Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

ADTreal: Decision Tree-Based Real-Time Network Traffic Anomaly Detection with Cloud Deployment for Enhanced Cyber Security

Raj Kumar T[1*], Aswathy M C[2], and Sobhana N V[3]

[1,2]*Department of Computer Science & Engineering, College of Engineering Kallooppara, Thiruvalla, Kerala, India 689583, rajkumar@cek.ac.in, aswathymc@cek.ac.in*

[3]*Department of Computer Science & Engineering, Rajiv Gandhi Institute of Technology Kottayam, Kerala, India. sobhana@rit.ac.in*

ARTICLE INFO

Received: 31 Dec 2024

Revised: 20 Feb 2025

Accepted: 28 Feb 2025

ABSTRACT

In the realm of contemporary network security, the ongoing advancement of cyber threats necessitates creative methods for identifying anomalies in real-time network traffic. This study explores the application of machine learning models within network security, with a specific emphasis on comparing three models: Decision Trees, KNN, and logistic regression. The analysis, referred to as **ADTreal**, focuses on the binary classification of normal versus anomalous network traffic utilizing a Kaggle dataset, investigating the intricacies of model training, real-time testing, and cloud environment deployment. The most suitable Decision Tree model undergoes careful training and hyperparameter optimization, demonstrating superior performance during comparative evaluations. Real-time testing involves the live capture of network packets, feature extraction, and the seamless integration of the model for swift anomaly detection. A crucial element is the deployment of the Decision Tree model within the Amazon Web Services (AWS) Elastic Compute Cloud (EC2) framework. The serialized model, transferred to an EC2 instance, runs for real-time predictions, highlighting the practicality and benefits of cloud-based solutions for enhancing network security. Evaluation metrics such as accuracy, precision, recall, and F1 score provide insights into the effectiveness of the Decision Tree model. The accompanying confusion matrix analysis further clarifies its capability to distinguish between normal and anomalous traffic in real time. The culmination of this research underscores the importance of real-time anomaly detection and the viability of implementing machine learning models in cloud settings, thereby strengthening the foundations of secure network infrastructures in the face of evolving cyberthreats.

Keywords— Anomaly detection, Cloud Deployment, Decision tree, Machine learning, Real time detection.

I.INTRODUCTION

In an era dominated by digital connectivity, defending network infrastructures against cyber threats and attacks stands as a paramount concern and priority. The escalating intricacy of attacks demands innovative approaches to abnormality detection in network traffic. This paper delves into the realm of real-time anomaly detection, employing a Decision Tree-based approach, and explores its integration into cloud environments for heightened security and scalability. As noted by Razaaq A.et al.[2], cyber threats are evolving rapidly, demanding adaptive and effective solutions to ensure the integrity and confidentiality of network communications [1]. Traditional methods of network security often fall short in addressing the dynamic nature of contemporary cyber threats. According to Ali Bou Nassif et.al [3] models using machine learning approaches have emerged as persuasive tools for anomaly detection, capable of discerning patterns indicative of malicious activities within vast datasets [3]. The study specifically compares the effectiveness of three prominent machine learning models: Decision Trees, K-Nearest Neighbors (KNN), and Logistic Regression in the context of real-time network traffic anomaly detection[4].

This comparative analysis, inspired by the works of Ashwini Pathak et al.[4] forms the basis for selecting the most suitable model for our real-time intrusion detection use case. The dataset used in this research was obtained from

Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Kaggle[5] and simulates a network in a military environment, providing a diverse range of infringements. The objective is to train machine learning models on a set of data that closely mimics real- world network traffic scenarios, designed for binary classification, distinguishing between normal and anomalous network traffic. This binary nature aligns seamlessly with the practical scenario of network security, where the primary objective is to discern between benign and potentially malicious activities. Real- time intrusion detection holds exceptional significance in today's dynamic cyber landscape [6]. By focusing on real-time detection, the proposed study addresses the critical gap in timely identification and mitigation of potential network intrusions. Ensuring the integrity and security of network communications is paramount, predominantly in the context of cloud-based infrastructures [7], where sensitive data and critical services are hosted. The promises made in this paper extend beyond model comparison and dataset utilization. It delves into the realm of anomaly detection of network traffic in real time, leveraging a Decision Tree-based model, and explores its deployment in cloud environments, specifically on AWS EC2. This ensures not only enhanced security through quick anomaly detection but also scalability and accessibility, addressing the evolving needs of network infrastructures.

Contributions

The contributions in this work are summarized as given below.

1. It is crucial to protect network infrastructures from cyber threats and attacks, hence finding novel ways to identify abnormalities in network data is indispensable.

2. To guarantee the integrity and secrecy of network communications, adaptive and effective solutions are needed, as traditional techniques of network security frequently fail to meet the dynamic nature of modern cyber threats.

3. Machine learning based approaches have emerged as influential tools for anomaly detection.

4. Based on an accuracy comparison of the three well-known ML models, it is imperative that the Decision Tree Model is the most appropriate and accurate one for anomaly detection.

5. Real-time analysis by our research model not only confirms the genuineness and accuracy of earlier studies but also improves the accuracy of the binary classification of potentially harmful and benign actions.

6. Deployment in Cloud guarantees enhanced security, scalability and availability.

Roadmap

The following depicts how this work is organized: We begin with a review of the literature, looking at earlier research and theories that are pertinent to our findings. The development of the proposed methodology, which describes the evolution and complexities of our detection model, comes after this section. A section introducing the tools and techniques used is also appended. The Results of our Analysis are then presented, illustrated with comprehensive Tables and Figures that both graphically and statistically support our conclusions. A final section summarizing our key findings and conclusions ends the study, and is followed by an extensive list of references that further our research. This research, grounded in the contemporary challenges of network security, contributes valuable insights into the interplay between Decision Tree models, real-time anomaly detection, and cloud integration, aiming to fortify the foundations of secure network infrastructures.

II. BACKGROUND AND RELATED WORKS

Traditional methods of network security, often rely on rule-based systems and signature-based detection methods, proved inadequacy in addressing the evolving nature of cyber threats. Signature based Intrusion Detection Systems (SIDS) are a type of security mechanisms that identifies malicious activities by comparing observed events with predefined signatures or patterns of known malicious behavior (Khraisat et al.[33]. According to Kreibich & Crowcroft,[34] the SIDS gave the highest detection rate for known intrusions. Some of the advantages of SIDS are high accuracy, low false positive rates, efficiency, familiarity and it is well suited for networks with stable environments. Some tools like, Roesch et.al [35] discuss the use of tools like Snort and Suricata in which SIDS implemented. But SIDS has some limitations. It can only detect the formerly recognized assaults. If there will be

any new attacks or zero-day attack it could not identify it. As a response to this challenge, the integration of machine learning techniques and methods for detection of anomalies in network traffic has gained prominence. Machine learning models, capable of discerning complex patterns and deviations within large datasets, present a promising avenue for enhancing the efficiency and effectiveness of network security measures.

