IoT BOTNET ATTACK DETECTION USING BIG DATA ANALYTICS

Bharat Kotwani,kotwanibharat7@gmail.com
Sai Krishna Rohith K,skrishrkat7@gmail.com
Dheepak ,dheepak2429@gmail.com
Sai Teja N,saitejusridhar@gmail.com
Nikhita bobba ,nikhita6446@gmail.com
katuru ashrith reddy,kashrithreddy@gmail.com
Uday Kiran Ramaraju,udaykiranramaraju@gmail.com
Raja Nithin Batchu,nitinbatchu15@gmail.com

ABSTRACT

Botnets pose a significant and escalating threat to both Internet and network security. The growing ability to identify suspicious online activities has led attackers to adopt more intricate and sophisticated methods of assault. Bots, essentially zombified computers manipulated by a malicious entity, serve as tools for carrying out attacks, spamming, phishing, and extracting information. The primary culprits behind major distributed denial of service (DDoS) botnet attacks have been Internet of Things (IoT) devices for quite some time. This persistent threat persists because numerous manufacturers continue to release IoT products lacking adequate security measures.

The vulnerability of IoT devices is exacerbated by their limited memory and computational resources, rendering them susceptible to security breaches. Furthermore, a multitude of security flaws persists due to the insufficient implementation of robust security mechanisms. The existing rule-based detection systems are often inadequate, as attackers find ways to circumvent them.

The predominant focus of botnet research revolves around the detection and prevention of malicious bot activities. Effective botnet detection requires advanced analytical capabilities, which are contingent on the nature of the data selected for analysis and the characteristics of the activities being examined. In this study, we aim to employ various algorithms for predicting botnet activity using big data and the Spark framework.

INTRODUCTION

In the contemporary landscape, Internet of Things (IoT) technology has found diverse applications in networks, captivating the attention of both research and industrial communities. However, this widespread adoption has brought about a surge in the number of vulnerable or unprotected IoT devices, leading to a significant rise in suspicious activities like IoT botnets and large-scale cyber-attacks. This research endeavors to construct a Big Data Analytics model designed for detecting IoT botnet attacks. Individual members will leverage distinct IoT Botnet Attack Datasets, treating them with diverse algorithms. Ultimately, the datasets will be amalgamated, resulting in a comprehensive model for detecting such attacks.

The imperative for promptly identifying IoT botnet attacks has become crucial in mitigating associated risks. Our research aims to assess the security and resilience of everyday IoT devices against the Mirai and Gafgyt botnets. The conclusion of the study will involve a comparative analysis of results obtained from the big data analysis of the provided datasets, evaluating the accuracy of botnet attack predictions.

Traditional botnet detection methods often struggle to scale up to meet the demands of multi-Gbps networks. This scalability challenge extends beyond the detection process, encompassing all facets of the botnet detection system, including data gathering, storage, feature extraction, and analysis.

Our research addresses this challenge by working with an extensive dataset and conducting big data analysis using Spark on the Azure environment. In the second phase of the review, each team member will focus on specific file types independently. To achieve this, each member will utilize 4-5 distinct CSV files, creating their own datasets by merging these files. Subsequently, individual analyses will be performed on these customized datasets, and the results will be compared to derive comprehensive insights.

LITERATURE SURVEY ON IOT DEVICESAGAINST MIRAI BOTNET

1. Testing And Hardening IoT Devices Against the Mirai Botnet, 2020

Numerous inexpensive Internet of Things (IoT) devices are being delivered brand new with default configurations, lacking proper security measures from manufacturers. This oversight makes them susceptible to existing malware present on the Internet, such as the infamous Mirai botnet. The leakage of Mirai's source code to the public has empowered malicious actors to easily configure and deploy it. The unsafe and inefficient use of software assets puts consumers at significant risk of complete compromise.

This study specifically utilized Mirai libraries to configure and launch attacks on four distinct IoT devices. The experiments revealed that 3 out of the 4 devices were vulnerable to the Mirai malware and became infected when deployed with their original configurations. This highlights a critical flaw in relying on default security settings, as they fail to provide adequate protection for consumers, leaving their devices exposed to potential threats.

By closely analyzing the Mirai libraries and their attack vectors, the researchers identified effective device configuration countermeasures to fortify the devices against this botnet. Subsequent experimentation successfully verified the efficacy of these countermeasures, emphasizing the importance of implementing proper security measures beyond default settings to safeguard IoT devices from potential compromises.

2. IoT Botnet Forensics: A Comprehensive Digital Forensic Case Study on Mirai Botnet Servers, 2020

This paper represents the inaugural comprehensive digital forensic case study on Mirai, a prominent type of IoT bot malware. While prior research extensively delved into the botnet architecture and analyzed Mirai's source code and its variants using traditional static and dynamic malware analysis techniques, there has been a notable absence of sufficient analysis of infected devices and Mirai network components. This study constructs a fully operational Mirai botnet network architecture and conducts a thorough forensic analysis on the Mirai botnet server.

The investigation covers forensic artifacts present on various elements of the Mirai ecosystem, including the attacker's terminal, CNC server, database server, scan receiver, and loader, along with insights into network

packets originating from these components. Importantly, the paper discusses methods by which a forensic investigator can remotely acquire some of these artifacts, even in the absence of immediate physical access to the botnet server.

This research yields valuable findings for forensic investigators, not only in terms of the data that can be obtained (such as IP addresses of bot members) but also crucial information guiding the selection of devices to target for procurement and research, enhancing the investigator's ability to extract useful insights from the Mirai botnet.

3. Securing an Internet of Things from Distributed Denial of Service and Mirai Botnet Attacks Using a Novel Hybrid Detection and Mitigation Mechanism, 2020

The prevalence of IoT in various applications and industries, including medical, military, and transportation, has become widespread. However, the inherent vulnerabilities in the communication channels of IoT devices pose significant challenges. Among the various threats, Distributed Denial of Service (DDoS) attacks emerge as particularly menacing, posing a substantial risk to the functionality of IoT devices. The presence of DDoS attacks in the network layer can result in severe damage to the data transmission channel, leading to data loss or system collapse. This research aims to address these challenges through innovative detection and mitigation strategies for Mirai and DDoS attacks in IoT environments.

The study begins by designing several IoT devices using a novel Hybrid Strawberry and African Buffalo Optimization (HSBABO) approach. Subsequently, different types of DDoS attacks are launched within the developed IoT network. The fitness of Strawberry and African Buffalo is utilized to identify and categorize the specific types of attacks. To counteract DDoS threats in IoT architectures, the researchers introduce a novel MCELIECE encryption with Cloud Shield scheme.

In the evaluation phase, the research demonstrates promising results, with an attack detection accuracy of 94%. The false-negative rate stands at 3%, indicating a low likelihood of missing actual attacks, while the false-positive rate is 5.5%, highlighting a relatively small number of false alarms. These findings underscore the effectiveness of the proposed approach in detecting and mitigating Mirai and DDoS attacks in IoT environments.

4. Protecting IoTs from Mirai Botnet Attacks using Blockchains, 2019

The Mirai Botnet malware exploits vulnerabilities in IoT devices, leading to widespread Distributed Denial of Service (DDoS) attacks. While several methodologies have been suggested to mitigate Mirai botnet attacks, many of them are centralized or merely offer precautionary measures for securing IoT devices. This paper presents a novel blockchain-based architecture for enhancing IoT security against Mirai Botnet attacks.

