

INTRUSION DETECTION IN SOFTWARE DEFINED NETWORK USING MACHINE LEARNING

Submitted in partial fulfilment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

PRASHANTH L (38110423)

M SAI VAMSI (38110308)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC**

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MARCH - 2022



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **PRASHANTH L (38110423), M SAI VAMSI (38110308)** who carried out the project entitled **"INTRUSION DETECTION IN SOFTWARE DEFINED NETWORK USING MACHINE LEARNING"** under my supervision from **November 2021 to March 2022**.

Internal Guide

Dr.M. SELVI, M.E, PH.D.,

Head of the Department

Dr.S. Vigneshwari, M.E., Ph.D.,

Dr.L. Lakshmanan, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

PRASHANTH L (38110423), M SAI VAMSI (38110308) hereby declare that the Project Report entitled “**INTRUSION DETECTION IN SOFTWARE DEFINED NETWORK USING MACHINE LEARNING**” done by me under the guidance of Dr.M. SELVI, M.E, PH.D., is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr.T. Sasikala M.E., Ph.D., Dean**, School of Computing **Dr.L. Lakshmanan M.E., Ph.D.**, and **Dr.S. Vigneshwari M.E., Ph.D. Heads** of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.M. SELVI, M.E, PH.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

The entrance framework (IDS) is right now exceptionally fascinating as a significant piece of framework security. The IDS gathers traffic data from the line or framework and afterward involves it for better security. Assaults are typically truly challenging and tedious to isolate street exercises. To screen the organization association, the examiner should survey all data, enormous and wide. Subsequently, an organization search strategy is expected to decide the recurrence of traffic. In this review, another strategy for looking for IDS identifiers was created utilizing a technique for concentrating on information mining procedures from a calculation machine. In this paper, we propose to use few machine learning approaches that include Decision Tree, Bayesian Network, K-Nearest Neighbour to detect network intrusions efficiently. We use NSL-KDD dataset, a variant of widely used KDDCup 1999 intrusion detection benchmark dataset, for evaluating our proposed machine learning approaches for network intrusion detection. Finally, Experimental results with 5-class classification are demonstrated that include: Detection rate, false positive rate, and average cost for misclassification. These are used to aid a better understanding for the researchers in the domain of network intrusion detection.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	i
	LIST OF FIGURES	ii
	LIST OF ABBREVIATIONS	iii
	LIST OF TABLES	iv
1.	INTRODUCTION	1
	1.1 OUTLINE	1
	1.2 DATA OVERVIEW	2
	1.3 PROBLEM DESCRIPTION	3
2.	LITERATURE REVIEW	4
3.	METHODOLOGY	9
	MACHINE LEARNING	
	BASED APPROACH	9
	WORKFLOW OF ML ALGORITHMS	9
	ALGORITHMS USED	14
	GENETIC ALGORITHM	18
4.	RESULTS AND DISCUSSION	20
	DT RESULTS	20
	EVALUATION METRICS	21
	ACCURACY	23
	CROSS VALIDATION	24
5.	CONCLUSION AND FUTURE WORK	25

REFERENCES	27
APPENDIX	31
A. SAMPLE CODE	31
B. SCREENSHOTS	38
C. PLAGARISM REPORT	40

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO.
3.1	Workflow of ML Algorithms	10
3.2	Supervised Learning Method Structure	14
3.3	KNN Intrusion detection algorithm	16
3.4	Genetic algorithm overview	18
3.5	Genetic algorithm flow diagram	19

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
MAE	Mean absolute error
MAPE	Mean absolute percentage error
KNN	K nearest neighbor
Dos	Denial of service
U2R	User to Root
R2L	Remote to Local
DT	Decision Tree

LIST OF TABLES

TABLE NO	TABLE NAME	PAGE NO
1.1	KDD datasets attacks	2
4.1	Distribution of datasets	21
4.2	Accuracy Comparison of different Algorithms	23
4.3	accuracy comparison for sub-feature dataset	24
4.4	Cross Validation Results	24

CHAPTER 1

INTRODUCTION

OUTLINE

For the past few years, network has played a significant role in communication. The computer network allows the computing network devices to exchange information among different systems and individuals. The services of various organizations, companies, colleges, universities are accessed throughout computer network. This leads to a massive growth in networking field. The accessibility of internet has acquired a lot of interest among individuals. In this context, security of information has become a great challenge in this modern area. The information or data that we would like to send is supposed to be secured in such a way that a third party should not take control over them. When we are talking about security, we have to keep three basic factors in our mind: Confidentiality, Integrity and availability. Confidentiality means privacy of information. It gives the formal users the right to access the system via internet. This can be performed suitably along with accountability services in order to identify the authorized individuals. The second key factor is integrity. The integrity service means exactness of information. It allows the users to have self- assurance that the information passed is acceptable and has not been changed by an illegal individual.

An Intrusion Detection System (IDS) is used to watch malicious activities over the network. It can sort the unfamiliar records as normal or attack class. First monitoring of the network traffic is done, and then the IDS sorts these network traffic records into either malicious class or regular class. It acts as an alarm system that reports when an illegal activity is detected. The exactness of the IDS depends upon detection rate. If the performance is high for the IDS, then the correctness of detection is also high. Some of the intrusion detection systems are marketed with the ability to stop attacks before they are successful. They are used to shield an association from attack. It is a relative concept that tries to identify a hacker when intrusion is attempted. Ideally, such a system will only alarm when a successful attack is made. Intrusion detection system is not a perfect solution to all attack types. The various goals that can be accomplished with an Intrusion Detection System are: The potential goals include the following:

- IDS detect attacks.
- IDS traces user activity from point of entry to.
- IDS generate alerts when required.
- Detect errors in system configuration.
- Provides security of the system without the need of non – expert staff.
- IDS can detect when the system is under attack. Provides evidences for attack.

DATA OVERVIEW

The dataset being used is the [KDD'99](#) dataset for network intrusion. It's a famous dataset being used by many researchers for the purpose of intrusion detection applying various learnings. The dataset contains many attack types like the DOS, U2R, R2L, Probe and normal (no attack). There are 21 types of attacks inside the main categories mentioned above.

Categories of Attack	Attack name	Number of instances
DOS	SMURF	2807886
	NEPTUNE	1072017
	Back	2203
	POD	264
	Teardrop	979
U2R	Buffer overflow	30
	Load Module	9
	PERL	3
	Rootkit	10
R2L	FTP Write	8
	Guess Passwd	53
	IMAP	12
	Multihop	7
	PHF	4
	SPY	2
	Warez client	1020
	Warez Master	20
PROBE	IPSWEET	12481
	NMAP	2316
	PORTSWEEP	10413
	SATAN	15892
normal		972781

Table 1.1 KDD Datasets Attacks

The dataset contains a total of 41 attributes which could be used to determine if the attack is malicious or not at all an attack.

PROBLEM DESCRIPTION

With the rapid development of information technology in the past two decades. Computer networks are widely used by industry, business and various fields of the human life. Therefore, building reliable networks is a very important task for IT administrators. On the other hand, the rapid development of information technology produced several challenges to build reliable networks which is a very difficult task. There are many types of attacks threatening the availability, integrity and confidentiality of computer networks. The Denial-of-service attack (DOS) considered as one of the most common harmful attacks.

CHAPTER 2

LITERATURE REVIEW

Software-defined Networking (SDNs) have as of late been created as a feasible and promising answer for the eventual fate of the Internet. Networks are made due, incorporated, and observed and adjusted utilizing SDN. These advantages, then again, bring us ecological dangers, for example, network crashes, framework incapacities, internet banking misrepresentation, and robbery. These issues can detrimentally affect families, organizations, and the economy. Truth, superior execution, and the genuine framework are fundamental to accomplishing this objective. The extension of wise AI calculations into the network intrusion detection system (NIDS) through a software-defined network (SDN) has been extremely invigorating over the previous decade. The accessibility of data, the distinction in information investigation, and the many advances in AI calculations assist us with making a superior, more dependable, and solid framework for distinguishing the various sorts of organization assaults. The review was essential for the NIDS SDN survey.