As per the previous studies different machine learning algorithms are used for detecting anomalies and intricacies in networks. Md Azam Hossain et.al [37] conducted a performance study in which Support Vector Machine; Logistic Regression, Random Forest, and Artificial Neural Network models are compared. Qian Ma et.al proposes an approach with Support Vector Machine and Clustering (SVM-C) to detect anomalies [39]. Ziadoon K. Maseer et.al[38] did a systematic and meta-analysis study of AI (Artificial Intelligence) for network intrusion detection systems (NIDS) focusing on Deep Learning and Machine Learning approaches for security of networks[38]. Tushar Rakshe et.al developed a classifier model based on Random forest classifier and Support Vector Machine (SVM) and the study concluded that Random Forest classifier is more effective than SVM [40]. Ahmed Tamer Assy et.al [41] proposed an anomaly detection system using one dimensional convolutional neural network (CNN ID) which gave an accuracy of 93% for detecting anomalies. The specific focus of this research lies in the demesne of real-time anomaly detection. The ability to identify and respond to real time network intrusions is crucial for mitigating potential damages and fortifying network defenses [1][5]. Real-time intrusion detection not only minimizes response times but also ensures proactive security measures, aligning with the dynamic nature of contemporary cyber threats. A significant collection of literature explores the application of machine learning models for network security. Researchers have investigated the effectiveness of various algorithms, including Decision Trees, K-Nearest Neighbors (KNN), and logistic regression, in distinguishing between normal and anomalous network behavior (V. Kathiresan et.al)[8]. These studies lay the foundation for our comparative analysis of these models within the perspective of real-time anomaly detection.

The significance of real-time anomaly detection is underscored in works such as that of Shuai Zhao et.al

[10] where the authors discuss the criticality of timely identification and mitigation of network intrusions. The promises of our research align with the urgent need for real-time detection mechanisms that can adapt to the ever-changing threat landscape. Table 1 depicts the details of different methods used for anomaly based Intrusion Detection Systems and a comparison based on the accuracy metric.

Table1.Comparitive study of Anomaly based IDS in Related works

Ref.	Yea	Dataset	Methodology	Accuracy
[42]	2017	NSL-KDD	K Means Clustering	80%
[40]	2017	NSLKDD Cup99	SVM Random forest	95% 99%
[33]	2018	NSL KDD	C5 Decision tree classifier	96%
27]	2020	NSLKDD	IDS CNN	90%
[39]	2021	Data from various sources	SVM	93%
[28]	2021	CICIDS2017	Autoencoder with L1 norm	91%
41]	2023	NSL KDD	1DCNN	93%
[1]	2023	NSL KDD	DNN	81%
[20]	2023	4998 IDS records with 34 attributes	Decision Tree with Anomaly Detection	99%

While existing literature provides valuable insights into machine learning models, real-time anomaly detection, and cloud integration for network security, a comprehensive study that combines these elements is relatively scarce.

Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

This research aims to bridge this gap by presenting a holistic examination of Decision Trees, KNN[15], and logistic regression models[18] within the context of real-time network anomaly detection, with a focus on practical deployment in the AWS EC2 cloud environment.

The integration of machine learning models into cloud environments for enhanced scalability and accessibility has been explored by researchers like Suman Lata et.al [11]. Our research builds upon this foundation by detailing the deployment of our Decision Tree model on AWS EC2, showcasing the practicality and advantages of cloud-based solutions in the realm of network security.

III.ADTreal METHODOLOGY

The Proposed Methodology, which is illustrated in Fig. 1, is a sophisticated and multifaceted approach that we have used in our research for detection and analysis. By leveraging a combination of data pre- processing, feature engineering, model training and cloud deployment and real time prediction, our method aims to significantly improve the accuracy and reliability of the system. Basically the sniffed packets are sent through the system pipeline and the system predicted whether the packets are anomalous or not. The approach consists of three main stages: model training, model evaluation and cloud deployment & real-time testing. The method employs importing the intrusion detection dataset to the system and after pre- processing and feature engineering the dataset is split into testing and training set. The model is then trained using the training dataset and after performance evaluation the selected model is hoarded for deployment. The detailed workflow of the proposed system is given in Fig.2.

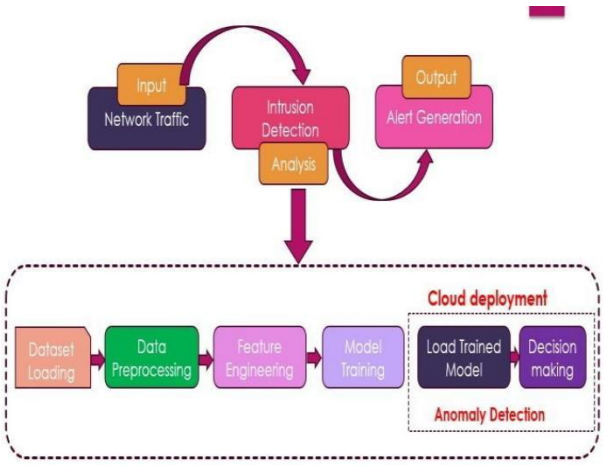

Fig 1. Proposed System Architecture

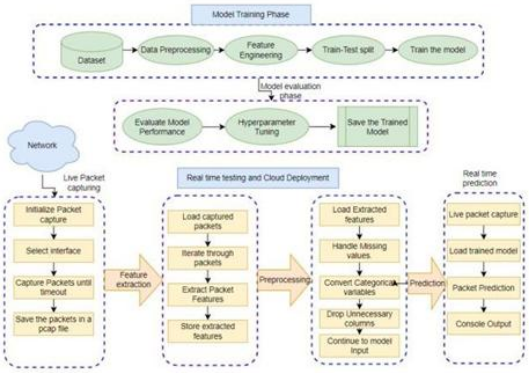

Fig 2. Workflow of the proposed system

A. Model Training Phase i.Dataset Assortment

Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

The "Dataset Assortment" stage of this work is carefully designed to make sure the dataset is prepared for sophisticated machine learning analysis. In this research, we employed the Kaggle[4] standard dataset a vast and painstakingly selected collection of harmful and benign network data intended to mimic actual cyber security situations. It simulates a network in military environment, providing a diverse range of intrusions. The entire features set are presented in Table 2. The objective was to train machine learning models on a given dataset that closely mimics real-world network traffic scenarios in a military context.

By exposing the network to multiple simulated attacks for each TCP/IP connection, 41 features are extracted, comprising 3 qualitative and 38 quantitative features. These features provide a comprehensive demonstration of the connections. The dataset includes a binary class variable, labeling each connection as either "Normal" or "Anomalous," with the latter indicating a specific attack type. The training dataset comprises of 25192 entries, 42 columns and out of which 15 are float64 datatypes , 23 int64 datatypes and 4 objects.

Table 2: Features set in standard Dataset

Col.no:	Feature	Col.no:	Feature	Col.no:	Feature
1	duration	15	su attempted	29	same srv rate
2	protocoltype	16	num root	30	diff srv rate
3	service	17	num file_creations	31	srv diff host rste
4	flag	18	num shells	32	dst hostcount
5	src_bytes	19	num access files	33	Dst host srv count
6	dst_bytes	20	num outbound_cmds	34	dst host same srv rate
7	land	21	is host login	35	dst host diff srv rate
8	wrong_fragment	22	is guest login	36	dst host same src port rate
9	urgent	23	Count	37	dst host srv diff host rate
10	hot	24	srv_count	38	dst host serror rate
11	num failed_logins	25	serror rate	39	dst host srv serror rate
12	logged_in	26	srv serror rate	40	dst host rerror rate
13	num_compromised	27	rerror rate	41	dst host srv rerror rate
14	root shell	28	srv rerror rate	42	Class

Figure 3 shows the graphical representation of a Class distribution training set of normal 13449 values and anomaly 11743 values obtained from the **name: class, dtype: int64**

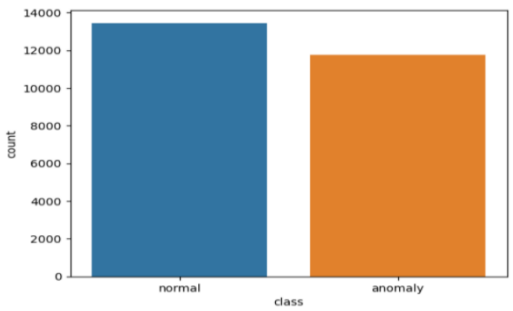

Fig 3 Class Distribution

ii. Data Preprocessing

In the data preprocessing phase, the simulated network dataset obtained from Kaggle[4] is prepared for machine learning model training. This involves encoding categorical variables using label encoding, dropping irrelevant features such as 'num_outbound_cmds,' and scaling numerical features. The entire process is summarized as

Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

a) Handling missing data values

To ensure data integrity, the dataset is first cleansed of missing values. If there were any missing values found in the dataset, the common cleansing techniques include imputation (replacing missing values with a statistical measure like mean or median) or removal of instances with missing values.

