NETWORK INTRUSION DETECTION SYSTEM USING ENSEMBLE METHODS AND ARTIFICIAL NEURAL NETWORKS

Kushagra Singh – 18BCE0017

Adhil Mohammed – 18BCE0056

Samyak Jain – 18BCE0083

Hariharan R S – 18BCE0671

A report submitted for the J component of

CSE3502 - Information Security Management

Supervisor: Dr. D. Ruby

School of Computer Science and Engineering

Vellore Institute of Technology, Vellore

May 2021

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count: 8023 words

Student Name: Kushagra Singh, Adhil Mohammed, Samyak Jain, Hariharan R S

Date of Submission: 31st May 2021

Signature: *Kushagra Singh, Adhil Mohammed, Samyak Jain, Hariharan R S*

TABLE OF CONTENTS

Abstract

In today's world, with rapid increase in the development of technology, internet usage and data generation have also increased significantly both in rural and urban environments. Huge amount of data is shared all over the internet and this data is one of the most valuable resources. With access to even a small part of this data, hackers can easily misuse it to get personal information like credit card and bank account details of the general public and perform any illegal actions like withdrawing it without their knowledge or blackmailing the data holder etc., thus resulting in an increase in the cybercrime cases in our society. Hence, safety and security of data through a network is considered as one of the most important needs of any company or organization, or even for personal networks used by members of a family. This calls for a Network Intrusion Detection System (IDS) to provide security and to detect if there is any anomaly in the network or not (virtual security threats).

IDS is implemented using the latest technologies such as Internet of Things and Machine Learning algorithms such as naive bayes, k-nearest neighbours, support vector machine, logical regression etc. to detect if any anomaly is present in the network or not in a real time environment. Also, choosing the best algorithm among the various ML algorithms available is a crucial task.

1. Problem Statement

1.1. Idea

We have proposed a multi-level IDS with seven ensemble machine learning algorithms that are running parallelly (level 1) and a deep learning algorithm - Forward Feedback ANN (level 2) which would help to overcome the problems of the existing IDS and optimally detect intrusion in any network.

1.2. Scope

Data these days has become an important part of our daily life. Starting with our personal details, bank transactions and business details, we need our data to be safe and secure when shared and used in any network. Business start-ups and companies are investing a huge amount of money in buying software and recruiting cyber security analysts to keep their data safe.

Intrusion detection network, shortly known as IDS plays an important role in securing computer networks by effectively detecting any anomaly in the network. A trained Intrusion detection system in real time, automatically gets details of the network, predicts outcome and alerts users in case of any intrusion detection.

Traditional technologies like firewalls are having limitations and with advancements in hacking tools, malware, adware and other security threats, network security is of high risk. This requires more advanced solutions for cyber security. Recent technologies like Intrusion detection system(IDS) and Intrusion prevention system(IPS) are successful in identifying anomalies in networks and preventing them beforehand. Studies show that the existing IDS have limitations and are not effective enough in identifying and classifying the attacks like Remote to Local(R2L), Denial of Service(DoS) and User to Root(U2R) on a network.

1.3. Novelty

The models that have been proposed have been found to be either underfitting or overfitting the datasets. Some of the models which worked well also had very long execution time and hence are not suitable to be used in real time.

We have proposed a new best Multi-level Network Intrusion Detection Model Design for real-time usage. It has two levels of training and data processing (level 1 - Machine Learning Algorithms and level 2 - Artificial neural network).

This model prevents overfitting and yields an optimal result in a short time duration.

1.4. Comparative Statement

Individual Classification Algorithms like Support Vector Machine, Logical Regression, Naïve Bayes, k-nearest neighbours do not predict accurately for large datasets. They result in underfitting IDS models to predict anomalies in all networks.

Bagging and Boosting ensemble algorithms have shown higher accuracy, precision and recall results in many research papers. But, these models are not tested in a real time environment. They result in overfitting models due to which the good results obtained are confined to this particular dataset. Their performance will not be as expected and high for other datasets.

Due to underfitting and overfitting individual algorithms, developing hybrid models and multi-level machine learning models doesn’t yield results as expected. They are highly influenced by the outcome of the previous level’s individual algorithm.

Running time to predict and detect if there is any intrusion in the network or not is very high for a dataset with larger instances if multiple algorithms are in the processing phase of proposed design.

Most of the proposed designs calculate their accuracy for part of the entire dataset (test-set) which is not used to train the model. Based on the results obtained, they directly come to a conclusion about their trained model. This is not an effective approach to validate ML algorithms. Their algorithm must be tested multiple times using techniques like K-fold cross validation and the average of all tests must be considered during analysis. Section 3 explains more about five different existing intrusion detection system, compares them with each other and points out their limitations

1.4.1. Introducing Deep Learning Self-Adaptive Misuse Network Intrusion Detection Systems [1]

Abstract

Intrusion detection systems (IDSs) are important when it comes to the security of ICT infrastructure. A stable process that can achieve high levels of attack detection (ADR) while putting false alarms below the levels available by the wrong IDS. The author proposes a method that distinguishes the benefits of self-study from MAPE-K frameworks for introducing the unstructured, self-serving and independent IDS.

Introduction

The author proposes a novel approach to creating a scary, flexible and independent IDS for misuse based on advanced intelligence (AI) techniques. IDS is active. These new pronouns are used to retrain the IDS automatically by adapting to the new environment. They incorporate their proposal into the context of the MAPE-K approach which draws the framework of independent and self-serving systems. They extensively reviewed their system in addition to many metrics and various environmental sites to provide conceptual evidence, supported by experimental results and demonstrating its potential for further expansion.

Methodology

The author describes their approach to abusing IDS in relation to environmental conditions, MAPE-K and Self-Taught Learning.
To keep the IDS detection rate at acceptable levels without thinking about environmental changes that may indicate the presence of previously unknown attacks. Their approach can empower IDSs to be independent and self-reliant by enabling them to adapt to their environment and contribute significantly to maintaining a high or at least acceptable level of security. This quality also greatly reduces safety professionals in the difficult task of re-learning IDS.
MARK-K is a trusted model for building independent and independent systems. MAPE-K has five functions that work beyond the Domain Specific System (DSS) and Context. The following figure shows the proposed program and details on its subparts, which are described as follows.

Algorithm

The author introduces an in-depth reading solution for NIDS. The authors use the self-taught reading method only as an unsupervised reading method to support a mathematically trained IDS. But their solution offers a powerful set of STLs in conjunction with the MAPE-K approach to delivering an in-depth learning approach to flexible IDS.

