlow Preparing Unit (TPU) silicon for assist acceleration.

TensorFlow gives a helpful front-end API for building applications utilizing Python or JavaScript, whereas the fundamental stage executes those applications in high-performance C++. It too gives libraries for numerous other dialects, in spite of the fact that Python tends to overwhelm. TensorFlow bolsters generation expectation at scale, with the same models utilized for preparing. It too has a wide library of pre-trained models accessible for utilize in your projects.

2. Keras:

Keras is an open-source neural arrange library composed in Python that runs on beat of TensorFlow, Theano, or CNTK. It was created by François Chollet and is presently kept up by Google. Keras gives a high-level, user-friendly interface for creating and prototyping profound learning models. It bolsters both convolutional and repetitive neural systems and offers a wide run of pre-built layers and models. Keras is outlined for simple and quick experimentation, permitting engineers to construct models with fair a few lines of code. It too gives a wide extend of instruments and libraries for visualizing and investigating models, making it a prevalent choice for both inquire about and generation environments.

3. Matplotlib:

Matplotlib is a comprehensive library for making inactive, vivified, and intuitively visualizations in Python. It is a well known instrument for making publication-quality figures in a assortment of designs and situations, counting Python scripts, shells, web application servers, and different graphical client interface toolkits. Matplotlib can be utilized to make a wide extend of visualizations, counting line plots, diffuse plots, histograms, 3D plots, and picture plots, among others. It is planned to be as usable as MATLAB, with the advantage of being free and open-source. Matplotlib is frequently utilized in logical computing, information investigation, and machine learning applications. It is too utilized in instruction and investigate, as well as in industry and government. Matplotlib is disseminated beneath a BSD-style permit and is accessible on all major stages, counting Windows, macOS, and Linux. It is a NumFOCUS financially supported extend and has a huge and dynamic advancement community.

4. NumPy:

NumPy is a numerical science expansion for the Python programming dialect. It gives bolster for expansive, multi-dimensional clusters and networks, along with a collection of scientific capacities to work on these clusters effectively. NumPy is a principal bundle for logical computing in Python and is broadly utilized in different areas such as machine learning, information examination, and numerical simulations

5. PANDAS:

Pandas is a Python library for information control and examination. It gives information structures and capacities for proficiently taking care of unthinkable information, such as information outlines, and is especially valuable for working with huge and complex datasets. The title "pandas" is determined from the term "board information", which alludes to a sort of information that is organized into a multidimensional array.

Pandas gives a wide run of highlights for working with information, counting information cleaning, information change, information blending, and information visualization. It moreover underpins different information groups, such as CSV, Exceed expectations, and SQL databases, and can be coordinates with other Python libraries, such as NumPy and Matplotlib, to perform more progressed information investigation tasks

5.2 MODEL IMPLEMENTATION

Data collection: Due to the unavailability of open-source data pertaining to cyber attacks, we curated a custom dataset by amalgamating diverse datasets sourced from reputable repositories like Zenodo. Employing rigorous preprocessing techniques, we transformed the raw data into a structured format, culminating in the creation of a comprehensive CSV file christened 'Storm.'

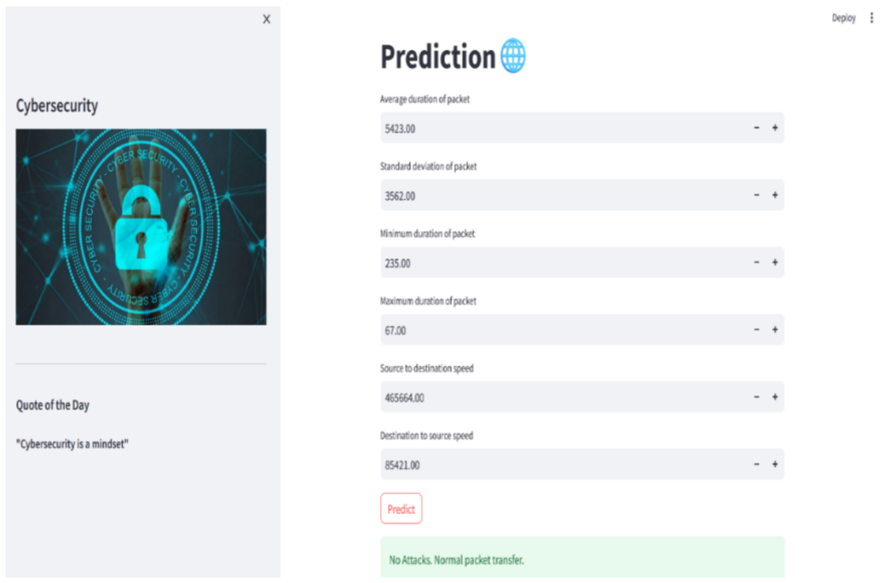
Data Preprocessing: At this stage the collected dataset undergoes few preprocessing steps for the purpose of cleaning of data. Which involves handling the missing values, duplicated values, Encoding techniques etc.