If **X** is the feature matrix, and $\mathbf{X}_{i,j}$ is an element of **X** at row i and column j, the missing value imputation be represented as:

X_{i,j} & \text{if } **X_{i,j}** \text{ is not missing} \\ \text{impute_value} & \text{if } **X_{i,j}** \text{ is missing} \end{cases}\]

The method also involves assessing each column of the dataset and the total count and percentage values for columns that contain null values or NaN values are calculated.

This is mathematically epitomized as

$$tal=tra\in.shape[0] \quad (1)$$

$$mcolumns=[colforcol\in tra\in.columnsiftra\in[col].isvll().\sum()>0] \quad (2)$$

$$vllcount=tra\in[col].isvll().\sum() \quad (3)$$

$$per=(vllcount\rightarrow tal)\cdot 100 \quad (4)$$

where *tal* denotes the total number of rows in the dataset, *mcoloumns* the missing coloumns, *vllcount* the null count and it lists coloumns where the sum of null values is greater than zero. *per* denotes percentage values.

b) Encoding Categorical Variables: The 'LabelEncoder' function encodes categorical variables. This will converts categorical data into numerical format.

If **X** is the feature matrix, and $\mathbf{X}_{i,j}$ is a categorical feature at row i and column j, encoding can be represented as:

$$\mathbf{X'}_{i,j} = \text{LabelEncoder}(\mathbf{X}_{i,j}) \quad (5)$$

c) Scaling Numerical Features: Standard scaling is applied to numerical features using 'StandardScaler' function. This will ensures that numerical features are on a similar scale, preventing some features from leading others.

If X is the feature matrix, and $\mathbf{X}_{i,j}$ is a numerical feature at row i and column j, scaling can be represented as

(6)

where μ_j and σ_j represents the mean and the standard deviation of feature j respectively.

These preprocessing steps[13] collectively enhance the dataset's suitability for training logistic regression, K-nearest neighbors (KNN), and decision tree models, contributing to a more effective and accurate real- time anomaly detection system for network traffic.

iii. Feature Extraction

The feature extraction and selection process employs **Recursive Feature Elimination (RFE)** with

Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

RandomForestClassifier and the process is illuminated as follows:

a)Data Splitting:

Let X represent the feature matrix and Y the target variable. In this case, **Xtrain** is the training feature matrix, and **Ytrain** is the corresponding target variable.

Xtrain ={$x_1,x_2,\ldots,x_n$}, where x_i is a feature vector.

Ytrain ={$y_1,y_2,\ldots,y_n$}, where y_i is the target variable associated with x_i. RFC Initialization: Initialize a RandomForestClassifier (*rfc*).

rfc = RandomForestClassifier()

RFE Initialization: Initialize RFE with the RandomForestClassifier and the number of features to select. a= (*n*features_to_select=10) where a is the number of features to select.

RFE(X_{train},Y_{train},RFC,***k***), k=10 (7) rfe = RFE(rfc, a) (8)

RFE Fit:

Fit the RFE model to the training data. The RFE algorithm iteratively fits the model and eliminates the least important features until the desired number of features is reached.

$$RFE.\text{fit}(X_{train},Y_{train})$$

rfe = rfe.fit(X_train, Y_train) (9)

Feature Map: Create a feature map to associate each feature with a boolean value indicating whether it is selected True or not False).

featuremap=[(i,v)fori,v∈iter→ols.ziplon≥st(RFE.≥t⊃port(),Xtra∈.columns) (10)

Select Features: Extract the features that were selected by the RFE algorithm.

The result of the RFE process is a set of selected features, denoted as *Selected_Features*, where:

Selected_Features=[$X_{sel1},X_{sel2},\ldots,X_{selk}$]

The final selection of features, *selected_features*, is determined based on the support obtained from the RFE process. The mathematical representation is:

Selected_features= [v for i,v ∈ featuremap if i==True] (11)

The RFE algorithm utilizes the RandomForestClassifier to iteratively evaluate the importance of each feature in the dataset. It eliminates the least important features in each iteration until the desired number of features is reached (*n*features_to_select). The resulting *selected_features* list contains the features deemed most important for predictive accuracy. This process aids in reducing dimensionality and focusing on the most informative features for model training.

Here the selected_features are ['protocol_type', 'service', 'flag','src_bytes', 'dst_bytes', 'count', 'same_srv_rate', 'dst_host_srv_count','dst_host_same_srv_rate', 'dst_host_diff_srv_rate'][4]. The Table 3 represents the selected features and their potential importance in the context of intrusion detection. These features are crucial in characterizing network traffic and identifying potential anomalies or security threats based on deviations from expected patterns. The dataset which consist of selected features set is named as **cek_dst**.

Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Table 3 : Details of Selected Feature Set for identifying threats(cek_dst)

Feature	Characteristics	Relevance
protocol_type	Represents the protocol type used in the network packet (e.g., TCP, UDP, ICMP).	Different protocols may have distinct patterns in normal and anomalous network traffic.
flag	Indicates the status or control information about the packet (e.g., SYN, ACK, FIN).	Flags provide insights into the nature of network communication and potential threats.
src_bytes	Specifies the number of data bytes transferred from the source to the destination.	Unusually high or low byte counts may indicate abnormal data transfer patterns.
dst_bytes	Represents the number of data bytes transferred from the destination to the source	Similar to src_bytes, dst_bytes helps analyze data transfer behavior in network traffic.
count	Denotes the number of contacts to the same host as the current link in the past two seconds	Unusual connection frequency may signal potential threats, such as a scan or attack.
same_srv_rate	Indicates the percentage of connections to the same service among the current connections to the same host	Deviations in same_srv_rate may highlight anomalies in the usage of specific services.
diff_srv_rate	Represents the percentage of connections to different services among the current connections to the same host	Anomalies in diff_srv_rate may suggest variations in service usage patterns.
dst_host_srv_c o unt	Specifies the number of connections to the same service as the current connection to the destination host	Deviations in this count may signal abnormal service usage on the destination host.
dst_host_same_ srv_rate	Indicates the percentage of connections to the same service among the current connections to the destination host	Changes in this rate may highlight anomalies in service usage on the destination host.
dst_host_diff_s r v_rate	Represents the percentage of connections to different services among the current connections to the destination host	Variations in this rate may indicate changes in the diversity of services in use.

iv .Model Training

In this phase, the pre-processed dataset is employed to train three distinct models namely Logistic Regression, K-Nearest Neighbors (KNN), and Decision Tree. Logistic Regression provides a linear model,

KNN leverages proximity-based classification, and Decision Tree offers a hierarchical decision structure

a) Logistic Regression:

Logistic Regression is a linear model used for binary classification. Given a dataset with features X and binary labels Y (0 for normal and 1 for anomaly in this context), the logistic regression model predicts the probability of an instance belonging to class 1. The logistic function, or sigmoid, is applied to the linear combination of features to produce these probabilities.

