Analyzing Feature Selection Techniques for Machine Learning Based Anomaly Detection in IOT System.

CAPSTONE PROJECT REPORT

BY

KRISHNA MURARI B(AP18110010533)

HEMANTH PHANI SAI M(AP18110010535)

AKHIL K(AP18110010588)

ABSTRACT:

As one of the greatest risks in online services, Distributed Denial of Service (DDoS) remains prevalent. In order to slow the process of the client's entrance, attackers can use a simple and efficient DDoS attack method. To recognize the DDoS attack, ML calculations are utilized. The administered machine learning calculations like k-nearest neighbors(k-NN), logistic regression, Naive Bayes, Decision tree, and random forest are utilized for recognition and alleviation of attack. By using these five stages data gathering, Pre-processing, feature engineering, data splitting, and feature selection to find the best features in the given datasets (CICIDS-2017, NSL-KDD) with better accuracy. We believe that comparing Machine Learning algorithms in terms of performance & relevance will always be necessary to get good accuracy from the given datasets. It requires the most effective model to distinguish malicious activities as quickly as expected and accurately. Various calculations show distinctive conduct dependent on the selected features. The presentation of DDOS attack detection is looked at and the best algorithm is proposed. The paper describes how to utilize feature importance, correlation matrix with heatmap, Principal Component Analysis (PCA), Linear Discriminant Analysis, and the Roulette wheel selection method is proposed to select important features of the original attack data with the genetic algorithm.

INTRODUCTION:

DDoS (Distributed Denial of Service) is a type of Denial-of-Service Attack. The DDoS detection model and defensive system software in this case are based on extensive learning in the environment. Patterns in network traffic can be used to train models, which can then be used to track network assault activities [1]. The results show that the model outperforms traditional machine learning methods significantly.

Existing DDoS detection methods are mainly divided into threshold detection and classification-based detection. Threshold detection frequently distinguishes between DDoS attacks and regular events by dividing the threshold of Internet features (e.g., the source IP and destination IP). By training a large number of network properties or attributes, classification-based detection separated traffic into attacks and normal occurrences. It increases computing demand and decreases detection speed. Recently, Machine Learning (ML) and data processing techniques are playing an important role in the detection and therefore the classification of intrusion attacks. Several machine learning studies are conducted in several domains. Many issues can influence machine learning performances, such as feature selection methods, etc. However, there are not any related works that perform a scientific analysis of those machine-learning techniques. Conducting a thorough assessment of the most recent CICIDS2017 dataset is based on the accuracy which is performed using different Machine Learning Techniques.

Typically, DDoS attacks are initiated with a botnet that is controlled by the Attacker. The botnet is usually made up of compromised machines. The exact nature of this attack is unpredictable. For so long Machine Learning (ML) methods have been a component of the scientific area of exploration. When it comes to prediction issues, the most used classification-based Machine Learning algorithms are Logistic Regression, Decision Tree, Random Forest, KNN, and Naive Bayes. The purpose of this research is to better understand how each of these Machine Learning algorithms works to find out the better accuracy among the features.

In this research paper, we use the correlation-based feature selection (CFS) which is a filter method that selects the most effective feature subset according to some evaluation

function where features are assumed to be conditionally independent. We use feature significance to illustrate how important a feature is for a taxonomic performance of a model.

the main idea of improving detection efficiency is a way to effectively reduce the dimension of features without reducing the flexibility of information expression to boost the efficiency of knowledge analysis, which consists of two main steps: feature selection and feature extraction. In addition, although feature selection methods are widely used and achieve good results, the redundancy caused by the correlation between features has not been taken into consideration, so feature extraction methods have to be introduced. A more efficient method called Principal Component Analysis (PCA) can reduce measurements by eliminating correlation. Linear Discriminant Analysis (LDA) is a common technology for dimensional reduction issues as a pre-processing step for machine learning and pattern division operations

An evolutionary algorithm is a problem-solving algorithm that is inspired by natural principles and emulates the behaviours of live creatures. Typically, evolutionary algorithms are developed to provide good approximate solutions to the issues that are difficult to address using other methods. This area encompasses a wide range of optimization issues. Finding a precise answer may be too computationally intensive, but sometimes a near-optimal solution can suffice. In some places like industries, hospitals, and militaries, evolutionary computation is becoming a standard technique for solving tough, real-world issues. The use of evolutionary algorithms will become more common as the speed of desktop computers increases. This study examines some of the practical benefits of employing evolutionary algorithms over traditional optimization or artificial intelligence methods. The processes' flexibility, as well as their ability to self-adapt the search for optimal solutions on the go, are two distinct advantages. Any implementation of an Evolutionary algorithm would necessitate the definition of a number of parameters, including population size, mutation rate, and maximum run time, as well as the design of selection, recombination, and mutation operations. Finding good options for these is a difficult task with little theoretical backing. In practice, researchers must rely on anecdotal tales from similar difficulties, as well as a lot of trial and error. Convergence is not guaranteed: There is no guarantee that the evolutionary algorithm will reach a global optimum. It's possible that it'll become caught in one of the local optima. This is why EAs cannot be used to solve real-time situations where the solution's correctness and validity cannot be compromised. The main task performed by evolutionary algorithms is optimization. The difference between traditional algorithms and evolutionary algorithms is that evolutionary algorithms are dynamic and thus they can evolve and can be efficiently used to represent frequently changed information.

The genetic algorithm is the mechanism that drives biological evolution, for addressing both limited and unconstrained optimization problems. A genetic algorithm selects individuals from the present population to be parents at each phase and uses them to generate the following generation's children. The GA maintains a population of chromosomes (solutions)

associated with fitness values [35]. Parents are selected to mate on the basis of their fitness, producing offspring via a reproductive plan. Consequently, highly fit solutions are given more opportunities to reproduce so that offspring inherit characteristics from each parent. Once an initial population is randomly generated, the algorithm evolves through operators:

- selection which equates to the survival of the fittest;
- crossover which represents mating between individuals;
- mutation which introduces random modifications [37].

Advantages:
•Genetic Algorithms do not necessitate the use of derivative data (There is every possibility that there may not be any information that we can rely on based on the problem that we choose to solve with these Genetic Algorithms).
•When compared to classic brute-force search methods, genetic algorithms are faster and more efficient.
•Genetic Algorithms have been shown to offer a wide range of parallel capabilities.
•Performs multi-objective optimization on both continuous and discrete functions.

Disadvantages:
•Genetic algorithms are not well adapted to basic issues with readily available derivative information.
•When employing Genetic Algorithms, the fitness value is evaluated across a number of generations, which can be a time-consuming process for a large number of problems.
•A Genetic algorithm may fail to converge to an optimal solution if it is not used properly.

USES
•DNA Analysis: They are used in DNA analysis to determine the structure of the DNA based on spectrometric data.
•Multimodal Optimization: They are utilized in multimodal optimization situations to provide numerous optimul solutions.
•Aviation Design: They're used to create parametric airplane designs. To produce superior designs, the aircraft's parameters are updated and upgraded.
•Economics: In economics, they are used to define numerous models such as game theory, cobweb model, asset pricing, and schedule optimization

DATASET:

Intrusion Detection Evaluation Dataset (CICIDS2017):

We have collected the DDOS dataset named CICIDS-2017 from the UNB website. this study uses Machine Learning CVE data consisting of eight (8) traffic monitoring sessions; each is in the form of a comma separated value (CSV) file. CICIDS-2017 has more complex types of attacks. Since the inception of CICIDS2017 dataset, the dataset started attracting researchers for analysis and development of new models and algorithms [42,43,44]. According to the author [45] of CICIDS2017, the dataset spanned over eight different files containing five days of normal and attacks traffic data of Canadian Institute of Cybersecurity. A short description of all those files are present-ed in Table 1.

It can be seen from Table 1 that the dataset contains attack information as five days traffic data. Thursday working hours afternoon and Friday data are well suited for binary classification. Similarly, Tuesday, Wednesday and Thursday morning data are best for designing multiclass detection model. However, it should be noted that a best detection model should be able to detect attacks of any type.

The CICIDS2017 dataset closely simulates real-world network data (PCAPs) and uses CICFlowmeter-V3.0 to extract 78 features and 79 labels. It also provides the results of a network traffic analysis using CIC Flow Meter, which comprises labelled flows based on the time stamp, source and destination IP addresses, source and destination ports, protocols, and attack vectors (CSV files). This dataset includes the abstract characteristic attitudes of 25 users according to the HTTP, HTTPS, FTP, SSH, and email protocols as shown in Table 2.

TABLE 1 CICIDS2017 Dataset Summary

File Name	Type of Traffic	Number of Record
Monday-WorkingHours.pcap_ISCX.csv	Benign	529,918
Tuesday-WorkingHours.pcap_ISCX.csv	Benign	432,074
	SSH-Patator	5,897
	FTP-Patator	7,938
Wednesday-WorkingHours.pcap_ISCX.csv	Benign	440,031
	DoS Hulk	231,073
	DoS GoldenEye	10,293
	DoS Slowloris	5,796
	DoS Slowhttptest	5,499
	Heartbleed	11
Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv	Benign	168,186
	Web Attack-Brute Force	1,507
	Web Attack-Sql Injection	21
	Web Attack-XSS	652
Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv	Benign	288,566
	Infiltration	36
Friday-WorkingHours-Morning.pcap_ISCX.csv	Benign	189,067
	Bot	1,966
Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv	Benign	127,537
	Portscan	158,930
Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv	Benign	97,718
	DdoS	128,027
Total Instance/ Record		**2,830,743**

TABLE 2 Description of files containing CICIDS2017 dataset

Name of Files	Class Found
Monday-Hours.pcap_ISCX.csv	Benign (Normal human activities)
Tuesday-Hours.pcap_ISCX.csv	Benign, FTP-Patator,SSH Patator
Wednesday-.pcap_ISCX.csv	Benign, DoS GoldenEye, DoSHulk, DoS lowhttptest, DoS slow loris, Heartbleed
Thursday-WebAttacks.pcap_ ISCX.csv	Benign, Brute Force, SQL Injection, XSS.
Thursday-Infilteration.pcap_.csv	Benign, Infiltration
Friday-pcap_ISCX.csv	Benign, Bot
Friday-PortScan.pcap_ISCX.csv	Benign, PortScan
Friday- DDos.pcap_ISCX. csv	Benign, DDoS

TABLE 3 Class wise instance occurrence of CICIDS2017 dataset

Class Labels	Number of instances
BENIGN	2359087
DoS Hulk	231072
PortScan	158930
DDoS	41835
DoS GoldenEye	10293
FTP-Patator	7938
SSH-Patator	5897
DoS slowloris	5796
DoS Slowhttptest	5499
Bot	1966
Web Attack - Brute Force	1507
Infiltration	36
Web Attack - XSS	652
Web Attack - Sql injection	21
Heartbleed	11

TABLE 4 Overall characteristics of CICIDS2017 dataset

Dataset Name	CICIDS2017
Dataset type	Multi class
Year of release	2017
Total number of distinct instances	2830540
Number of features	83
Number of distinct classes	15

Data are captured across different periods. Based on the 2016 McAfee Report, the attacks in this dataset are classified into brute force FTP, brute force SSH, DoS, heartbleed, web, infiltration, botnet, and DDoS attacks, which are not found in any of the previously mentioned datasets [38]. CICIDS2017 achieves an abstract characteristics profiling of human interactions by using the B-Profile system and applies the Alpha profile to simulate various multi-stage attack scenarios. The main features of this dataset are distinguished from those of others in terms of their realistic and reliable benchmarking. The benchmarking applies 11 criteria, including complete traffic and available protocols, to ensure the reliability of the evaluation [39].

NSL-KDD Dataset:

We have collected the DDOS dataset named NSL-KDD dataset from UNB website. The raw dataset contains total attacks in the training set is 4.8M and testing set has 311K attack. We transformed raw dataset to much cleaner dataset by adding the column names.
This is a refined version of KDD'99 dataset where duplicate records are removed from both training and test data. It suggested solving the inherent problem of the kdd'99 dataset.
The number of records is also reduced (125,973 training records and 22,544 test records) while keeping the same number of features [40]. moreover, the range of facts in the NSL-KDD education and test units is reasonable. This gain makes it low-priced to run the experiments at the entire set without the want to randomly pick out a small element.
therefore, evaluation results of different research paintings will be regular and similar.
Intrusion data set Our dataset is the NSL-KDD (http://iscx.ca/NSLKDD/) which is suggested to solve some of the problems in the original KDD99 dataset [40]. One of the most important deficiencies in the KDD data set is the huge number of redundant records, which causes the learning algorithms to be biased towards the frequent records.