Network Intrusion Detection Systems (NIDSs) are a significant device for network framework overseers to decide network security. NIDS screens and examines approaching and active calls from family network gadgets and cautions assuming that entrance is identified. As far as access control, NIDS is separated into two classifications: i) NIDS (SNIDS) based mark (abuse), and ii) NIDS (ADNIDS) based secrecy location. SNIDS and Drinking put assault marks first in NIDS. The helpful plan is made of against slip vehicle to permit admittance to the organization. Interestingly, ADNIDS permits network traffic to stream in when it is going to split away from typical traffic. Significant in characterizing SNIDS. notable, notable assault, non-salvage assault. Nonetheless, its unmistakable makes it extremely challenging to distinguish obscure or new assaults on the grounds that the marks of pre-introduced assaults on the IDS are decreased. However, ADNIDS is critical to be familiar with obscure and new assaults. In spite of the fact that ADNIDS estimates its adequacy well, its capacity to identify new assaults has prompted its far and wide acknowledgment. There are two issues that function admirably in the advancement of NIDS: gentle and direct assaults. Above all else, the strategy for choosing the right traffic information from the informational index line is hard to distinguish

peculiarities. Because of steady vacillations and changes, the capacities chose at a similar assault level may not be reasonable for other assault classes. Second, there is an absence of a bunch of traffic information from the genuine line of NIDS improvement. It requires a ton of work to separate a bunch of genuine or ongoing recorded information from the crude line of the gathered way.

The Network Access System (NIDS) assumes a significant part in ensuring PC organizations. Be that as it may, there are worries about the accessibility and maintainability of current innovation to meet present day network necessities. Specifically, these worries are connected with the increment in individuals' level of correspondence and the lessening in their level of information. This paper presents new top to bottom examination techniques to comprehend and resolve these issues. We plainly characterize non-standard encoder (NDAE) prerequisites for the investigation of uncontrolled items. Furthermore, we suggest a top to bottom investigation of the classes made utilizing the NDAE. Our proposals were carried out in GPU-TensorFlow and assessed utilizing the KDD Cup '99 scale and the NSL-KDD informational index.

Different Intrusion detection techniques have been explained and suggested over the last 2 decades (Catania and Garino 2012). To build robust and efficient IDS many types of research have been proposed with machine learning approaches and made some improvements in this area. An approach was introduced by Ingre and Yadav (2015) for ML-based IDS that combined the Decision Tree (DT) classifier and correlation input selection method for the classification task. A filter-based feature selection method was used to train and test their models. For their experimental process, the NSL-KDD dataset was analyzed. 14 significant features had been selected by filter-based feature selection technique to reduce the time complexity. However, this study was conducted for both binary and multiclass classification tasks. The result showed that they obtained an accuracy of 83.66% for a multiclass classification task that successfully identified five different attacks and 90.30% accuracy for binary classification.

Another filter-based approach was introduced to detect Distributed Denial of Service (DDoS) attacks by Osanaiye et al. (2016). They used multiple filters: Chi-Square, Information Gain, Gain Ratio, and ReliefF algorithm for selecting an optimum

number of features. For performance analysis of the system, they trained and evaluated the models on the NSL-KDD dataset. The classifier used in this approach was DT. The model had been validated with a tenfold cross-validation technique before testing. Their result indicated that DT was capable of detecting DDoS attacks with an accuracy of 99.67% while using 13 important features out of the whole feature set.

Alazzam et al. (2020) compared the performance of IDS using Decision Tree (DT) as a binary classifier. Pigeon Inspired Optimizer (PIO) is considered as the feature reduction technique in this study. PIO is a swarm intelligence algorithm inspired by pigeon's homing behavior. When a flock of pigeons flies a long way home, they use different navigation tools in different phases of their flight (Liu et al. 2019). The pigeons continuously change their position according to the navigation tool (map and compass operator) following the best pigeon that has the best position. Based on this philosophy of search and optimization, PIO has been further developed. In this experiment, two types of PIO were used for feature reduction: Cosine PIO and Sigmoid PIO. The authors carried out the simulation on UNSW-NB 15, NSL-KDD, and KDDCup99 datasets. Using Sigmoid PIO, 10 features from KDDCup99, 14 features from the UNSW-NB dataset, and 18 features from NSL-KDD were selected. Also, they selected 7 features from KDDCup99, 5 features from UNSW-NB-15, and NSL-KDD using Cosine PIO for comparing the result. Sigmoid PIO obtained an accuracy of 94.7% for KDDCup99, 86.9% using the NSL-KDD, and 91.3% over the UNSW-NB 15 dataset. On the contrary, Cosine PIO achieved 91.7% accuracy over UNSW-NB 15, 88.3% on NSL-KDD, and 96% on KDDCup99.

Another experiment was conducted by Khan et al. (2018) to enhance the performance of Random Forest, Extreme Gradient Boosting (XGBoost), Decision Tree, Bagging Meta Estimator, and K-Nearest Neighbors (KNN). The goal of this study was to decrease the computational time with reduced features while improving the accuracy of models. The subset of 11 features had been chosen according to the feature importance calculated by the Random Forest model. Random Forest obtained 74.87% accuracy, XGBoost achieved 71.43% detection accuracy, 74.64% for Bagging Meta Estimator using UNSW-NB Dataset. Also, Decision Tree and KNN attained an accuracy of 74.22% and 71.10% respectively. However, the training time for XgBoost was comparatively higher than other models in this experiment.

Decision Tree required the lowest prediction time and KNN needed significantly higher prediction time than other models.

An improved anomaly detection technique was proposed by Tama and Rhee (2019) using Gradient Boosting Machine (GBM) with complete features of NSL KDD, UNSW-NB 15, and GPRS dataset. The authors statistically assessed the superiority of GBM over other models: Support Vector Machine (SVM), Random Forest, Classification and Regression Tree (CART), and Deep Neural Network (DNN) in terms of area under the receiver operating characteristic curve (AUC), specificity, false-positive rate, sensitivity, and accuracy. The experiment was simulated in a Python environment with machine learning libraries. GBM outperformed other methods with an accuracy score of 91.31% for the UNSW-NB Dataset, 91.82% for KDDTest+, and 86.51% for KDDTest-21.

An SVM and Artificial Neural Network (ANN) based technique was proposed by Aboueata et al. (2019) for intrusion detection systems in cloud environments. Additionally, they considered Univariate and Principal Component Analysis (PCA) together for choosing an optimal set of features. They trained and tested the models on UNSW-NB 15 dataset and performance evaluation is carried out in terms of F1 score, Precision, Recall as well as Accuracy. However, feature engineering along with parameter tuning was performed by the authors to obtain maximum accuracy while reducing the time complexity of the models. The study indicated that with an appropriate set of features, ANN and SVM techniques were capable of detecting anomalies with 91% and 92% accuracy scores respectively

Jing and Chen (2019) introduced another SVM-based anomaly detection approach for both binary classification and multiclass classification. A nonlinear feature scaling method was applied in their simulation. However, they used Radial Basis Function (RBF) to map low dimensional space to high dimensional space in SVM. RBF kernel is suitable for large dataset evaluation in terms of computational time and accuracy. They achieved 85.99% testing accuracy for binary classification and 75.77% of accuracy for multiclass classification. Their preferred dataset for offline performance analysis was also the UNSW-NB 15 dataset. Kasongo and Sun (2020) implemented five supervised models: Support Vector Machine (SVM), Logistic Regression (LR), k-Nearest-Neighbour (kNN), Decision Tree (DT), and Artificial Neural Network

(ANN) using a filter-based feature reduction technique. The Extreme Gradient Boosting (XGBoost) technique was used for reducing the feature vector from 42 to 19 based on the feature importance score. This study compared the performance of models by training and testing them on UNSW-NB 15 dataset. However, they conducted the work for both binary and multiclass classification. With reduced features, they observed an increment in accuracy from 88.13 to 90.85% for the DT model while performing the binary classification task.

Another intrusion detection framework was proposed by Meftah et al. (2019) that included two stages. In the first stage, their approach performed binary classification. In the second stage, the attack traffic output was fed to multiclass classifiers for identifying each attack. The authors used Random Forest for the feature selection technique for the binary classification task. They trained the Gradient Boost Machine, Logistic Regression, and Support Vector Machine (SVM) and the test result revealed that SVM obtained the maximum detection accuracy of 82.11%. The Gradient Boost Machine and Logistic Regression obtained the accuracy score of 61.83% and 77.21% respectively. For multiclass classification, they applied multinomial Support Vector Machine, Decision Trees (C5.0), and Naïve Bayes. However, the Decision Tree achieved the highest accuracy of 74%, and 86% F1 score for multiclass classification. Moreover, they presented constructive criticism on UNSW-NB 15 dataset, which had been used in their experimental analysis.

