National College of Ireland

Security Vulnerability Detection with Enhanced Privacy Preservation for Edge Computing Using Hybrid Machine Learning Approach

MSc Research Project
Cloud Computing

Shubham Patil
Student ID: x21139261

School of Computing
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Shubham Patil
Student ID:	x21139261
Programme:	Cloud Computing
Year:	2022
Module:	MSc Research Project
Supervisor:	Sean Heeney
Submission Due Date:	15/12/2022
Project Title:	Security Vulnerability Detection with Enhanced Privacy Preservation for Edge Computing Using Hybrid Machine Learning Approach
Word Count:	7081
Page Count:	22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	15th December 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Security Vulnerability Detection with Enhanced Privacy Preservation for Edge Computing Using Hybrid Machine Learning Approach

Shubham Patil
x21139261

Abstract

Edge computing, which is now a slashing method, allows for the processing and calculation of upstream data in consideration of IoT properties and downstream data with the assistance of cloud services. The primary notion of edge computing is to move calculations closer to data sources, such as edge devices or IoT nodes, by bringing cloud computing to the network's edge. By using edge computing, services may be located close to the original data source in order to satisfy critical needs in agile connectivity, pragmatic optimization, smart or intelligent applications, reliability, and secrecy. One of the most pressing issues surrounding edge computing is the prevention of malware and other security breaches in networks. Several researchers have developed numerous ensemble approaches to network data for addressing security issues, however, these methods are unable to neutralize all modern network incursions in edge computing's rapidly developing and changing network traffic data pattern. As a consequence of this, there is a necessity for an Edge Intrusion attack classifier Framework (EIACF) that monitors the network for potentially harmful activity. In this study, we have demonstrated different classifiers such as Random Forest (RF), XGBoost, and KNN classifiers and proposed a hybrid classifier (EIACF) with the help of ensemble learning which is combined with Infinite feature selection and PCA for detecting edge network anomalies. The experimental outcomes illustrate that the EIHCF has an accuracy of 99.33%. When compared to the findings of the prior work, these results show an improvement in performance. In our trials, we utilized NSL-KDD datasets to assess the model's performance.

1 Introduction

In the foreseeable future, various things and information would be networked or interconnected, as well as the people that might anticipate experiencing better and brighter lifestyles. It is anticipated that the information and the things that are interconnected will be of greater value in the future. The Internet of Items refers to a network that connects things and information. IoT produces large amounts of partially processed data that must be analyzed quickly in order to react. In the present situation, the cloud has become a crucial component of this development. However, a cloud that is imposed globally from a central location must be able to handle massive amounts of data. But it also generates transmission delay, lengthy reaction time, and user tension rise as the user's

physical distance from the cloud increases. Additionally, the processing speed in this case heavily depends on the functionality of the consumers' devices. All of these problems can be effectively resolved using a technique called edge computing. Edge Computing is a decentralized computing system that integrates cellular mobile networking like mobile base stations with computer technology with networking-based components including hardware, memory, and hypervisor. (Yang et al. 2019)It increases the capability of IT services and cloud computing at the edge of the network. The main idea behind this technology was to place edge nodes on the mobile network adjacent to cellular subscribers and consumers so that software and related computer tasks could be carried out there. A cellular operator may provide effective services for a group of clients by using this capability.

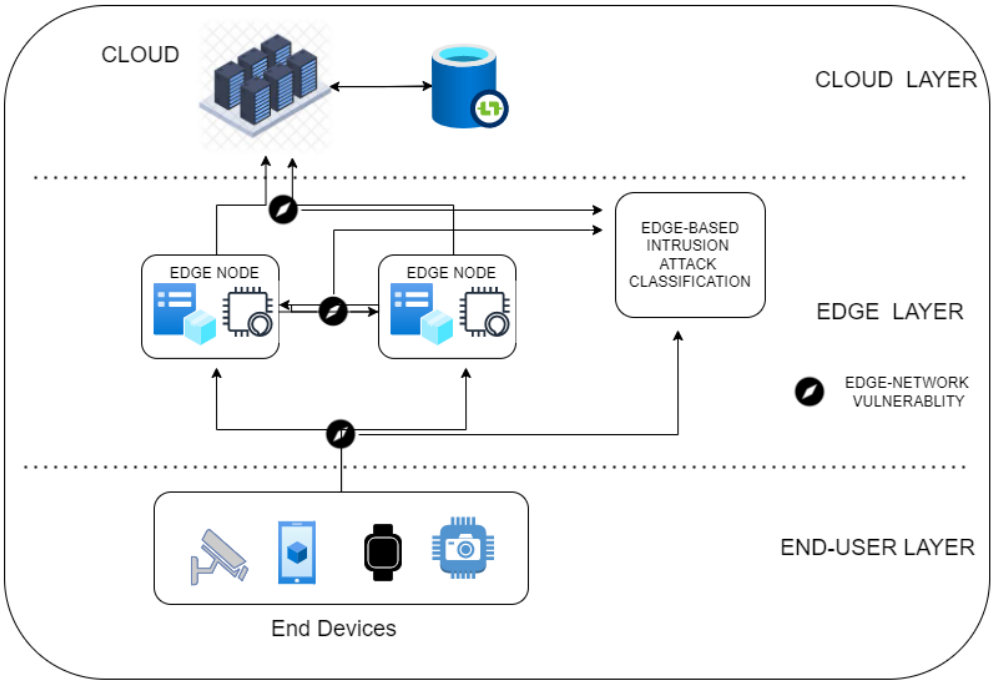


Figure 1: Basic architecture of Edge Computing

Edge computing, on the other hand, allows numerous smart apps to utilize the same edge device with several users at once. In turn, this makes edge devices more vulnerable to hacking, which might lead to fake output data, erroneous input data, and inaccurate outcomes.(Alwarafy et al. 2020) Any sensitive application's performance will be impacted by this. Additionally, the compromised edge device's data may be exploited for other nefarious or criminal activities. In general, edge computing is subject to two main categories of assaults. After authentication, the first one often involves the edge device being hacked. Because the device has particular rights for certain inside network activity, anybody inside may use it to carry out an undesirable action. These are unwanted attacks where the hacker/attacker attempts to target the edge layer or cloud layer. Secondly, it involves unauthenticated edge devices that sometimes attempt to use sophisticated attack methods to penetrate the device or cloud layers. An unauthenticated attack, often known as an outside attack, is this kind of assault. Since resources are shared across

several applications in an edge computing environment with a multitenant architecture, insider attacks are exceedingly difficult to detect. As a result, the primary disadvantage of mobile edge computing is malignant edge device assaults(Singh et al. 2022). In order to defend such networks, the firewall is the first line of defense. However, a conventional firewall system can only stop packets that are traveling over the internet and originating from outside sources. The malicious inside packets created by insider assault, however, cannot be filtered by this system. Such firewall-based solutions are ineffective because of the edge nodes' complexity and unpredictability. Additionally, managing many firewalls is expensive and resource-constrained on the majority of edge devices, which makes security solutions implementation difficult. As shown in 1 the black directional symbol represents the vulnerabilities in the network so to address this issue there is a need for security in the network.

1.1 Background and Research motivation

Edge computing acts as a resolution by functioning as an improved form of cloud computing that can reduce network latency and throughput. However, it often demonstrates a susceptibility to attacks that can jeopardize the security of the system's network. The most frequent attacks that weaken the security of edge networks include Distributed DoS, remote recording, ransomware, routing, as well as data leakage attacks.(Alzahrani & Alenazi 2021) (Xue et al. 2022) As part of this assault, the adversary sends an overwhelming number of invalid request packets to the edge server, thus blocking all traffic to and from it, rendering the resources inaccessible and lengthening the response time. To secure the edge network, several machine learning-based security measures have been developed, including intrusion detection algorithms and public key-based security algorithms. But there are no effective security options that secure the edge network completely. This inspired the study to try merging a hybrid classifier algorithm with infinite feature selection and PCA in an effort to obtain a better feature set for filtering and to get beyond the shortcomings of earlier ensemble techniques.

1.2 Research Objectives

• To implement a hybrid machine learning algorithm by combining the Tree-based algorithms and Hard Voting Classifier.
• Optimally enhancing the model's accuracy, precision, F1, and recall by integrating infinite feature selection with PCA for the voting classifier.
• To conduct a comparison of the proposed approach with existing machine learning techniques.