TABLE 5 List of NSL-KDD Dataset Files and Their Description

S.NO	Name of the file	Description
1	KDDTrain+.ARFF	The full NSL-KDD train set with binary labels in ARFF format
2	KDDTrain+.TXT	The full NSL-KDD train set including attack-type labels and difficulty level in CSV format
3	KDDTrain+_20Percent.ARFF	A 20% subset of the KDDTrain+.arff file
4	KDDTrain+_20Percent.TXT	A 20% subset of the KDDTrain+.txt file
5	KDDTest+.ARFF	The full NSL-KDD test set with binary labels in ARFF format
6	KDDTest+.TXT	The full NSL-KDD test set including attack-type labels and difficulty level in CSV format
7	KDDTest-21.ARFF	A subset of the KDDTest+.arff file which does not include records with difficulty level of 21 out of 21
8	KDDTest-21.TXT	A subset of the KDDTest+.txt file which does not include records with difficulty level of 21 out of 21

TABLE 6 Attribute Value Type (NSL-KDD)

Type	Features
Nominal	Protocol_type(2), Service(3), Flag(4), attack(42)
Binary	Land(7), logged_in(12), root_shell(14), su_attempted(15), is_host_login(21), is_guest_login(22)
Numeric	Duration(1), src_bytes(5), dst_bytes(6), wrong_fragment(8), urgent(9), hot(10), num_failed_logins(11), num_compromised(13), num_root(16), num_file_creations(17), num_shells(18), num_access_files(19), num_outbound_cmds(20), count(23) srv_count(24), serror_rate(25), srv_serror_rate(26), rerror_rate(27), srv_rerror_rate(28), same_srv_rate(29) diff_srv_rate(30), srv_diff_host_rate(31), dst_host_count(32), dst_host_srv_count(33), dst_host_same_srv_rate(34), dst_host_diff_srv_rate(35), dst_host_same_src_port_rate(36), dst_host_srv_diff_host_rate(37), dst_host_serror_rate(38), dst_host_srv_serror_rate(39), dst_host_rerror_rate(40), dst_host_srv_rerror_rate(41), level(43)

TABLE 7 Details of Normal and Attack Data in Different Types of NSL-KDD Dataset

Data set Type	Total No.of					
	Records	Normal class	DoS class	Probe class	U2R Class	R2L Class
KDD Train+ 20%	25192	13449	923	2289	11	209
		53.39%	36.65%	9.09%	0.04%	0.83%
KDD Train+	125973	67343	45927	11656	52	995
		53.46%	36.46	9.25%	0.04%	0.79%
KDD Test+	22544	9711	745	2421	200	2754
		43.08%	33.08%	10.74%	0.89%	12.22%

RELATED WORKS:

Punitha and mala proposed a flow-based intrusion detection system with flow features instead of request level parameters. The HTTP requests are differentiated from the normal DDoS attacks with flow features such as count of getting requests, GET request type and Service time. The low application-layer attacks are detected successfully with the Support Vector Machine (SVM) model and this model identifies request flood attacks only [2]. ML techniques are based on supervised, unsupervised, semi-supervised, and reinforcement learning. These techniques are applied to detect smart attacks and establish robust defensive policies. Supervised machine learning is the most common learning method in machine learning where the output is classified based on input using a trained data set. This method is also used in DDoS attack detection [3]. Applications in the Internet of Things are becoming pervasive in many domains around the world. However, this leads to many security threats. The proposed method allows the retrieval of missing features as well as feature reconstruction in case of incomplete data. The NSL-KDD dataset was used for the experiments considering features, achieving 98.27% of accuracy for 4-class detection, which is improved by increasing the number of fog nodes [4]. The machine learning method is based on an Intrusion detection system for securing IoT networks are discussed. The machine learning detection process is evolving with great results and considerations in the security field of IoT. The proposed IDS is responsible for DOS, data type probing, malicious control, malicious operation, scan, spying, and wrong setup attack detection occurring at various IoT sites. Among all the supervised algorithms used Random Forest gave the highest accuracy of 99.4% for all attack type detection [5]. Another example of anomaly detection is the rare-event detection system, which uses unsupervised learning on the IoT edge. The use of machine learning (ML) algorithms provides various solutions for edge devices, but they are restricted by computational capabilities. It is important to measure the accuracy of a dataset large enough to confirm the obtained results as valid and to select an optimal choice of hardware. This leads to new solutions where the use of machine learning algorithms or suitable pre-processing of raw data could be implemented [6]. Several classic machine learning methods such as one-class support vector machine (SVM) and principal component analysis (PCA) are served as out-of-box modules by the Azure cloud platform to accelerate the development of general-purpose anomaly detection solutions. The study applying machine learning-based anomaly detection approaches to vertical plant wall systems for indoor climate control is significant to the building and environment research community [7]. This may possess a significant segment of the organization’s transmission capacity of the casualty cloud foundations or devour a large part of the worker's time. Accordingly, in this work, we planned a DDoS location framework dependent on the C.4.5 calculation to relieve the DDoS danger [8]. The paper using the blend of the neural organization and the help vector machine presents the identification and the characterization strategy for the DDOS assaults in the media transmission network [9]. To get a higher True Negative Rate (TNR), exactness, and accuracy and to ensure the heartiness, dependability, and comprehensiveness of the recognition framework, in this paper, we propose a DDoS assault identification technique dependent on crossover heterogeneous multi classifier troupe learning and plan a heuristic discovery calculation dependent on Singular Value Decomposition (SVD) to develop our location framework [10]. We thought about Deep Neural Networks (DNN) for recognizing the assaults in IoT. An astute interruption discovery framework must be constructed if a compelling information set is accessible [11]. The DDoS assault was performed utilizing ping of death strategy and identified utilizing AI procedure by utilizing WEKA apparatus [12]. The distinctive AI calculations received for achieving the undertaking are Naive Bayes, K-Nearest neighbor (KNN), and Support vector machine (SVM) to recognize the strange conduct of the information traffic. These three calculations are contrasted agreeing with their exhibitions and

KNN is discovered to be the reasonable one over the other two [14]. Adetunmbi researched the relevance of the features using the KDD dataset in 2010. It presented there were some features that have no relationship to the class. It showed that removing some features was reasonable [29].The existing types of botnet-based DDoS attacks and the damage caused by DDoS attacks were detailed and analyzed by Alomari [30]. The knowledge illustrated in this paper helped the following researchers defend against such attacks efficiently. Revathi and Malathi conducted a detailed study on detecting attacks using different classification methods [31]. It pointed out that reducing features before classification was important. The result of NSL-KDD was better when using feature selection. In 2014, Aditya Harbola used sorting and greedy search methods to filter the NSL-KDD data. By using this method, 20 important attributes were selected, and the amount of data was reduced to half of the original. And a high detection rate has been obtained by using the KNN classification detection Algorithm [32]. Three different classification methods (J48, SVM, Naive Bayes) were performed on the NSL-KDD dataset by Dhanabal [33]. The result showed that the J48 classifier got a higher detection rate in the study. The prevention method for the attack in a cloud environment was introduced by Gupta [13]. The overview of DDoS attacks in the cloud computing environment presented the existing measure for DDoS attacks and provided the future scope of DDoS detection.

Background:

In Russia, Russian online media businesses were the most targeted industries in Q1. The Internet business was the second most targeted, followed by Cryptocurrency and Retail. DDoS attacks at random In January 2022, more than 17% of respondents who had been under assault reported being targeted by random DDoS attacks or received a threat in advance. We saw a record-breaking number of reported ransom DDoS assaults in the fourth quarter of 2021. (one out of every five customers). Looking at Q1 statistics, we can observe that the telecoms industry was the most targeted in terms of attack packets and attack bytes launched against Cloudflare clients. Looking at Q1 data, we can observe that the United States received the greatest percentage of DDoS attack traffic - more than 10% of total attack packets and almost 8% of all attack bytes. Most attacks last less than an hour, emphasizing the importance of automated always-on DDoS mitigation technologies. We already published a breakdown of 'attacks within an hour' and wider periods in earlier publications. However, in the majority of cases, over 90% of assaults last less than an hour. Companies should utilize automated, always-on DDoS security systems that monitor traffic and apply real-time fingerprinting quickly enough to prevent short-lived attacks.

Every day, the Fodcha DDoS botnet hits roughly 100 victims. Every day, a rapidly expanding botnet entraps routers, DVRs, and waiters across the Internet to target more than 100 victims in distributed denial-of-service (DDoS) attacks. " The global infection appears to be significant, with over diurnally active bots (IPs) in China and more than 100 DDoS victims being targeted every day."

Despite a large number of traditional recognition solutions currently available, DDoS attacks still increase with frequency and intensity. Many defence methods are conducted to detect and reduce the DoS attacks and these methods are based on the collection of flow characteristics received. A threat organization that targets crypto mining and distributed denial-of-service (DDoS) assaults has been connected to Enemy Bot, a new botnet found in March 2022 enslaving routers and Internet of Things (IoT) devices. The botnet has been linked to various botnets such as Simps, Ryuk (not to be confused with the ransomware of the same name), and Samael, and has a history of attacking cloud infrastructure to carry out crypto mining and DDoS activities [34].

PROPOSED IDEA:

To find DDoS attacks, Machine learning models supported with feature selection strategies and measures are utilized during this study. Machine learning is also a technique that pulls implications from existing knowledge by using mathematical and applied mathematics strategies and makes predictions regarding the unknown with these implications. Within the literature, the diffusion of machine learning models is projected. The taxonomy of models is usually summarized with kernel-based, distance-based, neural network-based, and probability-based characteristics. The continuous analysis suggests that there is no universal best model for all classification tasks. The literature suggests that logistic regression, naive Bayes, KNN (K Nearest Neighbours), Decision tree, and random forest models have shown smart performance for solutions to classification issues. During this study, as candidates of their classification cluster, the performance rate for detecting DDoS attacks from logistic regression, naive Bayes, KNN (K Nearest Neighbours), Decision tree, and random forest models were investigated.

We used a genetic algorithm to find the best solution. We can select features from the dataset by using different selection techniques. The metric-based classifier is tuned with Sklearn-genetic-opt. Param_grid is similar to scikit-learn, however, we have to specify what kind of data to use to sample parameters with Sklearn-genetic-opt. Scikit-learn classifiers or regressions must be used for the estimator, cv represents the number of splits in the cross-validation or a cross-validation generator, and scoring represents the measure chosen for optimization. It must be one of the sklearn metrics that are compatible with the estimator. By default, when we fit our model and use it in our test model, it will automatically select the most suitable set of hyperparameters from that model. During the execution of an algorithm, the accuracy that we choose is achieved in every generation. With the set of hyperparameters we used, we obtained accuracy in the parameters. The algorithm started with parameters around generation 0 which were generated randomly by hyperparameters. While the algorithm uses evolutionary strategies to choose the new set of hyperparameters, accuracy increases. The algorithm selects the best combinations of four hyperparameters from a wide range of parameters. The accuracy of the genetic algorithm can be predicted using logistic regression and decision trees. we used roulette wheel selection to find the best features in the dataset. Then we calculate the accuracy of the selected features.

Feature Selection Techniques:

Feature Importance:

Feature importance refers to techniques that assign a score to input options supported however helpful they're at predicting a target variable [15]. Feature importance scores can be calculated for problems that involve predicting a numerical value, called regression, and those problems that involve predicting a class label, called classification. The role of feature importance during a predictive modeling drawback. You will get the feature importance of each feature of your dataset by using the feature importance property of the model. It offers you a score for each feature of your knowledge. The upper score lots of significant or relevant is that the feature towards your output variable.

The scores are useful and can be used in a range of situations in a predictive modeling problem, such as:

1.Better understanding the data.

2.Better understanding a model.
3.Reducing the number of input features.
Feature importance scores can provide insight into the dataset. The relative scores can highlight which features may be most relevant to the target, and the converse, which features are the least relevant. This may be interpreted by a domain expert and could be used as the basis for gathering more or different data.

Correlation Matrix:

A matrix may be a table showing correlation coefficients between variables. Every cell within the table shows the correlation between 2 variables. A matrix is employed to summarize knowledge, as associate input into a lot of advanced analysis, and as a diagnostic for advanced analysis, Produce your matrix. Statisticians and knowledge analysts live with the correlation of 2 numerical variables to search out insight into their relationships. On a dataset with several attributes, the set of correlation values between pairs of its attributes type a matrix that is a termed matrix.

There square measures have many strategies for shrewd a correlation rate. The foremost standard one is the Pearson coefficient of correlation. Yet, it ought to be detected that it measures the sole linear relationship between 2 variables. In alternative words, it is going to not be ready to reveal a nonlinear relationship. The worth of Pearson correlation ranges from -1 to +1, wherever +/-1 describes an ideal positive/negative correlation and zero suggests that no correlation. The matrix may be an asymmetrical matrix with all diagonal parts adequate to +1. We might prefer to emphasize that a matrix solely provides insight to a knowledgeable man of science regarding correlation, and it's NOT a reliable tool to check indeed. The correlation values ought to be correctly taken by associate knowledgeable to avoid nonsense statements.