IDS algorithm based on Spectral Clustering (SC) and Deep Neural Networks (DNN). Authors use DNN and Long Short-Term Memory (LSTM) Networks Recurrent to enable anonymous detection.

Alternate Methodology

Some suggestions on alternate methodology will be that the internet of things is increasing its reach and now most of the applications use it so it will be better if we get realtime dataset by using iot in the systems and then use it to train it.

Mathematical Model

To highlight the benefits of our proposal and to compare it with other alternatives, they presented the results under different asset class metrics.

a: Accuracy: This metric measures the frequency of appropriate decisions.

$$Accuracy = \frac{\sum_{i=1}^{C} TP_i}{N}$$

b: Mean F-Measure (MFM) : Used to measure the balance between accuracy and memory.

$$MeanFMeasure = \frac{\sum_{i=1}^{C} FMeasure_i}{C}$$
$$FMeasure_i = \frac{2 \cdot Recall_i \cdot Precision_i}{Recall_i + Precision_i}$$
$$Precision_i = \frac{TP_i}{TP_i + FP_i}$$
$$Recall_i = \frac{TP_i}{TP_i + FN_i}$$

c: Average Accuracy: It is calculated as the normal recall between all categories of data.

$$AvgAccuracy = \frac{1}{C}\sum_{i=1}^{C} Recall_i$$

d: Attack Accuracy: This is used to calculate the model's ability to find only attack classes regardless of usual traffic. Index i = 1 represents the standard traffic category.

$$AttackAccuracy = \frac{1}{C-1}\sum_{i=2}^{C} Recall_i$$

e: Attack Detection Rate (ADR): Represents the accuracy of the attack stages.

$$ADR = \frac{\sum_{i=2}^{C} TP_i}{\sum_{i=2}^{C} TP_i + FP_i}$$

f: False Alarm Rate (FAR): This metric focuses on general traffic and measures FNs, that is, general conditions classified as offensive. Index i = 1 represents the standard traffic category.

$$FAR = \frac{FN_1}{TP_1 + FN_1}$$

Result

The author illustrates and compares the effectiveness of the flexible IDS by comparing it with the standard method, based on the metrics described in the paragraph. In all, they put the IDS into 100 natural variables, i.e., 100 differently compiled, associated with various attacks of each major category of KDDCup integrated databases.

Advantages

The author notes that it uses a process that is promoted by convolutional biology processes and also has the ability to solve performance problems during classification.
It is based on people's preconceived notions of opinion and gives a sense of uncertainty.

Limitations

After going through the paper we realized that it could have these limitations such as slowing down in training and looking for more memory space. It is computer-assisted because the separation of the test sample involves consideration of all training samples.

Conclusion

The author highlighted that it can provide any independent use of IDS independently, that is, to adapt to the ever-changing changes created in its network environment. In times of (and sometimes abrupt) changes, the IDS is able to maintain at least an acceptable level of attack, which can be attributed to a sudden decline, providing assistance to the IDS.

1.4.2. Semi-supervised multi-layered clustering model for intrusion detection [2]

Abstract

SMLC has the ability to learn from low-label data while achieving discovery performance compared to ML-monitored IDPS performance. The performance of SMLC is compared to the implementation of a moderately managed (triple-trained) model and integrated ML models, namely RandomForest, Bagging, and AdaboostM1 in two wireless benchmarks, NSL and Kyoto2006þ.

Introduction

The IDPS is an important component of network security infrastructure, as it monitors, detects, and identifies potential intruders. IDPSs are classified based on their ability to detect known and unknown attacks. The law-based IDPS makes decisions based on sets of rules defined by domain experts. The uniquely based IDPS also creates a standard model and detects deviations from this anat model attack.

Methodology

Local SMM learning styles are studied in emerging groups in various categories. The final forecast for the test situation is obtained by choosing to be divided by a majority of votes among all decisions from all levels. The contributions to this project are as follows: Design and construction of a moderately targeted model for network access activities. The proposed model assumes that the conditions of the same category remain close to the Euclidean space to reduce data input errors, which improves the final detection of ac-curacy. Based on the Euclidean estimation of the atomic particle's weight, we say that the time to model renewal and the time to split the proposed model can be significantly reduced by creating binary constructions in non-atomic groups only.), Bagging, and AdaBoostM1, as well as dual monitoring model, three training algorithms, depending on model accuracy, detection rate, false alarm, Mathews training time, training time, and test time in dual network information, NSL and Kyoto 2006p.

Proposed Design

The author discussed the SMLC model in terms of data integration. It follows the work shown by the author and takes advantage of the fact that consecutive collections of the K-Means algorithm are based on startup arguments to give distinction between their base dividers. Therefore, the data point / example may be in different parts with different layers. Collections with different layers may have overlapping but different data, but samelayer collections are individually integrated and may contain data points in one class or mul-tiple. To illustrate this, consider a two-dimensional database. The layer is defined as the K-Means element algorithm based on a single set of startup parameters (e.g., Seeds). Therefore, the point / sample of data may be in different collections with different layers.

Mathematical Model

The Information Gain Ratio (IGR) is used as the weight of the attribute, because it shows the use and importance of the attribute in determining the type of category, and is provided by

$$IGR(Y, A_j) = \frac{H(Y) - H(Y|A_j)}{H(A_j)}$$

When class Y and Ajis the jth attribute. Here, H (.) An entropy function provided by

$$H(X) = -\sum_{\forall i} P(x_i)\log_2[P(x_i)]$$

the whereP (.) operator of the opportunity and the index of the feasibility opportunities provided.

Result

With the rapid growth of data volumes, not only accu-racy acquisition but efficiency and robustness are also important. Although, in this paper, the use of the proposed model is done on a single machine, apparently, SMLC has the ability to effectively manage large amounts of data by distributing its computational costs to multiple devices as it provides excellent infrastructure for cutting large amounts of data on distributed computer systems.

Conclusion

SMLC has the ability to successfully manage large amounts of data by distributing its computing costs across multiple devices as it provides an excellent data processing infrastructure to a distributed computer system with multiple processing areas. However, SMLC has an extremely high testing time. Our future tasks will be to study the increase of SMLC, automation of the parameter adjustment process and reduce its testing time.

1.4.3. Building a Intrusion Detection System for IoT Environment using Machine Learning Techniques [3]

Abstract

An adversarial type system is built using a laptop type system that performs the tasks of sniffing and attacking toxins. The data of temperature, humidity and point of transmission transmitted to the ThinkSpeak platform using a wireless gateway taken from the sensor. In the standard category, sensor values are captured by the Node MCU and transferred to the Think Speak server which is stored and labeled as standard data.