Feature Engineering and Data analysis: In this step the features relevant for the botnet attack detection are extracted from the preprocessed data. Here we have performed some visualization on the features of the dataset to see the insights present in the data. We have plotted some Bar graphs, Pie charts, Histograms, Violin chart, line chart. These plots are used to see the attacks by various packets, port numbers and so on.

Model Training and Evaluation: For this project we are using the deep learning techniques like ANN, LSTM and along with these we are also using few ML models for training to get the accurate predictions. The models will be trained on a portion of the dataset and tested on a separate unseen portion to assess their generalizability.

Model selection: Based on the evaluation results, the best performing model will be selected. For model selection we consider the performance metrices like Accuracy, Precision, Recall, F1 score.

User Interface Screen



Average duration of packet
5423.00

Standard deviation of packet
3562.00

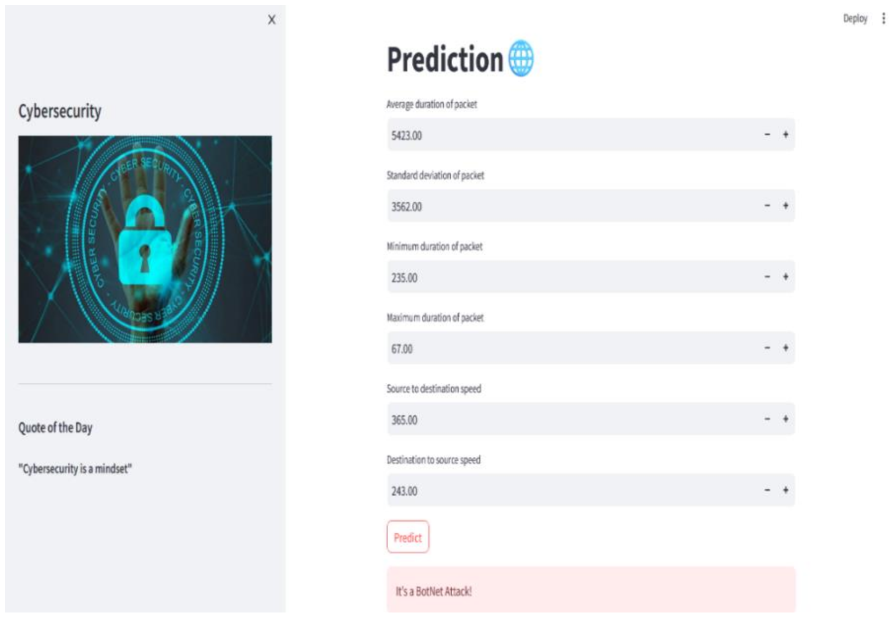
Minimum duration of packet
235.00

Maximum duration of packet
67.00

Source to destination speed
465664.00

Destination to source speed
85421.00

No Attacks. Normal packet transfer.



Average duration of packet
5423.00

Standard deviation of packet
3562.00

Minimum duration of packet
235.00

Maximum duration of packet
67.00

Source to destination speed
365.00

Destination to source speed
243.00

It's a BotNet Attack!

RESULT ANALYSIS

6.1-Accuracy: Accuracy refers to the proportion of correct predictions made by a model among the total number of predictions. It is a common evaluation metric used to assess the performance of classification models. The accuracy of a model is calculated by dividing the number of correct predictions (true positives and true negatives) by the total number of predictions (true positives, true negatives, false positives, and false negatives).

$$\text{Accuracy} = \frac{TP+FP+TN+FN}{TP+TN}$$

TP- True Positive, TN- True Negative, FP- False Positive, FN- False Negative

In this project we have used various DL and ML models for detection of Botnet Attacks. The table 6.1 shows the accuracy obtained by each model used.