Correlation Matrix with Heatmap:

A correlation heatmap may be a graphical illustration of a matrix representing the correlation between completely different variables. Correlation states however the options are measured associated with one another or the target variable. The correlation will be positive or negative. The rate of correlation will take any values from -1 to 1. Correlation between 2 random variables or quantity knowledge doesn't essentially imply a constructive relationship. Heatmap makes it straightforward to spot the options square measure most associated with the target variable. We'll plot a heatmap related to options exploitation in the seaborn library.

Feature Extraction Techniques:

Principal Component Analysis:

Principal component analysis (PCA) may be a statistical method that uses an orthogonal transformation, that converts a group of related variables to a group of uncorrelated variables. PCA is that the most generally used tool in exploratory knowledge analysis and machine learning for predictive models.

Principal Component Analysis is an associate unsupervised learning algorithmic program that's used for spatial property reduction in machine learning. It's an applied mathematics method that converts the observation of related options into a group of linearly unrelated options with the assistance of orthogonal transformation. These new remodelled options measure refers to the principal elements. It is one of all the favoured tools that is used for exploratory knowledge analysis and predictive modelling. It is a way to draw robust patterns from the given dataset by reducing the variances. PCA typically tries to search out the lower-dimensional surface to

project the high-dimensional knowledge. PCA works by considering the variance of every attribute as a result of the high attribute shows the great split between the categories, and hence it reduces the spatial property. Some real-world applications of PCA are measure image process, moving picture recommendation systems, optimizing the ability allocation in numerous communication channels [16]. It is a feature extraction technique; therefore, it contains the necessary variables and drops the smallest number of vital variables.

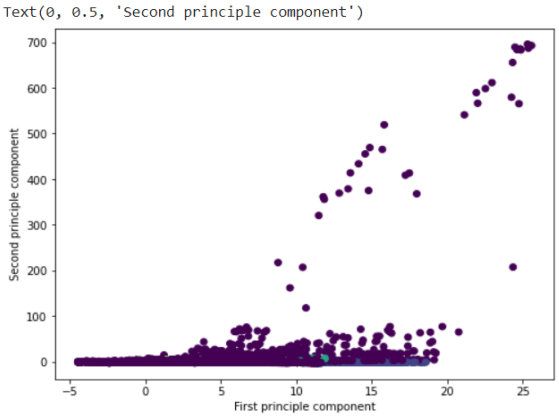

Fig. 1 The two-dimensional principal subspace for the CICIDS2017 data.

Linear Discriminant Analysis:

Linear discriminant analysis is also a supervised classification method that is used to create machine learning models. These models supported dimensionality reduction that are utilized within the application, like marketing predictive analysis and image recognition, amongst others. Linear Discriminant Analysis is also a dimensionality reduction technique. It is used as a pre-processing step in Machine Learning and applications of pattern classification [17]. The goal of LDA is to project the features in higher dimensional space into a lower-dimensional space to avoid the curse of dimensionality and also reduce resources and dimensional costs. The initial Linear Discriminant Analysis was described as a two-class technique. LDA is also a supervised classification technique that is considered a part of crafting competitive machine learning models. This category of dimensionality reduction is used in areas like image recognition and predictive analysis in marketing. The representation of LDA is pretty straightforward. The model consists of the statistical properties of your data that are calculated for each class. The identical properties are calculated over the multivariate within the case of multiple variables. Predictions are made by providing the statistical properties into the LDA equation [18].

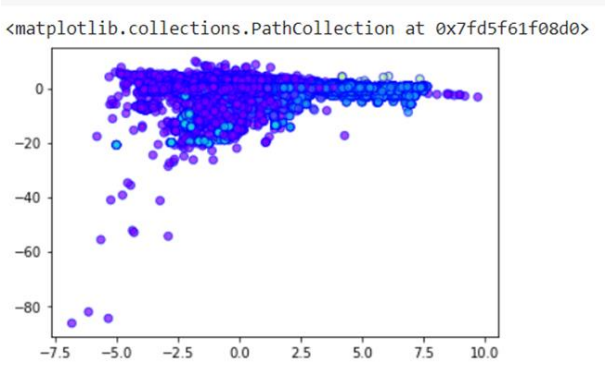

Fig. 2 Linear Discriminant Analysis for two categories

ROULETTE WHEEL SELECTION:

Roulette wheel, developed by Holland, is the first selection method. The probability Pi for each individual is defined by where Fi equals the fitness of individual i. The use of roulette wheel selection limits the genetic algorithm to maximization because the evaluation function must map the solutions to a fully ordered set of values on + ℜ [35]. Extensions, such as windowing and scaling, have been proposed to allow for minimization and negativity.
All of the individuals for the following generation are chosen using the roulette wheel method. In a genetic algorithm, it is a common selection strategy. Each individual's relative fitness (ratio of individual fitness to total fitness) is used to create a roulette wheel. the individual crossover probability is calculated by dividing the person's fitness by the sum of all population fitness.
Pi is the chance of each chromosome, which is the chromosomal frequency divided by the sum of all fitness on a roulette wheel. Assume the roulette wheel selection mechanism is similar to a pie chart. Each person has a fitness value, and the circle is the sum of all of them. As a result, your chances of finding a possible mate are determined by your overall fitness.
It is clear that a fitter individual has a greater pie on the wheel and therefore a greater chance of landing in front of the fixed point/pointer when the wheel is rotated. Therefore, the probability of choosing an individual depends directly on its fitness [36]. It has a low time complexity when implemented in parallel. There is no need for fitness grading or sorting.
A common selection approach assigns a probability of selection Pj to each individual j based on its fitness value. A series of N random numbers are generated and compared against the cumulative is selected and copied into the new population if Ci-1 < U(0, 1) ≤ Ci. Various methods exist to assign probabilities to individuals: roulette wheel, linear ranking, and geometric ranking.
the probability of an individual being selected as a parent for crossover is given by

$$P(i)=\frac{f(i)}{\sum_{i} f(i)}$$

$f(i)$: fitness of individual i
$P(i)$: probability that individual i is chosen

ML Algorithms:

Logistic Regression:

Logistic regression was used in the biological sciences in the early 20th century. It was then used in many social science applications. Logistic Regression is used when the dependent variable(target) is categorical [19].
For example,

- To predict whether an email is spam (1) or (0)
- Whether the tumor is malignant (1) or not (0)

Consider a scenario where we need to classify whether an email is a spam or not. If we use linear regression for this problem, there is a need for setting up a threshold based on which classification can be done.
it can be inferred that linear regression is not suitable for classification problems [20].

Naive Bayes:

Naive Bayes algorithm is a supervised learning algorithm, which is based on the Bayes theorem and used for solving classification problems [21]. This algorithm is an intuitive method that uses the probabilities of each attribute belonging to each class to make a prediction Naive Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building fast machine learning models that can make quick predictions. It is easy to build and particularly useful for very large datasets.

For example, your probability of getting a parking space is connected to the time of the day you park where you park and what conventions are you going at that time [22].

NB methods use a group of probabilistic classifiers that are created by implementing the Bayes theorem. These methods consider the naive conjectures of independence between each pair of features or attributes [41]. NB can be easily trained by using a supervised learning structure. In numerous realistic implementations, the maximal probability technique is applied to calculate the parameters of NB models. In other words, the NB model can function with the refusal of Bayesian probability or by using any Bayesian technique. Bayes theorem can be formulated as

$$p(A|B)=p(A|B)P(A)/p(B),$$

where A represents the active target attribute or dependent event, B denotes the active predictor attribute or prior event, P (A) represents the prior probability of A, P(A|B) represents the posterior probability of B, and P(B|A) represents the probability of B if hypothesis A holds true.

K Nearest Neighbours:

The KNN algorithm may be a simple, easily applicable, and supervised machine learning algorithm which will be used for solving both classification and regression problems [23]. When new data comes in, it determines the category of the new data by watching its nearest K neighbours. Manhattan, Makowski, and Euclidean distance functions are used for the space between two data. The Euclidean distance function was utilized in this study. The similarity between the sample to be classified and therefore the samples found within the classes were detected. When new data was encountered, the distance of this data in the training set was calculated individually by using the Euclidean function [24]. Then, the classification set was created by selecting the k-dataset from the smallest distance. The number of neighbouring KNN (k) is predicated on the worth of classification.

Decision Tree:

Decision Trees are a kind of Supervised Machine Learning where the information is continuously split in keeping with a specific parameter. A Decision Tree may be a Supervised learning technique that will be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and every leaf node represents the end result [25]. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions [26]. it is very fast and efficient compared to KNN and other classification algorithms. Entropy and Information Gain are the most commonly used Attribute Selection Measures.

Decision Tree works as follows:

a. Select the best attribute using Attribute Selection Measures (ASM) to split the records.
b. Make that attribute a decision node and breaks the dataset into smaller subsets.
c. It starts building the tree by repeating this process recursively for each child node until any one of the conditions matches:

1. All the tuples belong to the same attribute value.
2. There are no more remaining attributes.
3. There are no more instances.

Random forest:
Random forest is a supervised learning algorithm. The general plan of the bagging technique is the mix of learning models will increase the result. Put simply: random forest builds multiple decision trees and merges them to induce a lot of correct and stable prediction. One huge advantage of random forest is that it is typically used for each classification and regression issues. Let's examine the random forest in classification since classification is usually thought-about the building block of machine learning [27].

Performance Evaluation Metrics:

The confusion matrix consists of four values: True Positive, False Negative, False Positive, and True Negative.

a. **True Positive (TP):** It is the test result which detects the condition if the condition is present. The TP value in the confusion matrix is at cm [0][0].

b. **False Negative (FN):** It is the test result which does not detect the condition even if the condition is not present. The FN value in the confusion matrix is at cm [1][0].

c. **False Positive (FP):** It is the test result which detects the condition when the condition is absent. It is also called a False Alarm rate. The FP value in the confusion matrix is at cm [0][1].

d. **True Negative (TN):** It is the test result which does not detect the condition even if the condition is present. The TN value in the confusion matrix is at cm [1][1].

Performance can be measured using the

Accuracy: it refers to the ratio of correct predictions for both the true positives (TP) and true

negatives (TN) of attacks compared with the total number of tested cases.

Accuracy= TP+TN/ TP+TN+FP+FN

TABLE 8 Evaluation Parameters

Class	Predicted Positive	Predicted Negative
Class of Attack (Positive)	Predicted Attack as attacks (TP)	Predicted Attacks as Normal (FN)
Class of Normal (Negative)	Predicted Normal as Attacks (FP)	Predicted normal (TN) as Normal