Introduction

In the first section the author's test bed is designed to mimic an IoT-based environment. In the further phase a device such as a laptop is used to build resistance systems that cause network attacks. The information sent to the server is retained when the network is in normal and attack mode. In the last phase the machine learning models are designed to detect and distinguish network attacks and standard metrics are used to analyze the performance of classifiers.

Methodology

IoT testbeds are ideal for exploring various risks and ensuring solutions to reduce attacks.
A general design figure description of an IoT sleep-based program to study cyber attacks on an IoT device. The DHT11 sensor is connected to the Node MCU ESP8266 acting as a client. Information is transmitted by the wireless router to the ThinkSpeak platform a that acts as a gateway to the network. Attacker secretly captures data, converts and transmits messages between a Node MCU client and a ThinkSpeak server. This network mimics an IoT-enabled network that transmits encrypted and vulnerable data, powered by an attacker machine. The methods involved in building a system are explained as follows:
Create an imitation bed to mimic an IoT-based environment

Create attack programs to create an attack.

Algorithm

A machine learning algorithm used to detect and classify attacks. SVM, Naive Bayes, Decision Tree and Adaboost algorithm are the four algorithm categories used. SVM captures data points and finds the best data sharing aircraft using best practices. High-quality aircraft are an efficient separation tool with support vectors. Tree model based algo is Decision tree is a machine learning algorithm and resolutions defined in the tree nodes used for classification. Naïve Bayes uses the principle of Bayes law to create divisions using past opportunities, opportunities and opportunities backwards. Adaptive Boosting (Adaboost) is a reinforcing algorithm that combines the power of tree-based separators to create solid splits.

Evaluation

Data set: (Sensor480) Sensor Data use Think Speak server downloaded in CSV format to be compressed by Node MCU on the and used for analysis. The timestamp is removed from the downloaded data and 3 features (S1, S2, and S3) with the binary category label (Normal and Attack) are used to represent the data. The generated data has 480 records and is called Sensor480. To test the performance of classifiers, the Sensor480 developed by the authors is used for testing.Test Setup:.

Result

The confusion matrix as discussed by the author of the four algorithms is given below where 20% of the data in 480 records, 95 records taken to test the performance of the partition model. Table 2 shows the SVM results in data sharing.

Advantages

If we know the functionality and functionality of each device we are bound to maximize the efficient use of resources and recognize natural resources.

Limitations

As IoT systems are connected and interconnected by networks. The system does not provide minimal control despite security measures, and can lead to various types of network attacks.

Conclusion

Different devices must be used in the network so different data is considered in the design of the ML model. By overcoming these challenges a suitable natural ICT IDS can be developed. Safety factors must be considered at the beginning of the design phase of the process to make sure that IoT devices are saved from cyber threats.

1.4.4. Network Intrusion Detection System based PSOSVM for Cloud Computing [4]

Abstract

To mitigate this issue, intrusion detection systems (IDSs)play a crucial role in detecting attacks in the cloud space. The authors proposed an uncommon network access detection (NIDS) system that can monitor and analyze the flow of network traffic targeted at cloud space.

Introduction

Intervention programs are known for their high levels of false alarms and much of the research effort is still focused on finding effective, non-intrusive intrusions. However, there are many problems with current access systems. One of the biggest problems is the high number of fake alarms. False alarms are high and the ability to detect attacks is uncertain. The author has suggested that strategies should be put together to address some of these issues.

Methodology

User supports sequence S if S contains user sequence for this user. The support description is provided as a fraction of the total number obtained in sequence. In the SP prediction, the user's daily activities were taken as a sequence and each user's database which has daily user SP was created.

Algorithm

Many extensions have been recommended to make SVMs suitable for dealing with multiple classification related problems. While hardly any of the various methods mentioned in the literature are considered as a solution to common problems, SVM strategies these days are mature enough to work on many programmatic problems.

Alternate Methodology

Other suggestions for a different approach would be that the internet of things is expanding its reach and now most apps are using it so it would be better if we get real-time data using iot in programs and use it to train it.

Mathematical Model

The SVM method converts data into an F-element space that is usually large. It is interesting to keep in mind that the performance of SVM relies on the geometric features of the training data and not on the length of the input space. SVM training leads to the problem of using quadratic with boundary issues and one equation problem. The author demonstrates how its training for pattern recognition problem leads to give a quadratic application problem given below:

Minimize:

$$W(\alpha) = -\sum_{i=1}^{l} \alpha_i + \frac{1}{2}\sum_{i=1}^{l} \sum_{j=1}^{l} y_i \, y_j \alpha_i \alpha_j k(x_i, x_j)$$

$$\textit{Subject to} \;\; \sum_{i=1}^{l} y_i \, \alpha_i$$

$$\forall i: 0 \leq \alpha_i \leq C$$

Result

The LAN was operating in the real world, but was blown up by numerous attacks. For each TCP / IP connection, 41 different calculation and measurement features were extracted.

Advantages

The performance of SVM is an advantage because it depends not on the size of the input space but on the geometric features of the training set.

Limitations

A bad access system can have high accuracy but a high level of false positives.

Conclusion

A new approach was proposed and the test results show that the proposed approach improves the performance of the acquisition by reducing or eliminating both negative and negative profits. Proposed NIDS is trained and evaluated on the NSL-KDD data bench test results explained its effectiveness in detecting common behaviors and detecting high accuracy detection attacks and low levels of false alarms. In addition, comparisons with other related IDS were made and our proposed system surpassed these programs.

1.4.5. Performance Analysis of Machine Learning Algorithms in Intrusion Detection System [5]

Abstract

The IDS is one of the supporting layers working on information security. IDSs give a salty surrounding for business and keep them away from non suitable network activities. Recently, Machine Learning (ML) algorithms have been used in IDS to identify and classify security threats. The author examines studies of various ML algorithm comparisons used in IDS for several uses such as fog computing, Internet of Things (IoT), big data handling huge amounts of streaming data, smart city, and 5G network.

Introduction

The interpreter's level of data increase makes the old data management system more difficult due to time and resources. Big data is very huge and so complex in the environment to manage such type of data and they require powerful technology and smart algorithms developed.AIDS plays an important role in detecting attacks. IDS is a system that will monitor network traffic for the purpose of detecting any suspicious activity and known threats.