1.3 Research Question

In this work, we have proposed the following research question:- "To what extent Hybrid Machine Learning Algorithm can be utilized for edge computing networks to enhance the precision for security vulnerability detection?"

1.4 Document Struture

In addition to that, this study report is broken up into six pieces. In Section 2, we will discuss the work that was done in the past about network edge security and intrusions

that occurred in edge networks. In Section 3, we explain how the study was conducted, including the methodologies and computer programs that were used. In Section 4, the Design requirements of the proposed research are discussed. The hybrid machine learning method that was suggested is documented and its step-by-step implementation is shown in Section 5. Evaluation, findings, and a conclusion are presented in the last sections, which are numbered 6 and 7, respectively.

2 Related Work

2.1 Security of Edge Computing

The adversarial training method was introduced by (Zhang et al. 2021) as a security improvement approach for the DL model. This strategy can defend against counter-attacks, hide certain sensitive data features, and make edge devices more reliable. In order to determine whether or not the approach can be successfully implemented, it was applied to a phrase similarity analysis model that was built using a convolutional neural network (CNN) that was used in the CEC environment. The accuracy of the model that was created via the use of the confrontation training approach was enhanced by 4.8% in comparison to the CNN that was originally used. Next, (Singh et al. 2021) details the difficulties with security and privacy challenges presented by networking heterogeneous devices at differing stages of the Edge Computing architecture. Second, it goes through the variety of deep learning and machine learning methods used in EC application cases. This is followed by a general description of the many attack types that the Edge network faces, as well as the intrusion detection systems and accompanying machine learning techniques that address these security and privacy issues. The specifics of deep learning and machine learning approaches are summarized for EC security. The remaining challenges in protecting Edge networks are then listed, along with potential research topics. Next, (Chen 2020)suggested EC's integrated mobile-based sensitive information security architecture. The heterogeneous edge computing network was layered for mobile intelligent private information, as shown by its characteristics. HE (Homomorphism Encryption) enabled the edge network and mobile terminal to communicate and store sensitive data as ciphertext. From the simulation results, the framework assessed the consistency between the optimum defense methods of individual edge network devices and the complete edge network for diverse edge network devices. This method successfully reduced edge network device computing resource use.

Additionally, (Zhou et al. 2021) created a security architecture. Three operations identification, learning, and regulation—protected smart immunity. The author proposes a smart security defense that authorizes and authenticates users at the network edge server depending on criteria. Then, in the next evolutionary step, the information is checked against similar information and scored for learning, system accuracy, and security. The author suggested a three-phase smart algorithmic strategy for the smart immunity framework. Mathematical simulations verified the answer. Furthermore, due to the large amount of sensitive data stored in edge computing equipment, (Chen et al. 2021)indicated that it is vulnerable to various attacks. MATLAB program simulates the author's edge computing vulnerability management method. An object-oriented (C++) generator and edge devices create service assaults twenty times in this simulator. Memory, CPU, and data packet delivery and retrieval were simulated. This paper proposed a security-efficient edge computing technology for the IoT ecosystem. The trials showed that the method

improved the service-driven security of Edge computing IoT models.

2.2 Class imbalance in edge computing with machine learning

(Kozik et al. 2018) utilized the adaptability of cloud architectures with recent advancements in huge-scale computer vision from transformation to even more difficult computational as well as containers infrastructure to the cloud in utilizing edge computing techniques for incorporating internet traffic categorizations on extremely complicated machine learning models prepared for cloud infrastructure, For time-consuming and computationally challenging applications, the author presented a decentralized identification method employing Extreme Learning Machine (ELM) classifiers and cloud-based computing clusters. To validate the suggested methodologies, a comprehensive experiment is carried out on a real-world dataset linked to cyber security. ELMs have faster learning and generalization than other binary classification methods. Next, (Yue et al. 2021) created a federated learning architecture for secrecy education in class-imbalanced abnormal health detection applications. In order to increase model fidelity, it also created an original localized update strategy based on curiosity reinforcement learning as well as an adaptable global update technique. Results indicated that it performed better in terms of model utilization and accuracy than baselines. A generic framework for class-imbalanced federated learning was developed by the author. On a multi-board real-time collaborative edge test bed, the author constructed a prototype of the system. It was evaluated using a pair of real-time smart healthcare sensor apps for identifying anxiety and tracking driver fatigue. The technology significantly increases accuracy and energy consumption.

Additionally, (Feng et al. 2020) used edge computing to estimate assembly performance in an IIot scenario, guiding flexible industrial data collecting. The author predicted assembly quality using edge intelligent services. Edge computing allowed product assembly quality prediction. Second, Random Forest was used to identifying critical-to-quality qualities and organize them descending, giving management insights to address essential concerns. SMOTE with cooperatively set parameters and experimental values for imbalanced categorization is offered. Using our quality prediction technique, Random Forest evaluates assembly lines and provides guidance and reference for wheel bearing product assembly. Lastly, (Yuan et al. 2020) presented smart home edge nodes with intrusion detection systems. Generative Adversarial Networks mitigated the training dataset's imbalanced data. The author converts network traffic to graphics and builds a convolutional neural network for traffic classification. The researcher creates synthetic samples using a Generative Adversarial Network model to balance data between mild and severe classifications.

2.3 Existing Security Methods for Edge Computing

To secure radio networks, (Liu et al. 2020) created a security-enhanced traceback system. This operation split the network into three. Different nodes are marked differently. Device marking probability varied by area. Devices farthest from the sink had higher marking probability while those closer conserved power. This system also saved and moved data packet marking tuples to devices further from the sink to modify device memory. SET outperformed traceback in hypothetical and extensive trial simulations. Later, (Bai et al. 2022) introduced the Fine-Grained Access Control method to protect data. This design protected mobile edge computing data. The new design focused on meta-graph

rules-based active fine-grained dependable customer group system characteristics. By integrating with standard role-reliant access management systems, this system awarded customer roles based on customer group dependability. User verification was performed after feature matching to ensure data security. The testing revealed that the FGAC could recognize rivals, modify clusters, improve access control, and secure edge computing data.

Furthermore, in order to make MEC secure, Chen et al. (2022) introduced a beginner deep reinforcement learning technique together with convex optimization. The goal of deep reinforcement learning was to identify a useful method for the offloading ratio. Convex optimization was used to overcome issues with processing power and transmission energy distribution. By guaranteeing data security and secrecy, the shown deep reinforcement learning convex optimization approach outperforms more conventional strategies. Next, (Bai et al. 2022) introduced Multi-Core Federated Learning (MCFL) to assist FL intelligence to settle on actual MEC systems. MC-FL maintains and trains many global models with various learning performance-computational complexity tradeoffs. This simple update can manage device heterogeneity and device state fluctuations, improving FL compatibility and robustness. MC-FL also supports partial client involvement that changes over time. MC-FL can operate in unpredictable mobile situations. The author also proved MC-FL convergence. Specifically, the author proposed an online client scheduling strategy for MC-FL to intelligently plan clients for training various global models to decrease MC-FL completion time.

2.4 Network Intrusion Security at Edge computing

(Wang et al. 2022) suggested a deep reinforcement learning-based intrusion detection technique that can effectively discriminate aberrant traffic and categorize the different sorts of attacks in order to improve the security of Smart Vehicular Networks. The usefulness of the suggested intrusion detection technique and the classification performance increase over typical machine learning algorithms, which was up to 80% in test data and 97% in train data, was shown by the author's simulated tests on the NSL-KDD dataset. Additionally, Precision was a poor aspect. In order to conduct a lightweight security detection on a massive proportion of multimedia traffic, (Agrawal et al. 2022) proposed a distributed intrusion detection system and detection technique for multimedia traffic in an edge computing (EC) environment. Additionally, the difficult-to-detect traffic was identified using the C4.5 decision tree technique. The author suggested this feature so that DIDS may automatically modify the detection strength of multimedia packets in response to the queue length.