Injadat et al. (2020) proposed an ML-based NIDS framework that studied the oversampling technique on the training data and selected a suitable sample size for training the KNN and RF. They applied correlation-based (CBFS) and information gain (IGBFS) feature selection techniques and compared the performance of classifiers using CICIDS 2017 and UNSW-NB 2015 datasets. However, unlike other research works which used the benchmark UNSW-NB 15 training and UNSW-NB 15 test datasets, this study used the full UNSW-NB 15 data (2,540,044 instances) and made a random split for training and testing. Moreover, they used the oversampling technique, SMOTE to synthetically produce more instances to a smaller class. They analyzed the effect of feature selection on the training sample and feature set size. The performance of the models was evaluated based on Accuracy, Precision, Recall score.

CHAPTER 3

METHODOLOGY

MACHINE LEARNING BASED APPROACH

In this section we provide an ML based taxonomy to review the existing literatures that used to formulate IDS solution for SDN environment. The researchers investigated the use of shallow ML approaches to meet the criteria of efficient and functional IDS. ML fall under the broad umbrella of AI and seek to learn meaningful information from big data. ML methods learn from the knowledge it gathers from the data during the training of the model. In general, for IDS, there exist two types of ML algorithms: supervised ML algorithms and unsupervised ML algorithms. An IDS model attempts to differentiate between regular and malicious traffic patterns in the supervised ML method, provided a set of network flows with labels (normal/malicious) before training. The manual labelling of each data is costly as well as time intensive. However, unsupervised ML algorithms gather vital information or features directly from the unlabelled dataset. Supervised ML methods show much superiority than unsupervised ML methods in IDS efficiency as observed in the works of previous researchers.

IDS has been using mainstream ML algorithms such as LR, SVM, DT, NB, RF, CART, ID3, KNN, SOM, K-Means and many more for a long time. Often it is observed that a collection of ML models—supervised and unsupervised is used in ensemble to perform a task. Major problems within this methodology are the need of availability of data and retraining of the data. Often, they are prone to overfitting and fail to detect unknown attacks. In the following, we briefly discuss the workflow of ML algorithms.

WORKFLOW OF ML ALGORITHMS

Building a system through ML-DL algorithms generally pass through several steps before achieving the outcome. These steps are data collection, data processing, model selection, training, evaluation, parameter tuning, and testing. However, sometimes these steps could be varied subject to the particular task and available data and features. A workflow of a typical ML model is presented in Fig.3.1.

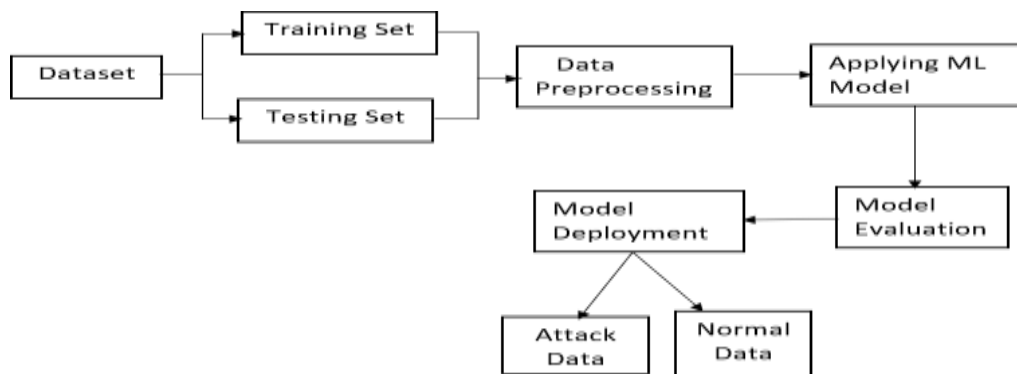


Fig 3.1 Workflow of ML Algorithms

Data Gathering

The process of gathering data depends on the type of project we desire to make, if we want to make an ML project that uses real-time data, then we can build an IoT system that using different sensors data. The data set can be collected from various sources such as a file, database, sensor and many other such sources but the collected data cannot be used directly for performing the analysis process as there might be a lot of missing data, extremely large values, unorganized text data or noisy data. Therefore, to solve this problem Data Preparation is done.

We can also use some free data sets which are present on the internet. Kaggle and UCI Machine learning Repository are the repositories that are used the most for making Machine learning models. Kaggle is one of the most visited websites that is used for practicing machine learning algorithms, they also host competitions in which people can participate and get to test their knowledge of machine learning.

Data Preprocessing

Collected raw data is generally noisy in nature. It may require some pre-processing tasks like removing duplicate data, identifying anomalies, discarding erroneous data, scaling, normalization, feature engineering, transformations, etc. Besides, the format of the data may require to be converted for some specific ML models. Data processing also enables data visualization to identify biases.

Why do we need it?

As we know that data pre-processing is a process of cleaning the raw data into clean data, so that can be used to train the model. So, we definitely need data pre-processing to achieve good results from the applied model in machine learning and deep learning projects. Most of the real-world data is messy, some of these types of data are:

1. *Missing data:*

Missing data can be found when it is not continuously created or due to technical issues in the application (IOT system).

2. *Noisy data:*

This type of data is also called outliers; this can occur due to human errors (human manually gathering the data) or some technical problem of the device at the time of collection of data.

3. *Inconsistent data:*

This type of data might be collected due to human errors (mistakes with the name or values) or duplication of data.

Three Types of Data

1. Numeric e.g., income, age
2. Categorical e.g., gender, nationality
3. Ordinal e.g., low/medium/high

How can data pre-processing be performed?

These are some of the basic pre — processing techniques that can be used to convert raw data.

1. *Conversion of data*

As we know that Machine Learning models can only handle numeric features, hence categorical and ordinal data must be somehow converted into numeric features.

2. *Ignoring the missing values*

Whenever we encounter missing data in the data set then we can remove the row or column of data depending on our need. This method is known to be efficient but it shouldn't be performed if there are a lot of missing values in the dataset.

3. Filling the missing values:

Whenever we encounter missing data in the data set then we can fill the missing data manually, most commonly the mean, median or highest frequency value is used.

Feature Selection & Extraction

Feature selection (FSS) is a pre-processing step commonly used in data mining. It is effective in dimensionality reduction and removes irrelevant features thus increases accuracy. It refers to the problem of identifying those features

that are useful in predicting class. Feature selection methods can be classified into three categories 1) filter method, 2) wrapper method and 3) embedded method

This is done to reduce the number of attributes in the dataset hence providing advantages like speeding up the training and accuracy improvements. In machine learning, pattern recognition, and image processing, feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations. Feature extraction is related to dimensionality reduction

When the input data to an algorithm is too large to be processed and it is suspected to be redundant (e.g., the same measurement in both feet and meters, or the repetitiveness of images presented as pixels), then it can be transformed into a reduced set of features (also named a feature vector). Determining a subset of the initial features is called feature selection. The selected features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data.

Model Selection and Training

A specific ML method is selected in this step. Results of different ML models can be varied significantly. That is why the model should be chosen carefully. A specific ML

method is selected in this step. Results of different ML models can be varied significantly. That is why the model should be chosen carefully. A training model is a dataset that is used to train an ML algorithm. It consists of the sample output data and the corresponding sets of input data that have an influence on the output.

The training model is used to run the input data through the algorithm to correlate the processed output against the sample output. The result from this correlation is used to modify the model. This iterative process is called “model fitting”. The accuracy of the training dataset or the validation dataset is critical for the precision of the model. Model training in machine language is the process of feeding an ML algorithm with data to help identify and learn good values for all attributes involved.

Evaluation

For the performance measurement of a selected model, various evaluation metrics and sampling techniques are used in this step. The trained model is often tested against unseen data using separate validation set or k-fold cross validation. Use of unseen data depicts the possible performance measure of the model in the practical scenario. In this module, we evaluate the performance of trained machine learning model using performance evaluation criteria such as F1 score, accuracy and classification error.