The proposed approach involves dividing the network into distinct Autonomous Systems (AS), establishing host connectivity within each AS. Blockchains are leveraged to store and share a list of IP addresses associated with hosts within an AS, indicating whether they are classified as malicious. Each AS monitors communication activities within the network, determining the presence of malware in a host by comparing its total sent packets with a predefined threshold value. The approach is implemented on a custom-developed simulator, which aids in determining an appropriate value for the malicious threshold. Results indicate the effectiveness of the proposed method in blocking malicious packets from infected hosts, preventing adverse impacts on the victim's response time. Additionally, scalability analysis and estimation of block propagation delay for various AS sizes and consensus algorithms are performed. The results highlight a true detection rate of 95%.

In essence, this research demonstrates that the blockchain-based architecture is an efficient and scalable solution for detecting and mitigating Mirai Botnet attacks, offering a robust defense mechanism against malicious activities targeting IoT devices.

5. A Game-Theoretic Technique for Securing IoT Devices against Mirai Botnet, 2018

The Internet of Things (IoT) has gained immense popularity in recent years, finding applications in various industries, households, and medical settings. However, the rise of the Mirai botnet poses a significant threat to IoT, representing the largest registered botnet known to infect IoT devices. At its peak, this botnet executed a successful attack affecting approximately 100 thousand devices. The aftermath revealed around 1.2 million infected IoT devices, with 170 thousand actively participating in the attack.

Safeguarding IoT devices from botnet activities is crucial, and this paper introduces a novel technique designed to detect attempts to compromise IoT devices. The approach utilizes a game-theoretic mathematical model specifically crafted to characterize attacks on IoT devices. By employing this technique, it becomes possible to identify and record attempts to compromise the security of IoT devices, enhancing overall cybersecurity in the IoT ecosystem.

LITERATURE SURVEY ABOUT DIFFERENT METHODS OF IOT BOTNET ATTACK DETECTION USED

6. N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders

The method proposed in this paper for detecting IoT botnet attacks relies on utilizing deep autoencoders for individual devices. These autoencoders are trained on statistical features extracted from benign traffic data. When applied to new data, potentially infected with IoT botnet attacks, anomalies detected by the autoencoders may indicate that the device is compromised. The botnets under consideration in this study are Mirai and Bashlite.

Following the training and optimization of the autoencoders using benign data, the same dataset was employed to train three other commonly used anomaly detection algorithms: Local Outlier Factor (LOF), One-Class SVM, and Isolation Forest. Similar to the autoencoders, the hyperparameters of these algorithms were optimized. Finally, the proposed method was tested against attacks with the same duration via Mirai and BASHLITE's Command and Control (C&C) servers.

RESULT:

In terms of True Positive Rate (TPR), False Positive Rate (FPR), and detection time, the deep autoencoders demonstrated superiority for most devices. This can be attributed to the capability of deep architectures to learn nonlinear structure mappings and approximate complex functions. Moreover, the constrained complexity of deep autoencoders, enforced by the reduced dimensionality in the hidden layers, prevents them from merely learning the trivial identity function. As a result, deep autoencoders are inclined to fit common patterns more effectively than uncommon ones. This ability to capture and represent nontrivial features contributes to their enhanced performance in detecting anomalies in IoT devices.

7. Improving IoT Botnet Investigation Using an Adaptive Network Layer

The proposed solution in this paper offers the capability to modify network traffic at the network layer based on the actions carried out by malware. The study focused on investigating the Mirai and Bashlite botnet families, showcasing the ability to block attacks on other systems, identify targets of attacks, and rewrite commands sent by the botnet controller to the infected devices.

The methodology introduced in the paper addresses the handling of network traffic generated by IoT malware within an analysis environment. Beyond merely mitigating the impact of the network traffic generated by the malware sample, the approach leverages the flexibility of the network layer. The primary objectives include implementing effective mechanisms to block attacks, fingerprinting the botnet Command and Control (C&C), monitoring the communication channel, and manipulating instructions directed at the infected device. This comprehensive approach aims to enhance the overall security posture by actively responding to and modifying malicious network activities associated with IoT malware.

RESULT:

To assess the efficacy of the proposed solution, the researchers conducted three experiments analyzing the Mirai and Bashlite malware families. In the first experiment, which concentrated on network traffic containment, they demonstrated the ability to discern malware communication by blocking attacks while allowing communication with the botnet Command and Control (C&C). In the second experiment, the

capabilities of the system were exploited to rewrite packet payloads. This showcased how the system can modify botnet instruction messages and manipulate network flows by redirecting a bot to an illegitimate botnet control infrastructure. These experiments collectively highlight the solution's effectiveness in distinguishing and actively intervening in malicious activities associated with these malware families.

Intelligent Detection of IoT Botnets Using Machine Learning and Deep Learning

Machine learning (ML) serves as an alternative technique, allowing the development of optimal security models based on empirical data from individual devices. This paper adopts the ML approach for detecting attacks on Internet of Things (IoT) devices, with a specific focus on botnet attacks. The research endeavors to create ML-based models tailored to different types of IoT devices that may be targeted by botnet attacks. The study utilizes the N-BaIoT dataset, which involves injecting botnet attacks (specifically Bashlite and Mirai) into various IoT devices such as Doorbells, Baby Monitors, Security Cameras, and Webcams.

The researchers develop distinct botnet detection models for each type of device using a range of ML models, including deep learning (DL) models. The ML models encompass five types: naïve Bayes (NB), K-nearest neighbors (KNN), logistic regression (LR), decision tree (DT), and random forest (RF). Additionally, three types of DL models, namely convolutional neural network (CNN), recurrent neural network (RNN), and long short-term memory (LSTM), are employed in the evaluation process. This comprehensive approach allows for the exploration of various ML and DL techniques to enhance the detection capabilities against botnet attacks on diverse IoT devices.

8. Botnets Attack Detection Using Machine Learning Approach for IoT Environment

In this study, the well-known machine learning algorithm, the Classification and Regression Tree (CART) algorithm, was employed as the detection model. To assess the proposed detection system, Python libraries, particularly scikit-learn, were utilized. The implementation was carried out in the Python programming language. Initially, the selected dataset was loaded into the learning program, with one-third of the dataset allocated for testing/validation and the remaining two-thirds for training purposes. The classification results, based on the CART algorithm, were extracted using this tool. Additionally, the model was trained using the Naïve Bayes Algorithm.

Following the extraction of the necessary features, these features underwent the detection phase, which constitutes the core component of the proposed system. For the detection phase, the CART algorithm was employed to construct the detection model, serving the dual purpose of building the model and classifying incoming patterns or features as either attack or normal. A selected subset of instances from the dataset was used to build the detection model using the machine learning classifier, the CART algorithm. This process allows the system to effectively distinguish between normal and attack patterns based on the learned features and classification model.

RESULT:

The results indicate that, on average, the detection accuracy of Naïve Bayes is 58%, while the average detection accuracy of the CART algorithm is 99% in the implementation of the detection system. These findings collectively suggest that the proposed system, particularly when utilizing the CART algorithm, is well-suited for detecting botnet attacks in the IoT environment. The significantly higher average detection accuracy of CART underscores its effectiveness in distinguishing between normal and attack patterns, showcasing its suitability for robust and accurate detection of botnet activities in the IoT ecosystem.