Later, (Khan et al. 2022) introduced an automated identification of suspicious network activity and intrusions. The primary objective of this research was to compare and contrast several intrusion detection systems that use deep learning to identify potential security breaches. The author also extensively investigated and evaluated publicly available network-based IDS datasets. Various performance criteria have been used in a critical analysis of deep learning approaches to IDS (accuracy, precision, recall, f-1 score, false alarm rate, and detection rate). In addition, current problems with networks' security and privacy have been examined, along with potential solutions to such problems. (Guezzaz et al. 2022) proposed the K-Nearest Neighbor classifier and Principal Component Analysis to create a powerful hybrid intrusion detection method. An innovative hybrid framework, PK-IDS, has been developed by combining Snort IDS for abuse detection with a K-NN classifier to enhance anomaly detection. Non-standardization and

heterogeneity of data are considered during data preparation. The author has used principal component analysis (PCA) for feature engineering to improve the created model's training time, accuracy, and detection rate. Several recommended techniques are put into practice to verify the PK-IDS architecture. For this purpose, the Bot-IoT and NSL-KDD datasets are used to conduct an empirical analysis of this innovative hybrid technique. The innovative framework provides superior performance and accuracy in comparison to state-of-the-art methods. Indeed, designing IDS for edge-based IIoT security is a challenging task filled with issues such as where to put IDS.

2.5 Summary of existing work

This assessment of the literature's first section focuses on works that employ various simulations and encryption techniques to try to protect the edge computing area. This approach has previously been used in research to minimize the need for edge computing resources, improve edge IoT system security, and provide encryption between two edge devices. The overall objective was to increase security at the edge servers, but the main focus was on reducing the amount of edge computing resources needed for processing to be applied at the edge. The emphasis of the research publications that come after this one is on class inequality in edge computing using machine learning. Many scholars put a lot of work into implementing various classifiers. The objective was to improve accuracy and reduce the class imbalance factor at the edge calculation using machine learning techniques. Later, we spoke about the current security techniques, with a particular emphasis on building data security techniques, security using deep reinforcement learning, and multi-core federated learning. Last but not least, a hybrid architectural technique for a network intrusion system is detailed in the most recent batch of articles. All previous research in this field aimed to use machine learning techniques to safeguard the network at edge devices. Additionally, edge-level incursion predictions were made using ensemble learning research, which combines two or more machine learning methods. There have also been many hybrid algorithms created, however, no study has combined hybrid machine learning with infinite feature selection and PCA approach. Therefore, in order to identify the network-level intrusion at edge computing, a hybrid machine learning technique is suggested in this study.

3 Methodology

3.1 Proposed Architecture:

Edge-Network traffic categorization(ENTC) involves identifying various sorts of application traffic data received by data packets that are examined. Nowadays, it is a vital piece of communication network technology. Input, pre-processing, attribute extraction, categorization, and performance analysis are all included in the traffic classification process. A trustworthy source is used to create the data for the input. Stage two of processing will purge the data and assist in achieving classification accuracy by removing inaccurate and ignored information. During the feature extraction stage, a link between each property and the objective is established, aiding in the identification of the key characteristics in the dataset. During the classification stage, a classification model will be employed for the training and test sets, and the outcome will be the predicted set. The accuracy,

precision, and recall performance analysis matrices will be used in the final step to evaluate the network traffic categorization model's performance. Rapid developments in ML have improved the value of learning algorithms for classifying network traffic, a crucial aspect of network management. Consistent datasets cause the imbalanced class to begin scattering due to fundamental characteristics of internet networks. ML-based ENTC has gained popularity as encrypted communication has become more widespread. But ML approaches often aim to get the maximum accuracy while disregarding class imbalance. Figure 2 depicts the recommended architecture for the research project.

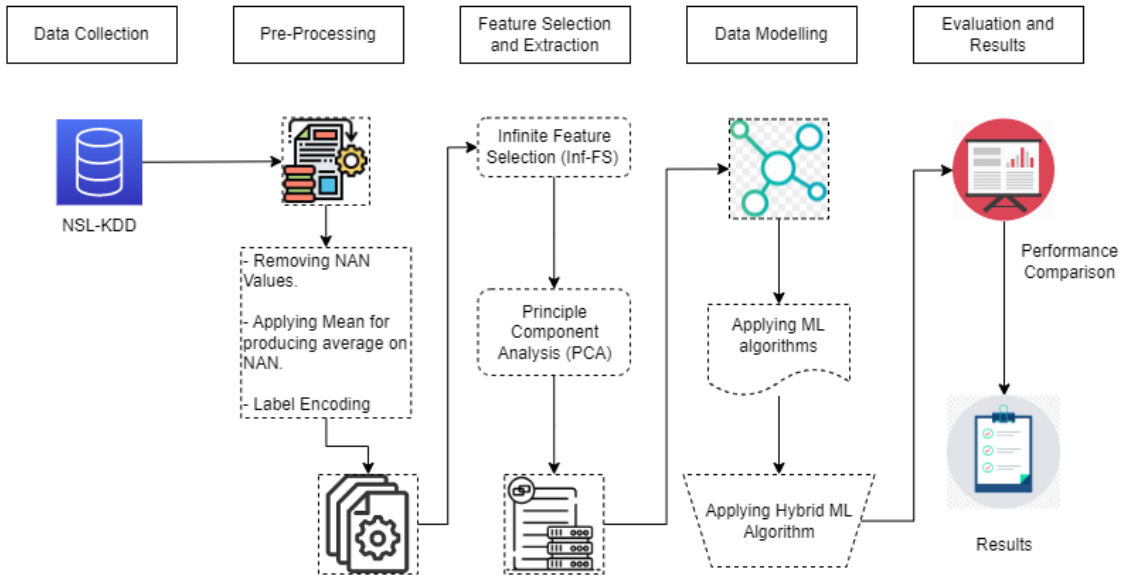


Figure 2: Proposed Architecture

In order to boost efficiency, this research proposes a unique paradigm for classifying network traffic. The recommended model incorporates three classifiers—KNN, XGB, and RF, which are integrated for classification using PCA, which minimizes the number of features. Then, a voting classifier is used, which is trained for a variety of base model or estimator combinations and makes predictions based on averaging the results of each base estimator. Voting for each estimates the output that is integrated with the aggregating criterion.

3.2 Data Gathering

The dataset named NSL-KDD, which was used in data collection includes information that is equivalent to that found in the KDDcup99 dataset. It is made up of about 1,07,992 single connection vectors, each of which comprises 41 characteristics. In addition to that, the type of attack labels and complexity level are provided in CSV format inside the NSL-KDD train data set. The most significant flaw in the KDD dataset is the abundance of duplicated records, which leads learning algorithms to be biased against frequent records as well as prevents those from learning infrequent records, which are typically more dangerous to networks, such as U2R & R2L attacks. Therefore, the NSL-KDD

dataset is selected over the original KDD dataset. Additionally, the placement of these recurring records inside the test set will distort the evaluation results in favor of systems that identify records more often.

3.2.1 NSL-Various KDD's Forms of Assault

- Denial of Service, sometimes known as DoS, is an attack that prevents users from accessing a network or other service.
- User to Root, often known as U2R, is a kind of attack where the attacker tries to get administrative access while simultaneously trying to enter the local computer with root rights.
- R2L: An instance of the Remote to Local attack type is triggered when an unauthorized person makes an effort to access the data by transferring data packets from the remote system to the local one. This kind of attack occurs when an unauthorized user is attempting to access the data.
- The probe attack falls under this category mainly when the attackers attempt to gather data on the local or target systems in order to launch the assault on those machines at a later stage.

3.3 Pre-processing Dataset

In the data processing process, the missing values are removed, the mean function is used to obtain the average of the missing values, the label data is separated, the label encoding is initialized, and the label data is transformed. Additionally, data is normalized to accommodate the corresponding input shapes of the type_of_protocol, service_type, flag, and attack-type_or_normal features in the pre-trained feature set. Additionally, we employed label encoding since we converted the labels into numeric form, which is a form that is readable by machines. The operation of such labels may then be better determined by machine learning techniques.