In case the model performs poorly, we optimize the machine learning algorithms to improve the performance. Performance Evaluation is defined as a formal and productive procedure to measure an employee’s work and results based on their job responsibilities. It is used to gauge the amount of value added by an employee in terms of increased business revenue, in comparison to industry standards and overall employee return on investment (ROI). All organizations that have learned the art of “winning from within” by focusing inward towards their employees, rely on a systematic performance evaluation process to measure and evaluate employee performance regularly. Ideally, employees are graded annually on their work anniversaries based on which they are either promoted or are given suitable distribution of salary raises. Performance evaluation also plays a direct role in providing periodic feedback to employees, such that they are more self-aware in terms of their performance metrics. In the next section, we discuss those studies which have used ML-based algorithms to develop IDS solutions specific to SDN.

ALGORITHMS USED

In the existing IDS literature in SDN, most of the researchers performed classification of network traffic using some supervised ML models. In SL, the machine is trained with well labelled dataset and aids in the prediction of unseen data as shown in Fig.3.2. It implies that some input data has already been marked with the accurate output label. DT, NB, RF, KNN and SVM are used more frequently in intrusion detection compared to other supervised models.

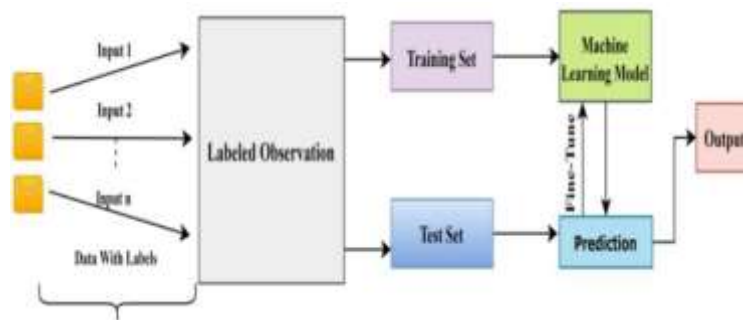


Fig 3.2 Supervised Learning Method Structure

Decision Tree

Decision tree is a classification technique. It is based on divide and conquer strategy. A decision tree consists decision nodes and leaf nodes, where decision node specifies a test over one of the attributes and a leaf node represents the class value. Every path from the root node to leaf node is rule. Classification error is the performance major factor for decision tree. Classification error is defined as the percentage of misclassified cases. In practice, the training data sets are usually large, which results in a greater number of branches and layers in the generated decision tree. In decision tree when the class categories are more, classification accuracy is significantly reduced.

The DT algorithm is a supervised tree-based method. It can be used for regression or classification tasks. DT creates and trains the model to predict the target's class value by learning simple tree rules. It starts from the first or the root node in the tree graph, compares the nodes' values with the given input, and searches in depth until it finds the required answer. DT can be constructed from the root node, which represents the entire record. The splitting process, which is the technique that makes the DT, works with it. The third decision node represents the node that has

been chosen by the model after one iteration selection — the fourth terminal node, which is the last level in the tree or the sub-tree. The ID3 metric is used to build the tree. This algorithm uses the entropy function and information gain as a metric, as shown in the following equation.

$$H(S) = \sum_{c \in C} (-P(c) \log_2 P(c))$$

Here, S is the current dataset for which the entropy is being calculated. C is the set of classes in S , $C = \text{Yes, No}$. $P(c)$ is the proportion of the number of elements in class c to the number of elements in set S . Using the ID3 and entropy metrics, the attributes are chosen to split the decision node. The smallest entropy is the first candidate to be chosen and split.

Intrusion detection can be considered as classification problem where each connection or user is identified either as one of the attack types or normal based on some existing data. Decision trees can solve this classification problem of intrusion detection as they learn the model from the data set and can classify the new data item into one of the classes specified in the data set. Decision trees can be used as a misuse intrusion detection as they can learn a model based on the training data and can predict the future data as one of the attack types or normal based on the learned model.

Decision trees work well with large data sets. This is important as large amounts of data flow across computer networks. The high performance of Decision trees makes them useful in real-time intrusion detection. Decision trees construct easily interpretable models, which is useful for a security officer to inspect and edit. These models can also be used in the rule-based models with minimum processing. Generalization accuracy of decision trees is another useful property for intrusion detection model. There will always be some new attacks on the system which are small variations of known attacks after the intrusion detection models are built. The ability to detect these new intrusions is possible due to the generalization accuracy of decision trees.

Random Forest

Random forest (RF) is an ensemble classifier used to improve the accuracy. Random forest consists of many decision trees. Random forest has low

classification error compared to other traditional classification algorithms. Number of trees, minimum node size and number of features used for splitting each node.

Advantages

- Generated forests can be saved for future reference.
- Random forest overcomes the problem over fitting.
- In RF accuracy and variable importance is automatically generated.

When constructing individual trees in random forest, randomization is applied to select the best node to split on. This value is equal to $\sqrt{A}$, where A is no. of attributes in the data set. However, RF will generate many noisy trees, which affect accuracy and wrong decision for new sample

K-Nearest Neighbor

K-nearest neighbor (KNN) classification algorithm is a data mining algorithm which is theoretically mature with low complexity. The basic idea is that, in a sample space, if most of its nearest neighbor samples belong to a category, then the sample belongs to the same category. The nearest neighbor refers to the single or multidimensional feature vector that is used to describe the sample on the closest, and the closest criteria can be the Euclidean distance of the feature vector.

K-nearest neighbor (KNN) classification algorithm is a data mining algorithm which is theoretically mature with low complexity. The basic idea is that, in a sample space, if most of its nearest neighbor samples belong to a category, then the sample belongs to the same category. The nearest neighbor refers to the single or multidimensional feature vector that is used to describe the sample on the closest, and the closest criteria can be the Euclidean distance of the feature vector.

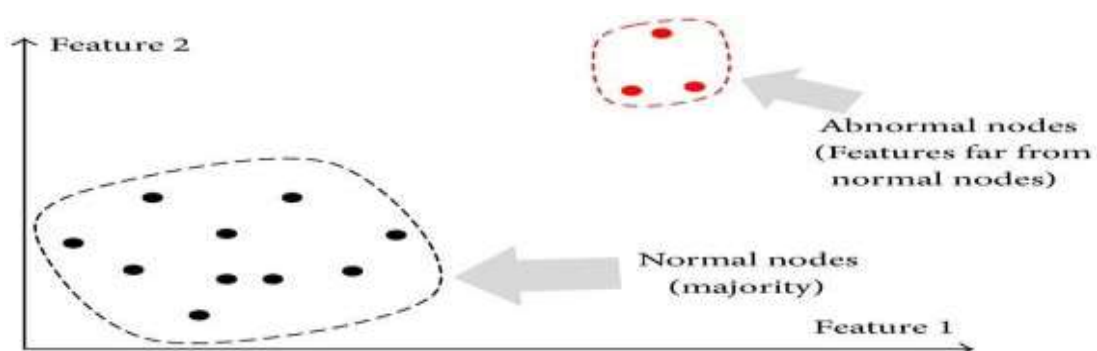


Fig 3.3 KNN Intrusion detection algorithm

In the intrusion detection algorithm, we use a n -dimensional vector to represent nodes, such as. These dimensions can be as follows: the routing message number can be sent in a period of time, the number of nodes with different destinations in the sending routing packets, the number of nodes with the same source node in the receiving routing packets, and so on. In general, the node of the same type has the same characteristics. Thus, the abnormal nodes will be distinguished.

Naïve Bayes Classifier

The naïve Bayes model is a heavily simplified Bayesian probability model. The naïve Bayes classifier operates on a strong independence assumption. This means that the probability of one attribute does not affect the probability of the other. Given a series of n attributes, the naïve Bayes classifier makes 2^n independent assumptions. Nevertheless, the results of the naïve Bayes classifier are often correct. The work reported in examines the circumstances under which the naïve Bayes classifier performs well and why. It states that the error is a result of three factors: training data noise, bias, and variance. Training data noise can only be minimised by choosing good training data. The training data must be divided into various groups by the machine learning algorithm. Bias is the error due to groupings in the training data being very large. Variance is the error due to those groupings being too small.