Machine Learning-Based IoT-Botnet Attack Detection with Sequential Architecture

This study introduces a machine learning (ML)-based framework for detecting botnet attacks, incorporating a sequential detection architecture. To streamline the system and enhance efficiency, a correlated-feature

selection approach is employed to eliminate irrelevant features, resulting in a lightweight implementation. The proposal adopts a versatile approach wherein classifiers based on different ML algorithms can be applied in various attack detection sub-engines. This approach aims to improve detection performance, reduce processing times, and achieve a lightweight design.

Several ML algorithms, such as Naïve Bayes, J48, and Artificial Neural Network (ANN), were experimented with in the study. The "Model Selector" module is responsible for implementing and evaluating multiple ML algorithms, selecting the most effective one for each sub-engine based on detection accuracy. This hybrid detection architecture allows different sub-engines to utilize distinct classifiers using diverse ML algorithms, contributing to a flexible and adaptive framework for botnet attack detection.

RESULT:

The results depicted in the figure demonstrate that the proposed detection scheme significantly outperforms the conventional detection architecture when using Naïve Bayes (NB) and Artificial Neural Network (ANN) classifiers. This improvement in performance suggests the efficacy of the sequential detection architecture with the correlated-feature selection approach. The results underscore the potential advantages of the proposed machine learning-based framework in achieving superior botnet attack detection compared to traditional architectures, especially when employing NB and ANN classifiers.

LITERATURE SURVEY ON SECURITY ATTACKS DETECTION USING BIG DATA ANALYSIS

9. Big Data Analytics framework for Peer-to-Peer Botnet detection using Random Forests, 2019

This paper builds upon the advancements made by open-source tools such as Hadoop, Hive, and Mahout to create a scalable implementation of a quasi-real-time intrusion detection system. The focus is on detecting Peer-to-Peer Botnet attacks using a machine learning approach. The implementation involves constructing a distributed framework utilizing Hive for sniffing and processing network traces, thereby extracting dynamic network features. The parallel processing capabilities of Mahout are leveraged to build a Random Forest-based Decision Tree model, which is then applied to the task of detecting Peer-to-Peer Botnets in quasi-real-time. The paper presents the implementation setup, performance metrics, initial observations, and proposes future extensions.

The technologies at the core of this framework include Libpcap, Hadoop, MapReduce, and Mahout. Libpcap is utilized for capturing packet data from a live network, and Tshark, a network protocol analyzer built on Libpcap, is employed to extract the required fields from the packets. This information is crucial for generating the feature set for the Machine Learning Module. After the Sniffer Module extracts the necessary data, the MapReduce paradigm is employed for feature extraction. Apache Hive is used in this context, providing a SQL-like language (HiveQL) to query the data and facilitate the extraction of features for further analysis.

10. A Fully Scalable Big Data Framework for Botnet Detection Based on Network Traffic Analysis, 2020

This paper addresses the scalability challenges faced by traditional Botnet detection methods, particularly in the context of multi-Gbps networks. The scalability issues extend beyond the detection process and affect various components of the Botnet detection system, including data gathering, storage, feature extraction, and analysis. The proposed solution introduces a fully scalable big data framework that allows scaling for each individual component of Botnet detection. This framework is designed to be compatible with various Botnet detection methods, including statistical methods, machine learning methods, and graph-based methods.

The experimental results demonstrate the success of the proposed framework in live tests conducted on a real network with 5 Gbps of traffic throughput and 50 million IP addresses visits. Notably, the runtime of the framework scales logarithmically with respect to the volume of the input data. For instance, when the scale of the input data increases by 4x, the total runtime only increases by 31 percent. This represents a significant improvement compared to other schemes, such as Botcluster, where runtime increases by 86 percent under similar scaling conditions. These results underscore the efficiency and scalability achieved by the proposed big data framework in addressing the challenges of Botnet detection in high-speed networks.

11. A Hybrid Deep Learning Model for Efficient Intrusion Detection in Big Data Environment, 2019

This paper introduces a hybrid deep learning model designed for efficient network intrusion detection, employing both a convolutional neural network (CNN) and a weight-dropped, long short-term memory (WDLSTM) network. The deep CNN is utilized to extract meaningful features from Intrusion Detection System (IDS) big data, while the WDLSTM network helps preserve long-term dependencies among these extracted features to prevent overfitting on recurrent connections. The hybrid model is compared with traditional approaches in terms of performance, showcasing its satisfactory results on a publicly available

dataset.

Two main techniques are employed for data pre-processing in this study: data conversion and data normalization. Data conversion is utilized to convert nominal features of traffic into numeric format, ensuring that all data are in a suitable numeric format for processing by the intrusion-detection model. Data normalization is applied to reduce the large variance of features into a specific range of values, also handling the removal of null values during the normalization process. To further normalize large values and mitigate their impact, a minimum–maximum scaling method is applied, placing values between zero and one.

The proposed model represents a hybrid deep learning approach for real-time intrusion classification of data traffic, combining the strengths of a deep CNN with weight-dropped long short-term memory (WDLSTM), creating what is termed a deep CNN–WDLSTM model.

12. Faster Detection and Prediction of DDoS attacks using MapReduce and Time Series Analysis, 2018

This paper addresses the significant challenge posed by Distributed Denial of Service (DDoS) attacks, especially in the context of emerging computing technologies like Cloud Computing and the Internet of Things. The primary issue lies in the fast detection and prevention of DDoS attacks due to the substantial processing required for large log files generated by every request and packet, often leading to denial of service for legitimate users and adversely impacting system resources.

The proposed method leverages Hadoop HDFS and MapReduce for expedited DDoS attack detection. The approach involves dividing the log file into multiple parts and distributing these segments across the Hadoop cluster for parallel processing and anomaly detection. A counter-based algorithm is employed to measure the number of requests related to different protocols (e.g., ICMP, TCP, HTTP) from a unique IP address. If the count exceeds a predefined threshold within a given time frame, the IP address is flagged as a potential attacker, allowing for temporary or permanent blocking based on detailed analysis.

To enhance detection efficiency in cases where DDoS attacks might go undetected due to misbehaving sources mimicking legitimate users, the paper introduces a technique based on Multiple-Window Peak Analysis. Additionally, the paper contributes by proposing the use of a time series prediction technique for