3.4 Infinite Feature Selection

There may be hundreds of characteristics in a data collection, many of which may not be connected to the data's output. The primary goal of feature selection (FS) techniques is to identify a subgroup of input factors that have a direct impact on the result of the data while decreasing noise and filtering out the irrelevant variables. In this study, we used an improved modified form of InfFS to filter out a unique collection of features from the dataset. This filtering approach uses an unsupervised method to do the ranking phase, and then a straightforward cross-validation method to choose the top m features. The approach's ability to assess the significance of a particular feature while considering all feasible subsets of characteristics is its most attractive feature. Additionally, each feature's score is affected by all the other characteristics in the collection; this method is like one for creating route integrals, which was developed for the area of data clustering or the investigation of graph centralities.(Roffo et al. 2015)

$$a_{i,j} = \alpha\sigma_{i,j} + (1-\alpha)c_{i,j}, \tag{1}$$

$$\xi_\gamma = \prod_{k=0}^{l-1} a_{v_k, v_{k+1}}, \tag{2}$$

where α stands for the loading coefficient, σ refers to standard deviation, and ξ_γ determines the power series matrics of all the features in comparison to other paths. Moreover, l specifies the path length, I and j denote the placement of features, and a refers to the feature. And all these are stored in the register Rl, which is calculated as follows:

$$R_l(i,j) = \sum_{\gamma \in P^l_{(i,j)}} \xi_\gamma = A^l(i,j), \tag{3}$$

here Pl(i,j) consists of every path length that is l between i as well as j. In addition to that, the single feature energy score of s(i) is denoted by:

$$S(i) = \sum_{j \in V} R_l(i,j) = \sum_{j \in V} A^l(i,j), \tag{4}$$

Lastly, the vector S is the sum of all the feature energies individually. Next, the feature energies are sorted from highest to lowest, starting with the highest energy feature, and then written to vector M.

After calculating all of the feature rank energies, the algorithm automatically decides how many features to maintain. As a result, certain features are judged unnecessary and are removed. It is created as a vector that holds the square of each rank energy. The calculation for each component of B is shown in equation 5. A feature is retained if its individual energy is greater than the threshold T; else, it is deleted. The remaining characteristics are then organized in vector M according to the order of their energy values. N stands for the number of features retained.(Roffo et al. 2015)

3.5 Principle Component Analysis

Problems with intrusion detection naturally involve large amounts of data. The dimensionality of the data must be decreased to facilitate exploration and subsequent analysis. This is often accomplished using PCA. PCA is a popular linear dimensionality compression approach that has been extensively used for datasets from a variety of scientific disciplines. (Tharwat 2016)

Using a small number of new variables that are linear combinations of the original variables, PCA attempts to describe the variance-covariance structure of the original set of data. The 's' random variables x1, x2, x3,..., xs form a linear combination known as a principal component because of three distinguishing characteristics. To begin, there is no correlation between the major components. Two, the variance of the first principal component is the largest, followed by the variance of the second principal component, and so on. One more equates the sum of the variances in all the PCs to the variance in the original variables (X1, X2, X3..., Xs). Eigenanalysis of the covariance or correlation matrix of X1, X2, X3..., Xs yields new variables with the desired characteristics. This encourages the use of feature extraction techniques designed to speed up the training process and provide more generalized features. Principal Components Analysis is used

in this case to extract features from the feature inputs.

3.6 Data Modelling

Training models, putting them into action as standalone classifiers, and creating a hybrid ensemble classifier are all part of the data modeling process. The models were trained using the datasets for both training and testing, which were divided with a ratio of 70 to 30 percent. In this part, the classifiers were first developed by using a variety of machine learning techniques in order to analyze anomalies and protect the edge platform from attacks in real-time scenarios. From among these models, only the ones with superior performance were considered for inclusion as possible component options. The component models were identified by an analysis of their erroneous assumptions. After then, an integrated model was developed in order to accomplish better levels of performance. Below are the individual classifier used and the hybrid one which is a combination of all individuals:

- KNN: K-closest neighbors, often known as kNN, is a form of controlled learning computation that is mostly used to deal with relapse and arrangement assignments. One possible interpretation of the idea that underlies kNN is that the value or category of a given information point is determined by the information points that are located nearby. The kNN classifier determines the category that an information point belongs to by using a greater percentage of the voting guideline.

- RF: The Random Forest is a collection of many different trees to choose from. Sacking is a method used in the construction of random forests. In this method, choice trees are used as equal assessors. Random forests are created using this method. When applied to a problem involving grouping, the result is determined based on the decision tree outcomes that received the most votes overall. When it comes to relapse, the expectation of a leaf hub is the mean value of the several objective attributes that make up that leaf. The average worth of the results from chosen trees is subtracted from the value of arbitrary woods.

- XGB: The XGBoost algorithm is a Gradient Boosting-based ensemble machine learning system that makes use of decision trees. Boosting is a class of algorithms that takes a poor learner and turns it into a strong learner by using weighted averages. Extreme Gradient Boosting is an improved gradient boosting approach that makes use of parallel processing, tree pruning, and regularization to prevent overfitting and bias. By taking into account the feature distribution in a leaf over all data points, Boost is able to minimize the search area for feature splits, addressing a significant shortcoming of gradient boost.

- Hybrid: The system employs a Hybrid (Voting) algorithm with the goal of improving the system's accuracy as a whole in order to better serve its users. Combining two or more different models into one algorithm is what hybrid algorithms do. A voting classifier is a kind of classifier that is made up of two or more different classifiers that collaborate with one another. Each classifier evaluates the information in its own unique way and comes up with a conclusion. The ultimate outcome is

decided upon by polling the majority of voters. In the method that I have presented, the Voting classifier is made by : (Random forest, KNN, and xgboost). The result that is acquired by voting will be the most effective of these three methods. Therefore, the purpose of this system is to use an ensemble-based hybrid algorithm in order to get superior outcomes compared to those achieved by employing a conventional machine learning method.

3.7 Evaluation and results Parameters

This section determines the evaluation parameter where it shows how well each of the machine learning classification models performs with respect to the accuracy, recall, F1-score, and precision. After that, separate classifiers were tested, and then they were combined with the voting classifier in order to get a greater level of performance. The suggested model was contrasted with the other three models, and the results were visualized. All assessment metrics are dependent on the various properties utilized. Accuracy (A), Precision (P), Recall (R), and F-Score (F) are calculated.

- Precision: Demonstrates how the attack classification model accurately categorized the different assaults, i.e., the model states that the different attacks belong to a class, and those attacks really do belong to that class. As seen in the following formula:

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

  TP, or True Positive, refers to the number of attacks that are part of a class and that were properly categorized, whereas FP, or False Positive, refers to the number of assaults that are part of a class but that were incorrectly classified.

- Recall: Demonstrates how the attack categorization model accurately categorizes different attacks if they fall under the same class. As seen in the following formula:

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

  where FN or False Negative is the number of attacks that were incorrectly categorized as belonging to a class of attack while TP or True Positive is the number of attacks that belong to a class and were properly classified.

- Accuracy: It provides the total number of attacks that the categorization models successfully identified as either true positives or true negatives. It is represented by the following formula.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

  where TN, or True Negative, is the number of offenses that were accurately identified as not falling under a certain category.

- F1 Score: This function returns the ratio of the precision to the recall. The following equation is a representation of it:

$$F1 - score = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall} \right) \tag{8}$$

4 Design Specification

The Edge-Based Intrusion Attack Classifier framework architecture combines an Infinite feature selection with PCA and a Hybrid Classifier as shown in Figure 3. The component infinite feature selection with PCA is discussed in 4.1. Then the component classification model includes three different tree-based models and voting models as discussed in 4.2.