Methodology

An IDS can be a program of hardware or software that automatically checks, identifies attacks or intruders, and alerts the computer or network. This warning report helps the head or administrator or user identify and resolve risks in the system or in a network. Other common methods of entry access are: Anomaly-based detection, Signature-based detection and Hybrid-based detection. Behavioral anomaly-induced intrusion detection is also called behavioral-based detection, as this process modifies user performance, network, and capture systems and after that creates an alarm or alert to the controller whenever it is unusual or the behavior deviates from normal behavior. Signature-based IDS is also called information-based acquisition. This method depends on data from a database that contains the before already occurred attack signature and previously known system vulnerabilities. The Hybrid-based detection system is a combination of anomaly-based detection and signature-based intrusion detection. Most IDSs use any incorrect login detection or signature.

Algorithms

For IDS the ML algos work precisely in getting large-scale data attacks in less time.

- Support Vector Machine (SVM)
- Logistic Regression (LR)
- Linear Discriminant Analysis (LDA)
- Classification and Regression Tree (CART)
- Random Forest (RF)
- K-means
- Principal Component Analysis (PCA)
- Semi-Supervised ML algorithm

Alternate Methodology

Semi-supervised Multi-Layered Clustering (SMLC)) is a model for network acquisition and blocking. SMLC has the ability to learn from low-label data while achieving discovery performance compared to ML-monitored IDPS performance. The performance of SMLC is compared to the implementation of a moderately managed (triple-trained) model and integrated

ML models, namely RandomForest, Bagging, and AdaboostM1 in two wireless benchmarks, NSL and Kyoto2006þ..

Mathematical Model

The efficiency of ML algorithms can be calculated using many metrics such as, precision, accuracy, memory and FScore etc

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

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

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

F-Measure is also called F-Score [11], which describes the average rate of Recall and Precision. Measured by formula:

$$F - Score = \frac{2(R*P)}{R+P}$$

The efficiency and effectiveness of detection of entry is calculated by the following metrics, in addition to detecting the findings of entry most researchers use metrics known as Detection rate and false positives. Good IDS should have a very high detection rate and a very low level of false identification. Calculated by formula:

$$Detection\ rate = \frac{TP}{TP+FN}$$

$$False\ Positive\ rate = \frac{FP}{FP+TP}$$

Result

Comparison of performance of various ML algorithms in IDS. After the comparison, the authors noted that the performance of the algos also relies on the length of the database and the applications used.

Advantages

Visibility. The IDS provides a clear view of what is happening within your network. It is an important source of information about the suspicious or malicious network involved. There are a number of methods that can be used in IDS that allow you to follow the depth of the network deeply. Protection. IDS adds a layer of protection to your security profile

Limitations

IDSs are notorious for removing fake stuff - alarm bells when nothing is wrong. Although you can change settings to reduce the number of false positives, you will never completely eliminate the need for positive feedback. False negatives. IDS can also be missed. Technology is evolving, but IDS does not always capture everything.

Conclusion

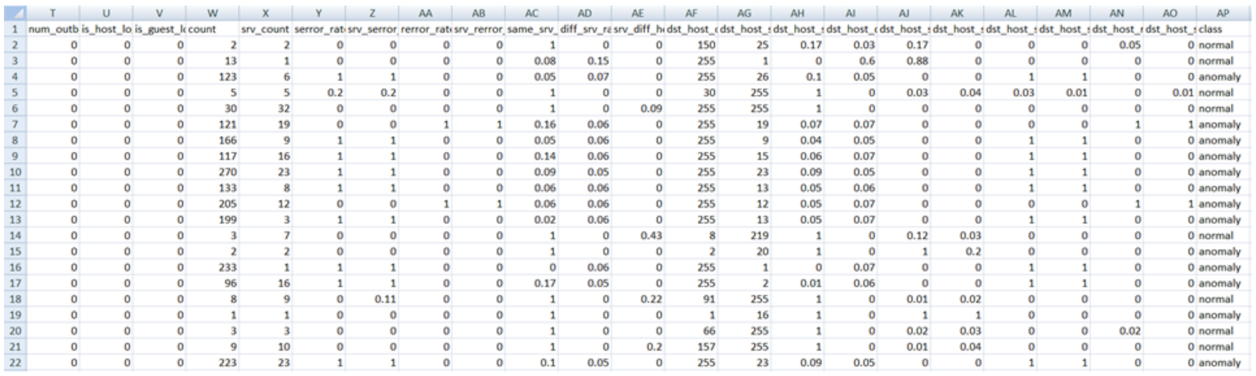
Detection rate & accuracy & false positive rating relies not only on the algorithm but also on the application area. In the future, the author will conduct an in-depth research on ML algorithms to provide better IDS solutions by taking real-time data.

1.5. Dataset

The dataset for Network Intrusion Detection System (NIDS) is taken from Kaggle. The dataset consists of a wide variety of intrusions simulated in a military network environment.

A connection is a sequence of TCP packets starting and ending at some time duration between which data flows to and from a source IP address to a target IP address under some well-defined protocol. Each connection record consists of about 100 bytes.

Each row which consists of 41 features are obtained from normal and attack data (3 qualitative and 38 quantitative features) .The class variable has two categories: Normal and Anomaly.