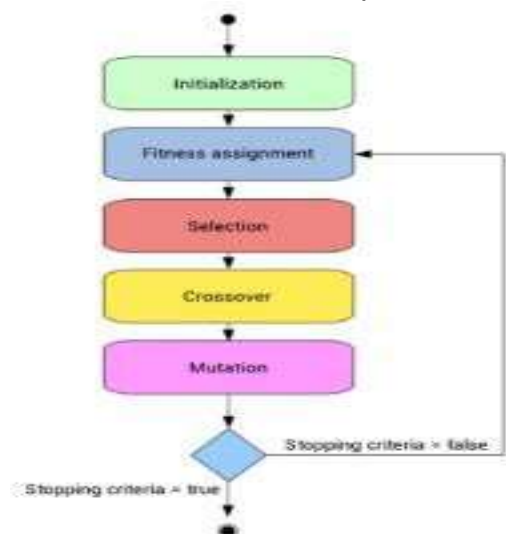
The naïve Bayes model is a heavily simplified Bayesian probability model. In this model, consider the probability of an end result given several related evidence variables. The probability of end result is encoded in the model along with the probability of the evidence variables occurring given that the end result occurs. The probability of an evidence variable given that the end result occurs is assumed to be independent of the probability of other evidence variables given that end results occur. Now we will consider the alarm example using a naïve Bayes classifier. Assume that we have a set of examples that monitor some attributes such as whether it is raining, whether an earthquake has occurred etc. Let's assume that we also know, using the monitor, about the behaviour of the alarm under these conditions. In addition, having knowledge of these attributes, we record whether or not a theft actually occurred. We will consider the category of whether a theft

occurred or not as the class for the naïve Bayes classifier. This is the knowledge that we are interested in. The other attributes will be considered as knowledge that may give us evidence that the theft has occurred. Fig 3.2 below shows the framework for a Naïve Bayesian model to perform intrusion detection.

The naïve Bayes classifier operates on a strong independence assumption. This means that the probability of one attribute does not affect the probability of the other. Given a series of n attributes, the naïve Bayes classifier makes 2^n independent assumptions. Nevertheless, the results of the naïve Bayes classifier are often correct. The work reported in examines the circumstances under which the naïve Bayes classifier performs well and why. It states that the error is a result of three factors: training data noise, bias, and variance. Training data noise can only be minimised by choosing good training data. The training data must be divided into various groups by the machine learning algorithm. Bias is the error due to groupings in the training data being very large. Variance is the error due to those groupings being too small.

GENETIC ALGORITHM

Firstly, we defined our initial population as our countrymen. We defined a function to classify whether it is a person is good or bad. Then we selected good people for mating to produce their off-springs. Finally, these off-springs replace the bad people from the population and this process repeats. This is how the genetic algorithm actually works, which basically tries to mimic human evolution to some extent. So, to formalize a definition of a genetic algorithm, we can say that it is an optimization technique, which tries to find out such values of input so that we get the best output



values or results. The working of a genetic algorithm is also derived from biology, which is as shown in the image below.

Fig 3.4- Genetic Algorithm overview

Genetic algorithm is one of the most commonly used machine learning approach in the field of intrusion detection, which consists of its natural selection. Decision node represents to testing a single attribute of the given instances whereas the leaf node presents the idea about whether the output of a classifier falls in to either normal or intrusion (any of the possible attacks) category during the classification phase. A novel method to find intrusion characteristic for IDS using decision tree machine learning of data mining technique was proposed. Method used to generate of rules is classification by genetic algorithm of decision tree.

Advantages

- Intrusion detection can be performed manually or automatically.
- IDS need to be able to deal with a large and ever-increasing flow of alerts and events.
- Using Decision trees is more essential. Automatic procedures for detecting and responding to intrusion are becoming increasingly essential.

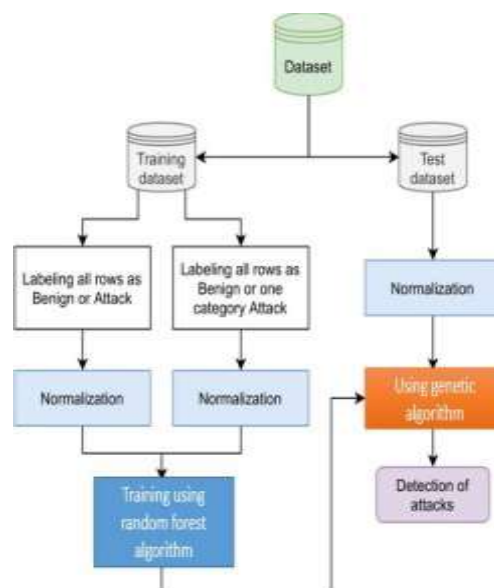


Fig 3.5 Genetic Algorithm flow diagram

CHAPTER 4

RESULTS AND DISCUSSION

DT RESULTS

Decision trees and other classification algorithms have been tested and applied to many practical intrusion detection challenges in academia. Some researchers have successfully created decision tree classifiers using data from honeypots (i.e., decoy systems put on a network as bait for attackers) and data from normal network activities. This analysis leads to the creation of decision rules to identify malicious activity. Other classification techniques have also been used to classify and characterize scanning activity as well as to detect novel attacks from cyber adversaries. In the corporate world, researchers have also used these techniques to prioritize IDS alarms and determine their root causes.

Summarizing the benefits of decision tree analysis to cyber security, decision trees can help perform the following functions for an organization:

- Supplement honeypot analysis by learning from adversary trends and creating rules to detect malicious activity.
- Supplement penetration testing efforts by learning from the pen tester's actions and creating rules to detect their tactics, techniques, and procedures.
- Identify and highlight malicious traffic.
- Prioritize alerting by identifying and tagging low priority alerts.
- Identify and characterize known scanning activity Supplement an incident response team's recovery and monitoring process by flagging a repeated intrusion attempt early.
- Detect previously unknown network anomalies.
- Support IDS signature development by providing a set of rules to identify malicious activity.

After decision trees are built, they have the potential to reduce the amount of data required for analysis, help identify anomalous malicious activity, and provide analytic insight into the differences between malicious and benign network traffic. In addition, as the amount of data handled by IT security experts increases, flagging the priority of malicious activity in an automated fashion becomes more and more beneficial for the mission.

Dataset	Type	Total	Imbalance ratio	Training set	Testing set
NSL-KDD	NORMAL	77054	-	67343	9711
	DoS	53385	1.44	45927	7458
	R2L	3882	19.85	995	2887
	Probe	14077	5.47	11656	2421
	U2R	119	647.51	52	67

Table 4.1 Distribution of datasets

EVALUATION METRICS

The Performance of Intrusion detection system is measured in terms of Precision, Accuracy, Recall, F-measure. Intrusion detection system requires high detection rate, high accuracy. A confusion matrix is used to calculate these all parameters.

- True Positive (TP) - Attack data is correctly classified as an attack.
- False Positive (FP) - Normal data is incorrectly classified as an attack.
- True Negative (TN) - Normal data is correctly classified as normal.
- False Negative (FN) - Attack data is incorrectly classified as normal.

The firewall has turned into a significant security strategy issue overall. Firewalls can be permitted or denied over the organization, however since firewalls can't be recognized or assaulted, signing in and applying to a firewall is a method for controlling how you forestall it. Access location Firewall innovation is viewed as an extra answer for identify interruptions in an organization without a firewall. Firewalls and IDS address the old as far as data innovation security. A firewall is great for ensuring frameworks and networks and lessens the danger of organization assaults. IDS can identify endurance or assault. Capacity to interface IDS and firewalls called IPS. That is the fair thing to do, and it should end there. There is at least one distinct

standard for every retailer. Each organization parcel that arrives at the firewall should be tried by characterized rules until an appropriate rule is found. Under current law, bundles will be permitted or restricted from arriving at the line. Every law determines a particular kind of vehicle. The points of interest of how the pipeline will be sold should be visible from the lines of vehicles from people's perspective. This review plans to try not to attempt to sign in to look for Internet-based substance, like IDS, and afterward implementing firewall rules like impeding. Need to find out about our information mining machine security strategy. The technique used to make the standard is to rank the ID3 calculation by tree endorsement. It's a decent and great practice to implement firewalls.