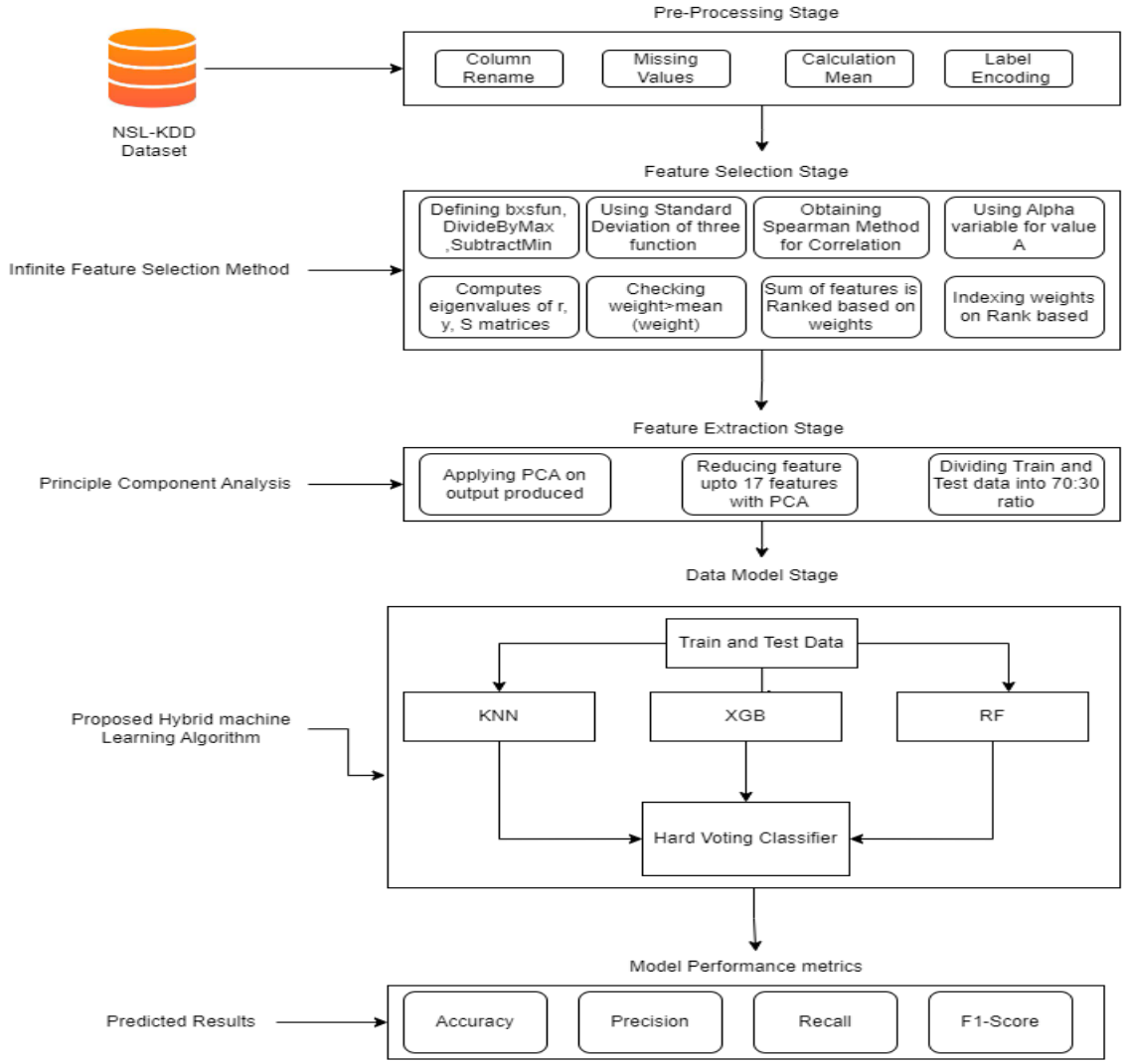


Figure 3: Edge-Based Intrusion Attack Classifier Framework

4.1 Infinite Feature Selection with Principle Component Analysis

As illustrated in Figure 3, the procedure is started by utilizing the NSL-KDD network traffic dataset. Following data retrieval, Null values are verified. At this point, the average of the missing data is used to derive the mean. Further, we used a label encoding approach to pre-process the data. The label values are transformed into a machine-readable

format via label encoding. The next step is feature selection, which is the most crucial component in this scenario. We employed infinite feature selection. where the mixing parameter, which is the alpha variable, is defined there. Then, three functions—bsxfun, DivideByMax, and SubtractMin were defined. Here, the vector is expanded into matrices of the same size and sorted by order using the bsxfun function. The next function is Subtract min, which subtracts the minimal value from the data, and DivideByMax, which gets the maximum value and normalizes the data. Additionally, we used the spearman approach to create the correlation. Then, we determined whether or not a null value was present. The standard deviation has been mostly utilized for the function we previously created. Additionally, we used the alpha variable to get the weights. Additionally, we have calculated the values of r, y, and S, where r is calculated using eigval, y using I and its inverse, and S is the total of all weights. To do so, we have ranked-ordered the weight and saved it as a feature.

All of the features have been chosen at this point. Additionally, as seen in Figure 3, we also employed principle component analysis, and a number of experiments with a feature set of 10, 15, and 20 have been performed. Here, PCA minimizes information loss while reducing the dataset's dimensionality. This is accomplished by repeatedly generating new, independent variables with the highest possible variance.

4.2 Hybrid Machine Learning Algorithm

As can be seen in figure 3, after the variables have been included, the dataset is next split into a training portion and a testing portion in the proportion of seventy percent to thirty percent. During the phase in which classification takes place, the vote classification approach is used when it comes to danger prediction. The voting classification strategy utilizes a combination of a wide variety of classifiers, such as XGBoost and Random Forest Classifier, in addition to a variety of other potential choices. The Hard Voting Classifier is a machine learning (ML) model that is applied to train an ensemble of many models and forecast an output (class) on the basis of their largest likelihood of picking class as the output. This is done in order to improve the accuracy of the prediction. A tool called a Hard Voting Classifier is used in order to accomplish this goal. The sum of the results from each classifier is computed, and then they are sent to the classifier responsible for voting. The voting results that garnered the biggest majority of votes are then used to make a determination about which output class will be produced. A single model is built, the training of which is performed by the deployment of these models, and the output is projected based on the aggregated majority of votes that each of these models has received for each output class. We were able to achieve this result by using a hybrid machine learning method rather than building entirely distinct models and evaluating the degree of accuracy achieved by each of those models on an individual basis.

5 Implementation

Considering the expense of establishing a real-world cloud edge computing infrastructure, we have opted to create the proposed hybrid technique using a jupyter notebook. The Google Colaboratory tool is utilized in this instance to demonstrate and perform the intricate calculations required for an algorithm. The bulk of faults may be found using a raspberry pi or any expense edge device before the proposed solution is used in the real world. Therefore, using this arrangement has the advantage of allowing us

to take preventive action without having to spend money on putting up a live testing environment.

The Google Colaboratory, a free tool provided by Google Research that enables users to develop and run Python code in their web browsers, is also used. Moreover, Python version 3 was used along with libraries like pandas, numpy, math, Scikit learn, matplotlib, and ensemble were used in the development of this project.

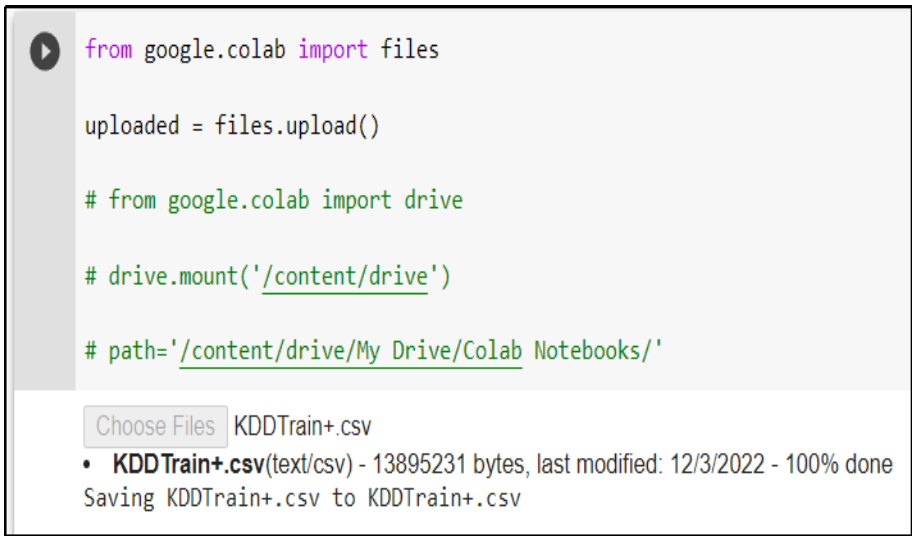
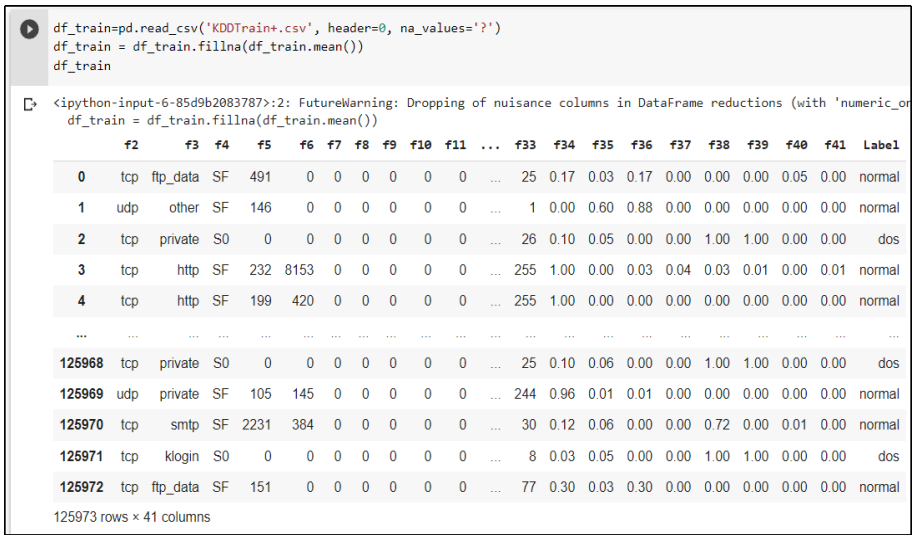