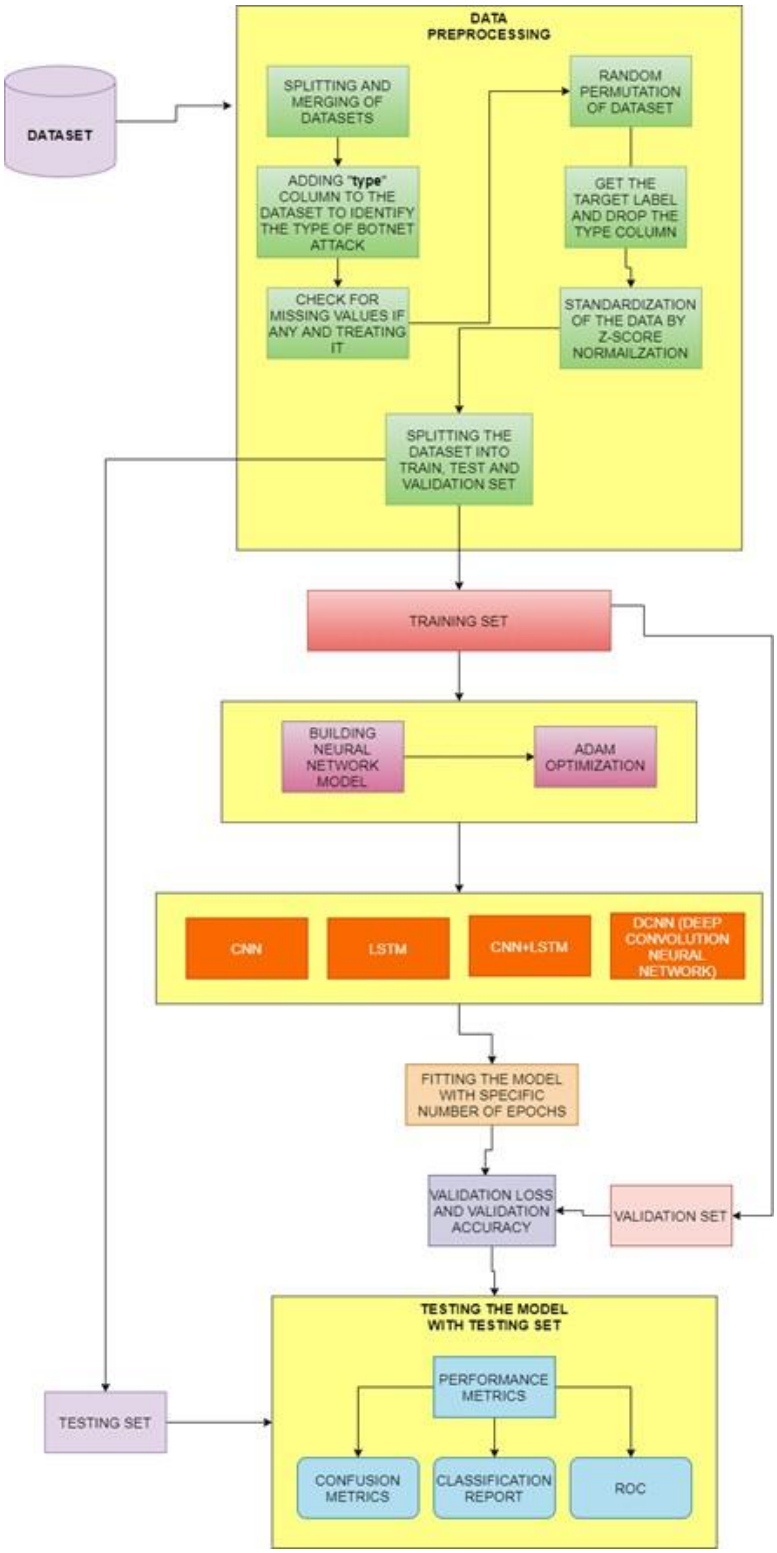
early detection of misbehaving IPs that could be part of potential DDoS attacks. Hadoop is employed for time series analysis, enabling MapReduce operations on log files at frequent intervals, as short as one minute. This comprehensive approach aims to significantly improve the speed and accuracy of DDoS attack detection in large-scale environments..

13. Distributing Extreme Learning Machines with Apache Spark for NetFlow-Based Malware Activity Detection, 2018

In the current landscape, information technology systems face complex cyber threats, including sophisticated malware. Infected computers often connect to form a 'botnet,' which can be centrally controlled by cybercriminals for various malicious activities like DDoS attacks, SPAM distribution, ransomware, and data theft. Traditional signature-based detection techniques are becoming less effective against these advanced threats. Consequently, advanced technologies, including Big Data, Distributed Data Mining, and Machine Learning, are emerging as viable solutions to counter cyber-attacks and cybercrime.

This paper introduces a method that combines NetFlows with an Extreme Learning Machines (ELM) classifier, trained in a distributed environment using the Apache Spark framework. The key contribution of this research lies in an algorithm that utilizes the Map-Reduce programming model to scale and distribute the training process of an ELM classifier for the detection of malware activities based on NetFlows. By leveraging distributed computing capabilities, the proposed method aims to enhance the efficiency and scalability of malware detection in large-scale network environments.

PROPOSED MODEL



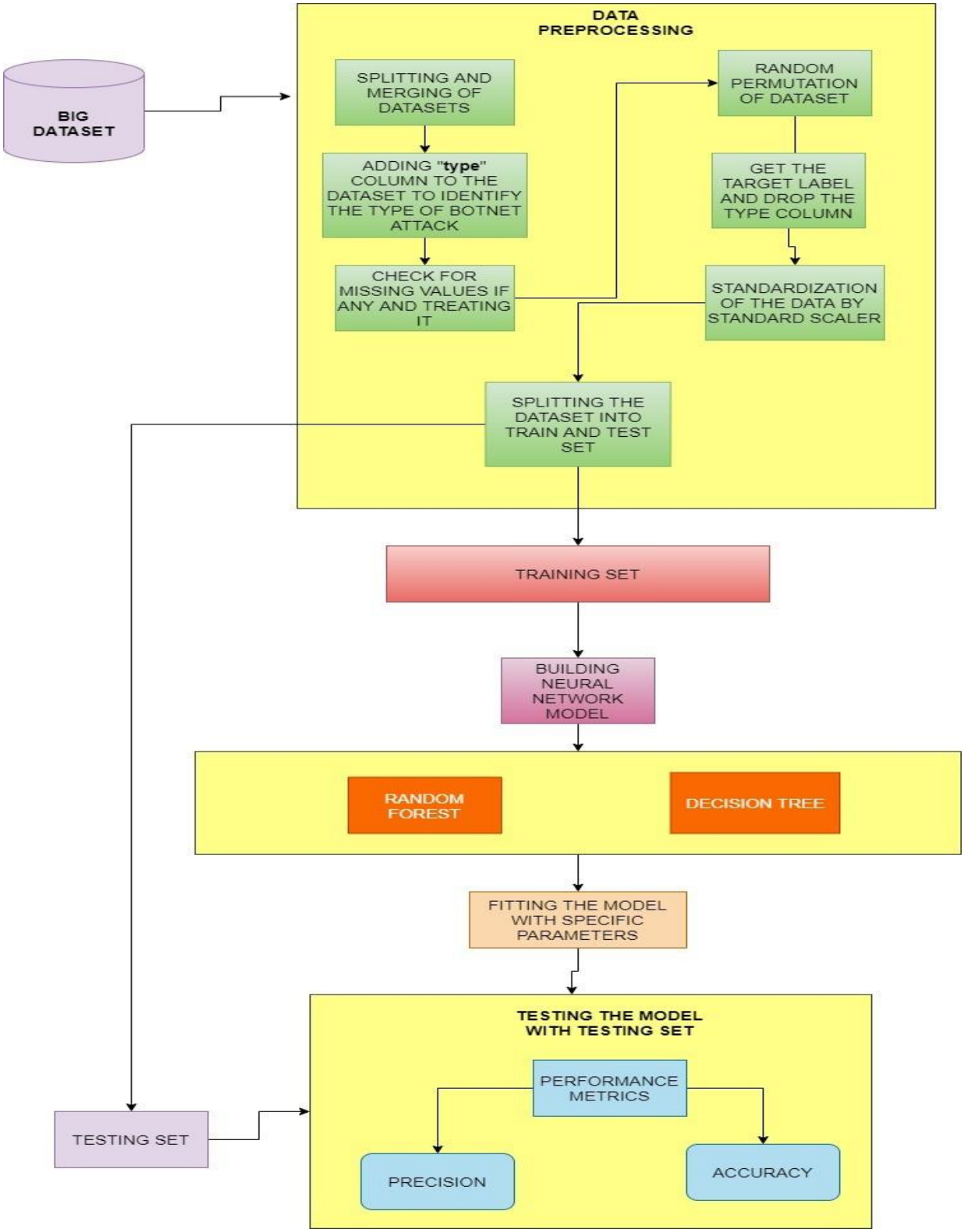
DESCRIPTION OF MODULES IN REVIEW 2:

The process involved loading and preprocessing the data. Initially, a "type" column was added to the datasets. Subsequently, a check for missing values was conducted, revealing the absence of any in our dataset. Following this, the rows were shuffled to prevent bias during model training. Additionally, data standardization was performed using the MIN-MAX SCALER.