Table 6.1. Predicted Accuracy of all Models

S.NO	**Model Implemented**	**Accuracy**
1.	Hybrid model (Random Forest and KNN)	0.94
2.	ANN	0.99
3.	LSTM	0.98
4.	Support Vector Machine	0.56
5.	Logistic Regression	0.66
6.	Decision tree Classifier	1.00
7.	Gradient Boosting Classifier	1.00
8.	Gaussian Naïve Bayes	0.61

6.2 Sensitivity and Specificity:

Sensitivity (also known as recall or true positive rate) measures the proportion of actual positive cases that were correctly identified by the model:

$$\text{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$

Specificity measures the proportion of actual negative cases that were correctly identified by the model:

$$\text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives + False Positives}}$$

These metrics help assess a model's ability to correctly identify positive and negative instances, respectively, providing insight into its performance beyond overall accuracy, especially in imbalanced datasets.

Fig 6.1 shows the sensitivity and specificity plot for all the Machine Learning models

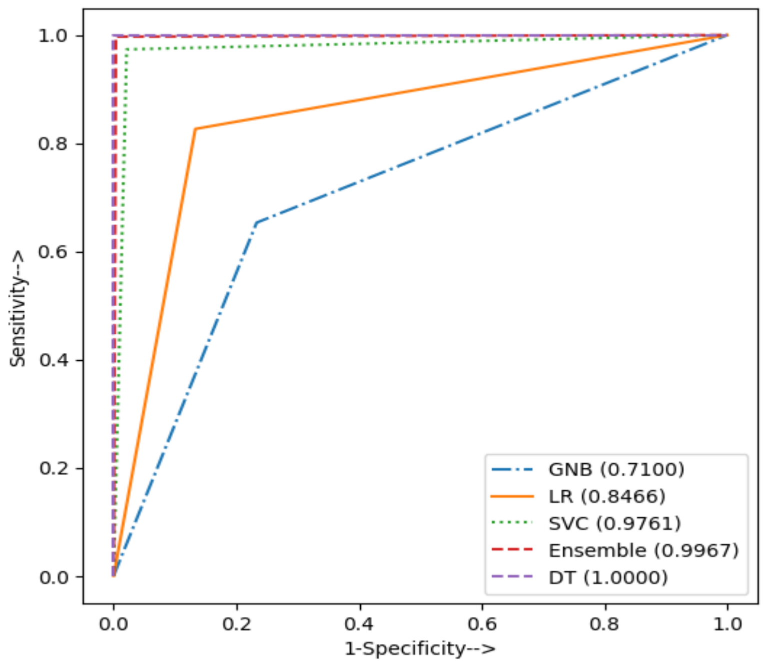


Fig 6.2.1 Sensitivity and Specificity graph plotted for ML model

6.3 Accuracy and Loss plotting:

Plotting accuracy and loss curves for both training and validation data provides insights into the model performance and training progress.

Accuracy Plot:

Training Accuracy: This curve shows how the accuracy of the model changes over epochs during training on the training dataset. It indicates how well the model is learning to correctly classify the training instances.

Validation Accuracy: This curve demonstrates the accuracy of the model on a separate validation dataset not used during training. It helps in assessing the model's ability to generalize to unseen data.

Loss Plot:

Training Loss: This curve depicts how the loss (error) of the model decreases over epochs during training on the training dataset. It reflects the convergence of the model during training.

Validation Loss: This curve represents the loss of the model on the validation dataset. It assists in monitoring overfitting; a decreasing training loss alongside an increasing validation loss may indicate overfitting, suggesting adjustments to the model architecture or training process.

Interpretation:

A consistent increase in both training and validation accuracy with decreasing loss signifies that the model is effectively learning and generalizing from the training data.

If the training accuracy continues to increase while the validation accuracy stagnates or decreases, it may indicate overfitting.

Divergence between training and validation loss or accuracy could suggest issues such as overfitting, underfitting, or data mismatch between training and validation sets.

In summary, analyzing these plots helps in understanding the training dynamics, identifying potential issues, and optimizing the ANN model for better performance and generalization.

The Fig 6.3.1 shows the Accuracy and Loss plotting for Training and Validation data in ANN model. The model is trained on 50 epochs resulting the Accuracy of 0.996 and Loss of 0.003.