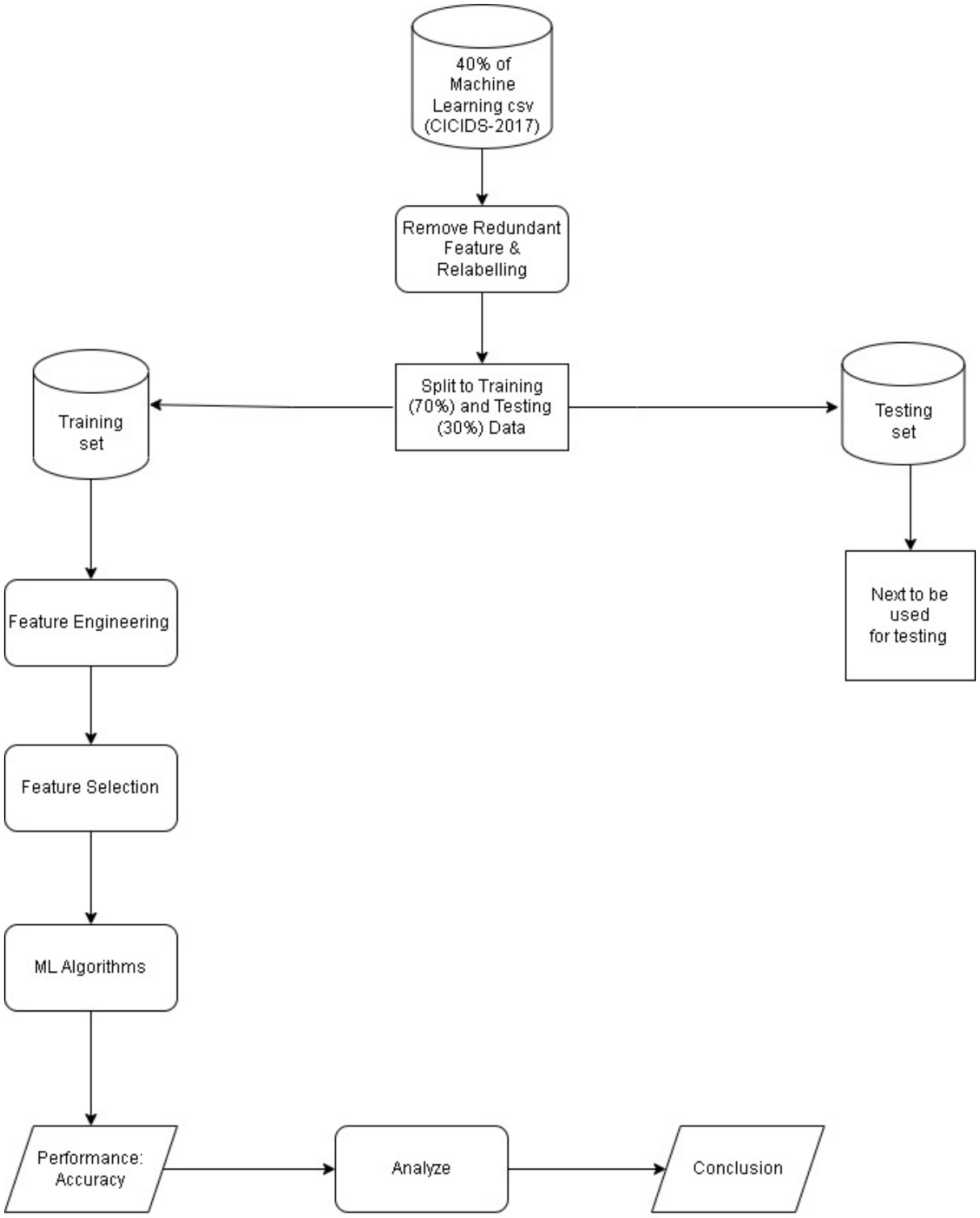

Fig. 3 The Process of Evaluation-Based Algorithms

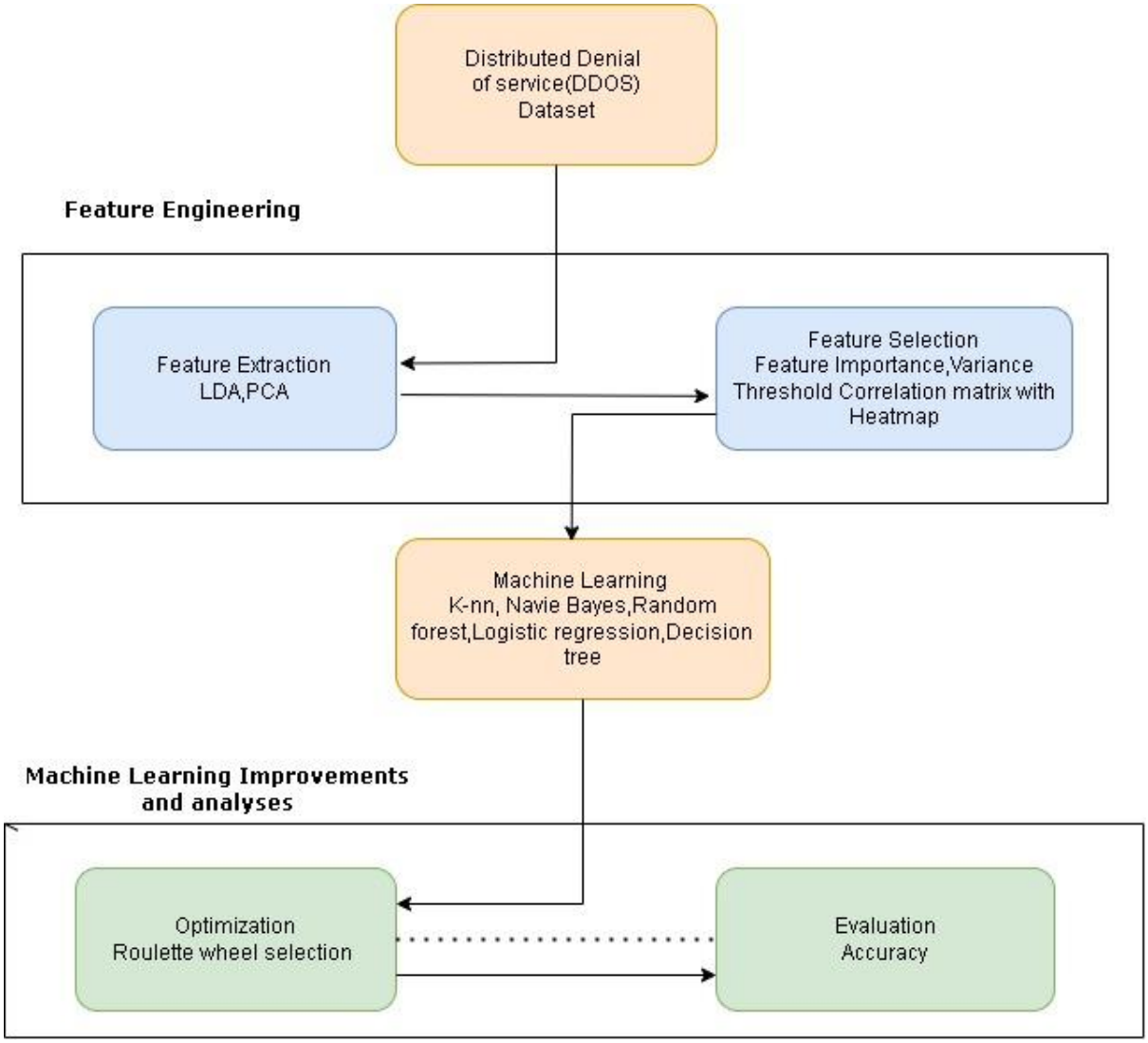

Fig. 4 Strategic-Level framework for DDoS attack detection

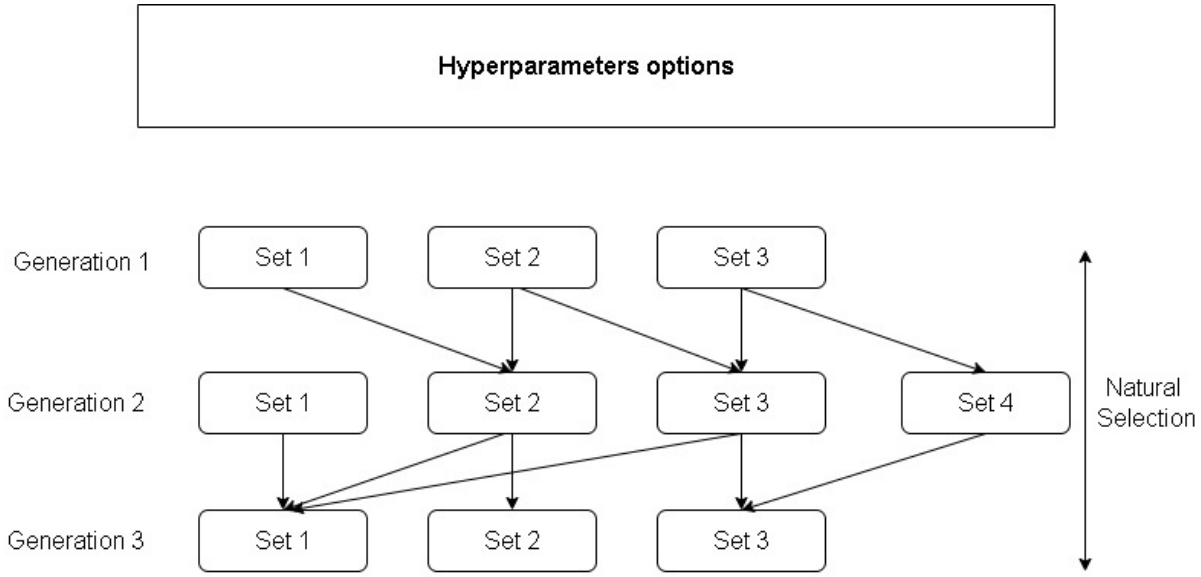

Fig. 5 Evolutionary Cross-Validation

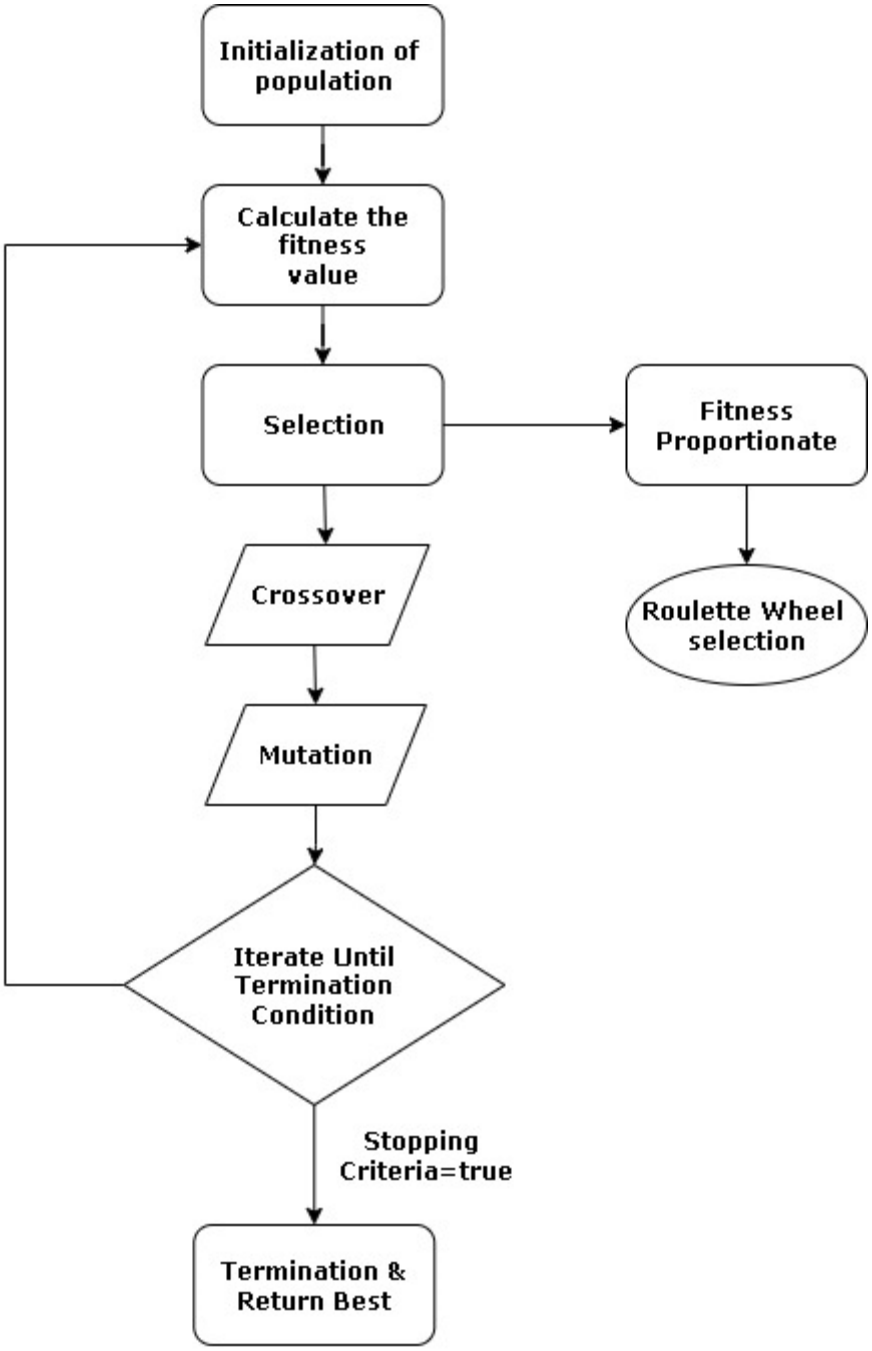

Fig. 6 Steps involved in genetic algorithm

Evaluation of result:

Histogram groups the information in bins is the fastest way to get a concept about the distribution of every attribute within the dataset. The subsequent are the number of characteristics of histograms.

It provides us a count of the quantity of observations in each bin created for a visualization. From the shape of the bin, we are able to easily observe the distribution i.e., whether it's Gaussian, skewed, or exponential.

Histograms also help us to work out possible outliers. We are using histogram for the given dataset in Fig 7.