Figure 4: Uploading the Dataset

Firstly, we have uploaded our NSL-KDD dataset to jupyter notebook, as seen in figure 4 above. Additionally, we can retrieve data from the drive as well.



```
df_train=pd.read_csv('KDDTrain+.csv', header=0, na_values='?')
df_train = df_train.fillna(df_train.mean())
df_train
```

```
<ipython-input-6-85d9b2083787>:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_or
  df_train = df_train.fillna(df_train.mean())
```

	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	...	f33	f34	f35	f36	f37	f38	f39	f40	f41	Label
0	tcp	ftp_data	SF	491	0	0	0	0	0	0	...	25	0.17	0.03	0.17	0.00	0.00	0.00	0.05	0.00	normal
1	udp	other	SF	146	0	0	0	0	0	0	...	1	0.00	0.60	0.88	0.00	0.00	0.00	0.00	0.00	normal
2	tcp	private	S0	0	0	0	0	0	0	0	...	26	0.10	0.05	0.00	0.00	1.00	1.00	0.00	0.00	dos
3	tcp	http	SF	232	8153	0	0	0	0	0	...	255	1.00	0.00	0.03	0.04	0.03	0.01	0.00	0.01	normal
4	tcp	http	SF	199	420	0	0	0	0	0	...	255	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	normal
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
125968	tcp	private	S0	0	0	0	0	0	0	0	...	25	0.10	0.06	0.00	0.00	1.00	1.00	0.00	0.00	dos
125969	udp	private	SF	105	145	0	0	0	0	0	...	244	0.96	0.01	0.01	0.00	0.00	0.00	0.00	0.00	normal
125970	tcp	smtp	SF	2231	384	0	0	0	0	0	...	30	0.12	0.06	0.00	0.00	0.72	0.00	0.01	0.00	normal
125971	tcp	klogin	S0	0	0	0	0	0	0	0	...	8	0.03	0.05	0.00	0.00	1.00	1.00	0.00	0.00	dos
125972	tcp	ftp_data	SF	151	0	0	0	0	0	0	...	77	0.30	0.03	0.30	0.00	0.00	0.00	0.00	0.00	normal

125973 rows × 41 columns

Figure 5: Reading the Dataset

Additionally, we have imported the necessary libraries, read the CSV file as shown in figure 5, and displayed the data. To further clean up our dataset, we additionally looked for features that may contain null values and obtained an average of nan values, by calculating the mean.

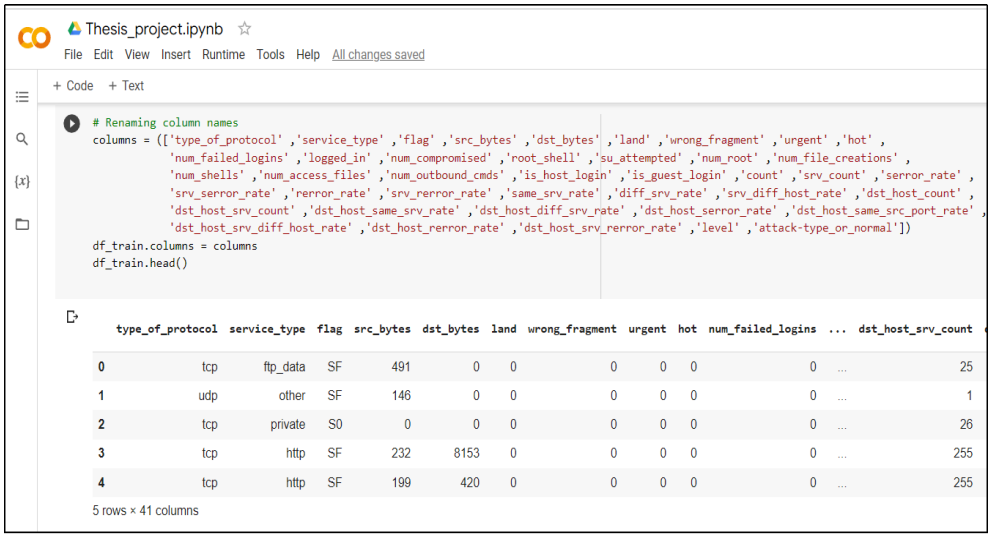


Figure 6: Renaming the columns

In order to investigate and preprocess the data, we have renamed the column names and presented the data, as seen in the figure 6 that is located above.

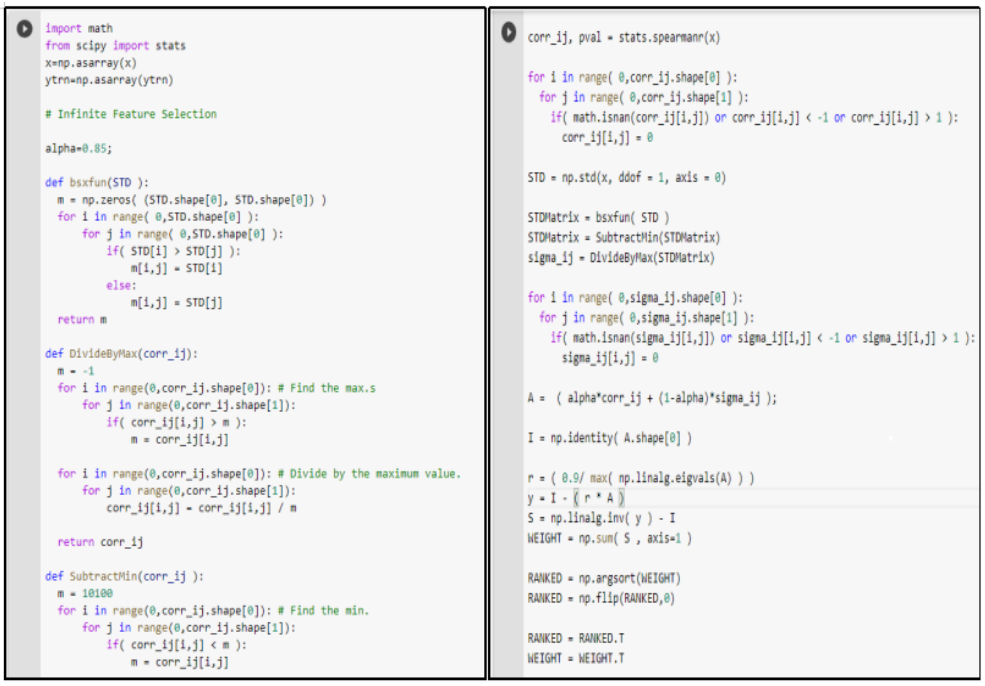
```
# Pre-processing Dataset
lbEn = LabelEncoder()
x['type_of_protocol']=lbEn.fit_transform(x['type_of_protocol'])
x['service_type']=lbEn.fit_transform(x['service_type'])
x['flag']=lbEn.fit_transform(x['flag'])
x['attack-type_or_normal']=lbEn.fit_transform(x['attack-type_or_normal'])

ytrn=lbEn.fit_transform(ytrn)
```

Figure 7: Label Encoding

We have performed some preliminary processing on the dataset using Label encoding, as can be seen in figure 7. The target column includes the kind of protocol, flag, service, and attack type or normal when we apply Label Encoding to the NSL-KDDdataset. The labels are transformed into machine-readable numeric values.