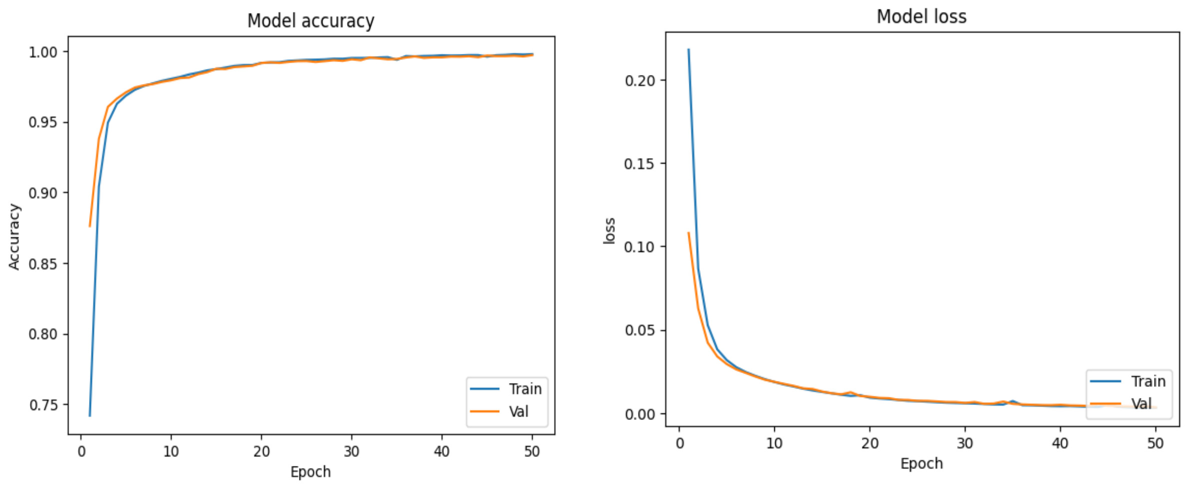


Fig 6.3.1 Accuracy and Loss plotting for Training and Validation data in ANN model

The Fig 6.3.2 shows the Accuracy and Loss plotting for Training and Validation data in LSTM model. The model is trained on 50 epochs resulting the Accuracy of 0.9907 and Loss of 0.0247.

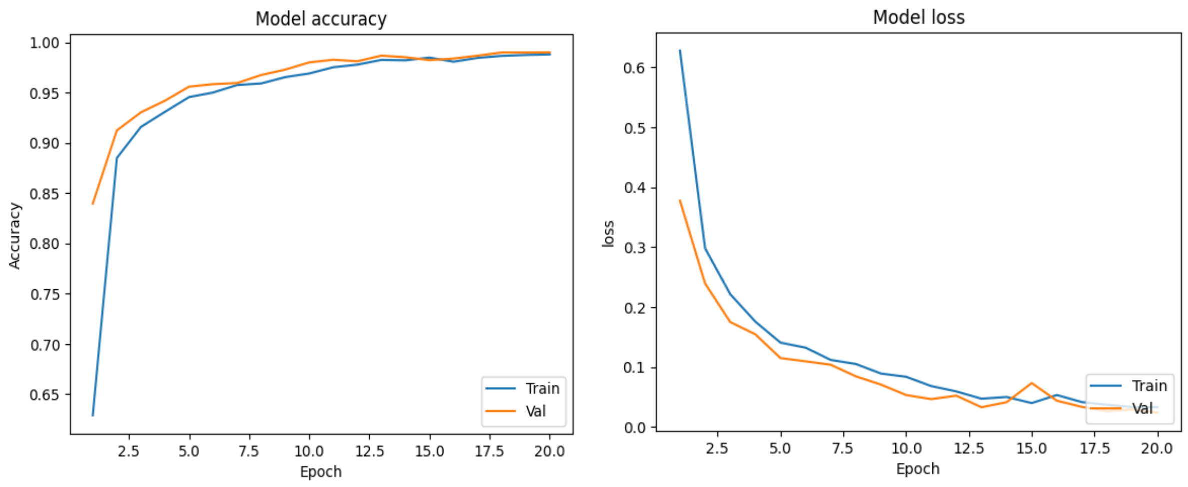


Fig 6.3.2 Accuracy and Loss plotting for Training and Validation data in LSTM model

CONCLUSION & FUTURE SCOPE

7.1 CONCLUSION

In summary, “Detecting and Mitigating Botnet Attacks in Software Defined Network” project presents an effective approach towards detecting of botnet attacks in SDN Environment. The proposed system can be able to surpass the existing systems in terms of accuracy, speed and robustness. In this system we selected few efficient parameters for detecting the Attacks. Along with DL models we have used few ML models which gives the accurate results. We observe both DL and ML models performed well with the accuracy of above 90 percent. We have used the Hybrid model which is the combination of Random Forest and KNN. The proposed system also contains a frontend for giving the features of the system like Average duration of packet, standard deviation of the packet, Source to destination speed, Destination to source speed, etc for the detection of the Botnet Attacks.