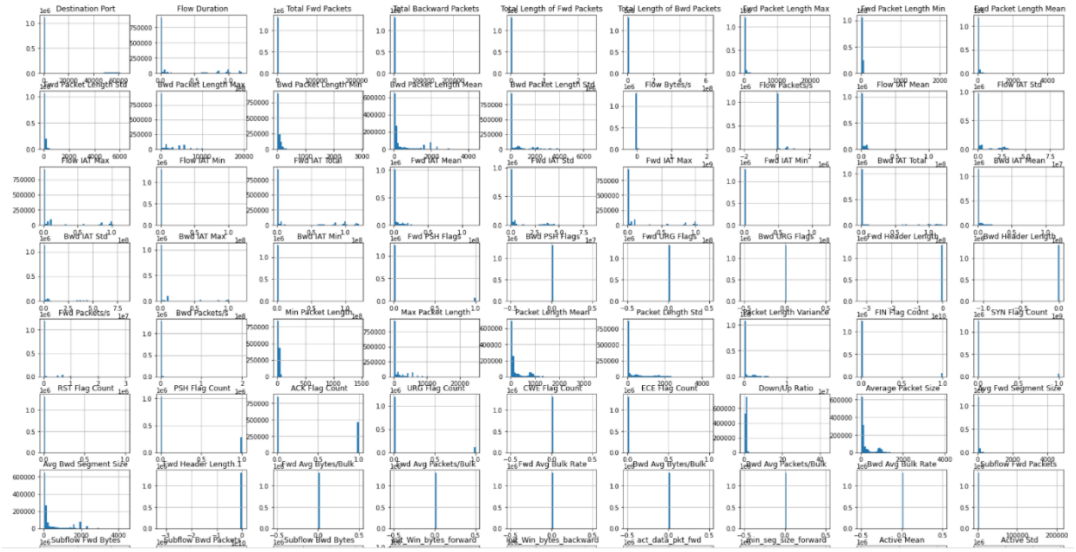

Fig. 7 Count of the quantity of observations in each bin created for CICIDS2017 dataset seaborn.countplot() technique is used to point out the counts of observations in every categorical bin using bars. For Fig 8 we are finding how many numbers of different attacks are there in the Attacks column.

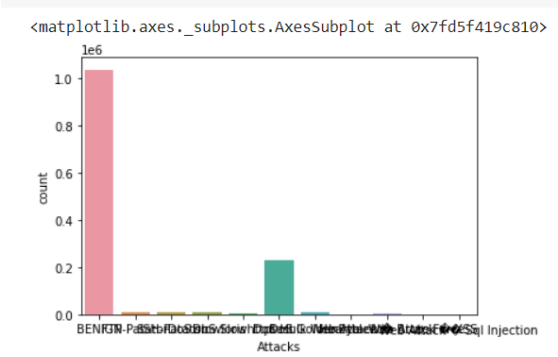

Fig. 8 Count of observations in every categorical bin using bars

Seaborn Distplot represents the overall distribution of continuous data variables. The distplot represents the univariate distribution of data i.e. data distribution of a variable against the density distribution. The seaborn.distplot() function accepts the data variable as associate argument and returns the plot with the density distribution [28]. We have used displot function for Attacks_enc as associate argument and returns the plot with the density distribution in Fig 9.

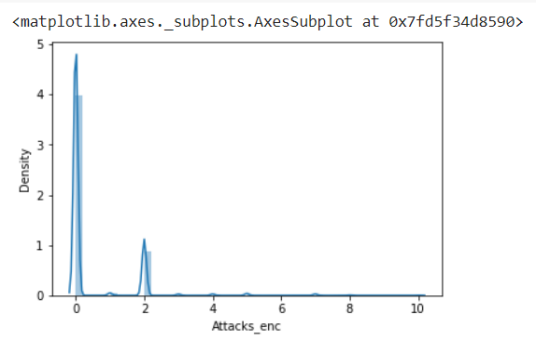

Fig. 9 Data distribution of a variable against the density distribution

Feature Selection is tough in the starting, then figure out to eliminate the non-numerical columns and applied a confusion matrix to find the best features to train the model.

Fig. 10 We took all the features of attack value is more than 0.85 then we got the features.

Feature importance is the associate built-in category that comes with a Tree-Based Classifier that will extract the best ten options from the CICIDS2017 dataset.