	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP
1	num_outb	is_host_lo	is_guest_l	count	srv_count	serror_rat	srv_serror	rerror_rat	srv_rerror	same_srv_	diff_srv_ra	srv_diff_h	dst_host_c	dst_host_s	dst_host_s	dst_host_c	dst_host_s	dst_host_s	dst_host_s	dst_host_s	dst_host_r	dst_host_s	class
2	0	0	0	2	2	0	0	0	0	1	0	0	150	25	0.17	0.03	0.17	0	0	0	0.05	0	normal
3	0	0	0	13	1	0	0	0	0	0.08	0.15	0	255	1	0	0.6	0.88	0	0	0	0	0	normal
4	0	0	0	123	6	1	1	0	0	0.05	0.07	0	255	26	0.1	0.05	0	0	1	1	0	0	anomaly
5	0	0	0	5	5	0.2	0.2	0	0	1	0	0	30	255	1	0	0.03	0.04	0.03	0.01	0	0.01	normal
6	0	0	0	30	32	0	0	0	0	1	0	0.09	255	255	1	0	0	0	0	0	0	0	normal
7	0	0	0	121	19	0	0	1	1	0.16	0.06	0	255	19	0.07	0.07	0	0	0	0	1	1	anomaly
8	0	0	0	166	9	1	1	0	0	0.05	0.06	0	255	9	0.04	0.05	0	0	1	1	0	0	anomaly
9	0	0	0	117	16	1	1	0	0	0.14	0.06	0	255	15	0.06	0.07	0	0	1	1	0	0	anomaly
10	0	0	0	270	23	1	1	0	0	0.09	0.05	0	255	23	0.09	0.05	0	0	1	1	0	0	anomaly
11	0	0	0	133	8	1	1	0	0	0.06	0.06	0	255	13	0.05	0.06	0	0	1	1	0	0	anomaly
12	0	0	0	205	12	0	0	1	1	0.06	0.06	0	255	12	0.05	0.07	0	0	0	0	1	1	anomaly
13	0	0	0	199	3	1	1	0	0	0.02	0.06	0	255	13	0.05	0.07	0	0	1	1	0	0	anomaly
14	0	0	0	3	7	0	0	0	0	1	0	0.43	8	219	1	0	0.12	0.03	0	0	0	0	normal
15	0	0	0	2	2	0	0	0	0	1	0	0	2	20	1	0	1	0.2	0	0	0	0	anomaly
16	0	0	0	233	1	1	1	0	0	0	0.06	0	255	1	0	0.07	0	0	1	1	0	0	anomaly
17	0	0	0	96	16	1	1	0	0	0.17	0.05	0	255	2	0.01	0.06	0	0	1	1	0	0	anomaly
18	0	0	0	8	9	0	0.11	0	0	1	0	0.22	91	255	1	0	0.01	0.02	0	0	0	0	normal
19	0	0	0	1	1	0	0	0	0	1	0	0	1	16	1	0	1	1	0	0	0	0	anomaly
20	0	0	0	3	3	0	0	0	0	1	0	0	66	255	1	0	0.02	0.03	0	0	0.02	0	normal
21	0	0	0	9	10	0	0	0	0	1	0	0.2	157	255	1	0	0.01	0.04	0	0	0	0	normal
22	0	0	0	223	23	1	1	0	0	0.1	0.05	0	255	23	0.09	0.05	0	0	1	1	0	0	anomaly

1.6. Test Bed

Hardware Requirements:

- Intel i5 Processor
- Minimum 4GB of RAM
- 10 GB of storage space (for software and dataset)

Software Requirements:

- Python 3.6
- Tensorflow
- Scikit-Learn

1.7. Expected Result

The final model obtained after processing through proposed design is expected to perform better than individual normal classification models like support vector machine, naïve bayes, logical regression etc. and individual ensemble classification algorithms like random forest, XG boost etc. This eliminates the problem of resulting in an underfitting model to detect intrusion in a network.

The resultant IDS model will be trained multiple times in a multi-level design, due to which the overfitting nature of the model in one level can be overcome in subsequent levels. Using multiple algorithms in the same level ensures that at-least one of those algorithms predicts correctly for any data instance.

From analysis we can understand that using ML models again after level 1 would result in getting influenced by the result of previous levels. For instance, using a voting classifier that determines output based on what the majority of models say in the previous level will result in an ineffective, highly influenced model due to which if the majority of previous level ML models goes wrong, then the entire system goes wrong. To overcome it, we have used a deep learning algorithm (artificial neural network) which cannot be influenced by previous levels and is self-tuned according to dataset and works efficiently.

Sequentially running all ensembled algorithms in level 1 of proposed design is time consuming. Hence, all seven models as seven different threads would run in each core of the computer. Due to this, time to finish level 1 processing is reduced from sum of run time of all algorithms to maximum run time of all algorithms.

For each data instance, the results are obtained in level 1 and level 2 of proposed design after processing it an optimal number of times. In Artificial neural network(feed-forward), the data as a signal moves forward and backward multiple times and updates values accordingly. This ensures that the result obtained after ANN can be directly considered for validating the proposed system.