7.2 FUTURE SCOPE

The future scope of this project is outlined as follows:

Advanced Analytics: Implementing machine learning and AI algorithms to predict and detect botnet attacks in real-time, enabling faster response times and improving network security.

Integration with IoT: With the increasing number of IoT devices, integrating the hybrid model with IoT networks can help detect and prevent botnet attacks, ensuring the security of IoT devices and networks.

Multi-layer Security: Implementing multi-layer security measures, including intrusion detection systems, firewalls, and access control policies, can enhance the effectiveness of the hybrid model in detecting and preventing botnet attacks.

REFERENCES

NOE MARCELO YUNGAICELA-NAULA , CESAR VARGAS-ROSALES , (Senior Member, IEEE),AND JESUS ARTURO PEREZ-DIAZ. "SDN-Based Architecture for Transport and Application Layer DDoS Attack Detection by Using Machine and Deep Learning." School of Engineering and Sciences, Tecnologico de Monterrey, Monterrey 64849, Mexico. IEEE, 2021

Ivan Letteri, Massimo Del Rosso, Pasquale Caianiello, and Dajana Cassioli, "Performance of Botnet Detection by Neural Networks in Software-Defined Networks." Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, via Vetoio snc, 67100 L'Aquila, Italy

Ancy Sherin Jose, Latha R Nair, Varghese Paul, "Mitigation of Distributed Denial of Service (DDoS) Attacks over Software Defined Networks (SDN) using Machine Learning and Deep Learning Techniques." International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8 Issue-8S3, June 2019

Amran Mansoor , Mohammed Anbar, Abdullah Ahmed Bahashwan , Basim Ahmad Alabsi and Shaza Dawood Ahmed Rihan, "Deep Learning-Based Approach for Detecting DDoS Attack on Software-Defined Networking Controller." Deep Learning-Based Approach for Detecting DDoS Attack on Software-Defined Networking Controller. Systems 2023, 11, 296.

Sanjeetha R, Anant Raj, Kolli Saivenu, Mumtaz Irteqa Ahmed, Sathvik B and Anita Kanavalli, "Detection and mitigation of botnet based DDoS attacks using catboost machine learning algorithm in SDN environment." International Journal of Advanced Technology and Engineering Exploration, Vol 8(76) ISSN (Print): 2394-5443 ISSN (Online): 2394-7454, March 2021

Misbachul Munir , Ipung Ardiansyah 2) , Joko Dwi Santoso 3) , Ali Mustopa 4) , Sri Mulyatun, "DETECTION AND MITIGATION OF DISTRIBUTED DENIAL OF SERVICE ATTACKS ON NETWORK ARCHITECTURE SOFTWARE DEFINED NETWORKING USING THE NAIVE BAYES ALGORITHM." Journal of Information System Management (JOISM) e-ISSN: 2715-3088 Vol. 3, No. 2 (2022)

MUHAMMAD WAQAS NADEEM 1, HOCK GUAN GOH1, YICHIET AUN1, AND VASAKI PONNUSAMY, "Detecting and Mitigating Botnet Attacks in Software-Defined Networks Using Deep Learning Techniques." Digital Object Identifier 10.1109/ACCESS.2023.3277397, May 2023

Narmeen Zakaria Bawany, Jawwad A. Shamsi, Khaled Salah, "DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions." King Fahd University of Petroleum & Minerals 2017, 2017

ANDREA MELIS, AMIR AL SADI, DAVIDE BERARDI, FRANCO CALLEGATI, (Senior Member, IEEE), AND MARCO PRANDINI, "A systematic Literature Review of Offensive and Defensive Security Solutions With Software Defined Network", partment of Computer Science and Engineering, University of Bologna, 40126 Bologna, Italy. IEEE, 2023.

Seyed Mohammad Mousavi, "Early Detection of DDoS Attacks in Software Defined Networks Controller", Carleton iversity Ottawa, Ontario.IEEE, 2014.