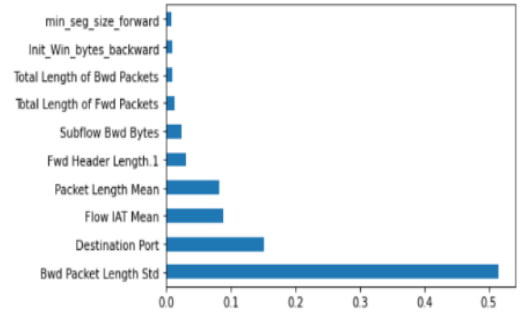

Fig. 11 We use Decision Tree Classifier to get top 10 features

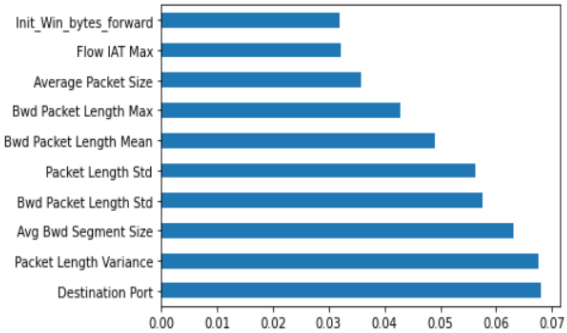

Fig. 12 We use Random Forest Classifier to get top 10 features

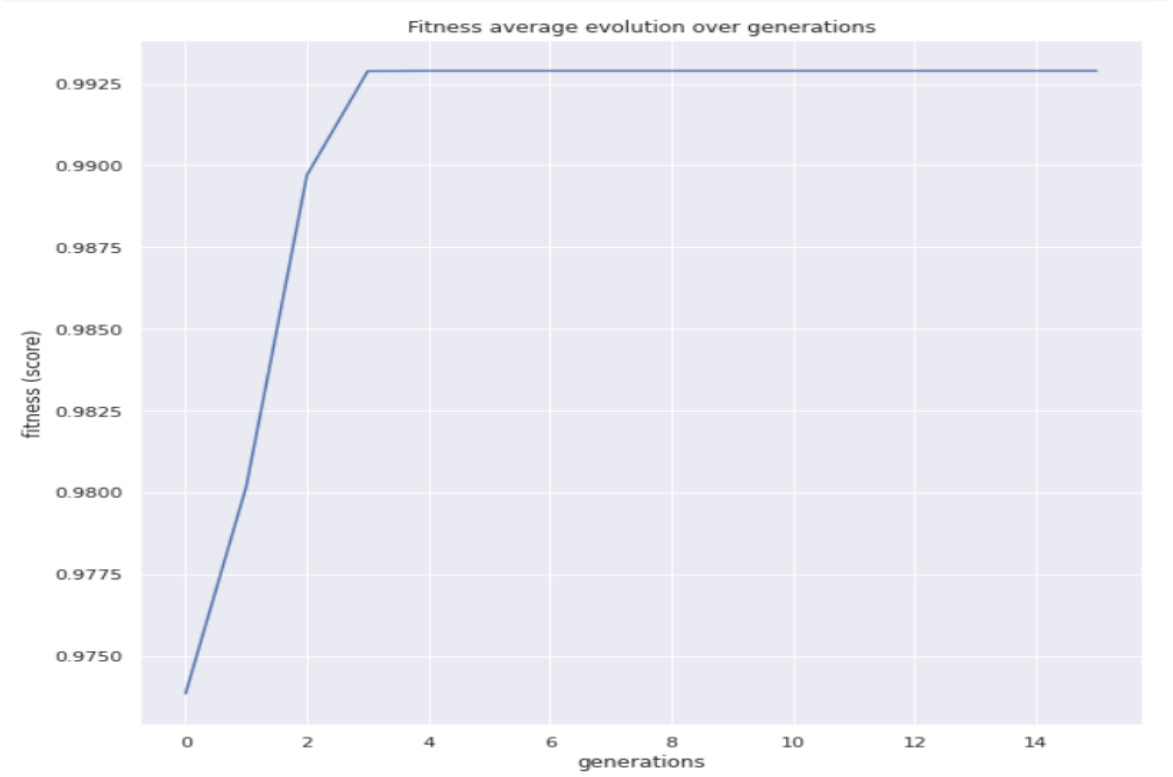

Fig. 13 Fitness Evolution Plotting of CICIDS2017 Dataset

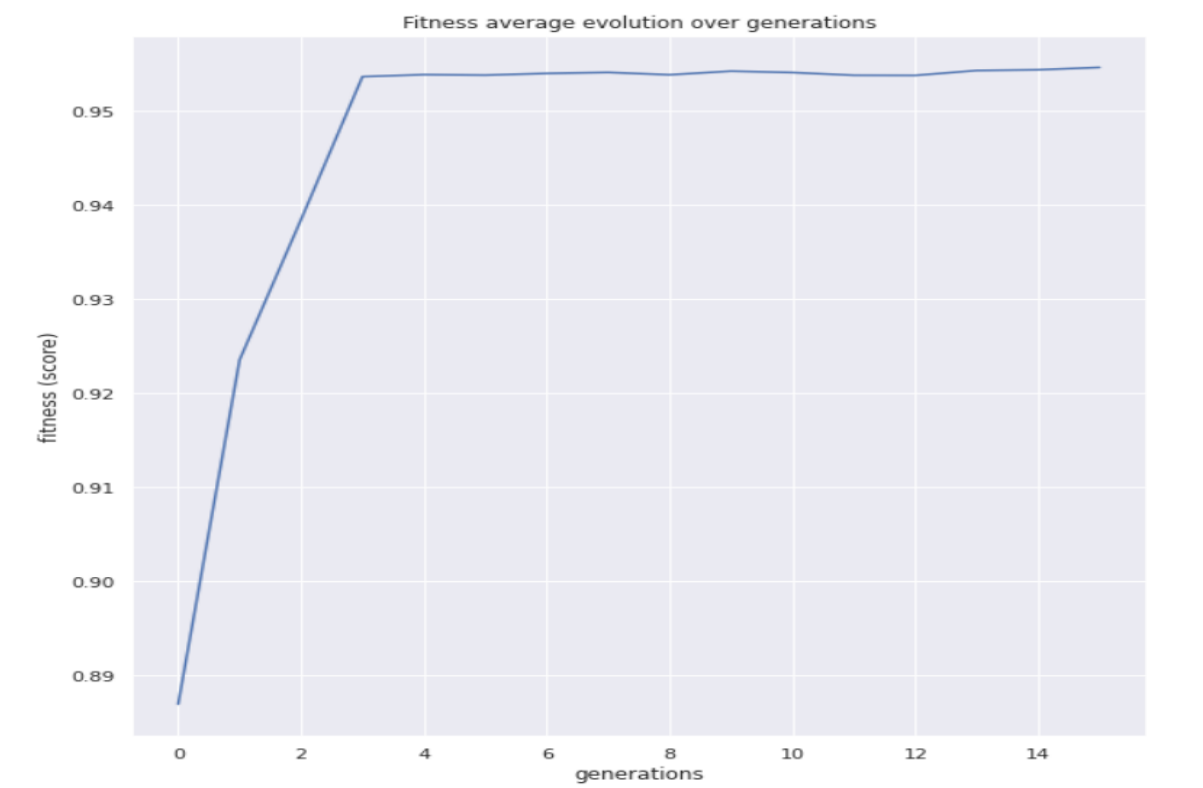

Fig. 14 Fitness Evolution Plotting of NSL-KDD Dataset

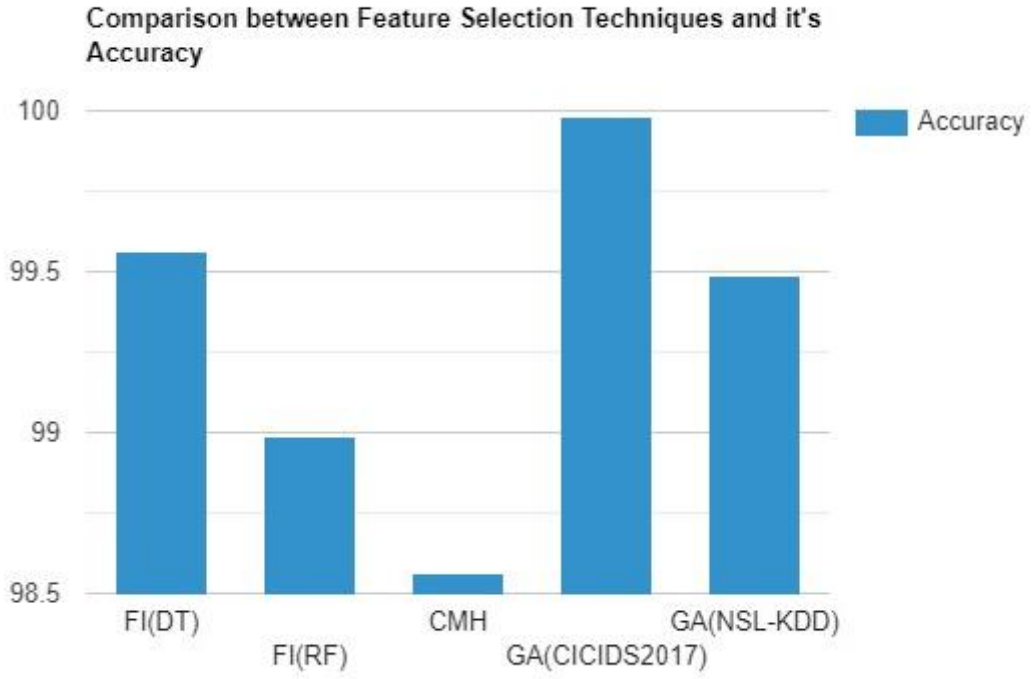

Fig. 15 Comparison between feature selection techniques and accuracy

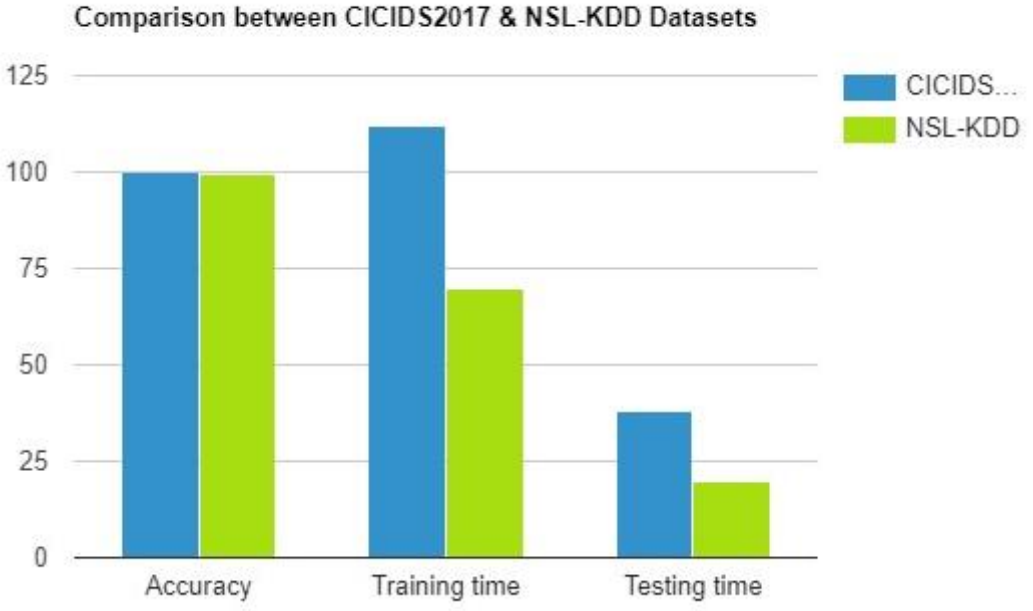

Fig. 16 Comparison between CICIDS2017 & NSL-KDD Datasets

TABLE 9 Accuracy of ML Algorithms of CICIDS2017 Dataset

Algorithms	Accuracy
Decision Tree	100%
Logistic Regression	98.79%

TABLE 10 Accuracy of ML Algorithms of NSL-KDD Dataset

Algorithms	Accuracy
Decision Tree	99.97%
Logistic Regression	86.41%

TABLE 11 Feature Selection Techniques with Accuracy of CICIDS2017 Dataset

Feature Selection	Algorithm	Accuracy
Principal Component Analysis(PCA)	Naïve Bayes	96.8%
	Logistic Regression	96.8%
Linear Discriminant Analysis(LDA)	Knn	99.74%
	Logistic Regression	97.35%

TABLE 12 Parameters assumed in evolutionary algorithm

Parameter	Value
Number of generations	15
Chromosome population Size	60
Crossover probability	0.6
Crossover type	Single
Mutation probability	0.05
Mutation type	Uniform
Evolutionary algorithm	eaMuPlusLambda

TABLE 13 Fitness Evolution Table of CICIDS2017 Dataset

Generation	Fitness(accuracy)
0	0.973849
1	0.98018
2	0.989705
3	0.992883
4	0.992891
5	0.992891
6	0.992891
7	0.992891
8	0.992891
9	0.992891
10	0.992891

TABLE 14 Fitness Evolution Table of NSL-KDD Dataset

Geneartion	Fitness(accuracy)
0	0.886952
1	0.923541
2	0.938454
3	0.953646
4	0.953857
5	0.953804
6	0.95399
7	0.954096
8	0.953844
9	0.954228
10	0.954082

TABLE 15 Features Selected By Roulette Wheel with accuracy of CICIDS2017 Dataset

Feature Selection	Algorithm	Feature Selected	Accuarcy
Feature importance	Decision Tree	Destination port Subflow Fwd Packets Fwd IAT Min Packet Length Variance Init_Win_bytes_backward Bwd Packets/s Fwd Packet Length Max Flow IAT Min Bwd IAT Total Bwd IAT std	99.98%
	Random Forest	Destination port Avg Fwd Segment Size Fwd Packet Length Std Packet Length Variance Fwd Packet Length Max Average Packet Size Packet Length Std Packet Length Mean Max Packet Length Flow IAT Min	98.34%
Roulette Wheel	Decision Tree	Destination Port Flow IAT Min Fwd IAT Total Fwd IAT Min Bwd URG Flags CWE Flag Count Fwd Header Length.1 Subflow Bwd Bytes Init_Win_bytes_forward Init_Win_Bytes_backward	99.88%

TABLE 16 Features Selected By Roulette Wheel with accuracy of NSL-KDD Dataset

Feature Selection	Algorithm	Feature Selected	Accuarcy
Roulette Wheel	Decision Tree	service flag src_bytes is_host_login same_srv_rate diff_srv_rate dst_host_count dst_host_srv_diff_host_rate dst_host_error_rate level	99.49%

TABLE 17 Comparision of different selection methods

Feature Selection Methods	Number Of Features	Features Selected	Accuarcy
Feature-Importance(Decision Tree)	10	28,26,20,7,38,68,43, 25,63,1	99.56%
Feature-Importance(Random Forest)	10	20,40,41,42,53,7,43, 10,54,1	98.99%
Correlation matrix with heatmap	35	74,53,54,55,27,29,30,13,14, 51,18,19,56,22,24,25,10,77, 78,40,41,42,43,45,65,66,64, 4,6,69,26,21,37,75,63	98.56%
GA-RouleteeWheel(CICIDS2	10	1,20,21,25,34,50,56,66,67,68	99.98%
GA-RouleteeWheel(NSL-KDD)	10	3,4,5,21,29,30,32,37,38,43	99.49%

CONCLUSION:

In this work, a strategy for including a choice is called Feature Selection for CICIDS2017. Intrusion Dataset is introduced and it was able to classify different types of DoS/DDoS attacks. An online solution for detecting DoS/DDoS attacks was introduced in this research. According to the testing results, when we use feature importance with the decision tree it is giving 99.98% accuracy. When we use feature importance with the random forest it is given 98.34%. A decision tree is giving better accuracy with feature importance. When we use principal component analysis with naive Bayes it is giving 96.8% accuracy. When we use principal component analysis with logistic regression it is giving 96.8%. Both Logistic regression and Naïve Bayes are giving same accuracy with principal component analysis. When we use linear discriminant analysis with logistic regression it is giving 97.35%. When we use linear discriminant analysis with KNN it is giving 99.74%. KNN is giving better accuracy with linear discriminant analysis. After experimenting on the CICIDS2017 and NSL-KDD datasets, the features were reduced from 79 to 10 and the amount of data was roughly reduced to 13% of the original in the CICIDS2017 dataset. the features were reduced from 43 to 10 and the amount of data was roughly reduced to 24% of the original IN NSL-KDD dataset. Both the efficiency and accuracy were improved with the proposed feature selection method. the detection rate was not simply increased with the number of features. It depends on whether the selected features were related to the event. the proposed detection method concentrates on improving the detection model by selecting valuable features. The time of computational resource of the detection part is saved because the cost of selection is increased.

FUTURE WORK:

This work gives the foundations for several future systems and more experiments may also be conducted to include the diversity of the machine learning algorithms, for instance, supervised, unsupervised, and semi-supervised models across multiple DDoS Attack related datasets. Feature selection is a largely open exploration area and we believe that hybrid methods of feature selection using various approaches of statistical tools and multi objectives such as random search or some additional algorithm can be the most effective way of feature selection for DDoS attack findings. Our Future work is to investigate ways for improving the discovery rate of various attacks. It'll include analyzing DDoS attacks based on service vulnerabilities like adding multiple-class classification and putting in place precautionary measures. Changes in Optimization Techniques like Ant Colony Optimization Algorithm, PSO (Particle swarm optimization), etc. make more experiments in genetic algorithms, Compare with different datasets like Caida, SDN, etc. Change the selection process and test with different data like changing the number of generations Thus, we hope to simplify the investigation process by joining researchers' efforts.

REFERENCES:

[1] Pawar, Mohan V., and J. Anuradha. "Network security and types of attacks in network." Procedia Computer Science 48 (2015): 503-506.

[2] Chandra, G. Ramesh, et al. "Flow based intrusion Detection system with Whale Optimization and Evolutionary Algorithms (EA) for Diversified Traffic Streams." (2021).

[3] Aamir, Muhammad, and Syed Mustafa Ali Zaidi. "Clustering based semi-supervised machine learning for DDoS attack classification." Journal of King Saud University-Computer and Information Sciences 33.4 (2021): 436-446.

[4] Aversano, Lerina, et al. "Effective Anomaly Detection Using Deep Learning in IoT Systems." Wireless Communications and Mobile Computing 2021 (2021).

[5] Tyagi, Himani, and Rajendra Kumar. "Attack and Anomaly Detection in IoT Networks Using Supervised Machine Learning Approaches." Rev. d'Intelligence Artif. 35.1 (2021): 11-21.

[6] Huč, Aleks, Jakob Šalej, and Mira Trebar. "Analysis of Machine Learning Algorithms for Anomaly Detection on Edge Devices." Sensors 21.14 (2021): 4946.

[7] Liu, Yu, et al. "Anomaly detection based on machine learning in IoT-based vertical plant wall for indoor climate control." Building and Environment 183 (2020): 107212.

[8] Zekri, Marwane, et al. "DDoS attack detection using machine learning techniques in cloud computing environments." 2017 3rd international conference of cloud computing technologies and applications (CloudTech). IEEE, 2017.

[9] Smys, S. "DDOS attack detection in telecommunication network using machine learning." Journal of Ubiquitous Computing and Communication Technologies (UCCT) 1.01 (2019): 33-44.

[10] Jia, Bin, et al. "A DDoS attack detection method based on hybrid heterogeneous multiclassifier ensemble learning." Journal of Electrical and Computer Engineering 2017 (2017).

[11] Choudhary, Sarika, and Nishtha Kesswani. "Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 datasets using deep learning in IoT." Procedia Computer Science 167 (2020): 1561-1573.

[12] Pande, Sagar, et al. "DDOS detection using machine learning technique." Recent Studies on Computational Intelligence. Springer, Singapore, 2021. 59-68.

[13] Gupta, B.B., Badve, O.P.: Taxonomy of dos and ddos attacks and desirable defense mechanism in a cloud computing environment. Neural Comput. Appl. 28(12), 1–28 (2017)

[14] Prakash, Aditya, and Rojalina Priyadarshini. "An intelligent software defined network controller for preventing distributed denial of service attack." 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT). IEEE, 2018.

[15] Brownlee, J. "How to Calculate Feature Importance With Python." Machine Learning (2020).

[16] "Principal Component Analysis - Javatpoint." Www.javatpoint.com, https://www.javatpoint.com/principal-component-analysis.

[17] "1.4 Dimensionality Reduction Method - Machine Learning Concepts." Machine Learning Concepts - How to Crack a Data Science Interview, 18 Oct. 2021.

[18] Sarkar, Priyankur, et al. "What Is Linear Discriminant Analysis (Lda)?" Knowledgehut, 30 Sept. 2019, https://www.knowledgehut.com/blog/data-science/linear-discriminant-analysis-for-machine-learning.

[19] Robson, Winston. "Beginner's Guide to Logistic Regression with Cuml." Medium, Dropout Analytics, 28 Sept. 2020, https://medium.com/dropout-analytics/beginners-guide-to-logistic-regression-with-cuml-5061086d8694.

[20] Raj, Ashwin. "The Perfect Recipe for Classification Using Logistic Regression." Medium, Towards Data Science, 5 Jan. 2021, https://towardsdatascience.com/the-perfect-recipe-for-classification-using-logistic-regression-f8648e267592.

[21] "Naive Bayes Classifier in Machine Learning - Javatpoint." Www.javatpoint.com, https://www.javatpoint.com/machine-learning-naive-bayes-classifier.

[22] “Bayes' Theorem Problems, Definition and Examples.” Statistics How To, 7 June 2021, https://www.statisticshowto.com/probability-and-statistics/probability-main-index/bayes-theorem-problems.

[23] Harrison, O. "Machine Learning Basics with the K-Nearest Neighbors Algorithm.[online] Towards Data Science." (2019).

[24] Polat, Huseyin, Onur Polat, and Aydin Cetin. "Detecting DDoS attacks in software-defined networks through feature selection methods and machine learning models." Sustainability 12.3 (2020): 1035.