As shown in the below figure 8, we have used a feature selection strategy known as infinite feature selection as it makes use of the convergence characteristics of power series of matrices to determine the significance of a feature in comparison to all of the other features taken together, and after that, we have ranked the output that has been created according to the weights of the features.



```
import math
from scipy import stats
x=np.asarray(x)
ytrn=np.asarray(ytrn)

# Infinite Feature Selection

alpha=0.85;

def bsxfun(STD ):
  m = np.zeros( (STD.shape[0], STD.shape[0]) )
  for i in range( 0,STD.shape[0] ):
      for j in range( 0,STD.shape[0] ):
          if( STD[i] > STD[j] ):
              m[i,j] = STD[i]
          else:
              m[i,j] = STD[j]
  return m

def DivideByMax(corr_ij):
  m = -1
  for i in range(0,corr_ij.shape[0]): # Find the max.s
      for j in range(0,corr_ij.shape[1]):
          if( corr_ij[i,j] > m ):
              m = corr_ij[i,j]

  for i in range(0,corr_ij.shape[0]): # Divide by the maximum value.
      for j in range(0,corr_ij.shape[1]):
          corr_ij[i,j] = corr_ij[i,j] / m

  return corr_ij

def SubtractMin(corr_ij ):
  m = 10100
  for i in range(0,corr_ij.shape[0]): # Find the min.
      for j in range(0,corr_ij.shape[1]):
          if( corr_ij[i,j] < m ):
              m = corr_ij[i,j]
```

```
corr_ij, pval = stats.spearmanr(x)

for i in range( 0,corr_ij.shape[0] ):
  for j in range( 0,corr_ij.shape[1] ):
    if( math.isnan(corr_ij[i,j]) or corr_ij[i,j] < -1 or corr_ij[i,j] > 1 ):
      corr_ij[i,j] = 0

STD = np.std(x, ddof = 1, axis = 0)

STDMatrix = bsxfun( STD )
STDMatrix = SubtractMin(STDMatrix)
sigma_ij = DivideByMax(STDMatrix)

for i in range( 0,sigma_ij.shape[0] ):
  for j in range( 0,sigma_ij.shape[1] ):
    if( math.isnan(sigma_ij[i,j]) or sigma_ij[i,j] < -1 or sigma_ij[i,j] > 1 ):
      sigma_ij[i,j] = 0

A =  ( alpha*corr_ij + (1-alpha)*sigma_ij );

I = np.identity( A.shape[0] )

r = ( 0.9/ max( np.linalg.eigvals(A) ) )
y = I - ( r * A )
S = np.linalg.inv( y ) - I
WEIGHT = np.sum( S , axis=1 )

RANKED = np.argsort(WEIGHT)
RANKED = np.flip(RANKED,0)

RANKED = RANKED.T
WEIGHT = WEIGHT.T
```

Figure 8: Infinite feature selection

The preceding figure 9 demonstrates that we extracted features using principal component analysis. After that, we utilized values as 17 features by altering the output generated by infinite feature selection, which is the indexed rank weight. This was done so that we could use the result as features. In addition to that, we have also split the data from the test and the train into a percentage ratio of 70:30.

```
# Applying Feature Extraction [ Princple Component Analysis]
pca = PCA(n_components=15)
nSelx = pca.fit_transform(Selx);

# Dividing data into training and testing
X_train,X_test,Y_train,Y_test=train_test_split(nSelx,ytrn,random_state=42,test_size=0.3)
```

Figure 9: Principle Component Analysis

In conclusion, we have utilized data modeling as shown in figure 10, which consists of three individual classifiers called KNN, RF, and XGB. After that, we utilized a hard voting classifier to calculate the voting given by separate classifiers for the prediction of X and Y train data.

```
# Applying Data Modelling
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
model_1 = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
model_2 = XGBClassifier()
model_3 = RandomForestClassifier()


classifier = VotingClassifier(
    estimators=[('KNN', model_1), ('XGB', model_2), ('RF', model_3)], voting='hard')

classifier.fit(X_train, Y_train)
Output=classifier.predict(X_test)
```

Figure 10: Data Modelling

6 Evaluation

In this part, a thorough evaluation of the suggested hybrid machine-learning method is carried out so that future improvements may be made. In order to generate results, the algorithm is put into practice and carried out by using Google Colab's Jupyter Notebook. After that, the suggested method is evaluated with regard to a number of criteria, including Recall, Accuracy, F1-score, and Precision. The suggested approach for hybrid machine learning is based on ensemble learning techniques, which are then contrasted with multiple tree-based model techniques that are already in use. In order to carry out the tests, the current models were implemented.

6.1 Results

Following the successful development of the hybrid machine learning algorithm, we proceeded to test and assess the performance of various algorithms by taking into consideration a range of metrics including recall, accuracy, f1-score, and precision.

```
ACC  = metrics.accuracy_score(Y_test, Output)
P  = metrics.precision_score(Y_test, Output,average='weighted')
R  = metrics.recall_score(Y_test, Output,average='weighted')
F  = metrics.f1_score(Y_test, Output,average='weighted')

plotdata = pd.DataFrame({
    "Performance Parameter Comparison":[ACC*100,P*100,R*100,F*100],},
    index=["Accuracy","Precision" ,"Recall","F1-Score"])
print(plotdata)
```

```
           Performance Parameter Comparison
Accuracy                          99.338484
Precision                         99.337542
Recall                            99.338484
F1-Score                          99.321603
```

Figure 11: Proposed algorithms results

Based on the findings and figure 11, 12, it is clear that the performance of the proposed hybrid machine-learning algorithm is superior to that of other machine-learning models when it comes to the identification of harmful or anomalous behavior.

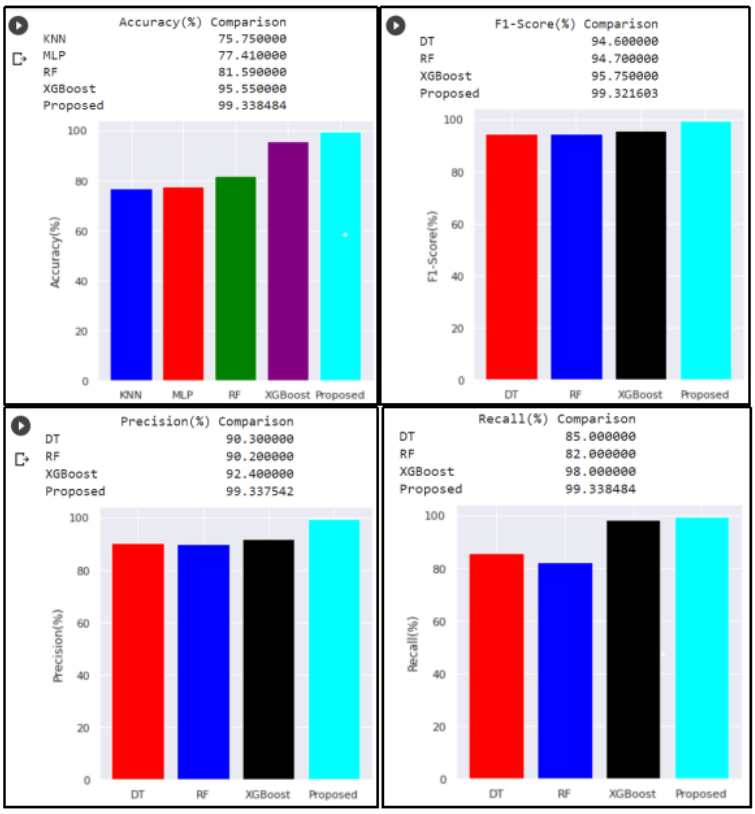


Figure 12: Accuracy, Recall, Precision, F1 score Comparison

Moreover, for the experiment purpose, we put the suggested method to the test by attempting modification with the feature up to twenty features. Key considerations in the study of the results from this experiment were determining whether or not an increase in the number of characteristics causes algorithms to have a greater influence.

```
ACC  = metrics.accuracy_score(Y_test, Output)
P  = metrics.precision_score(Y_test, Output,average='weighted')
R  = metrics.recall_score(Y_test, Output,average='weighted')
F  = metrics.f1_score(Y_test, Output,average='weighted')

plotdata = pd.DataFrame({
    "Performance Parameter Comparison":[ACC*100,P*100,R*100,F*100],},
    index=["Accuracy","Precision" ,"Recall","F1-Score"])
print(plotdata)
```

```
           Performance Parameter Comparison
Accuracy                          99.296147
Precision                         99.294820
Recall                            99.296147
F1-Score                          99.279134
```

Figure 13: Proposed algorithms results with 20 features

Upon implementing 20 features, still, the performance is 99.29% which is evident that our model is a successful and effective classifier that is very close to one hundred percent

and also can immediately represent its use when utilized in an actual work environment.