Farhan Tariq, Shamim Baig, "Machine Learning Based Botnet Detection in Software Defined works", International Journal of security and its applications, Vol.11.IEEE, 2017

APPENDIX-I

SAMPLE CODE

st.py:

```
import streamlit as st
import pandas as pd
import numpy as np
import joblib

# Load the model
model = joblib.load(open("hybrid_model.pkl", "rb"))

def predict_with_selected_features(pktcount, src_10, src_1, bytecount, src_12, src_2, model):
    # Create dummy values for the remaining features
    dummy_values = [0] * 65  # Assuming the remaining 65 features are numerical

    # Combine the provided features with dummy values
    all_features = [pktcount, src_10, src_1, bytecount, src_12, src_2] + dummy_values

    # Reshape the data to match the format expected by the model
    input_data = np.array(all_features).reshape(1, -1)

    # Predict the outcome using the provided model
    prediction = model.predict(input_data)

    return prediction[0]  # Return the predicted outcome

def main():
    st.sidebar.title("Cybersecurity")
    st.sidebar.image("cybersecurity_logo.jpg", use_column_width=True)
    st.sidebar.markdown("---")
    st.sidebar.markdown("### Quote of the Day")
    st.sidebar.markdown("#### \"Cybersecurity is a mindset\"")

    st.title("Category of BotNet Attack Prediction:globe_with_meridians:")

    # Input form
    avg_dur = st.number_input("Average duration of packet")
    stddev_dur = st.number_input("Standard deviation of packet")
    min_dur = st.number_input("Minimum duration of packet")
    max_dur = st.number_input("Maximum duration of packet")
    srate = st.number_input("Source to destination speed")
    drate = st.number_input("Destination to source speed")

    # Predict button
    if st.button("Predict"):
        # Make prediction
        with st.spinner('Predicting...'):
            output = predict_with_selected_features(avg_dur, stddev_dur, min_dur, max_dur,
srate, drate, model)

        # Show prediction result
        if output == 1:
            st.error("It's a BotNet Attack!")
        else:
            st.success("No Attacks. Normal packet transfer.")

if __name__ == "__main__":
    main()
```

Botnet_Detection.ipynb:

```
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Assuming you have loaded your preprocessed data into X_train, y_train, X_test, y_test

# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

# Train base models
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(x_train, y_train)

knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(x_train, y_train)

# Combine predictions from base models
def hybrid_predict(X):
    rf_preds = rf_model.predict(X)
    knn_preds = knn_model.predict(X)
    combined_preds = []
    for rf_pred, knn_pred in zip(rf_preds, knn_preds):
        # Use majority voting for simplicity
        combined_pred = 1 if (rf_pred + knn_pred) >= 1 else 0
        combined_preds.append(combined_pred)
    return np.array(combined_preds)

# Evaluate the hybrid model
y_pred = hybrid_predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
print("Hybrid Model Accuracy:", accuracy)
```

```
from joblib import dump

# Save the trained base models
dump(rf_model, 'random_forest_model.joblib')
dump(knn_model, 'knn_model.joblib')

# Save the hybrid model function
def hybrid_predict(X):
    rf_preds = rf_model.predict(X)
    knn_preds = knn_model.predict(X)
    combined_preds = []
    for rf_pred, knn_pred in zip(rf_preds, knn_preds):
        # Use majority voting for simplicity
        combined_pred = 1 if (rf_pred + knn_pred) >= 1 else 0
        combined_preds.append(combined_pred)
    return np.array(combined_preds)

# Save the hybrid model function
dump(hybrid_predict, 'hybrid_model.joblib')


from joblib import load

# Load the trained base models
loaded_rf_model = load('random_forest_model.joblib')
loaded_knn_model = load('knn_model.joblib')

# Load the hybrid model function
loaded_hybrid_predict = load('hybrid_model.joblib')


y_pred = loaded_hybrid_predict(x_test)

# Evaluate the predictions
accuracy = accuracy_score(y_test, y_pred)
print("Hybrid Model Accuracy:", accuracy)
```