After the preprocessing phase, the data was divided into training and testing sets with an 8:2 ratio. The training set was further subdivided into a validation set, employed for training the deep learning models. Models were constructed for the respective algorithms, and Adam optimization was applied to each model, serving a role akin to hyperparameter tuning. The models were then trained using a specific number of epochs on the training dataset.

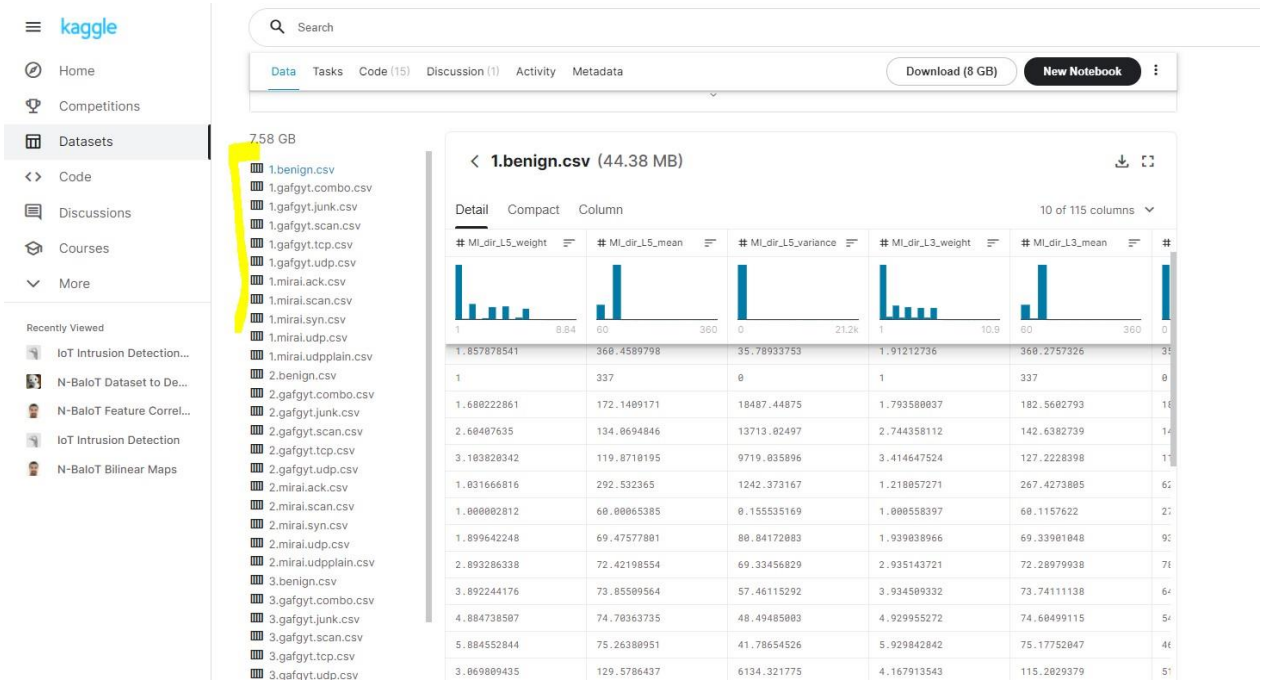
Post-training, the models underwent validation for validation loss and accuracy. Subsequently, the trained models were used for predictions on a testing dataset. The evaluation metrics employed for the obtained results included a Classification report, Confusion matrix, and ROC curve.

REVIEW 3:


DATA PREPROCESSING
BIG DATASET
SPLITTING AND MERGING OF DATASETS
ADDING "type" COLUMN TO THE DATASET TO IDENTIFY THE TYPE OF BOTNET ATTACK
CHECK FOR MISSING VALUES IF ANY AND TREATING IT
RANDOM PERMUTATION OF DATASET
GET THE TARGET LABEL AND DROP THE TYPE COLUMN
STANDARDIZATION OF THE DATA BY STANDARD SCALER
SPLITTING THE DATASET INTO TRAIN AND TEST SET
TRAINING SET
BUILDING NEURAL NETWORK MODEL
RANDOM FOREST
DECISION TREE
FITTING THE MODEL WITH SPECIFIC PARAMETERS
TESTING THE MODEL WITH TESTING SET
PERFORMANCE METRICS
PRECISION
ACCURACY
TESTING SET

PLAN FOR REVIEW 3

In review 3, we have merged 33 different datasets of categories 1, 2, 4 from the following link as provided by ma'am. https://www.kaggle.com/mkashifn/nbaiot-dataset



We performed analysis on that single .csv file and compared the result with review 2 analysis. we tried using Hadoop And Databricks. Finally we have performed big data implementation using Microsoft Azure cloud environment. We explored the Pyspark library and used Random forest and Decision tree classifier models for our analysis.

DESCRIPTION OF MODULES IN REVIEW 3:

We transitioned to Microsoft Azure for Big Data management. In the provided datasets, we introduced an additional column in each file named "class," containing integral values corresponding to different types of botnets and attacks. Combining 33 datasets resulted in a merged dataset of 1.4 GB. Initially, we loaded this data into the Synapse workspace analytics using Azure Data Studio.

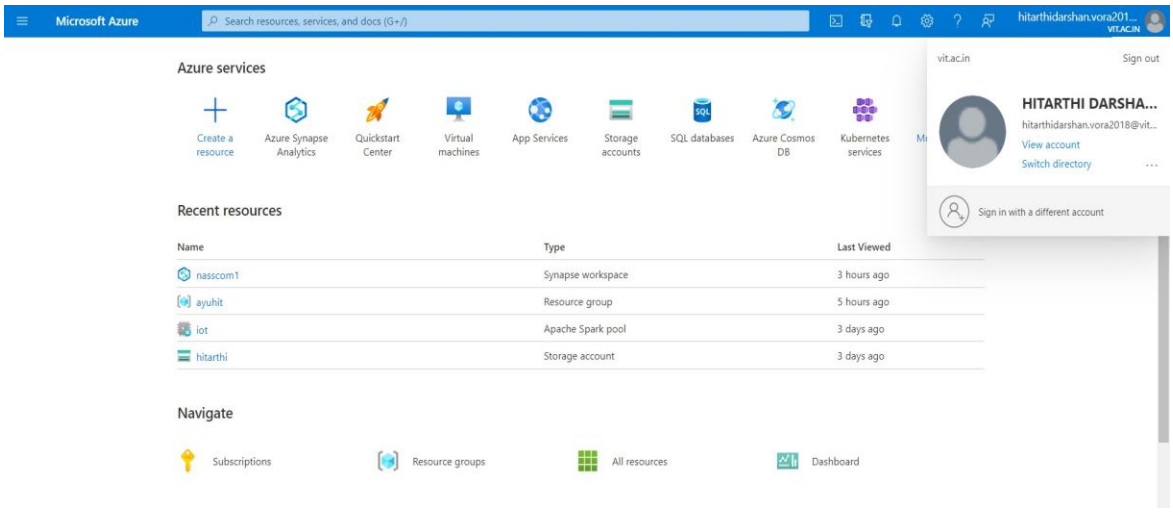
Our data processing involved the utilization of Vector Assembler and Standard Scaler. Subsequently, we

partitioned our dataset into training and testing sets with an 8:2 ratio. The train and test sets were further segregated into features and labels.