In general, the performance of NIDS is evaluated in terms of accuracy (AC), precision (P), recall (R) and F-measure (F). A NIDS requires high accuracy, high detection rate and low false alarm rate. A confusion matrix is used to calculate these parameters. In the confusion matrix, True Positive (TP) is the number of attack records correctly classified. True Negative (TN) is the number of normal records correctly classified. False Positive (FP) is the number of normal records incorrectly classified. False Negative (FN) is the number of attack records incorrectly classified. Then we can say:

- Accuracy (AC): shows the percentage of true detection over total traffic trace:

$$AC = TP + TN / (TP + TN + FP + FN)$$

- Precision (P): shows how many intrusions predicted by a NIDS are actual intrusions. The higher P is, the lower false alarm is:

$$P = TP / (TP + FP)$$

- Recall (R): shows the percentage of predicted intrusions versus all intrusion presented. We want a high R value:

$$R = TP / (TP + FN)$$

- F-measure (F): gives a better measure of the accuracy of a NIDS by considering both the precision (P) and the recall (R). We also aim for a high F value:

$$F = 2 / (1/P + 1/R)$$

ACCURACY

We also evaluated our work by comparing the results with that of other works and other machine learning algorithms. In, the authors use 6-tuple features: APf, ABf, ADf, PPf, GSf and GDP for DDoS detection. Their result is good with a detection rate of 99.11% and a false alarm rate of 0.46%. Our results are quite low compared to their results. The main reason may be our feature selection for training and testing. Their six features are mainly specified for detecting DDoS attacks. In 4 our case, we just chose the six most basic features and we did not focus on any kind of attacks. In our future research, we will try to apply their 6-tuple features to our model for further evaluation. We compared our results with the result in from different machine learning algorithms. In, the authors train and test different algorithms with a full training and testing set of forty-one features. From these experiments, the authors can evaluate the performance of these algorithms in their dataset. From Table V, we can see that our DNN approach is quite low compared with other algorithms with 75.75% accuracy. The most accurate machine learning algorithm is the NB tree with 82.02%. This is the accuracy obtained from full feature training set, and so the NB tree algorithm can generalize the characteristics of the normal and anomaly traffic very well. In our simulation, it must be noted that only the six basic features which are mentioned in Table 4.2 are used for training and testing. This sub-feature set cannot provide enough information for our algorithm to generalize the characteristics of some sophisticated or new attacks. However, it can be seen that algorithm performs reasonably compared with other algorithms.

Algorithm	Accuracy (%)
Decision Tree	95.05
Naive Bayes (NB)	88.56
Random Forest	92.67
KNN	85.52

Table 4.2 Accuracy Comparison of different Algorithms

For further evaluation, we just use the sub-feature set for training and testing. In the next experiment, the proposed DNN algorithm in this study was compared with other machine learning algorithms: NB, Random Forest and Decision Tree. Table 5.2 gives us an overview about the performance of each machine learning algorithm

using sub-feature dataset. The proposed DNN algorithm gives us the best results amongst all algorithms. The other machine learning algorithms cannot generalize the characteristic of training samples well with just six features, so they achieve quite poor performance.

Algorithm	Accuracy (%)
Naive Bayes	92
Decision Tree	94

Table 4.3 ACCURACY COMPARISON FOR SUB-FEATURE DATASET

CROSS VALIDATION

The Random Forest, KNN and the Decision Table give high accuracy but the accuracy achieved in case of the Decision tree is the highest. Therefore, to further our project we choose the Random Forest algorithm, use it practically to find out the error rate using the Python Programming language.

	Cross Validation (10 Folds)		Percentage Split (70%)	
	Correctly Classified Instances	Incorrectly Classified Instances	Correctly Classified Instances	Incorrectly Classified Instances
Naive Bayes	58866 (76.16%)	18425 (23.84%)	17987 (77.57%)	5200 (22.43%)
Random forest	76141 (98.51%)	1150 (1.49%)	22868 (98.51%)	319 (1.36%)
KNN	76474 (98.94%)	817 (1.06%)	22944 (98.95%)	243 (1.05%)
Decision Tree	76829 (99.40%)	462 (0.60%)	23040 (99.37%)	147 (0.63%)

Table 4.4 Cross Validation Results

CHAPTER 5

CONCLUSION AND FUTURE WORK

CONCLUSION

Initially, we implemented the NIDS for 2-class classification (normal and anomaly class). The performance of the model depends on the value of the hyper-parameter initiated. Firstly, the model was optimized by finding the hyper-parameter which leads to optimal classification results. Thus, we tried to optimize the model by varying the value of learning rate in the range {0.1, 0.01, 0.001, 0.0001}. When training a model, we tried to minimize the loss and maximize the accuracy. By comparing loss and accuracy of the training phase (see Table III), we can see that along with the decrease of the learning rate, the loss will decrease and the accuracy will increase. However, in the testing phase, when we decreased the learning rate to 0.0001, the results are not as good as the learning rate of 0.001. This is because if the learning rate is too small, the NIDS model will be trained too accurately. That is the reason for the best results in the training phase of the learning rate of 0.0001. Because of this too accurate training phase, the model cannot generalize the characteristic of the training samples well. So, while the model can easily catch the intrusion instances in the training set, it cannot catch the new intrusion instances in the test set. As a result, the NIDS performance decreases in the testing phase.

Before directly using all the 41 attributes onto the R we do feature selection to select the attributes using the InfoGain method to reduce the time of computation whereas the accuracy is not much compromised.

The attributes elected are

- SrcBytes - number of data bytes from source to destination
- DstBytes - number of data bytes from destination to source
- DstHostSameSrvRate – Destination host same server rate.
- Count - number of connections to the same host as the current connection in the past two seconds
- DstHostDiffSrvRate - Destination host different server rate.

The number of attributes reduce from 41 to 5. Using the InfoGain method which makes it much faster in terms of execution time. The Python language is then used to see the OOB Estimate of Error that is retrieved after using the Random Forest

machine-learning algorithm. The OOB Estimate of Error was found to be 1.46% when the half of dataset was used in training and the other half in testing the algorithm. For each of the testing dataset we can predict if the connection setup is normal or malicious and even pin point to the particular type of attack (example – Neptune, smurf, Saturn, teardrop, rootkit etc.) This means that we can be 98.54% sure of the prediction value shown by the algorithm running on.

FUTURE WORK

In this paper, we have implemented a Supervised learning algorithm for detecting network intrusion and evaluated our NIDS model. Although our results are not yet good enough to be adopted in any commercial product or an alternative solution for signature-based IDS, our approach still has significant potential and advantages for further development. By comparing the results with those of other classifiers, we have shown the potential of using deep learning for the flow-based anomaly detection system. In the context of the SDN environment, the deep learning approach also has potential. This is attributed to the centralized nature of the controller and the flexible structure of SDN. The basic information about network traffic can be extracted easily by the controller and evaluated by the deep learning intrusion detection module. To improve the accuracy, we will analyse the traffic and propose other types of features. With the flexibility of the SDN structure, we can extract many features that contain more valuable information or focus on one specific type of attack, like DDoS, to increase the accuracy of the NIDS. We will also try to adjust our DNN model for better performance (e.g., varying the number of hidden layers and hidden neurons). In the near future, we will also try to implement this approach in a real SDN environment with real network traffic and evaluate the performance of the whole network in terms of latency and throughput.

Decision tree assists the network administrator to decide about the incoming traffic, i.e., whether the coming data is malicious or not by providing a model that separates malicious and non-malicious traffic. By modified the split value calculation by taking the average of all the values in the domain of an attribute. The algorithm provides uniform weightage to all the values in the domain. It allows taking a smaller number of attributes and provides acceptable accuracy in reasonable account of time. From the results of the experiments, it is concluded that the proposed algorithm for

signature-based intrusion detection is more efficient with respect to finding attacks in the network with less number of features and it takes less time to construct the model. It is also concluded that the efficiency depends on the size of the data set and the number of features used to construct the decision tree. The formula used in DTS to calculate gain ratio can also be used in attribute selection for feature reduction. Our future scope of work is to improve the split value by using concepts such as geometric mean which also gives uniform weightage to the domain values.

REFERENCES

- [1] H. J. Liao, C. H. Richard Lin, Y. C. Lin, and K. Y. Tung, "Intrusion detection system: A comprehensive review," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 16–24, 2013.
- [2] "Software Defined Networking Market Size, Share and Global Market Forecast to 2025 | COVID-19 Impact Analysis," *Marketsandmarkets.com*, 2020. [Online]. Available: 14 Feb 2021.
- [3] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, 2015.
- [4] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A Survey on Software-Defined Networking," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 1, pp. 27–51, 2015.
- [5] S. W. Lee et al., "Towards secure intrusion detection systems using deep learning techniques: Comprehensive analysis and review," *J. Netw. Comput. Appl.*, vol. 187, no. December 2020, p. 103111, 2021.
- [6] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Appl. Sci.*, vol. 9, no. 20, 2019.
- [7] P. Mishra, V. Varadharajan, S. Member, and U. Tupakula, "A Detailed Investigation and Analysis of using Machine Learning Techniques for Intrusion Detection," *IEEE Commun. Surv. Tutorials*, vol. 21, no. 1, pp. 686-728., 2018.
- [8] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, 2019.