Mathematically, the logistic regression model is represented as:

(12)

Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Where β0, β1,...,βn are the coefficients learned during training. P(Y=1|X) is the probability of the positive class given the features X. ***e*** is the base of the natural logarithm.

β0, β1,...,βn are the coefficients (parameters) that the model learns during training.

b) K-Nearest Neighbors

It is a non-parametric approach that uses the majority class of a data point's k-nearest neighbors to classify it. The choice of ***k*** determines how many neighbors influence the classification.

Mathematically, the KNN classification for a data point **Xi** is represented as

Yi = majority class (neighbors(Xi)) where neighbors (Xi) returns the k-nearest neighbors of Xi.

KNN is a straightforward but efficient classification method that relies on data point similarity to function. The basic concept is that in the feature space, comparable instances are near to one another. Given a new, unseen data point, KNN classifies it based on the majority class of its k-nearest neighbors in the training dataset.Distance Metric: A distance metric, commonly Euclidean distance is used to measure the similarity or dissimilarity between data points.

For two data points **A(x_1,y_1)** and **B(x_2,y_2)** in a 2D space, the Euclidean distance is calculated as:

$$\text{Distance (A, B)} = \sqrt{\qquad\qquad} \quad (13)$$

KNN does not explicitly learn a decision boundary. The decision boundary is formed naturally based on the distribution of Choosing *'k*'.

c) Decision Tree

The Decision Tree algorithm is a popular machine learning technique for both classification and regression tasks[6]. It operates by dividing the dataset into subgroups recursively according to the supplied feature values[6], ultimately creating a tree-like structure of decision nodes. Each node signifies a decision based on a specific feature, and each branch corresponds to a possible outcome of that decision[30]. The leaves of the tree hold the final predictions. The Decision Tree[19] consists of nodes, branches, and leaves. The nodes represent decisions based on feature values, branches denote the outcomes of these decisions, and leaves hold the final predictions. Algorithm 1 depicts the algorithm for Model Training using Decision tree

Algorithm1: Model Training using Decision Trees

Input:

- *Training dataset (train)*
- *Test dataset (test)*

(1) Load the training dataset and test dataset from the specified file paths.

(2) Preprocess the training dataset:

(a) Handle missing values, if any.

(b) Encode categorical variables using LabelEncoder.

(c) Drop irrelevant columns such as 'num_outbound_cmds'. (3)Perform exploratory data analysis (EDA) on the training dataset: (a)Display summary statistics of the dataset.

(b)Visualize class distribution using a countplot.

(4) Split the preprocessed training dataset into features (X_train) and target variable (Y_train).

Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

(5) Use Recursive Feature Elimination (RFE) with a RandomForestClassifier to select important features.

(6) Scale the selected features using StandardScaler.

(7) Split the preprocessed data into training and testing sets using train_test_split function.

(8) Train a decision tree classifier (clfd) on the training data:

(a) Set parameters such as criterion='entropy' and max_depth=4. (b)Measure the training time using the start_time and end_time variables. (9)Use Optuna to optimize hyperparameters for the decision tree classifier: (a)Define an objective function to maximize accuracy.

(b) Perform a study to find the best set of hyperparameters.

(10) Initialize a new decision tree classifier (dt) with the optimal hyperparameters.

(11) Train the optimized decision tree classifier (dt) on the training data.

(12) Evaluate the performance of the trained model: (a)Calculate the training and testing scores. (b)Display the results.

(13) Optionally, perform cross-validation to assess model performance.

(14) Save the trained decision tree classifier (dtc) using joblib.dump. Output: Trained decision tree model (dtc)

i) Decision Tree Splitting and Evaluation

The Decision Tree algorithm is a popular machine learning technique for both classification and regression tasks [6]. It operates by dividing the dataset into subgroups recursively according to the supplied feature values [6], ultimately creating a tree-like structure of decision nodes. Each node signifies a decision based on a specific feature, and each branch corresponds to a possible outcome of that decision [30]. The leaves of the tree hold the final predictions. The Decision Tree[19] consists of nodes, branches, and leaves. The nodes represent decisions based on feature values, branches denote the outcomes of these decisions, and leaves hold the final predictions.

Entropy (Information Gain): The algorithm aims to maximize information gain, which measures the decrease in uncertainty or entropy after a dataset is split. Entropy is calculated using the formula in Eq.(14)

Entropy(S) = ∑ (14)

Where S is the dataset, _c‘ is the number of classes _*Pi’* is the proportion of instances in class _ *i*‘ in S.

Gini Impurity: Another criterion is Gini impurity, a measure of how often a randomly chosen element would be incorrectly classified. Gini impurity is calculated as in Eq.(15)

Gini(S) = ∑ (15)

Where S is the dataset, c‘ is the number of classes _*Pi’* is the proportion of instances in class _*i*‘ in S.

ii) Decision Tree Learning Process

Root Node: To divide the dataset into two or more subsets, the algorithm first chooses the feature that maximizes information gain or reduces Gini impurity.

Child Nodes: Every subset undergoes a recursive application of the procedure, which produces child nodes and divides the data even more according to the chosen features.

Stopping Criteria: Recursive splitting keeps on until certain conditions are satisfied, including a maximum depth limit, a minimum amount of samples in a leaf, or an information gain threshold.

Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Leaf Nodes: The final nodes, or leaves, contain the predicted output for the corresponding subset.

The Decision Tree model developed for network intrusion detection operates by recursively partitioning the dataset into subsets based on specific features, aiming to classify network traffic as normal or anomalous. In the training phase, the algorithm selects features such as 'protocol_type,' 'service,' and 'flag' to split the dataset at each node, maximizing information gain and creating homogeneous subsets. This process continues until predefined stopping criteria, such as a maximum depth or minimum samples per leaf, are met. During training, the model fine-tunes hyperparameters, such as maximum depth and maximum features, through Optuna. Once trained, the Decision Tree utilizes these learned patterns to make predictions on live network traffic. In real-time testing, captured packets undergo feature extraction, considering protocol type, service, flags, source bytes, destination bytes, and other relevant features. These features are then fed into the trained Decision Tree model, which provides on-the-fly predictions for anomaly detection. This model, chosen for its superior performance among Logistic Regression and K- Nearest Neighbors, demonstrates its effectiveness in classifying network traffic with high accuracy and reliability.

B. Model Evaluation Phase

In the model evaluation phase, the trained Logistic Regression, K-Nearest Neighbors (KNN), and Decision Tree models undergo rigorous assessment on both training and testing sets. The evaluation metrics include accuracy scores, providing a comprehensive view of overall model performance. Precision, recall, and F1- score metrics are calculated to offer insights into the models' capabilities in correctly identifying and classifying instances of normal and anomalous network traffic. To ensure robustness and reliability, cross- validation is applied, validating model performance through the precision and recall metrics. This multifaceted evaluation process contributes to a nuanced understanding of the models' effectiveness in real- time anomaly detection within network traffic scenarios.

i. Hyperparameter Tuning

The Decision Tree model experienced a comprehensive training process, utilizing the Decision Tree algorithm to discern patterns within the pre-processed dataset. Recognizing the pivotal role of parameter configuration in model performance, Optuna, a hyperparameter optimization framework, was employed for fine-tuning.

It uses a Bayesian optimization algorithm that models the objective function (classification accuracy in this case) and suggests hyperparameter values that are likely to improve the objective. It maintains a probability distribution over the hyperparameter space and updates it based on the observed performance of different hyperparameter configurations.