6.2 Discussion

On the basis of the findings of the experiment described above, it is evident that the Edge-Based Intrusion Attack Classifier Framework, which consists of a Hybrid machine method with infinite feature selection and PCA, is successful at detecting edge-based intrusion attacks based on evaluation criteria. It also demonstrates that the strategy we recommended would stop the abnormalities in the experimental network dataset. In addition, the studies demonstrate that the proposed method is robust enough to survive any other ensemble machine learning methodologies that, on top of that variable feature set, have provided results that are superior to those produced by the proposed method. Additionally, as the world becomes more aware of edge computing and energy consumption, increasing computation and usage at the edge may lead to more edge-based solutions for both security and energy consumption. This can happen because the world is becoming more conscious of edge computing. On the other hand, the suggested algorithm will be effective in accomplishing its goal of getting higher outcomes in a real-time setting. When implemented as part of a feature set, it has the potential to reach even better outcomes.

7 Conclusion and Future Work

Edge network security is quickly becoming one of the most critical pieces of technology for smart applications across a wide variety of industries. This is mostly attributable to the growing demand for edge computing services and applications. However, when it comes to the security of networks, there are significant vulnerability risks in the network traffic sector. In order to address these concerns, we carried out research aimed at enhancing the network security of edge computing and securing edge devices via the use of a suggested framework. Combining three tree-based methods with PCA and infinite feature selection was one of the suggestions that we made in order to achieve the best potential outcomes. The performance of safeguarding the network traffic between the edge environment was effectively improved by the technique presented in this research when compared to a variety of other current ensemble algorithms. Additionally, the goal of minimizing network intrusion by achieving 99.33% is the highest possible percentage that was accomplished by this study. The outcomes are visible and can be substantiated by reviewing the outputs that were generated by the Jupyter Notebook.

Furthermore, this, in turn, is increasing the amount of workload that is being done at edge devices at any given point in time, which in turn leads to an increase in the overall consumption of energy and usage. Therefore, there is room for extra research to be conducted in order to make the suggested method more efficient. Furthermore, the employment of datasets at the edge may also be regarded as a factor that puts forth a greater scope of development.

References

Agrawal, S., Sarkar, S., Aouedi, O., Yenduri, G., Piamrat, K., Alazab, M., Bhattacharya, S., Maddikunta, P. K. R. & Gadekallu, T. R. (2022), 'Federated learning for intrusion

detection system: Concepts, challenges and future directions', *Computer Communications* .

Alwarafy, A., Al-Thelaya, K. A., Abdallah, M., Schneider, J. & Hamdi, M. (2020), 'A survey on security and privacy issues in edge-computing-assisted internet of things', *IEEE Internet of Things Journal* **8**(6), 4004–4022.

Alzahrani, A. O. & Alenazi, M. J. (2021), 'Designing a network intrusion detection system based on machine learning for software defined networks', *Future Internet* **13**(5), 111.

Bai, Y., Chen, L., Li, J., Wu, J., Zhou, P., Xu, Z. & Xu, J. (2022), 'Multi-core federated learning for mobile edge computing platforms', *IEEE Internet of Things Journal* .

Chen, L., Tang, S., Balasubramanian, V., Xia, J., Zhou, F. & Fan, L. (2022), 'Physical-layer security based mobile edge computing for emerging cyber physical systems', *Computer Communications* **194**, 180–188.

Chen, W., Chen, Y., Jiao, Y. & Liu, Q. (2021), Security awareness scheme of edge computing in iot systems, *in* '2021 IEEE 4th International Conference on Computer and Communication Engineering Technology (CCET)', IEEE, pp. 332–335.

Chen, X. (2020), 'A security integration model for private data of intelligent mobile communication based on edge computing', *Computer Communications* **162**, 204–211.

Feng, Y., Wang, T., Hu, B., Yang, C. & Tan, J. (2020), 'An integrated method for high-dimensional imbalanced assembly quality prediction supported by edge computing', *IEEE Access* **8**, 71279–71290.

Guezzaz, A., Benkirane, S., Mohyeddine, M., Attou, H. & Douiba, M. (2022), 'A lightweight hybrid intrusion detection framework using machine learning for edge-based iiot security', *International Arab Journal of Information Technology* **19**(5).

Khan, A. R., Kashif, M., Jhaveri, R. H., Raut, R., Saba, T. & Bahaj, S. A. (2022), 'Deep learning for intrusion detection and security of internet of things (iot): current analysis, challenges, and possible solutions', *Security and Communication Networks* **2022**.

Kozik, R., Choraś, M., Ficco, M. & Palmieri, F. (2018), 'A scalable distributed machine learning approach for attack detection in edge computing environments', *Journal of Parallel and Distributed Computing* **119**, 18–26.

Liu, Y., Wang, T., Zhang, S., Liu, X. & Liu, X. (2020), 'Artificial intelligence aware and security-enhanced traceback technique in mobile edge computing', *Computer Communications* **161**, 375–386.

Roffo, G., Melzi, S. & Cristani, M. (2015), Infinite feature selection, *in* 'Proceedings of the IEEE International Conference on Computer Vision', pp. 4202–4210.

Singh, A., Chatterjee, K. & Satapathy, S. C. (2022), 'An edge based hybrid intrusion detection framework for mobile edge computing', *Complex & Intelligent Systems* **8**(5), 3719–3746.

Singh, S., Sulthana, R., Shewale, T., Chamola, V., Benslimane, A. & Sikdar, B. (2021), 'Machine-learning-assisted security and privacy provisioning for edge computing: A survey', *IEEE Internet of Things Journal* **9**(1), 236–260.

Tharwat, A. (2016), 'Principal component analysis: an overview', *Pattern Recognit* **3**(3), 197–240.

Wang, Z., Jiang, D., Lv, Z. & Song, H. (2022), A deep reinforcement learning based intrusion detection strategy for smart vehicular networks, *in* 'IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)', IEEE, pp. 1–6.

Xue, W., Shen, Y., Luo, C., Xu, W., Hu, W. & Seneviratne, A. (2022), 'A differential privacy-based classification system for edge computing in iot', *Computer Communications* **182**, 117–128.

Yang, R., Yu, F. R., Si, P., Yang, Z. & Zhang, Y. (2019), 'Integrated blockchain and edge computing systems: A survey, some research issues and challenges', *IEEE Communications Surveys & Tutorials* **21**(2), 1508–1532.

Yuan, D., Ota, K., Dong, M., Zhu, X., Wu, T., Zhang, L. & Ma, J. (2020), Intrusion detection for smart home security based on data augmentation with edge computing, *in* 'ICC 2020-2020 IEEE International Conference on Communications (ICC)', IEEE, pp. 1–6.

Yue, S., Ren, J., Qiao, N., Zhang, Y., Jiang, H., Zhang, Y. & Yang, Y. (2021), 'Todg: Distributed task offloading with delay guarantees for edge computing', *IEEE Transactions on Parallel and Distributed Systems* **33**(7), 1650–1665.

Zhang, P., Wang, Y., Kumar, N., Jiang, C. & Shi, G. (2021), 'A security-and privacy-preserving approach based on data disturbance for collaborative edge computing in social iot systems', *IEEE Transactions on Computational Social Systems* **9**(1), 97–108.

Zhou, C., Yu, Y., Yang, S. & Xu, H. (2021), 'Intelligent immunity based security defense system for multi-access edge computing network', *China Communications* **18**(1), 100–107.