- [9] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 1, pp. 1–29, 2021.
- [10] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on SDN based network intrusion detection system using machine learning approaches," *Peer-to-Peer Netw. Appl.*, vol. 12, no. 2, pp. 493–501, 2019.
- [11] J. Xie et al., "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges," *IEEE Commun. Surv. Tutorials*, vol. 21, no. 1, pp. 393–430, 2019.
- [12] J. Singh and S. Behal, "Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions," *Comput. Sci. Rev.*, vol. 37, p. 100279, 2020.
- [13] T. Jafarian, M. Masdari, A. Ghaffari, and K. Majidzadeh, "A survey and classification of the security anomaly detection mechanisms in software defined networks," *Cluster Comput.*, vol. 1, 2020.
- [14] Y. Hande and A. Muddana, "A survey on intrusion detection system for software defined networks (SDN)," *Int. J. Bus. Data Commun. Netw.*, vol. 16, no. 1, pp. 28–47, 2020.
- [15] L. F. Eliyan and R. Di Pietro, "DoS and DDoS attacks in Software Defined Networks: A survey of existing solutions and research challenges," *Futur. Gener. Comput. Syst.*, vol. 122, pp. 149–171, 2021.
- [16] Y. Cui et al., "Towards DDoS detection mechanisms in SoftwareDefined Networking," *J. Netw. Comput. Appl.*, vol. 190, no. July, p. 103156, 2021.
- [17] "Open Networking Foundation. 2021. Software-Defined Networking (SDN) Definition - Open Networking Foundation. [online]," Open Networking Foundation, 2013. [Online]. Available: <https://searchnetworking.techtarget.com/definition/software-definednetworking-SDN>. [Accessed: 08-Feb-2021].

- [18] W. Braun and M. Menth, "Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices," *Futur. Internet*, vol. 6, no. 2, pp. 302–336, 2014.
- [19] "Open vSwitch." [Online]. Available: <https://www.openvswitch.org/>. [Accessed: 07-Nov-2021].
- [20] "Pantou - OpenFlow 1.3 for OpenWRT." [Online]. Available: <https://github.com/CPqD/openflow-openwrt>. [Accessed: 07-Nov2021].
- [21] "Indigo - Core OpenFlow agent and infrastructure modules." [Online]. Available: <https://floodlight.atlassian.net/wiki/spaces/Indigo/overview>. 41 [Accessed: 07-Nov-2021].
- [22] A. Voellmy and P. Hudak, "Nettle: Functional Reactive Programming of OpenFlow Networks," in *YALE UNIV NEW HAVEN CT DEPT OF COMPUTER SCIENCE*, 2010.
- [23] J. W. Lockwood et al., "NetFPGA-an open platform for gigabit-rate network switching and routing," in *IEEE International Conference on Microelectronic Systems Education (MSE'07)*, 2007, p. 177.
- [24] G. Lu et al., "Serverswitch: a programmable and high performance platform for data center networks," *USENIX Symp. Networked Syst. Des. Implement.*, vol. 11, pp. 1–14, 2011.
- [25] M. B. Anwer, M. Motiwala, M. Bin Tariq, and N. Feamster, "SwitchBlade: A platform for rapid deployment of network protocols on programmable hardware," in *Proceedings of the ACM SIGCOMM 2010 conference*, pp. 183–194.
- [26] A. Ben Letaifa, *SSIM and ML based QoE enhancement approach in SDN context*, 1st ed., vol. 114. Elsevier Inc., 2019.
- [27] S. Gao, Z. Li, B. Xiao, and G. Wei, "Security Threats in the Data Plane of Software-Defined Networks," *IEEE Netw.*, vol. 32, no. 4, pp. 108– 113, 2018.
- [28] D. B. Rawat and S. R. Reddy, "Software defined networking architecture, security and energy efficiency: A survey," *Softw. Defin. Netw. Archit. Secur. Energy Effic. A Surv.*, vol. 19, no. 1, pp. 325– 346, 2017.

- [29] “RYU Controller.” [Online]. Available: <https://ryu-sdn.org/>. [Accessed: 25-Dec-2020].
- [30] L. R. Prete, A. A. Shinoda, C. M. Schweitzer, and R. L. S. De Oliveira, “Simulation in an SDN network scenario using the POX Controller,” in 2014 IEEE Colombian Conference on Communications and Computing, COLCOM 2014 - Conference Proceedings, 2014, no. June, pp. 1–6.
- [31] N. Gude et al., “NOX: towards an operating system for networks,” ACM SIGCOMM Comput. Commun. Rev., vol. 38, no. 3, pp. 105– 110, 2008.
- [32] “Opendaylight Controller.” [Online]. Available: <https://www.opendaylight.org/about/platform-overview>. [Accessed: 28-Dec-2020].
- [33] FloodLight Controller, “FloodLight Controller.” [Online]. Available: <https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview>. [Accessed: 27-Dec-2020].
- [34] “ONOS Controller.” [Online]. Available: <https://opennetworking.org/onos/>. [Accessed: 27-Dec-2020].
- [35] A. Dawoud, S. Shahrstani, and C. Raun, “Deep learning and softwaredefined networks: Towards secure IoT architecture,” Internet of Things, vol. 3–4, pp. 82–89, 2018.
- [36] A. Yazdinejadna, R. M. Parizi, A. Dehghantanha, and M. S. Khan, “A kangaroo-based intrusion detection system on software-defined networks,” Comput. Networks, vol. 184, p. 107688, 2021.
- [37] P. Dong, X. Du, H. Zhang, and T. Xu, “A detection method for a novel DDoS attack against SDN controllers by vast new low-traffic flows,” in 2016 IEEE International Conference on Communications, ICC 2016, 2016.
- [38] “OpenFlow Switch Specification Version 1.3.5 (Protocol version 0x04),” 2015. [Online]. Available: <https://opennetworking.org/wpcontent/uploads/2014/10/openflow-switch-v1.3.5.pdf>. [Accessed: 19- Dec-2021].
- [39] “‘Openflow – The Workflow Introduction’. SDN For You, 2020,” Openflow – The Workflow Introduction, 2015. [Online]. Available:

<https://sdnforyou.wordpress.com/2015/08/12/openflow-theworkflow-introduction/>.
[Accessed: 19-Nov-2020].

[40] A. Bahnasse, F. E. Louhab, H. Ait Oulahyane, M. Talea, and A. Bakali, "Novel SDN architecture for smart MPLS Traffic EngineeringDiffServ Aware management," *Futur. Gener. Comput. Syst.*, vol. 87, pp. 115–126, 2018.

APPENDIX

A. SAMPLE CODE

```
import numpy as np

import pandas as pd

from sklearn.kernel_approximation import RBFSampler

from sklearn.linear_model import SGDClassifier

from sklearn.model_selection import train_test_split

from sklearn import svm

from sklearn.metrics import classification_report

from sklearn import metrics

from sklearn.linear_model import LogisticRegression

from sklearn.naive_bayes import GaussianNB

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import (precision_score, recall_score, f1_score,
accuracy_score, mean_squared_error, mean_absolute_error)

from sklearn.ensemble import AdaBoostClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn import preprocessing

from sklearn.preprocessing import Normalizer
```

```

from sklearn.model_selection import GridSearchCV

from sklearn.svm import SVC

from sklearn.metrics import confusion_matrix

from sklearn.metrics import (precision_score, recall_score, f1_score,
accuracy_score, mean_squared_error, mean_absolute_error, roc_curve,
classification_report, auc)

import warnings

warnings.filterwarnings("ignore")

traindata = pd.read_csv('tr.csv', header=None)

testdata = pd.read_csv('ts.csv', header=None)


X = traindata.iloc[:,1:42]

Y = traindata.iloc[:,0]

C = testdata.iloc[:,0]

T = testdata.iloc[:,1:42]


scaler = Normalizer().fit(X)

trainX = scaler.transform(X)


scaler = Normalizer().fit(T)

testT = scaler.transform(T)


traindata = np.array(trainX)

trainlabel = np.array(Y)