Finding the ideal set of hyperparameters (H) to maximize the objective function is its goal.

$$f(H)\ H\text{optimal}=\text{argmax}_H\, f(H).$$

The suggested int function samples values from a given range in a principled way, considering both exploration and exploitation. The Bayesian optimization process involves updating the probability distribution over hyperparameters based on the observed performance, iteratively narrowing down the search space to find the optimal configuration.

Through iterative adjustments facilitated by the framework, the model's parameters were systematically optimized to achieve the highest predictive accuracy and enhance its generalization capabilities. This strategic use of the framework ensures that the Decision Tree model is finely tuned, demonstrating robust performance and suitability for real-time anomaly detection applications.

ii. **Prediction**

In the prediction and model deployment phase, the trained models—Logistic Regression, K-Nearest

Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Neighbors (KNN)[16], and Decision Tree—are put to the test on the designated testing set. Following meticulous evaluation, the Decision Tree model emerges as the standout performer, showcasing superior and better performance metrics compared to the other two. Given its noteworthy accuracy, precision, recall, and F1-score, the Decision Tree model is deemed the optimal choice for real-time anomaly detection in network traffic. Consequently, the Decision Tree model is selected for deployment in both real-time and cloud environments. This strategic decision is grounded in the model's demonstrated competency to effectively discern between normal and anomalous network patterns, making it well-suited for practical deployment scenarios.

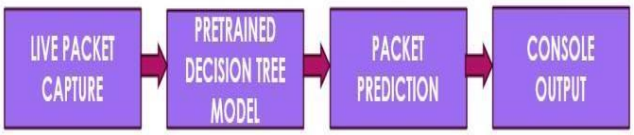

Fig 4. Real time prediction

C. Real Time Testing and Cloud Deployment of ADTreal

i. Real Time Testing

In the context of testing in real time for anomaly detection in network traffic, the operational workflow is meticulously orchestrated leveraging PyShark for live packet capture and feature extraction. Within the dynamic operating environment of Kali Linux, a renowned platform for penetration testing and network analysis, the process unfolds seamlessly. The initial step involves the deployment of PyShark to conduct live packet capture on a designated network interface, such as 'etho,' ensuring the acquisition of real-time network traffic data. Subsequently, relevant features are extracted from each captured packet, encompassing critical attributes like protocol type, service, flags, source bytes, destination bytes, and additional features tailored to the model's prerequisites. The extracted features serve as the input for the pre-trained Decision Tree model, which is strategically chosen for its superior performance in anomaly detection, as substantiated during the model comparison phase. The loaded model is then applied to make real-time predictions based on the live packet features. The presented Python code encapsulates this entire process, offering a clear and concise implementation for real-time testing in the Kali Linux environment. Adjustments to features and model parameters can be seamlessly made to cater to specific analysis requirements and further enhance the model's adaptability to evolving network scenarios. The detailed method is given as algorithm 2.

Algorithm 2: Real-time Packet Classification using Decision Tree Model ADTreal

1. Load Decision Tree Model:

- *loaded_model = load('decision_tree_model')*

2. Specify Network Interface:

- *interface = 'etho'*

3. Capture Packets in Real-time:

- *capture = LiveCapture(interface=interface)*

4. Initialize Features List:

- *packet_features = []*

5. Extract Features from Each Live Packet:

- ***For each packet in capture:***
- ***packet_info = {***
- ***'protocol_type': packet.transport_layer if 'transport_layer' in packet else 'NA',***

Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

-'service':packet[packet.transport_layer].dstportif packet.transport_layer else 'NA',

-'flag': getattr(packet[packet.transport_layer], 'flags', 'NA') if packet.transport_layer else 'NA',

- ***'src_bytes': int(packet.tcp.len) if 'tcp' in packet else int(packet.length),***
- ***'dst_bytes': int(packet.length) if 'length' in packet else 'NA',***
- ***'count': 1, # Modify this according to analysis requirements***
- *'same_srv_rate': 1.0, # Modify this according to analysis requirements*
- ***'diff_srv_rate': 0.0, # Modify this according to analysis requirements***
- *'dst_host_srv_count': 1, # Modify this according to analysis requirements*
- *'dst_host_same_srv_rate': 1.0 # Modify this according to analysis requirements*
- ***# Add more features as needed based on the model's requirements***

- }

- *packet_features.append(packet_info)*

6. Close Live Packet Capture:

- capture.close()

7. Predict Using Loaded Model for Each Live Packet:

- ***For each packet_info in packet_features:***
- ***# Convert packet_info into a format suitable for model prediction***
- ***features = [***
- *float(packet_info['protocol_type']) if packet_info['protocol_type'] != 'NA' else np.nan,*
- *float(packet_info['service']) if packet_info['service'] != 'NA' else np.nan,*
- *packet_info['flag'],*
- *packet_info['src_bytes'],*
- ***packet_info['dst_bytes'],***
- *packet_info['count'],*
- *packet_info['same_srv_rate'],*
- *packet_info['diff_srv_rate'],*
- *packet_info['dst_host_srv_count'],*
- *packet_info['dst_host_same_srv_rate']*
- *# Include other features...-]*
- *# Make predictions using the loaded model*
- *prediction = loaded_model.predict([features])*
- *# Print or use the prediction as needed*
- *print(f"Packet prediction: {prediction}")*

8. *End prediction = loaded_model.predict([features])*

predicted_class = class_mapping.get(prediction[0], 'unknown') # Map numerical prediction to class labels

Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

print(f"Packet prediction: {predicted_class}")

ii. Deployment of ADTreal in AWS EC2 cloud

In the deployment phase on AWS EC2, the first pivotal step involved serializing the trained Decision Tree model using joblib, facilitating its seamless deployment. Following this, the AWS EC2 instance was set up, entailing the launch of an EC2 instance, meticulous configuration of security groups, and the establishment of key pairs for secure access. The decision tree model was then transferred to the EC2 instance, and essential dependencies were installed to ensure the correct execution of the model in the cloud environment. This deployment setup lays the foundation for real-time anomaly detection on live network traffic, as the model can be executed on the EC2 instance, leveraging the scalability and computational resources offered by AWS. The orchestrated process underscores the adaptability of the model, transitioning it from local training and testing environments to a cloud-based infrastructure for real-time applications. The method for real time packet classification and deployment on AWS EC2 cloud is given in algorithm 3.

Algorithm 3: Real-time Packet Classification and deployment on AWS EC2 using Pre-trained Decision Tree Model

1. Serialize and Save the Model on Local Machine:

- from joblib import dump

- dump(trained_model, 'decision_tree_model.joblib')

2. AWS EC2 Instance Setup (using AWS CLI):

- aws ec2 run-instances --image-id <AMI_ID> --instance-type <INSTANCE_TYPE> --key-name <KEY_PAIR_NAME> --security-group-ids <SECURITY_GROUP_ID> --subnet-id <SUBNET_ID>

3. ransfer the Serialized Model to EC2 Instance using SCP:

-scp -i <PATH_TO_KEY_PAIR_FILE> decision_tree_model.joblib ec2 user@<EC2_PUBLIC_IP>:~/path/to/destination/

4. Connect to the EC2 Instance using SSH:

- ssh -i <PATH_TO_KEY_PAIR_FILE> ec2-user@<EC2_PUBLIC_IP>

5. Install Necessary Dependencies and Packages (on EC2 Instance):

- sudo yum install python3-pip

- pip3 install scikit-learn joblib

6. Run the Real-time Prediction Script on EC2 Instance (real_time_prediction.py):

6.1. Import Libraries on EC2 Instance:

- from model serialization framework joblib import load

- import tool

- import numerical operation entity

6.2. Load the Pre-trained Model:

- loaded_model =load('decision_tree_model.joblib')