APPENDIX II
PLAGIARISM REPORT

ORIGINALITY REPORT

23%
SIMILARITY INDEX

PRIMARY SOURCES

1	www.techtarget.com Internet	149 words – 2%
2	www.geeksforgeeks.org Internet	144 words – 2%
3	fastercapital.com Internet	139 words – 2%
4	www.mdpi.com Internet	134 words – 2%
5	researchbank.swinburne.edu.au Internet	129 words – 2%
6	www.researchsquare.com Internet	123 words – 2%
7	mgnirsa.ac.in Internet	70 words – 1%
8	jurnal.polgan.ac.id Internet	46 words – 1%
9	"Machine Learning: Theoretical Foundations and Practical Applications", Springer Science and Business Media LLC, 2021 Crossref	43 words – 1%

10	dokumen.pub Internet	43 words – 1%
11	open-innovation-projects.org Internet	42 words – 1%
12	www.grafiati.com Internet	38 words – < 1%
13	www.coursehero.com Internet	37 words – < 1%
14	hackmd.io Internet	27 words – < 1%
15	medium.com Internet	24 words – < 1%
16	1library.net Internet	22 words – < 1%
17	"Cyberspace Safety and Security", Springer Science and Business Media LLC, 2018 Crossref	21 words – < 1%
18	Yerragudipadu Subbarayudu, Gurram Vijendar Reddy, Masuram Raj Kumar, Mudavath Aravind Naik, R.V.S.S. Nagini, Balpreet Singh. "An efficient novel approach for glaucoma classification on retinal fundus images through machine learning paradigm", MATEC Web of Conferences, 2024 Crossref	21 words – < 1%
19	doaj.org Internet	21 words – < 1%
20	saisahithineelamth522.sites.umassd.edu	

	Internet	20 words – < 1%
21	randomresearchai.medium.com Internet	19 words – < 1%
22	romanpub.com Internet	19 words – < 1%
23	dataaspirant.com Internet	18 words – < 1%
24	www.warse.org Internet	18 words – < 1%
25	Christina, Indra Aulia, Sarah Purnamawati. "Automating News Tweet Writing Based on Indonesia Language Using NLG Approach", 2023 IEEE International Conference on Cryptography, Informatics, and Cybersecurity (ICoCICs), 2023 Crossref	17 words – < 1%
26	Deepthi K. Prasad, L. Vibha, K.R. Venugopal. "Early detection of diabetic retinopathy from digital retinal fundus images", 2015 IEEE Recent Advances in Intelligent Computational Systems (RAICS), 2015 Crossref	17 words – < 1%
27	gredos.usal.es Internet	17 words – < 1%
28	webthesis.biblio.polito.it Internet	17 words – < 1%
29	lup.lub.lu.se Internet	16 words – < 1%

30 www.ijraset.com
Internet
16 words – < 1%

31 T.S. Balaji, S. Srinivasan. "Detection of safety wearable’s of the industry workers using deep neural network", Materials Today: Proceedings, 2021
Crossref
15 words – < 1%

32 F Parente. "Oral contrast enhanced bowel ultrasonography in the assessment of small intestine Crohn's disease. A prospective comparison with conventional ultrasound, x ray studies, and ileocolonoscopy", Gut, 2004
Crossref
14 words – < 1%

33 Sabila Nawshin, Salekul Islam, Swakkhar Shatabda. "PCA-ANN: Feature selection based hybrid intrusion detection system in software defined network", Journal of Intelligent & Fuzzy Systems, 2024
Crossref
14 words – < 1%

34 Syed Safdar Hussain, Syed Sajjad Haider Zaidi. "AdaBoost Ensemble Approach with Weak Classifiers for Gear Fault Diagnosis and Prognosis in DC Motors", Applied Sciences, 2024
Crossref
14 words – < 1%

35 papers.ssrn.com
Internet
14 words – < 1%

36 Abdullahi Aishatu Wabi, Ismaila Idris, Olayemi Mikail Olaniyi, Joseph A. Ojeniyi. "DDOS attack detection in SDN: Method of attacks, detection techniques, challenges and research gaps", Computers & Security, 2024
Crossref
12 words – < 1%

37 ceur-ws.org

	Internet	12 words – < 1%
38	www.daytrading.com Internet	12 words – < 1%
39	diposit.ub.edu Internet	11 words – < 1%
40	searchsdn.techtarget.com Internet	11 words – < 1%
41	zg104.github.io Internet	11 words – < 1%
42	Iweriebor, Lawrence Ejime. "Approach to Medicare Provider Fraud Detection and Prevention", Capitol Technology University, 2023 ProQuest	10 words – < 1%
43	link.springer.com Internet	10 words – < 1%
44	repositorium.sdum.uminho.pt Internet	10 words – < 1%
45	trilogi.ac.id Internet	10 words – < 1%
46	www.section.io Internet	10 words – < 1%
47	www.turcomat.org Internet	10 words – < 1%
48	M. D. Amala Dhaya, R. Ravi. "Multi feature behavior approximation model based efficient	9 words – < 1%

botnet detection to mitigate financial frauds", Journal of Ambient Intelligence and Humanized Computing, 2020
Crossref