```

```

testdata = np.array(testT)

testlabel = np.array(C)


model = LogisticRegression()

model.fit(traindata, trainlabel)


# make predictions

expected = testlabel

predicted = model.predict(testdata)

np.savetxt('predictedLR.txt', predicted, fmt='%01d')

accuracy = accuracy_score(expected, predicted)

recall = recall_score(expected, predicted, average="binary")

precision = precision_score(expected, predicted , average="binary")

f1 = f1_score(expected, predicted , average="binary")

cm = metrics.confusion_matrix(expected, predicted)


tpr = float(cm[0][0])/np.sum(cm[0])

fpr = float(cm[1][1])/np.sum(cm[1])


# fit a Naive Bayes model to the data

model = GaussianNB()

model.fit(traindata, trainlabel)

```

```

print(model)

# make predictions

expected = testlabel

predicted = model.predict(testdata)

np.savetxt('predictedNB.txt', predicted, fmt='%01d')

accuracy = accuracy_score(expected, predicted)

recall = recall_score(expected, predicted, average="binary")

precision = precision_score(expected, predicted , average="binary")

f1 = f1_score(expected, predicted , average="binary")

cm = metrics.confusion_matrix(expected, predicted)

print(cm)

tpr = float(cm[0][0])/np.sum(cm[0])

fpr = float(cm[1][1])/np.sum(cm[1])

print("%.3f" %tpr)

print("%.3f" %fpr)

print("Accuracy")

print("%.3f" %accuracy)

print("precision")

print("%.3f" %precision)

print("recall")

print("%.3f" %recall)

print("f-score")

print("%.3f" %f1)

```

```

print("*****")

# fit a k-nearest neighbor model to the data

model = KNeighborsClassifier()

model.fit(traindata, trainlabel)

print(model)

# make predictions

expected = testlabel

predicted = model.predict(testdata)

np.savetxt('predictedKNN.txt', predicted, fmt='%01d')

# summarize the fit of the model

accuracy = accuracy_score(expected, predicted)

recall = recall_score(expected, predicted, average="binary")

precision = precision_score(expected, predicted , average="binary")

f1 = f1_score(expected, predicted , average="binary")


cm = metrics.confusion_matrix(expected, predicted)

print(cm)

tpr = float(cm[0][0])/np.sum(cm[0])

fpr = float(cm[1][1])/np.sum(cm[1])

print("%.3f" %tpr)

print("%.3f" %fpr)

print("Accuracy")

print("%.3f" %accuracy)

```

```

print("precision")

print("%.3f" %precision)

print("recall")

print("%.3f" %recall)

print("f-score")

print("%.3f" %f1)


print("*****")

model = DecisionTreeClassifier()

model.fit(traindata, trainlabel)

print(model)

# make predictions

expected = testlabel

predicted = model.predict(testdata)

np.savetxt('predictedDT.txt', predicted, fmt='%01d')

# summarize the fit of the model

accuracy = accuracy_score(expected, predicted)

recall = recall_score(expected, predicted, average="binary")

precision = precision_score(expected, predicted , average="binary")

f1 = f1_score(expected, predicted , average="binary")


cm = metrics.confusion_matrix(expected, predicted)

print(cm)

tpr = float(cm[0][0])/np.sum(cm[0])

```

```
fpr = float(cm[1][1])/np.sum(cm[1])

print("%.3f" %tpr)

print("%.3f" %fpr)

print("Accuracy")

print("%.3f" %accuracy)

print("precision")

print("%.3f" %precision)

print("recall")

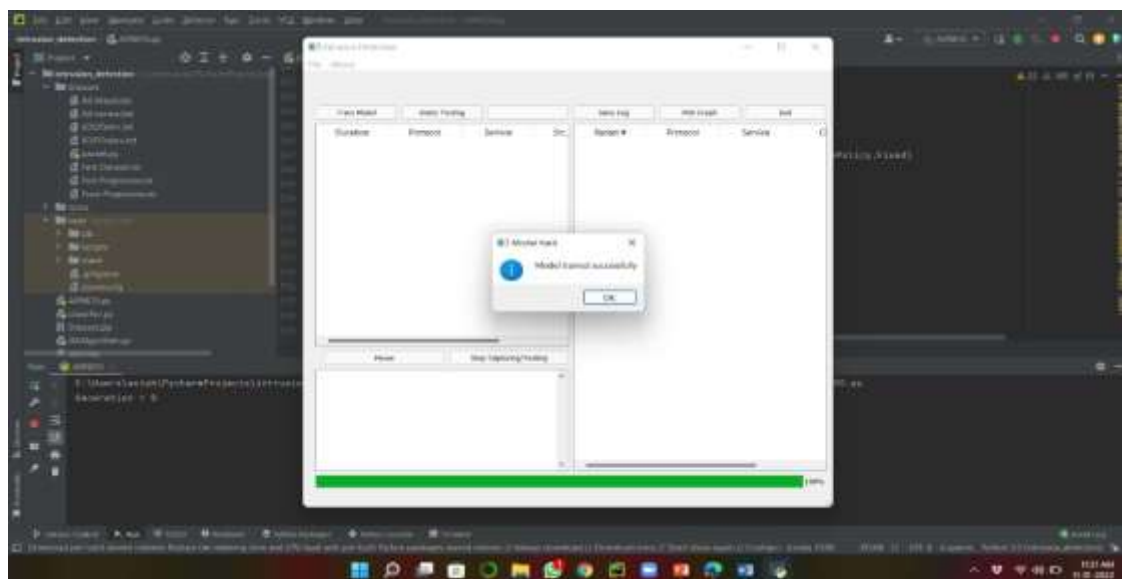
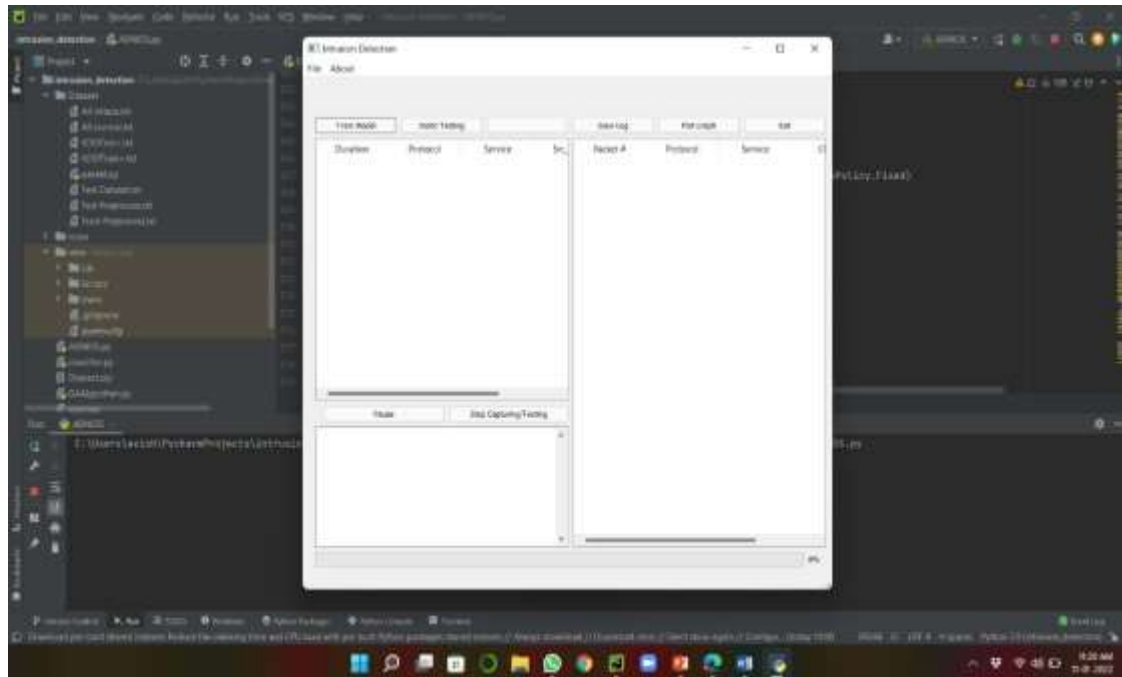
print("%.3f" %recall)

print("f-score")

print("%.3f" %f1)


print("*****")
```

B. SCREENSHOTS



Train and test dataset

C. PLAGARISM REPORT