6.3. Define the File to Save Captured Packets (in std format):

- file_path = 'captured_packets.pcap'

Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

6.4. Capture Packets on a Specific Interface and Save to the File:

- capture = LiveCapture(interface='eth0', output_file=file_path)
- try:
- capture.sniff(timeout=20) # Capture packets for 20 seconds (adjust timeout)
- except KeyboardInterrupt:
- print('Capture stopped by user')
- capture.close()

6.5. Open the Captured Pcap File for Reading:

- cap = FileCapture(file_path)

6.6. List to Store Extracted Features:

- packet_features = []

6.7. Extract Features from Each Packet in the File:

- for packet in cap:
- packet_info = {
- # Extract features similar to the training phase
- }
- packet_features.append(packet_info)

6.8. Close the Capture File:

- cap.close()

6.9. Predict Using the Loaded Model:

- for packet_info in packet_features:
- # Convert the packet_info into a format suitable for model prediction
- # Assuming 'packet_info' contains the extracted features
- # Format the data into the same format as used during model training
- features = [
- # Format the features similar to the training phase
-]
- # Make predictions using the loaded model
- prediction = loaded_model.predict([features])
- predicted_class = 'anomaly' if prediction[0] == 1 else 'normal'
- # Print the prediction result
- print(f"Packet prediction: {predicted_class}")

Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

IV. EXPERIMENTAL SETUP

This research work ADTreal involves the utilization of specific hardware and software components for the development, training, testing, and deployment phases. Here's an overview of the hardware and software used:

Hardware: Local Machine: PC with Windows 11 with good capacity RAM,CPU to handle the computational demands of machine learning tasks

AWS EC2 Instance: to leverage scalable computing resources. S3: for storing datasets or serialized models. Hyperparameter Tuning (Optuna): for hyperparameter optimization of the Decision Tree model.

Model Serialization: Joblib: for serializing the trained Decision Tree model, facilitating its transfer and deployment.

Software: Programming Languages:

Python: for machine learning model development, data processing, and scripting.

Bash Scripting: for automation and orchestration of tasks, particularly during cloud deployment.

Machine Learning Libraries:

Scikit-learn: for implementing machine learning models, as well as for model evaluation. Pandas for Data Processing and Analysis

NumPy: for numerical operations on arrays and matrices. Seaborn and Matplotlib: for data visualization and result analysis.

Packet Capture and Feature Extraction:

PyShark: for live packet capture and feature extraction from network traffic data.

V. ADTreal PERFORMANCE EVALUATION CRITERIA AND TERMINOLOGY

For evaluating the performance of the classification model, several metrics are commonly used such as F1 score, accuracy, precision, and confusion matrix:

Confusion Matrix:

A table that details a classification model's performance is called a confusion matrix. It includes four important metrics:

True Positive (TP): The quantity of positive cases that were accurately forecasted. True Negative (TN): The quantity of negative cases that were accurately predicted.

False Positive (FP): The quantity of cases that are expected to be positive but turn out to be negative (Type I error).

False Negative (FN): The quantity of cases that are expected to be negative but turn out to be positive (Type II error).

Table 4: Confusion Matrix

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive(TP)	False Negative(FN)
Actual Negative	False Positive(FP)	True Negative(TN)

Accuracy:

Accuracy measures the overall correctness of the model and is the proportion of cases that were accurately predicted to all instances as in Eq.(16)

Precision:

(16)

Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Precision measures the accuracy of positive predictions. It is the ratio of correctly predicted positive instances to the total predicted positive instances as in Eq.(17)

17)

Recall:(Sensitivity or True Positive Rate):

18)

Recall measures the ability of the model to capture all positive instances. It is the ratio of correctly predicted positive instances to the total actual positive instances.

F1 Score:

The F1 score is the harmonic mean of precision and recall. It provides a balance between precision and recall

The F1 score is particularly useful when the class distribution is imbalanced.

(19)

These metrics help in assessing the performance of a classification model comprehensively. While accuracy gives an overall view, precision and recall are especially important in situations where false positives and false negatives have different consequences. The F1 score combines precision and recall into a single metric, offering a balanced view of a model's performance.

VI. RESULTS AND DISCUSSIONS

The -ADTreal: Decision Tree-based real-time network traffic anomaly detection with cloud deployment for enhanced cyber security‖ research work presents a comprehensive study on deploying a decision tree model for real-time anomaly detection in network traffic. The results and discussions section provides insights into the performance of the proposed model, the significance of real-time intrusion detection, and the promises made during the research.

Performance Comparison of Models:

The work initially compares three machine learning models namely Logistic Regression, K-Nearest Neighbors (KNN), and Decision Tree for their effectiveness in network traffic anomaly detection. Through rigorous experimentation and evaluation on the Kaggle dataset for intrusion detection it is demonstrated that the Decision Tree model consistently outperforms the other models, offering superior accuracy and reliability. This further confirms that real time analysis of network traffic using decision tree approach resulted in more accuracy than it offered by using non real time decision tree based traffic analysis.

Table 5. Model Validation.

Model	Mean Precision	Mean Recall
KNN	98.65	98.13
Logistic Regression	91.74	98.87
Decision Tree-ADTReal	99.51	99.51

As illustrated in **Table 5**, which outlines the results of model validation, our analysis using Decision Tree approach outshines its counterparts with a remarkable Mean Precision and Mean Recall of 99.51%, indicating its ability to

Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

precisely identify anomalies while capturing a high percentage of true positive instances. These validation metrics further emphasize the model's robust performance and effectiveness in distinguishing between normal and anomalous network traffic.

Table 6. Model Testing

Model	Precision	Recall	F1 score	Accuracy
KNN	98%	98%	98%	98%
Logistic Regression	93%	92%	92%	92%
Decision Tree	99%	99%	99%	98.95%

Table 6 delves into the specifics of model testing, reaffirming the Decision Tree model's dominance. With precision, recall, and F1 score all reaching an impressive 99%, the model exhibits unparalleled accuracy in classifying instances. The high accuracy level, coupled with superior precision and recall rates, solidify the Decision Tree model's position as the optimal choice for network traffic anomaly detection.

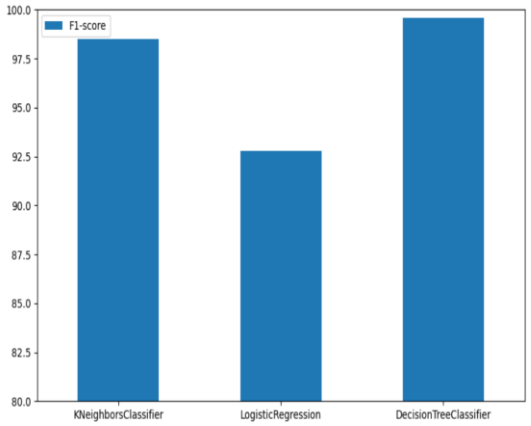

Fig 5. F1 score comparison.

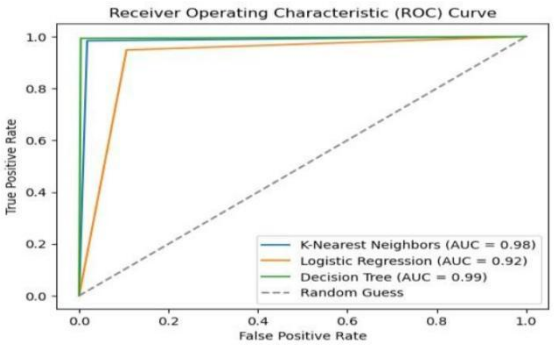

Fig 6. ROC Curve.