2. Architecture

2.1. High Level Design

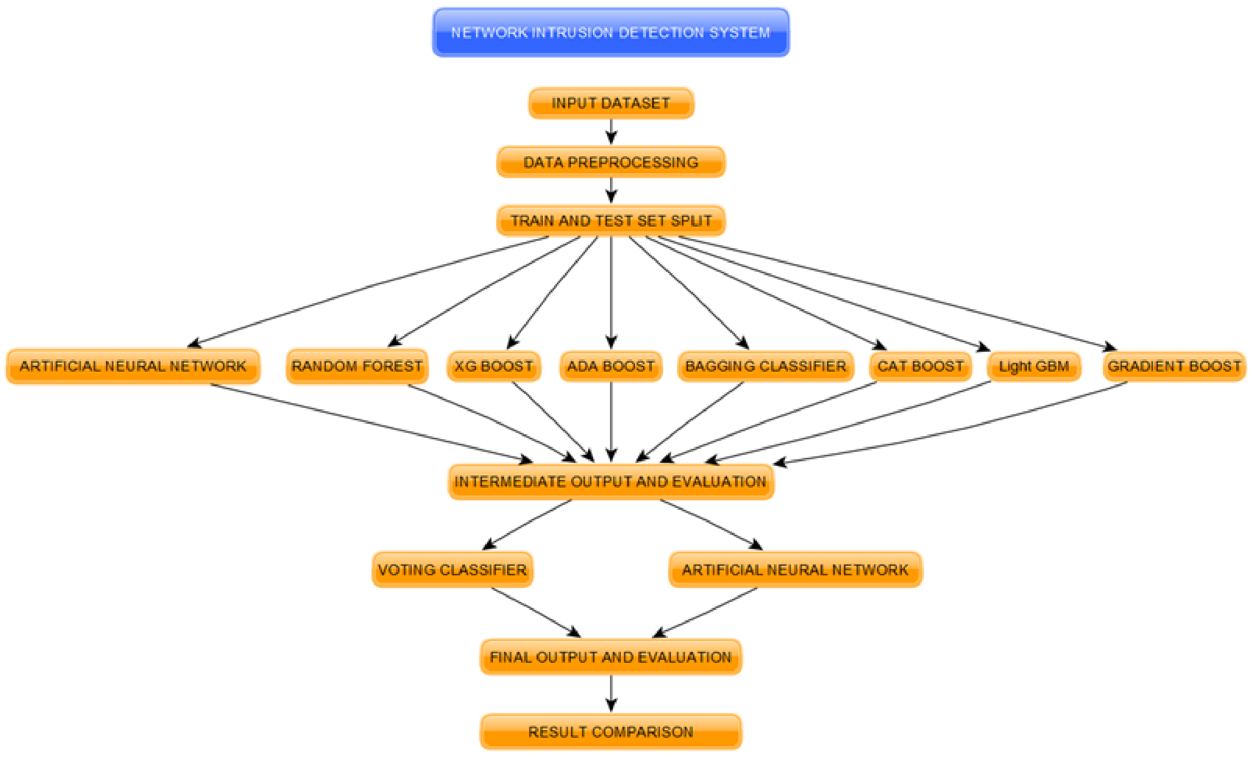


Fig. 1. Detailed diagram explaining the components of the Proposed System

Figure 1 shows the architecture of the proposed intrusion detection system. From the problem statement, we understood the need for an effective intrusion detection system in a real time environment. We propose a system which has two levels of training: the model using machine learning ensemble methods in level 1 and deep learning artificial neural network in level 2. The proposed system gets input dataset, loads it and pre-process it to remove noise, fill missing values, eliminate unwanted columns like name of company from where the instance generated, reference number of instance etc.. Once the preprocessing is done, it encodes the strings into numbers using label encoding which is machine understandable form and splits the entire dataset into train and test set in ratio of 8:2. The train set is now given as input to level 1 of model training phase where seven ensemble methods (bagging and boosting) is considered as seven different threads and runs parallely in each core of the computer (multi-processing). Then, the trained model is tested against the test set and the result obtained is given as input to artificial neural network in input layer where after giving equal weightage to all input nodes (ensemble machine learning models), the data is processed through hidden layer and result is obtained through output nodes as 0(normal) or 1(anomaly in network). Now, the obtained result can be compared with the actual results to validate the proposed solution for an Intrusion Detection System.

2.2. Low Level Design

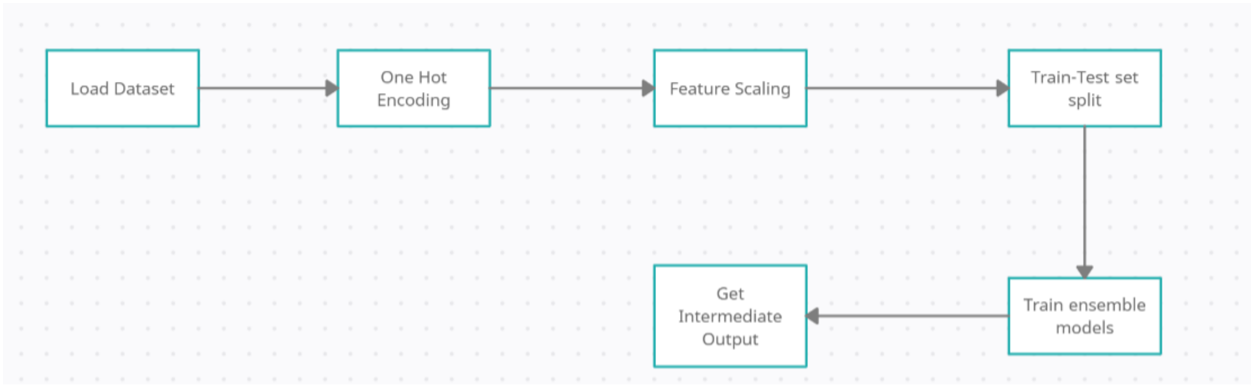


Fig 2: Design for training ensemble model

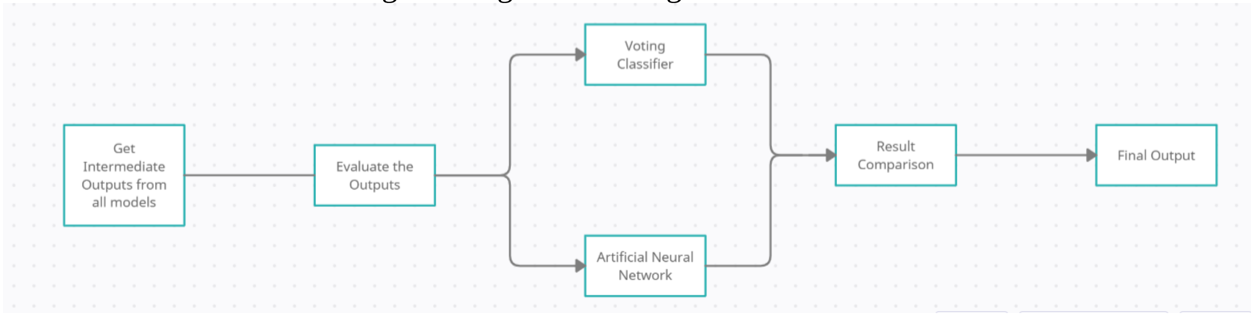


Fig 3: Design for Comparative analysis

2.2.1. Data Acquisition

The first phase of implementation is to select an appropriate dataset to train machine learning models for detecting anomalies in any network. Based on the correctness of data provided by the dataset, the trained model will behave in a real time environment. Once a model is set in a real time environment, it can automatically generate a dataset from real time data and get trained after a regular interval of time from the same.

Sample dataset : https://www.kaggle.com/sampadab17/network-intrusion-detection

2.2.2. Data Preprocessing

A good dataset must have a very low number of missing values and minimal noise. This phase aims at pre-processing the input dataset by selecting columns from dataset to be given as input, removing noises that falsely train (wrongly train) the machine learning model, filling missing values of any cell with average/minimum/maximum value to the cell attribute column. Also, this phase label encodes columns with strings into numerical format for the machine to understand the dataset better (strings cannot be given as input for training models).

2.2.3. Splitting into Training and Test Set

After the data pre- processing phase, the dataset is split into two parts of instances. One part is called the training set which is given as input to the machine learning model to get trained from that data instances. Another part is called the test set which is used to evaluate the performance of trained machine learning models. Once the model is trained with both features and results of data instances, it will be made to predict results for the features of the test set's data instances. This result will be compared with the original result to evaluate the model. The dataset is split into a train and test set with a standard ratio of 8:2.

2.2.4. Model Selection

In this phase of implementation, we select a model for training the dataset. The proposed design ignores normal classification algorithms like support vector machine, naïve bayes, logical regression, k-nearest neighbour etc.. and uses ensemble algorithms for training the model.

Ensemble algorithms include bagging and boosting techniques which splits input dataset into multiple random clusters of input dataset instances and trains them parallelly and sequentially respectively. Bagging algorithms in proposed design include random forest and bagging classifiers using decision trees . Boosting algorithms in the proposed design include gradient boost, CAT boost, XG boost, ADA boost and light GBM.

2.2.5. Artificial Neural Network

Once the machine learning model is trained for the first time using the ensemble algorithms selected, all the models are tested against the test set. The ML algorithm results for features in data instances are then given as input to a deep learning algorithm - Artificial neural network which processes it and gives 0 and 1 as output. 0 denotes that no anomaly is detected and 1 denotes that anomaly is detected in the network.