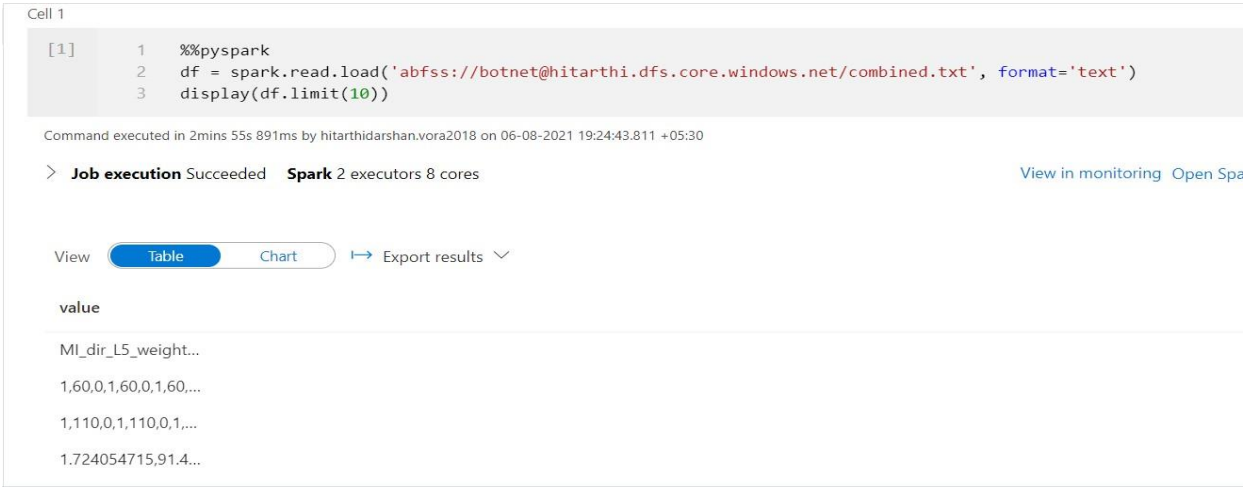
For model implementation, we employed Random Forest and Decision Tree Classifiers on the dataset. Additionally, we calculated accuracy and precision metrics for both classifiers to evaluate their performance.

IMPLEMENTATION AND EXPLANATION OF CODE:

PLATFORM USED - MICROSOFT AZURE



LOAD THE DATA



Cell 2

```
[2]
1  from pyspark.sql import SQLContext
2  from pyspark.sql.types import *
3  sqlContext = SQLContext(sc)
4
5  df = sqlContext.read.load('abfss://botnet@hitarthi.dfs.core.windows.net/combined.txt',
6                            format='com.databricks.spark.csv',
7                            header='true',
8                            inferSchema='true')
9
10 df.count()
```

Command executed in 3mins 38s 204ms by hitarthidarshan.vora2018 on 06-08-2021 19:25:26.182 +05:30

> **Job execution** Succeeded **Spark** 2 executors 8 cores View in monitoring Open Spark UI

```
2209674
```

Cell 8

```
[8]
1  df.schema.names
```

Command executed in 3mins 50s 699ms by hitarthidarshan.vora2018 on 06-08-2021 19:25:38.818 +05:30

```
['MI_dir_L5_weight', 'MI_dir_L5_mean', 'MI_dir_L5_variance', 'MI_dir_L3_weight', 'MI_dir_L3_mean', 'MI_dir_L3_variance',
'MI_dir_L1_weight', 'MI_dir_L1_mean', 'MI_dir_L1_variance', 'MI_dir_L0_1_weight', 'MI_dir_L0_1_mean', 'MI_dir_L0_1_variance',
'MI_dir_L0_01_weight', 'MI_dir_L0_01_mean', 'MI_dir_L0_01_variance', 'H_L5_weight', 'H_L5_mean', 'H_L5_variance', 'H_L3_weight',
'H_L3_mean', 'H_L3_variance', 'H_L1_weight', 'H_L1_mean', 'H_L1_variance', 'H_L0_1_weight', 'H_L0_1_mean', 'H_L0_1_variance',
'H_L0_01_weight', 'H_L0_01_mean', 'H_L0_01_variance', 'HH_L5_weight', 'HH_L5_mean', 'HH_L5_std', 'HH_L5_magnitude', 'HH_L5_radius',
'HH_L5_covariance', 'HH_L5_pcc', 'HH_L3_weight', 'HH_L3_mean', 'HH_L3_std', 'HH_L3_magnitude', 'HH_L3_radius', 'HH_L3_covariance',
'HH_L3_pcc', 'HH_L1_weight', 'HH_L1_mean', 'HH_L1_std', 'HH_L1_magnitude', 'HH_L1_radius', 'HH_L1_covariance', 'HH_L1_pcc',
'HH_L0_1_weight', 'HH_L0_1_mean', 'HH_L0_1_std', 'HH_L0_1_magnitude', 'HH_L0_1_radius', 'HH_L0_1_covariance', 'HH_L0_1_pcc',
'HH_L0_01_weight', 'HH_L0_01_mean', 'HH_L0_01_std', 'HH_L0_01_magnitude', 'HH_L0_01_radius', 'HH_L0_01_covariance', 'HH_L0_01_pcc',
'HH_jit_L5_weight', 'HH_jit_L5_mean', 'HH_jit_L5_variance', 'HH_jit_L3_weight', 'HH_jit_L3_mean', 'HH_jit_L3_variance',
'HH_jit_L1_weight', 'HH_jit_L1_mean', 'HH_jit_L1_variance', 'HH_jit_L0_1_weight', 'HH_jit_L0_1_mean', 'HH_jit_L0_1_variance',
'HH_jit_L0_01_weight', 'HH_jit_L0_01_mean', 'HH_jit_L0_01_variance', 'HpHp_L5_weight', 'HpHp_L5_mean', 'HpHp_L5_std',
'HpHp_L5_magnitude', 'HpHp_L5_radius', 'HpHp_L5_covariance', 'HpHp_L5_pcc', 'HpHp_L3_weight', 'HpHp_L3_mean', 'HpHp_L3_std',
'HpHp_L3_magnitude', 'HpHp_L3_radius', 'HpHp_L3_covariance', 'HpHp_L3_pcc', 'HpHp_L1_weight', 'HpHp_L1_mean', 'HpHp_L1_std',
'HpHp_L1_magnitude', 'HpHp_L1_radius', 'HpHp_L1_covariance', 'HpHp_L1_pcc', 'HpHp_L0_1_weight', 'HpHp_L0_1_mean', 'HpHp_L0_1_std',
'HpHp_L0_1_magnitude', 'HpHp_L0_1_radius', 'HpHp_L0_1_covariance', 'HpHp_L0_1_pcc', 'HpHp_L0_01_weight', 'HpHp_L0_01_mean',
'HpHp_L0_01_std', 'HpHp_L0_01_magnitude', 'HpHp_L0_01_radius', 'HpHp_L0_01_covariance', 'HpHp_L0_01_pcc', 'class']
```