49	Sai Pranavi Kamaraju, Kritiprasanna Das, Ram Bilas Pachori. "EEG Based Biometric Authentication System Using Multivariate FBSE Entropy", Institute of Electrical and Electronics Engineers (IEEE), 2023 Crossref Posted Content	9 words – < 1%
50	github.com Internet	9 words – < 1%
51	ia902502.us.archive.org Internet	9 words – < 1%
52	yusera khan, Tathagat Banerjee, Gagandeep Singh Narula, Ritika Wason. "HHO-UNet-IAA: Harris Hawks Optimization based Novel UNet-Inception Attention Architecture for Glaucoma Segmentation", Research Square Platform LLC, 2023 Crossref Posted Content	9 words – < 1%
53	"Future Data and Security Engineering", Springer Science and Business Media LLC, 2021 Crossref	8 words – < 1%
54	"Web, Artificial Intelligence and Network Applications", Springer Science and Business Media LLC, 2020 Crossref	8 words – < 1%
55	Anagha Patil, Arti Deshpande. "Evaluating ML models on CTU-13 and IOT-23 Datasets", 2023 International Conference on Advanced Computing Technologies and Applications (ICACTA), 2023 Crossref	8 words – < 1%

56 Soumya Shrivastava, Ravi Shukla, Shinu Abhi, Rashmi Agarwal. "Chapter 14 Assessing Student Engagement Levels Using Speech Emotion Recognition", Springer Science and Business Media LLC, 2024
Crossref
8 words – < 1%

57 Sridhar Alla, Suman Kalyan Adari. "Beginning MLOps with MLFlow", Springer Science and Business Media LLC, 2021
Crossref
8 words – < 1%

58 Worku Gachena Negera, Friedhelm Schwenker, Taye Girma Debelee, Henock Mulugeta Melaku, Yehualashet Megeresa Ayano. "Review of Botnet Attack Detection in SDN-Enabled IoT Using Machine Learning", Sensors, 2022
Crossref
8 words – < 1%

59 Za'ter, Muhy Eddin. "Machine Learning Framework for Power System Security Assessment", University of Colorado at Boulder, 2023
ProQuest
8 words – < 1%

60 curve.carleton.ca
Internet
8 words – < 1%

61 financedocbox.com
Internet
8 words – < 1%

62 kylo.tv
Internet
8 words – < 1%

63 www.researchgate.net
Internet
8 words – < 1%

64 Jin Wang, Liping Wang, Ruiqing Wang. "A Method of DDoS Attack Detection and Mitigation for the
7 words – < 1%

Comprehensive Coordinated Protection of SDN Controllers", Entropy, 2023
Crossref

65 Nematzadeh, Peyman. "Development of In-Field Data Acquisition Systems and Machine Learning-Based Data Processing and Analysis Approaches for Turfgrass Quality Rating and Peanut Flower Detection", Oklahoma State University, 2023 — 7 words – < 1%
ProQuest

66 Parneet Kaur, Manish Kumar, Abhinav Bhandari. "A review of detection approaches for distributed denial of service attacks", Systems Science & Control Engineering, 2017 — 7 words – < 1%
Crossref

67 Shchubelka, Khrystyna. "Medically Relevant Genome Diversity in Ukraine", Oakland University, 2023 — 7 words – < 1%
ProQuest

68 Hansen, Kristen Antonia. "Toward Precise Statistical Inference in Spatial Environmental Epidemiology", University of California, San Diego, 2023 — 6 words – < 1%
ProQuest

69 Pablo Ferri Borredà. "Deeep Continual Multimodal Multitask Modells for Out-of-Hospital Emergency Medical Call Incidents Triage Support in the Presence of Dataset Shifts", Universitat Politecnica de Valencia, 2024 — 6 words – < 1%
Crossref Posted Content

EXCLUDE QUOTES	OFF	EXCLUDE SOURCES	OFF
EXCLUDE BIBLIOGRAPHY	OFF	EXCLUDE MATCHES	OFF