[25] “Machine Learning Decision Tree Classification Algorithm - Javatpoint.” Www.javatpoint.com, https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm.

[26] Machine Learning Decision Tree Classification Algorithm, https://thedeveloperblog.com/machine/machine-learning-decision-tree-classification-algorithm.

[27] Donges, N. "A Complete Guide to the Random Forest Algorithm. 2019." Retrieved from URL https://builtin. com/data-science/random-forest-algorithm (2021).

[28] “Seaborn Distplot: A Comprehensive Guide.” JournalDev, 15 May 2020, https://www.journaldev.com/39993/seaborn-distplot.

[29] Olusola, A.A., Oladele, A.S., Abosede, D.O.: Analysis of kdd’99 intrusion detection dataset for selection of relevance features.Lecture Notes Eng. Comput. Sci. 2186(1), 1371–1379 (2010)

[30] Alomari, E., Manickam, S., Gupta, B.B., Karuppayah, S., Alfaris,R.: Botnet-based distributed denial of service (ddos) attacks onweb servers: classification and art. Int. J. Comput. Appl. 49(7),24–32 (2012)

[31] Revathi, S., Malathi, A.: A detailed analysis on nsl-kdd dataset using various machine learning techniques for intrusion detection.In: International Journal of Engineering Research and Technology(2013)

[32] Harbola, A., Harbola, J., Vaisla, K.S.: Improved intrusion detection in ddos applying feature selection using rank & score of attributes in kdd-99 data set. In: International Conference on Computational Intelligence and Communication Networks, pp. 840–845 (2014)

[33] Dhanabal, L., Shantharajah, S.P.: A study on nsl-kdd dataset for intrusion detection system based on classification algorithms. In: International Journal of Advanced Research in Computer and Communication Engineering, vol. 4 (2015)

[34] Forza, A. (2022, April 14). *New* EnemyBot ddos botnet borrows exploit code from Mirai and gafgyt - the hacker news - JN-66 data analytics. JN. Retrieved April 22, 2022, from https://www.jn66dataanalytics.com/news/new-enemybot-ddos-botnet-borrows-exploit-code-from-mirai-and-gafgyt-the-hacker-news

[35] Pencheva, Tania, Krassimir Atanassov, and Anthony Shannon. "Generalized net model of selection function choice in genetic algorithms." Recent Advances in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics 2 (2010): 193-201.

[36] Shyalika, C. (2019, March 2). Genetic algorithms -selection. Medium. Retrieved April 22, 2022, from https://medium.datadriveninvestor.com/genetic-algorithms-selection-5634cfc45d78

[37] Barekatain, Behrang, Shahrzad Dehghani, and Mohammad Pourzaferani. "An energy-aware routing protocol for wireless sensor networks based on new combination of genetic algorithm & k-means." Procedia Computer Science 72 (2015): 552-560.

[38] I. Sharafaldin, A. H. Lashkari and A. A. Ghorbani, A Detailed Analysis of the CICIDS2017 Data Set, Springer, 2019.

[39] Z. Pelletier and M. Abualkibash, "Evaluating the CIC IDS-2017 dataset using machine learning methods and creating multiple predictive models in the statistical computing language R", Science, vol. 5, no. 2, pp. 187-191, 2020.

[40] Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the KDD CUP 99 data set. In: Symposium on Computational Intelligence for Security and Defense Applications (CISDA), IEEE, pp. 1–6 (2009)

[41] D. D. Lewis, "Naive (Bayes) at forty: The independence assumption in information retrieval", Proc. ECML, pp. 16, 1998.

[42] Benjamin J. Radford, Bartley D. Richardson, Shawn E. Davis. Se-quence Aggregation Rules for Anomaly Detection in Com-puter Network Traffic. American Statistical Association 2018 Symposi-um on Data Science and Statistics, May 2018, pp. 1-13.

[43] R. Vijayanand, D. Devaraj, B. Kannapiran, Intrusion detection sys-tem for wireless mesh network using multiple support vec-tor ma-chine classifiers with genetic-algorithm-based feature selection, Computers & Security, Volume 77, 2018, Pages 304-314, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2018.04.010.

[44] L.Nicholas, S.Y. Ooi, Y.H. Pang, S.O. Hwang, S.Tan, "Study of long short-term memory in flow-based network intrusion detection system", Journal of Intelligent & Fuzzy Systems, vol. Pre-press, no. Pre-press, pp. 1-11, 2018, doi: 10.3233/JIFS-169836

[45] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorba-ni, “Toward Generating a New Intrusion Detection Dataset and Intru-sion Traffic Characterization”, 4th International Confer-ence on In-formation Systems Security and Privacy (ICISSP), Purtogal, Janu-ary 2018

APPENDIX

```
#Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
#Load the data
df1 = pd.read_csv('Tuesday-WorkingHours.pcap_ISCX.csv')
df2 = pd.read_csv('Wednesday-workingHours.pcap_ISCX.csv')
df3 = pd.read_csv('Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv')
#Combining of three Dataframes into one Dataframe
frames = [df1,df2,df3]
df = pd.concat(frames)
#It gives an overview about a Dataframe columns
df.info()
#By default the head function returns the first 5 rows
df.head()
#By default the tail function returns the first 5 rows
df.tail()
#Count the number of rows and column in the data set
df.shape
#Explore the data
df.columns
#It calculates some basic statistical details
df.describe()
#It shows how many unique values are there in every column
df.nunique()
#It will check if any value is NaN in a Dataframe
np.isnan(df.any())
#The function tests element-wise whether it is finite or not and return the result as a boolean
array.
np.isfinite(df.all())
#It will replace NaN values by Zeroes in a column of a Dataframe
df = df.fillna(0)
#It will remove nan and inf values in a Dataframe
df =df[~df.isin([np.nan, np.inf]).any(1)]
#Renaming the Dataframe column from Label to Attacks
df.rename({' Label':'Attacks'},axis=1,inplace=True)
#Returns a list of unique values
print(df['Attacks'].unique())
#dataframe Visualization
```

```
#Let's plot a histogram to get the feel of type of data we are dealing with
#We can plot histogram only for numerical attributres
df.hist(bins=50, figsize=(30,20))
plt.show()
sns.countplot(df['Attacks'])
#Return the data types of each column in the DataFrame.
df.dtypes
from sklearn.preprocessing import LabelEncoder
Attacks_encoder = LabelEncoder()
Attacks_encoder.fit(df['Attacks'])
df['Attacks_enc'] = Attacks_encoder.transform(df['Attacks'])
df[['Attacks_enc','Attacks']]
#Dropping the Attacks column from the Dataframe
df.drop(['Attacks'], axis =1,inplace=True)
sns.distplot(df['Attacks_enc'])
#Split the data set into independent (X) and dependent (Y) data sets
# X --> contains the dataframe without the target i.e price
X = df.drop('Attacks_enc',axis=1)
# Y --> contains only the target value
Y = df['Attacks_enc']
#Split the data set into 75% training and 25% testing
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test= train_test_split(X, Y, test_size = 0.25, random_state= 0)
X_train = X_train.copy()
Y_train = Y_train.copy()
X_test = X_test.copy()
Y_test = Y_test.copy()
X_train.shape,Y_train.shape
X_test.shape,Y_test.shape
# it will remove zero variance features
from sklearn.feature_selection import VarianceThreshold
var_thres = VarianceThreshold(threshold=0)
var_thres.fit(df)
#Getting all number of columns that have constant values
constant_columns = [column for column in df.columns
                    if column not in df.columns[var_thres.get_support()]]
print(len(constant_columns))
#Dropping constant_columns from the Dataframe
df.drop(constant_columns,axis=1,inplace=True)
#Returning the columns that are having constant values
for feature in constant_columns:
  print(feature)
# decision tree for feature importance on a classification problem
from sklearn.datasets import make_classification
from sklearn.tree import DecisionTreeClassifier
from matplotlib import pyplot
```

```
# define the model
model = DecisionTreeClassifier()
# fit the model
model.fit(X_train, Y_train)
# get importance
importance = pd.Series(model.feature_importances_,index=X.columns)
# plot feature importance
importance.nlargest(10).plot(kind='barh')
pyplot.show()
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
# Create Decision Tree classifer object
model = DecisionTreeClassifier()
# Train Decision Tree Classifer
model = model.fit(X_train,Y_train)
#Predict the response for test dataset
y_pred = model.predict(X_test)
# Model Accuracy, how often is the classifier correct?
print("Decision Tree Training Accuracy: ",metrics.accuracy_score(Y_test, y_pred))
# random forest for feature importance on a classification problem
from sklearn.datasets import make_classification
from sklearn.ensemble import RandomForestClassifier
from matplotlib import pyplot
# define the model
model = RandomForestClassifier()
model.fit(X_train,Y_train)
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_,index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
pyplot.show()
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
# classify using random forest classifier
classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(X_train, Y_train)
y_pred = classifier.predict(X_test)
# print the accuracy
print('Random Forest Training Accuracy : ' + str(accuracy_score(Y_test, y_pred)))
#get correlations of each features in dataset
corrmat = df.iloc[:,:-1].corr()
top_corr_features = corrmat.index
plt.figure(figsize=(55,40))
#plot heat map
g=sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="RdYlGn")
# with the following function we can select highly correlated features
# it will remove the first feature that is correlated with anything other feature
```

```
def correlation(df, threshold):
    col_corr = set()  # Set of all the names of correlated columns
    corr_matrix = df.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold: # we are interested in absolute coeff value
                colname = corr_matrix.columns[i]  # getting the name of column
                col_corr.add(colname)
    return col_corr
corr_features = correlation(X_train, 0.85)
len(set(corr_features))
corr_features
X_train.drop(corr_features,axis=1,inplace=True)
X_test.drop(corr_features,axis=1,inplace=True)
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(df)
scaled_data=scaler.transform(df)
scaled_data
from sklearn.decomposition import PCA
pca=PCA(n_components=2)
pca.fit(scaled_data)
x_pca=pca.transform(scaled_data)
scaled_data.shape
x_pca.shape
x_pca
plt.figure(figsize=(8,6))
plt.scatter(x_pca[:,0],x_pca[:,1],c=df['Attacks_enc'])
plt.xlabel('First principle component')
plt.ylabel('Second principle component')
from sklearn.dummy import DummyClassifier
from sklearn.metrics import accuracy_score
# define model
model = DummyClassifier(strategy='most_frequent')
# fit model
model.fit(x_pca, Y)
# make predictions
y = model.predict(x_pca)
# calculate accuracy
accuracy = accuracy_score(Y, y)
print('Naive Bayes Classifier Training Accuracy: %.3f' % accuracy)
from sklearn.linear_model import LogisticRegression
log = LogisticRegression(random_state=0,solver='lbfgs')
log.fit(x_pca, Y)
# print the accuracy
```

```
print('Logistic Regression Training Accuracy: ', log.score (x_pca, Y))
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

# apply Linear Discriminant Analysis
lda = LinearDiscriminantAnalysis(n_components=2)
X_train = lda.fit_transform(X_train, Y_train)
X_test = lda.transform(X_test)
# plot the scatterplot
plt.scatter(
    X_train[:,0],X_train[:,1],c=Y_train,cmap='rainbow',
  alpha=0.7,edgecolors='b'
)
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
log = LogisticRegression(random_state=0,solver='lbfgs')
log.fit(X_train, Y_train)
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, Y_train)
y_pred = knn.predict(X_test)
# print the accuracy
print('Logistic Regression Training Accuracy: ', log.score (X_train, Y_train))
print('KNN Training Accuracy: ', metrics.accuracy_score(Y_test, y_pred))
```