STANDARDIZATION OF DATA WITH STANDARD SCALER

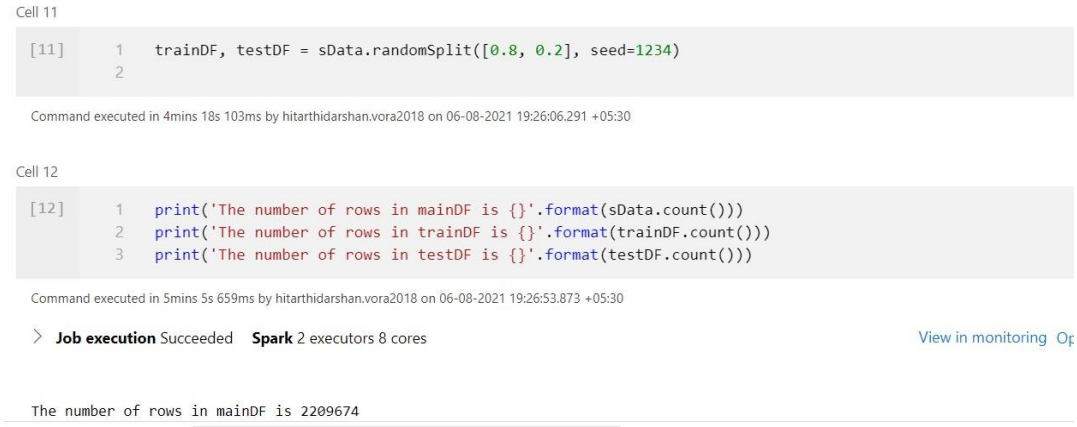
Cell 9

```
[9]
1  from pyspark.ml.feature import StandardScaler
2  from pyspark.ml.feature import VectorAssembler
3
4  features = ['MI_dir_L5_weight', 'MI_dir_L5_mean', 'MI_dir_L5_variance', 'MI_dir_L3_weight', 'MI_dir_L3_mean', 'MI_(
5  assembler = VectorAssembler(inputCols=features,outputCol='unscaled')
6  assembled = assembler.transform(df)
7
8  scaler = StandardScaler(inputCol='unscaled',outputCol='features',withStd=True, withMean=True)
9  scalerModel = scaler.fit(assembled)
10 sData=scalerModel.transform(assembled)
```

Command executed in 4mins 12s 982ms by hitarthidarshan.vora2018 on 06-08-2021 19:26:01.124 +05:30

> **Job execution** Succeeded **Spark** 2 executors 8 cores View in monitoring Open Spark UI

SPLITTING THE DATA INTO TRAIN AND TEST SET WITH 8:2 RATIO WITH RANDOM SPLIT AND SEED = 1234



SEGREGATING THE TRAIN AND TEST INTO FEATURES AND LABELS

RANDOM FOREST CLASSIFIER:

Cell 17

```
1  from pyspark.ml.classification import RandomForestClassifier
2
```

Command executed in 9mins 43s 361ms by hitarthidarshan.vora2018 on 06-08-2021 19:31:31.680 +05:30

Cell 18

```
[18]
1  rf_classifier=RandomForestClassifier(labelCol='class',numTrees=50).fit(trainDF)
2
3
```

Command executed in 11mins 41s 728ms by hitarthidarshan.vora2018 on 06-08-2021 19:33:30.068 +05:30

Job execution Succeeded **Spark** 2 executors 8 cores View in monitoring Open Spark U

Cell 19

```
[19]
1  rf_predictions=rf_classifier.transform(testDF)
2
```

Command executed in 11mins 42s 948ms by hitarthidarshan.vora2018 on 06-08-2021 19:33:31.308 +05:30

Ready

```
[22]
1  from pyspark.ml.evaluation import MulticlassClassificationEvaluator
2  rf_accuracy=MulticlassClassificationEvaluator(labelCol='class',metricName='accuracy').evaluate(rf_predictions)
3
```

Command executed in 14mins 37s 244ms by hitarthidarshan.vora2018 on 06-08-2021 19:36:25.665 +05:30

Job execution Succeeded **Spark** 2 executors 8 cores View in monitoring Open Spark UI

Cell 23

```
[23]
1  print('The accuracy of RF on test data is {0:.0%}'.format(rf_accuracy))
2
3  print(rf_accuracy)
4
```

Command executed in 14mins 38s 455ms by hitarthidarshan.vora2018 on 06-08-2021 19:36:26.886 +05:30

```
The accuracy of RF on test data is 94%
0.9350214613616381
```

Cell 24

```
[24]
1  rf_precision=MulticlassClassificationEvaluator(labelCol='class',metricName='weightedPrecision').evaluate(rf_predi
2  print('The precision rate on test data is {0:.0%}'.format(rf_precision))
3
```

Command executed in 15mins 44s 80ms by hitarthidarshan.vora2018 on 06-08-2021 19:37:32.521 +05:30

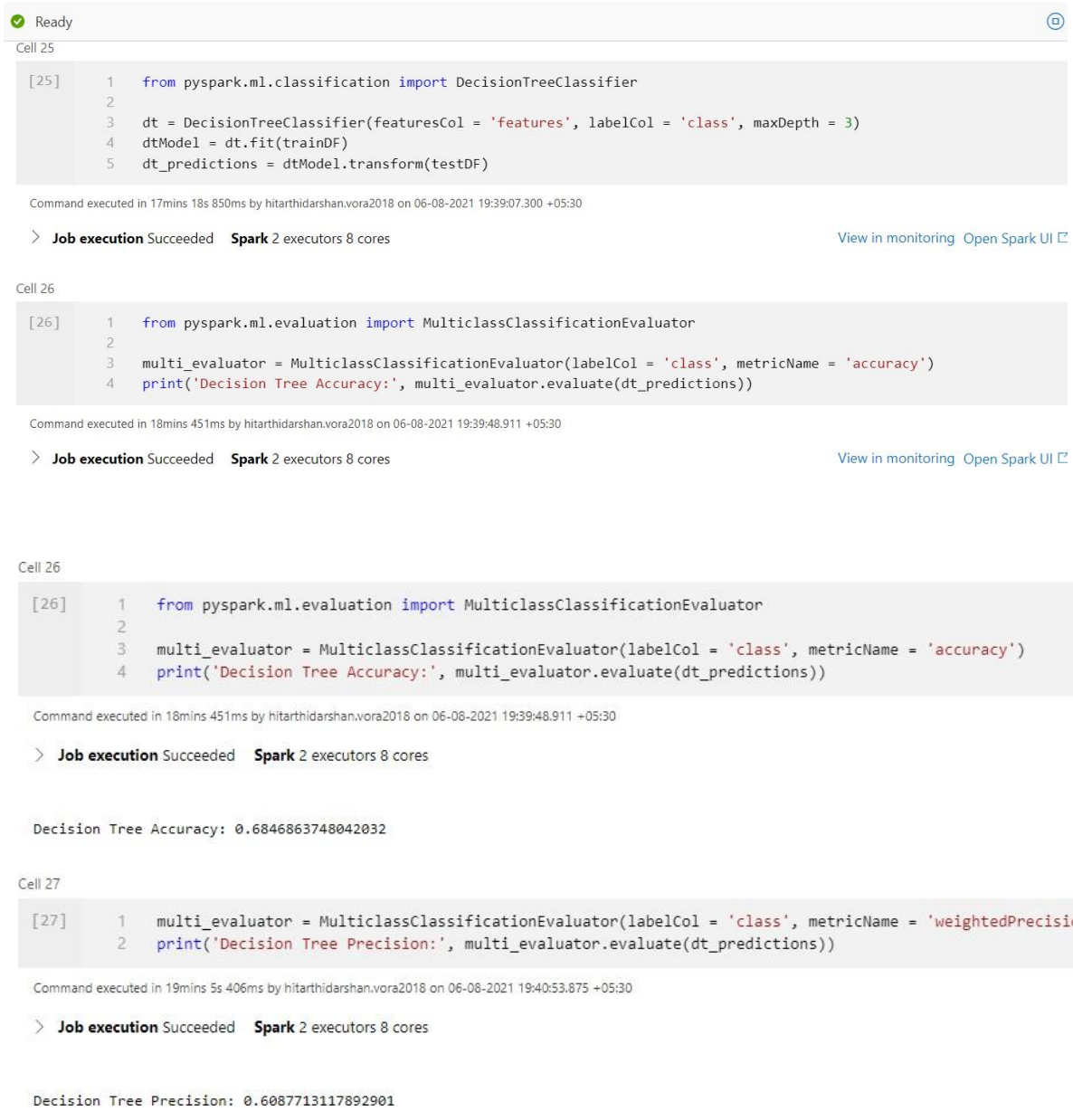
Job execution Succeeded **Spark** 2 executors 8 cores View in monitoring Open Spark UI

```
The precision rate on test data is 95%
```