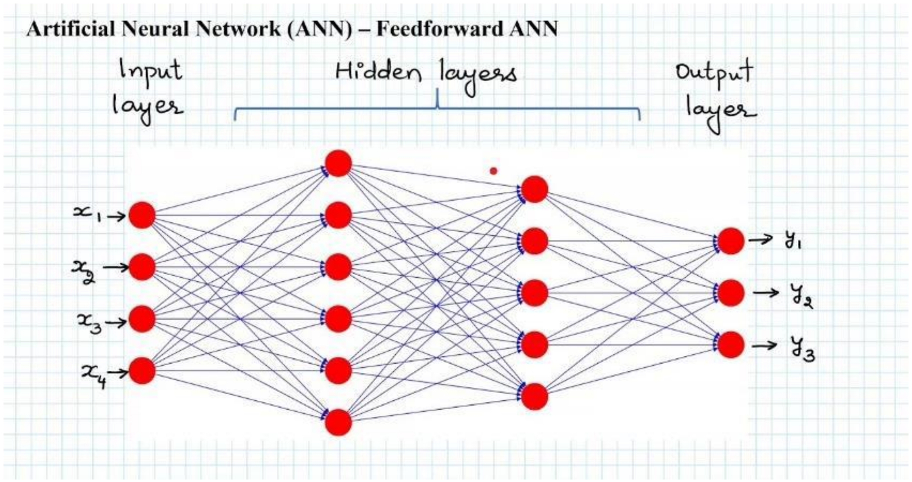


Fig. 4. Diagram for a Feedforward Artificial Neural Network

Unlike other machine learning algorithms like the voting classifier in the second level of training the IDS model, an artificial neural network would effectively process it and detect anomalies in the network better. Voting classifier's output for an instance is based on output of ML algorithms in level 1 which constitutes the majority. Thus, it is highly influenced by them and doesn't show any improvement in results. On the other hand, ANN has its unique algorithm with input layer, output layer and customizable hidden layers which can be tuned in multiple ways according to different dataset, to provide good results all the time.

2.2.6. Performance Metrics

After processing the result through an Artificial Neural Network, the output obtained has to be compared with the actual result. The proposed model need not give 100% accuracy. A good amount of deviation in the result ensures that the model is not overfitting and not influenced by ensemble algorithm results from level 1. The accuracy, precision and recall can be found by forming a confusion matrix as shown in Figure 4.

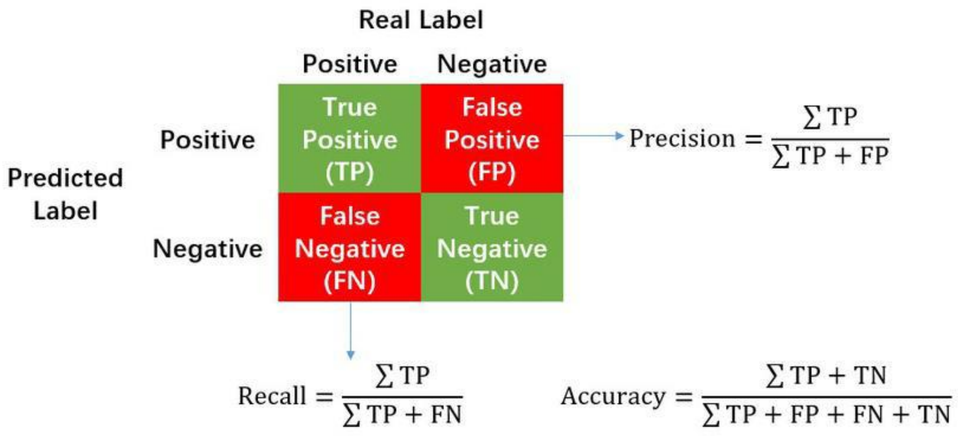

Real Label
Positive
Negative
Predicted Label
Positive
Negative
True Positive (TP)
False Positive (FP)
False Negative (FN)
True Negative (TN)
Precision = $\frac{\sum TP}{\sum TP + FP}$
Recall = $\frac{\sum TP}{\sum TP + FN}$
Accuracy = $\frac{\sum TP + TN}{\sum TP + FP + FN + TN}$

Fig.5. Diagram explaining the components of a Confusion Matrix

True positive means that the proposed model result and actual result is Normal(0). True negative means that the proposed model result and actual result is Anomaly(1). False Positive means that the proposed model result is normal(0) and actual result is anomaly. False negative means that the proposed model result is anomaly(1) and actual result is normal(0). On comparing we can categorise the result obtained from ANN into these 4 categories and based on the count of instances in each category, we can find accuracy, precision and recall of the proposed model. Also, we can plot a graph for the designed model using principal component analysis.

3. Implementation

3.1. Algorithms Used

We have implemented the following ensemble methods in the project to train the IDS dataset:

1. XG Boost
2. Random Forest

3. Bagging Classifier- Decision Tree
4. Ada Boost
5. Gradient Boost
6. Cat Boost
7. Light GBM
8. Voting Classifier

1. XG Boost

The XGBoost (eXtreme Gradient Boosting) algorithm is an implementation of gradient boosted decision trees. It focuses on increasing the performance of training and testing. The algorithm features three types of gradient boosting: Gradient Boosting, Stochastic Gradient Boosting and Regularized Gradient Boosting. The main reason that we chose the XGBoost algorithm is because the algorithm is very fast when compared to other gradient boosting algorithms. It is also very effective for training structured datasets for classification and regression problems.

2. Random Forest

The Random Forest classifier is an estimator that trains a number of decision trees on different subsets of the features. Classification is performed by taking the majority result from the prediction of all trees. Averaging is used to improve the prediction accuracy along with reducing overfitting.

3. Bagging Classifier - Decision Tree

Bagging Classifier is an ensemble learning method in which the base classifiers are fit on random subsets of the original dataset. The aggregate of the individual predictions is taken for the final classification. This type of classification model helps in reducing the variance with the help of the randomization technique. In our project, we have used the bagging method with the decision tree classifier.

4. Ada Boost

AdaBoost, short for Adaptive Boosting, is a meta-estimating classifier that works by fitting a model on the original dataset. This is followed by creating multiple copies of the classifier where the weights are adjusted in order to detect those cases that were incorrectly classified. This algorithm can be combined with other algorithms to improve performance and reduce overfitting.