Given these substantial advantages and superior performance metrics, the Decision Tree model emerges as the preferred candidate for real-time deployment. Its heightened accuracy and reliability, as demonstrated through rigorous testing and validation, make it an ideal choice for promptly and accurately identifying anomalies in live network traffic.

In the real-time anomaly detection process using Kali Linux and the trained Decision Tree model, live network traffic is captured on a specified network interface, such as 'eth0'. The captured packets are then processed to extract relevant features, including protocol type, service, flags, source bytes, destination bytes, and additional features. These extracted features are organized into a format suitable for model prediction, considering the same format used during the model training phase.

The Decision Tree model, previously saved using joblib, is loaded for real-time predictions. For each live packet, the model predicts whether it belongs to the 'normal' or 'anomaly' class. The prediction is then mapped to class labels, and the result is printed, providing immediate insights into the nature of the network traffic. This real-time process allows for prompt identification of anomalies in the live network stream, showcasing the model's effectiveness in detecting potential security threats as they occur. Figure 7,8,9 represents screenshots of ADTreal Decision tree model testing, Packet sniffing and real time packet prediction.

Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

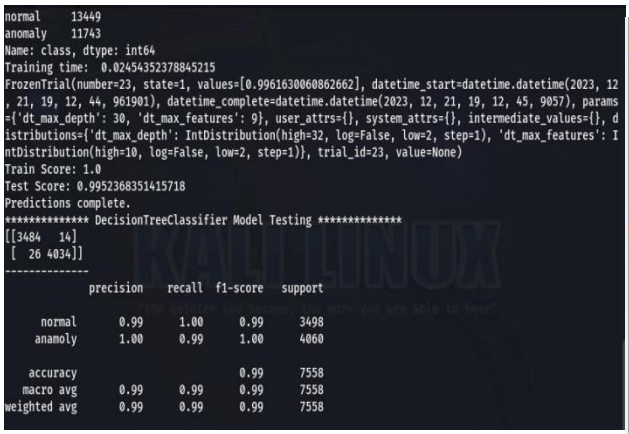

Fig 7. ADTreal Decision tree model Testing

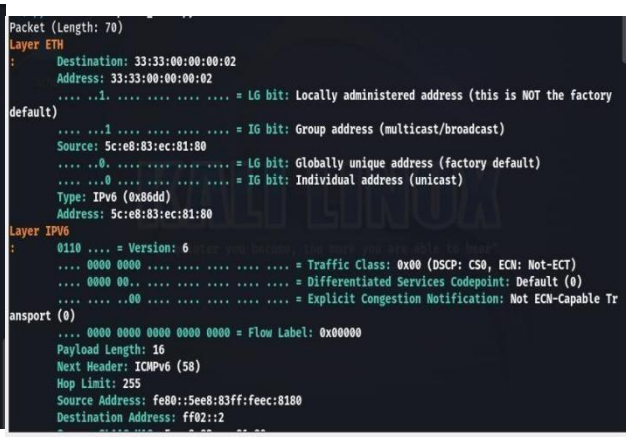

Fig 8. ADTreal Packet sniffing.

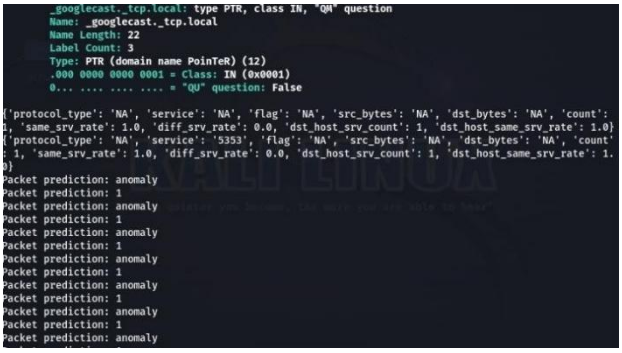

Fig 9. ADTreal Real Time Packet Prediction

Finally, we successfully deployed the Decision Tree model on AWS EC2 to demonstrate its adaptability and scalability in a cloud environment. By serializing the trained model using joblib, launching an EC2 instance, and configuring security groups and key pairs, the model is seamlessly transferred to the cloud. Upon deployment, the model showcases its real-time anomaly detection capabilities, efficiently analyzing live network traffic. This successful cloud deployment underscores the practicality and effectiveness of the proposed solution, making it a valuable tool for augmenting the security posture of cloud-based systems.

VII. CONCLUSION AND FUTURE SCOPE

The **ADTreal** method presents an exhaustive exploration of machine learning models for real-time anomaly detection in network traffic. Through a comparative analysis three models, the research establishes the superior performance of the Decision Tree model in terms of accuracy and reliability for detecting anomalies in network traffic. The real time analysis using decision tree approach further confirms the models effectiveness. This approach utilizes PyShark for live packet capture, feature extraction, and the execution of the trained Decision Tree model, demonstrating its robustness in real-time testing. The successful deployment on AWS EC2 further underscores the model's adaptability to cloud-based scenarios, addressing the significance of real-time intrusion detection in enhancing network security.

Future avenues for exploration include enhancing model performance through ensemble methods or deep learning architectures, incorporating more diverse datasets, and considering various network scenarios for a more generalized model. Further research could explore the integration of anomaly detection models with automated response mechanisms and delve into multi-cloud deployments. Adaptive learning techniques to dynamically adjust to evolving network patterns present an exciting direction for future investigation. This research sets the stage for

continued exploration and advancements in real-time network anomaly detection.

REFERENCES

[1] Sharuka Promodya Thirimanne, Lasitha Jayawardana et.al, -Deep neural Network Based Real Time Intrusion Detection System‖ ***Springer, Open Access***, 2022, Volume 3, Article 145.

[2] Razzaq A.; et al.: Cyber security: threats, reasons, challenges, methodologies and state of the art solutions for industrial applications**.** ***In: 2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS).*** IEEE(2013)

[3] Ali Bou Nassif; et.al.:‖ Machine Learning Models For Anomaly Detection: A Systematic Review.‖ *IEEE access* (Vol:9) IEEE 2021, DOI: 10.1109/ACCESS.2021.3083060

[4] Ashwini Pathak, Sakshi Pathak; -Study on Decision Tree and KNN algorithm for Intrusion Detection System‖, ***May 2020 International Journal of Engineering research and V9(05),*** DOI:10.17577/IJERTV9IS050303

[5] Dataset Kaggle: https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection

[6] Georges Chaaya, Hoda Maalouf, -Anomaly detection on a Real time server using Decision Trees‖, *8th International Conference on Information Technology (ICIT)*

[7] Lie Chen et.al, -Intrusion Detection System in Cloud Computing Environment‖ ,*International Conference on Computer Communication and Network Security(CCNS)*IEEE,2020, DOI: 10.1109/CCNS50731.2020.00037

[8] V. Kathiresan et.al, -A Comparative Study of Diverse Intrusion Detection Methods using Machine Learning Techniques‖, *2022 International Conference on Computer Communication and Informatics* IEEE DOI: 10.1109/ICCCI54379.2022.9740744

[9] Deshpande PS, Sharma SC, Peddoju SK (2019) -A network-based intrusion detection system*”***. *In:* *Proceedings of security and data storage aspect in cloud computing*. *Springer*, Singapore, pp 35-48

[10] Shuai Zhao et. al, -Real-time Network Anomaly detection System Using Machine learning‖, *2015,11th International Conference on the Design of Reliable Communication Networks (DRCN),* IEEE, DOI: 10.1109/DRCN.2015.7149025

[11] Yansen Shou et.al, -Research of Network Traffic Anomaly Detection Model Based on Multilevel Autoregression‖ , *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)* , DOI:10.1109/ICCSNT47585.2019.8962517

[12] Suman Lata, Dheerendra Singh, -Intrusion Detection System in Cloud Enviornment, Literature Survey and future research directions‖ , *International Journal of Information Management data Insights*, Vol2, Issue 2, November 2022, 100134, Springer

[13] Faisal Shahzad, Abdul Mannan et al, -Cloud based Multiclass anomaly detection and Categorization using ensemble learning‖, ***Springer,*** *Journal of Cloud Computing* volume 11, Article number: 74 (2022)

[14] Jonathan J. Davis et. al, Computers & Security , -Data Preprocessing for Anomaly based Network Intrusion Detection‖, ***Elsevier*** , Volume 30, Issues 6– 7, September–October 2011, Pages 353-375

[15] Aditya Vikram, Mohana et.al, -Anomaly Detection in Network Traffic Using Unsupervised Machine Learning Approach- ,IEEE,2020

[16] Brao, Bobba et al., -Fast KNN Classifiers for Network Intrusion Detection System‖, ***Indian Journal of Science and Technology***. 2017.