ACCURACY - 0.935

PRECISION - 0.95

DECISION TREE CLASSIFIER:



ACCURACY - 0.684

PRECISION - 0.608

PERFORMANCE ANALYSIS TOTAL TIME OF EXECUTION

Environment	Time for uploading the data	Time for model building
Google Colab	3 hours	2 hours
Microsoft Azure	15 minutes	19min 6sec

NOTE: The time of uploading the data in google colab corresponds to a much smaller amount of data (maximum 150 MB), whereas in Azure we used a huge amount of data (1.4 GB).

PERFORMANCE METRICS REVIEW 2:

DATASET	ALGORITHM	Accuracy	Precision	Recall	F1- score
1 (with Adam optimisation)	CNN	99.9%	1.00	1.00	1.00
	LSTM	34.1%	0.22	0.34	0.27
	CNN+LSTM	99.8%	1.00	1.00	1.00
	DCNN	99.9%	1.00	1.00	1.00

2 (with Adam optimisation)	CNN	99.9%	1.00	1.00	1.00
	LSTM	67.2%	0.56	0.67	0.59
	CNN+LSTM	99.2%	0.99	0.99	0.99
	DCNN	99.9%	1.00	1.00	1.00
4 (with Adam optimisation)	CNN	99.9%	1.00	1.00	1.00
	LSTM	29.1%	0.03	0.29	0.19
	CNN+LSTM	99.9%	1.00	1.00	1.00
	DCNN	99.9%	1.00	1.00	1.00

REVIEW 3:

Algorithm	Accuracy	Precision
Random Forest	93.5%	0.95
Decision Tree	68.4%	0.608

CONCLUSION

We made a transition to Microsoft Azure to enhance our capabilities in managing Big Data. In the given datasets, we introduced an extra column in each file labeled "class," assigning integral values that represent different types of botnets and attacks. The amalgamation of 33 datasets resulted in a consolidated dataset of substantial size, totaling 1.4 GB. Our initial step involved loading this data into the Synapse workspace analytics within Azure Data Studio.

To streamline data processing, we leveraged Vector Assembler and Standard Scaler techniques. Following this, we partitioned the dataset into distinct training and testing sets, maintaining an 8:2 ratio. The subsequent step involved segregating these sets into their respective features and labels.

Moving on to model application, we implemented both Random Forest and Decision Tree Classifiers on our dataset. Beyond this, we computed accuracy and precision metrics for both classifiers, serving as vital indicators of their performance without relying on review-oriented language.

REFERENCES

[1] Christopher Kelly, Nikolaos Pitropakis, Sean McKeown and Costas Lambrinoudakis “Testing And Hardening IoT Devices Against the Mirai Botnet” (2020), DOI: 10.1109/CyberSecurity49315.2020.9138887

[2] Xiaolu Zhang, Oren Upton, Nicole Lang Beebe, Kim-Kwang Raymond Choo “IoT Botnet Forensics: A Comprehensive Digital Forensic Case Study on Mirai Botnet Servers” (2020), DOI: https://doi.org/10.1016/j.fsidi.2020.300926

[3] Ganesh Karthik and M. B. Mukesh Krishnan “Securing an Internet of Things from Distributed Denial of Service and Mirai Botnet Attacks Using a Novel Hybrid Detection and Mitigation Mechanism” (2020)

[4] Zohaib Ahmed, Syed Muhammad Danish, Hassaan Khaliq Qureshi and Marios Lestas “Protecting IoTs from Mirai Botnet Attacks using Blockchains” (2019), DOI: 10.1109/CAMAD.2019.8858484

[5] Olga Hachinyan, Anastasiya Khorina, Sergey Zapechnikov “A Game-Theoretic Technique for Securing IoT Devices against Mirai Botnet” (2018), DOI: 10.1109/EIConRus.2018.8317382

[6] Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Dominik Breitenbacher, Asaf Shabtai, and Yuval Elovici, “N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders”, IEEE PERVASIVE COMPUTING, VOL. 13, NO. 9, JULY-SEPTEMBER 2018

[7] João Marcelo Ceron , Klaus Steding-Jessen , Cristine Hoepers , Lisandro Zambenedetti Granville , Cíntia Borges Margi, "Improving IoT Botnet Investigation Using an Adaptive Network Layer", Sensors(2019).

[8] Jiyeon Kim, Minsun Shim, Seungah Hong, Yulim Shin and Eunjung Choi, "Intelligent Detection of IoT Botnets Using Machine Learning and Deep Learning", MDPI Applied Sciences(2020).

[9] Yan Naung Soe, Yaokai Feng, Paulus Insap Santosa, Rudy Hartanto and Kouichi Sakurai, "Machine Learning-Based IoT-Botnet Attack Detection with Sequential Architecture", Sensors(2020).

[10] Chaw Su Htwe, Yee Mon Thant and Mie Mie Su Thwin, "Botnets Attack Detection Using Machine Learning Approach for IoT Environment", ICNISC 2020.

[11] Singh K, Guntuku SC, Thakur A, Hota C. Big data analytics framework for peer-to-peer botnet detection using random forests. Information Sciences. 2019 Sep.

[12] Mousavi, S. H., Mohammad Khansari, and R. Rahmani. "A fully scalable big data framework for botnet detection based on network traffic analysis." *Information Sciences* 512 (2020): 629-640.

[13] Mohammad Mehedi Hassan , Abdu Gumaei , Ahmed Alsanad , Majed Alrubaian and Giancarlo Fortino "A Hybrid Deep Learning Model for Efficient Intrusion Detection in Big Data Environment" *Information Sciences November, 2019*

[14] Maheshwari, Vishal, Ashutosh Bhatia, and Kuldeep Kumar. "Faster detection and prediction of DDoS attacks using MapReduce and time series analysis." 2018 International Conference on Information Networking (ICOIN). IEEE, 2018.

[15] Kozik, Rafał. "Distributing extreme learning machines with Apache Spark for NetFlow-based malware activity detection." Pattern Recognition Letters 101 (2018): 14-20.