Genetic Algorithm Implementation (NSL-KDD Dataset)

```
pip install scikit-learn
pip install sklearn-genetic
pip install sklearn-genetic-opt
#Import libraries
import numpy as np
import pandas as pd
import random
import matplotlib.pyplot
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import metrics
df = pd.read_csv('/content/KDDTrain+_20Percent.txt')
##  adding column names to data frame
```

```
columns = (['duration'
,'protocol_type'
,'service'
,'flag'
,'src_bytes'
,'dst_bytes'
,'land'
,'wrong_fragment'
,'urgent'
,'hot'
,'num_failed_logins'
,'logged_in'
,'num_compromised'
,'root_shell'
,'su_attempted'
,'num_root'
,'num_file_creations'
,'num_shells'
,'num_access_files'
,'num_outbound_cmds'
,'is_host_login'
,'is_guest_login'
,'count'
,'srv_count'
,'serror_rate'
,'srv_serror_rate'
,'rerror_rate'
,'srv_rerror_rate'
,'same_srv_rate'
,'diff_srv_rate'
,'srv_diff_host_rate'
,'dst_host_count'
,'dst_host_srv_count'
,'dst_host_same_srv_rate'
,'dst_host_diff_srv_rate'
,'dst_host_same_src_port_rate'
,'dst_host_srv_diff_host_rate'
,'dst_host_serror_rate'
,'dst_host_srv_serror_rate'
,'dst_host_rerror_rate'
,'dst_host_srv_rerror_rate'
,'attack'
,'level'])
df.columns = columns

#It gives an overview about a Dataframe columns
```

```
df.info()
df = df.replace((np.inf, -np.inf, np.nan), 0).reset_index(drop=True)
from sklearn.preprocessing import LabelEncoder
protocol_type_encoder = LabelEncoder()
df['protocol_type']=protocol_type_encoder.fit_transform(df['protocol_type'].astype(str))
service_encoder = LabelEncoder()
df['service']=service_encoder.fit_transform(df['service'].astype(str))
flag_encoder = LabelEncoder()
df['flag']=flag_encoder.fit_transform(df['flag'].astype(str))
attack_encoder = LabelEncoder()
df['attack']=attack_encoder.fit_transform(df['attack'].astype(str))
label=df['attack']
#splitting the model into training and testing set
X_train, X_test, y_train, y_test = train_test_split(df,
                                    label, test_size=0.30,
                                    random_state=101)
clf = DecisionTreeClassifier()
# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)
#Predict the response for test dataset
y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
#training a logistics regression model
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
predictions = logmodel.predict(X_test)
print("Accuracy = "+ str(accuracy_score(y_test,predictions)))
#defining various steps required for the genetic algorithm
def initilization_of_population(size,n_feat):
    population = []
    for i in range(size):
        chromosome = np.ones(n_feat,dtype=np.bool)
        chromosome[:int(0.3*n_feat)]=False
        np.random.shuffle(chromosome)
        population.append(chromosome)
    return population
def fitness_score(population):
    scores = []
    for chromosome in population:
        logmodel.fit(X_train.iloc[:,chromosome],y_train)
        predictions = logmodel.predict(X_test.iloc[:,chromosome])
        scores.append(accuracy_score(y_test,predictions))
    scores, population = np.array(scores), np.array(population)
    inds = np.argsort(scores)
    return list(scores[inds][::-1]), list(population[inds,:][::-1])
def selection(pop_after_fit,n_parents):
```

```
    population_nextgen = []
    for i in range(n_parents):
        population_nextgen.append(pop_after_fit[i])
    return population_nextgen
def crossover(pop_after_sel):
    population_nextgen=pop_after_sel
    for i in range(len(pop_after_sel)):
        child=pop_after_sel[i]
        child[3:7]=pop_after_sel[(i+1)%len(pop_after_sel)][3:7]
        population_nextgen.append(child)
    return population_nextgen
def mutation(pop_after_cross,mutation_rate):
    population_nextgen = []
    for i in range(0,len(pop_after_cross)):
        chromosome = pop_after_cross[i]
        for j in range(len(chromosome)):
            if random.random() < mutation_rate:
                chromosome[j]= not chromosome[j]
        population_nextgen.append(chromosome)
    #print(population_nextgen)
    return population_nextgen
def generations(size,n_feat,n_parents,mutation_rate,n_gen,X_train,
                                X_test, y_train, y_test):
    best_chromo= []
    best_score= []
    population_nextgen=initilization_of_population(size,n_feat)
    for i in range(n_gen):
        scores, pop_after_fit = fitness_score(population_nextgen)
        print(scores[:2])
        pop_after_sel = selection(pop_after_fit,n_parents)
        pop_after_cross = crossover(pop_after_sel)
        population_nextgen = mutation(pop_after_cross,mutation_rate)
        best_chromo.append(pop_after_fit[0])
        best_score.append(scores[0])
    return best_chromo,best_score
from sklearn.svm import LinearSVC
from __future__ import print_function
from sklearn.preprocessing import LabelEncoder
from sklearn import datasets, linear_model
from genetic_selection import GeneticSelectionCV
def main():
    df = pd.read_csv('/content/KDDTrain+_20Percent.txt')
    df = df.replace((np.inf, -np.inf, np.nan), 0).reset_index(drop=True)
    ##  adding column names to data frame
```

```
 columns =
(['duration','protocol_type','service','flag','src_bytes','dst_bytes','land','wrong_fragment','urgent
','hot','num_failed_logins'
,'logged_in','num_compromised','root_shell','su_attempted','num_root','num_file_creations','n
um_shells','num_access_files','num_outbound_cmds'
,'is_host_login','is_guest_login','count','srv_count','serror_rate','srv_serror_rate','rerror_rate','sr
v_rerror_rate','same_srv_rate'
,'diff_srv_rate','srv_diff_host_rate','dst_host_count','dst_host_srv_count','dst_host_same_srv_
rate','dst_host_diff_srv_rate'
,'dst_host_same_src_port_rate','dst_host_srv_diff_host_rate','dst_host_serror_rate','dst_host_s
rv_serror_rate','dst_host_rerror_rate','dst_host_srv_rerror_rate'
    ,'attack','level'])
    df.columns = columns
    protocol_type_encoder = LabelEncoder()
    df['protocol_type']=protocol_type_encoder.fit_transform(df['protocol_type'].astype(str))
    service_encoder = LabelEncoder()
    df['service']=service_encoder.fit_transform(df['service'].astype(str))
    flag_encoder = LabelEncoder()
    df['flag']=flag_encoder.fit_transform(df['flag'].astype(str))
    attack_encoder = LabelEncoder()
    df['attack']=attack_encoder.fit_transform(df['attack'].astype(str))
    X = df
    y = df['attack']
    estimators = linear_model.LogisticRegression(solver="liblinear", multi_class="ovr")
    selectors = GeneticSelectionCV(estimators,
                                   cv=6,
                                   verbose=2,
                                   scoring="accuracy",
                                   max_features=10,
                                   n_population=60,
                                   crossover_proba=0.6,
                                   mutation_proba=0.05,
                                   n_generations=15,
                                   crossover_independent_proba=0.6,
                                   mutation_independent_proba=0.05,
                                   tournament_size=4,
                                   n_gen_no_change=20,
                                   caching=True,
                                   n_jobs=-2)
    selectors = selectors.fit(X, y)
    print(selectors.support_)
if __name__ == "__main__":
    main()
from sklearn import datasets
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
#Load the data
data = pd.read_csv('/content/KDDTrain+_20Percent.txt')
data = data.replace((np.inf, -np.inf, np.nan), 0).reset_index(drop=True)
columns =
(['duration','protocol_type','service','flag','src_bytes','dst_bytes','land','wrong_fragment','urgent
','hot','num_failed_logins'
,'logged_in','num_compromised','root_shell','su_attempted','num_root','num_file_creations','n
um_shells','num_access_files','num_outbound_cmds'
,'is_host_login','is_guest_login','count','srv_count','serror_rate','srv_serror_rate','rerror_rate','sr
v_rerror_rate','same_srv_rate'
,'diff_srv_rate','srv_diff_host_rate','dst_host_count','dst_host_srv_count','dst_host_same_srv_
rate','dst_host_diff_srv_rate'
,'dst_host_same_src_port_rate','dst_host_srv_diff_host_rate','dst_host_serror_rate','dst_host_s
rv_serror_rate','dst_host_rerror_rate','dst_host_srv_rerror_rate'
,'attack','level'])
data.columns = columns
protocol_type_encoder = LabelEncoder()
data['protocol_type']=protocol_type_encoder.fit_transform(data['protocol_type'].astype(str))
service_encoder = LabelEncoder()
data['service']=service_encoder.fit_transform(data['service'].astype(str))
flag_encoder = LabelEncoder()
data['flag']=flag_encoder.fit_transform(data['flag'].astype(str))
attack_encoder = LabelEncoder()
data['attack']=attack_encoder.fit_transform(data['attack'].astype(str))
n_samples = len(data)
X = data
y = data['attack']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7)
clf = RandomForestClassifier()
from sklearn_genetic import GASearchCV
from sklearn_genetic.space import Continuous, Categorical, Integer
from sklearn_genetic.plots import plot_fitness_evolution, plot_search_space
from sklearn.model_selection import StratifiedKFold
import matplotlib.pyplot as plt
param_grid = {'min_weight_fraction_leaf': Continuous(0.01, 0.5, distribution='log-uniform'),
              'bootstrap': Categorical([True, False]),
              'max_depth': Integer(2, 30),
              'max_leaf_nodes': Integer(2, 35),
              'n_estimators': Integer(100, 300)}
cv = StratifiedKFold(n_splits=3, shuffle=True)
```

```
evolved_estimator = GASearchCV(estimator=clf,
                               cv=cv,
                               scoring='accuracy',
                               population_size=10,
                               generations=15,
                               tournament_size=3,
                               elitism=True,
                               crossover_probability=0.6,
                               mutation_probability=0.05,
                               param_grid=param_grid,
                               criteria='max',
                               algorithm='eaMuPlusLambda',
                               n_jobs=-1,
                               verbose=True,
                               keep_top_k=4)
evolved_estimator.fit(X_train,y_train)
y_predicy_ga = evolved_estimator.predict(X_test)
accuracy_score(y_test,y_predicy_ga)
evolved_estimator.best_params_
plot_fitness_evolution(evolved_estimator)
plt.show()
print(evolved_estimator.logbook)
evolved_estimator.hof
chromo,score=generations(size=20,n_feat=43,n_parents=10,mutation_rate=0.10,
                         n_gen=15,X_train=X_train,X_test=X_test,y_train=y_train,y_test=y_test)
clf.fit(X_train.iloc[:,chromo[-1]],y_train)
predictions = clf.predict(X_test.iloc[:,chromo[-1]])
print("Accuracy score after genetic algorithm is= "+str(accuracy_score(y_test,predictions)))
chromo,score=generations(size=20,n_feat=43,n_parents=10,mutation_rate=0.10,
                         n_gen=15,X_train=X_train,X_test=X_test,y_train=y_train,y_test=y_test)
logmodel.fit(X_train.iloc[:,chromo[-1]],y_train)
predictions = logmodel.predict(X_test.iloc[:,chromo[-1]])
print("Accuracy score after genetic algorithm is= "+str(accuracy_score(y_test,predictions)))
from genetic_selection import GeneticSelectionCV
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
import pandas as pd
import numpy as np
df = pd.read_csv('/content/KDDTrain+_20Percent.txt')
df = df.replace((np.inf, -np.inf, np.nan), 0).reset_index(drop=True)
columns =
(['duration','protocol_type','service','flag','src_bytes','dst_bytes','land','wrong_fragment','urgent
','hot','num_failed_logins'
,'logged_in','num_compromised','root_shell','su_attempted','num_root','num_file_creations','n
um_shells','num_access_files','num_outbound_cmds'
```

```
,'is_host_login','is_guest_login','count','srv_count','serror_rate','srv_serror_rate','rerror_rate','sr
v_rerror_rate','same_srv_rate'
,'diff_srv_rate','srv_diff_host_rate','dst_host_count','dst_host_srv_count','dst_host_same_srv_
rate','dst_host_diff_srv_rate'
,'dst_host_same_src_port_rate','dst_host_srv_diff_host_rate','dst_host_serror_rate','dst_host_s
rv_serror_rate','dst_host_rerror_rate','dst_host_srv_rerror_rate'
,'attack','level'])
df.columns = columns
protocol_type_encoder = LabelEncoder()
df['protocol_type']=protocol_type_encoder.fit_transform(df['protocol_type'].astype(str))
service_encoder = LabelEncoder()
df['service']=service_encoder.fit_transform(df['service'].astype(str))
flag_encoder = LabelEncoder()
df['flag']=flag_encoder.fit_transform(df['flag'].astype(str))
attack_encoder = LabelEncoder()
df['attack']=attack_encoder.fit_transform(df['attack'].astype(str))
x = df.drop('attack',axis=1)
Y = df['attack']
estimators = DecisionTreeClassifier()
models = GeneticSelectionCV(
    estimators, cv=5, verbose=0,
    scoring="accuracy", max_features=10,
    n_population=100, crossover_proba=0.6,
    mutation_proba=0.05, n_generations=15,
    crossover_independent_proba=0.5,
    mutation_independent_proba=0.04,
    tournament_size=3, n_gen_no_change=10,
    caching=True, n_jobs=-1)
models = models.fit(x, Y)
print('Feature Selection:', x.columns[models.support_])
#split dataset in features and target variable
feature_cols = ['service', 'flag', 'src_bytes', 'is_host_login', 'same_srv_rate',
       'diff_srv_rate', 'dst_host_count', 'dst_host_srv_diff_host_rate',
       'dst_host_serror_rate', 'level']
x = df[feature_cols] # Features
Y = df['attack'] # Target variable
# Split dataset into training set and test set
x_train, x_test, Y_train, Y_test = train_test_split(x, Y, test_size=0.3, random_state=1) # 70%
training and 30% test
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()
# Train Decision Tree Classifer
clf = clf.fit(x_train,Y_train)
#Predict the response for test dataset
Y_pred = clf.predict(x_test)
print("Accuracy:",metrics.accuracy_score(Y_test, Y_pred))
```