[17] Azwar, Hassan et all., -Intrusion Detection in secure network for Cybersecurity systems using Machine Learning and Data Mining‖, 2018.

[18] Y. Chang et al., -Network Intrusion Detection Based on Random Forest and Support Vector Machine,‖ *IEEE International Conference on Computational Science and Engineering* , 2017.

[19] Francesco Palmieri, -Network Anomaly detection Using Logistic Regression of Nonlinear chaotic invariants‖, ***Elsevier ,*** *Journal of Network and Computer Applications* ,Volume 148, 15 Dec 2019.

[20] Alifiannisa Alyahasna Wighneswara et.al, -Network Behavior Anomaly Detection Using Decision Tree***”,*** *IEEE 12th International Conference on Communication Systems and Network Technologies (CSNT)*, 2023,DOI: 10.1109/CSNT57126.2023.10134589

[21] K. Samunnisa et.al , – Intrusion Detection system in Distributed cloud Computing: Hybrid clustering and

Classification methods‖, ***Elsevier***, Measurement Sensors: Volume 25, February 2023, 100612

[22] Sandip Sonawane et.al, ―Rule based learning intrusion detection system using KDD and NSL KDD dataset -, *Prestige International Journal of Management & IT - Sanchayan* (2015), pp. 135-145, 10.37922/PIJMIT.2015.V04i02.009

[23] Rajesh T, Deepa P. -A survey of intrusion detection models based on NSL-KDD data set‖. *In: **Proceedings of the 5th HCT information technology trends (ITT),*** Dubai, United Arab Emirates, November 2018 https://doi.org/10.1109/CTIT.2018.8649498.

[24] Hayoung O, Kijoon C.‖ Real-time intrusion detection system based self-organized maps and feature correlations‖, ***In: Proceedings of the 3rd international conference on convergence and hybrid information technology*** ,Korea(South), November,2008. https://doi.org/10.1109/ICCIT.2008.32.

[25] Komviriyavut T, Sangkatsanee P, Wattanapongsa-korn N, Charnsripinyo C. -Network intrusion detection and classification with Decision Tree and rule based approaches***". In: Proceedings of the 9th international symposium on communication and information technology***, Korea, September 2009. https://doi.org/10.1109/ISCIT.2009.5341005.

[26] Sangkatsanee P, Wattanapongsakorn N, Charnsripinyo C. -Practical real- time intrusion detection using machine learning approaches‖. Comput Commun.2011; 34(18):222735. https://doi.org/10.1016/j.comcom.2011.07.001

[27] Hui W, Zijian C, Bo H. -A network intrusion detection system based on convolutional neural Network‖. J Intell Fuzzy Syst. 2020;38:7623– 37. https://doi.org/10.3233/JIFS-179833

[28] Zhang C, Chen Y, Meng Y, Ruan F, Chen R, Li Y, Yang Y. -A novel framework design of network intrusion detection based on machine learning techniques‖. Secur Commun Netw. 2021 . https://doi.org/10.1155/2021/6610675.

[29] Nour M, Jill S. UNSW-NB15: -A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set***)". In: Proceedings of the military communications and information Systems conference***, Australia, November 2015 https://doi.org/10.1109/MilCIS.2015.7348942.

[30] Abbes, T., Bouhoula, A., Rusinowitch, M.:‖ Efficient decision tree for protocol analysis in intrusion detection‖. Int. J. Secur. Netw. 5(4), 220- 235,(2010)

[31] Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: -An effective unsupervised network anomaly detection method‖. ***In: Proceedings of the International Conference on Advances in Computing, Communications and Informatics***, pp. 533–539. ACM, New York(2012)

[32] Denning, D.E., Neumann, P.G.: -Requirements and Model for IDES – A Real-time Intrusion Detection System‖. Tech. Rep. 83F83-01-00, Computer Science Laboratory, SRI Inc. (1985)

[33] Khraisat A, Gondal I, Vamplew P (2018) -An anomaly intrusion detection system using C5 decision tree classifier‖. ***In: Trends and applications in knowledge discovery and data mining.*** *Springer International Publishing*,Cham, pp 149–155

[34] Kreibich C, Crowcroft J (2004) -Honeycomb: creating intrusion detection signatures using Honeypots***". SIGCOMM Comput Commun Rev*** 34(1):51–56

[35] Roesch M (1999) -Snort-lightweight intrusion detection for networks‖. ***In:Proceedings of the 13th USENIX conference on system administration***. Seattle,Washington, pp 229–238

[36] Adeeb M. Alhomoud et. al(2011), -Performance evaluation study of Intrusion Detection Systems‖, December 2011, *Procedia Computer Science* 5:173-180, DOI:10.1016/j.procs.2011.07.024

[37] Md Azam Hossain et.al(2022) ,‖ Network Traffic anomalies Detection using Machine Learning Algorithm: A Performance Study‖, ***In*** *Proceedings of 2nd International Conference on Smart Computing and Cyber Security* pp 274–282

[38] Ziadoon K. Maseer, Robiah Yusof,et.al, -Systematic Review for Anomaly Network Intrusion Detection Systems: Detection Methods, Dataset,Validation Methodology, and Challenges‖ , https://Doi.Org/10.48550/Arxiv.2308.02805

[39] Qian Ma et.al, -A Novel Model for Anomaly Detection in Network Traffic based on Support Vector Machine and Clustering‖(2021), ***Research Article | Open Access*** Volume 2021 | Article ID 2170788 | https://doi.org/10.1155/2021/2170788

[40] Tushar Rakshe, Vishal Gonjari, -Anomaly based Network Intrusion Detection using Machine Learning

Research Article

Techniques‖, ***International Journal of Engineering Research & Technology (IJERT)*** Published by : http://www.ijert.org ISSN: 2278-0181 Vol. 6 Issue 05, May – 2017

[41] Ahmed Tamer Assy et.al, -Anomaly-based Intrusion detection system using One Dimensional Convolutional Neural Network‖, ***Elsevier: The 14th International Conference on Ambien t Systems, Networks and Technologies(ANT)*** March15-17, 2023, Leuven, Belgium, Procedia Computer Science 220 (2023) 78–85 Procedia ,ComputerScience 00(2019)000–00

[42] M. Elif KarsligEl et.al , -Network Intrusion Detection using Machine learning Anomaly Detection Algorithms***”,*** ***In Preceedings of*** *2017,25th Signal Processing and Communications Applications Conference(SIU)* IEEE,2017, DOI:10.1109/SIU.2017.7960616