5. Gradient Boost

Gradient Boost is an algorithm that works in a forward stage-wise method, where the main objective is to optimize the loss functions. The three main elements that are required are: a differentiable loss function, a weak learner for classification, and an additive model that reduces the loss. Since it is a greedy algorithm, it can lead to overfitting of the training data.

6. Cat Boost

Cat Boost, an ensemble learning method, is a variation of the gradient boosting algorithm where gradient boosting is performed on the decision trees. It is a very powerful algorithm that can perform better than other ensemble methods.

7. Light GBM

LightGBM (Light Gradient Boosting Machine), is an ensemble learning method, based on the gradient boosting algorithms. The algorithm works mainly on tree-based learning algorithms like decision trees. Since it is a distributed algorithm, it has certain advantages like high efficiency, low memory usage, improved accuracy, GPU support and capacity to handle large-scale data.

8. Voting Classifier

Voting Classifier is a ML model that is used to predict the output class based on the probability of each class predicted by the different estimators. Multiple classifiers are required to be trained and then passed on to the voting classifier for it to work. The aggregate of all predictions is taken and the majority prediction is then taken as the output of the voting classifier.

3.2. Mathematical Model Followed

Since the proposed method is using the connection details to determine whether a connection is Normal or Anomalous, we have used popular classification algorithms in order to classify the output. Input data consists of 41 features on which label encoding and feature scaling have been applied in order to give equal weightage to all features. These features are trained on the ensemble methods using various hyperparameters to get the best possible output while also trying to prevent overfitting.

4. Results and Discussion

4.1. Implementation

4.1.1. Load Dataset and Data Preprocessing

```
dataset =pd.read_csv('IDS_dataset.csv')
X=dataset.iloc[:,0:13].values
y=dataset.iloc[:,13].values
def Multilabelencoder(X,k):
    from sklearn.preprocessing import LabelEncoder
    X[:,k]= LabelEncoder().fit_transform(X[:,k])
    return X
for i in range(1,4):
    X=Multilabelencoder(X,i)
from sklearn.preprocessing import LabelEncoder
y= LabelEncoder().fit_transform(y)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =train_test_split(X,y,test_size=0.2,random_state=0)
from sklearn.preprocessing import StandardScaler
sc_X= StandardScaler()
X_train=sc_X.fit_transform(X_train)
X_test=sc_X.transform(X_test)
```

4.1.2. XGBoost

```
import xgboost as xgb

classifier1=xgb.XGBClassifier(random_state=1,learning_rate=0.01)

classifier1.fit(X_train, y_train)
```

4.1.3. Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier

classifier2 = RandomForestClassifier(n_estimators=10,
criterion='entropy',random_state=0)

classifier2.fit(X_train,y_train)
```

4.1.4. Bagging

```
from sklearn.ensemble import BaggingClassifier

from sklearn import tree

classifier3 = BaggingClassifier(tree.DecisionTreeClassifier(random_state=1))

classifier3.fit(X_train, y_train)
```

4.1.5. AdaBoost

```
from sklearn.ensemble import AdaBoostClassifier

classifier4 = AdaBoostClassifier(random_state=1)

classifier4.fit(X_train, y_train)
```

4.1.6. Gradient Boosting

```
from sklearn.ensemble import GradientBoostingClassifier

classifier5= GradientBoostingClassifier(learning_rate=0.01,random_state=1)

classifier5.fit(X_train, y_train)
```

4.1.7. CatBoost

```
from catboost import CatBoostClassifier

classifier6=CatBoostClassifier()

categorical_features_indices = np.where(dataset.dtypes != np.float)[0]

classifier6.fit(X_train,y_train,eval_set=(X_test, y_test))
```

4.1.8. LightGBM

```
import lightgbm as lgb
```

```
train_data=lgb.Dataset(X_train,label=y_train)
params = {'learning_rate':0.001}
classifier7 = lgb.train(params, train_data, 100)
```

4.1.9. Voting Classifier

```
from sklearn.ensemble import VotingClassifier
classifier8 = VotingClassifier(estimators=[('classifier1', classifier1),
('classifier2', classifier2),('classifier3', classifier3),('classifier4',
classifier4),('classifier5', classifier5),('classifier6', classifier6)],
voting='hard')
classifier8.fit(X_train,y_train)
```

4.1.10. Artificial Neural Network

```
 classifier.add(Dense(output_dim=4,init='uniform',activation='relu',input_dim=8))
    classifier.add(Dropout(p=0.1))


    #Adding output layer
    classifier.add(Dense(output_dim=1,init='uniform',activation='sigmoid'))


    #Compiling ANN
    classifier.compile(optimizer='adam',
loss='binary_crossentropy',metrics=['accuracy'])


    #Fitting ANN to training set
    classifier.fit(X_trainn,y_trainn,batch_size=10,nb_epoch=1)


    #Making Prediction and Evaluating model prediction
    y_pred = classifier.predict(X_testn)
    y_pred = (y_pred > 0.5)


    #confusion matrix
    from sklearn.metrics import confusion_matrix
    cm=confusion_matrix(y_testn,y_pred)


    #Evaluating ANN
    from keras.wrappers.scikit_learn import KerasClassifier
    from sklearn.model_selection import cross_val_score
    def build_classifier():
        classifier = Sequential()

classifier.add(Dense(output_dim=4,init='uniform',activation='relu',input_dim=8))
        classifier.add(Dense(output_dim=1,init='uniform',activation='sigmoid'))
        classifier.compile(optimizer='adam',
loss='binary_crossentropy',metrics=['accuracy'])
        return classifier

    classifier = KerasClassifier(build_fn = build_classifier,batch_size=10,nb_epoch=1)
    accuracies = cross_val_score(estimator=classifier,X=X_trainn, y=y_trainn,cv=10)
    mean=accuracies.mean()
    variance=accuracies.std()

    #Tuning the ANN
    from keras.wrappers.scikit_learn import KerasClassifier
    from sklearn.model_selection import GridSearchCV
    from keras.models import Sequential
    from keras.layers import Dense

    def build_classifier(optimizer):
        classifier = Sequential()

classifier.add(Dense(output_dim=4,init='uniform',activation='relu',input_dim=8))
        classifier.add(Dense(output_dim=1,init='uniform',activation='sigmoid'))
        classifier.compile(optimizer='adam',
loss='binary_crossentropy',metrics=['accuracy'])
        return classifier

    classifier = KerasClassifier(build_fn = build_classifier)
    grid_param={'batch_size':[25,32],
                'nb_epoch':[100,500],
                'optimizer':['adam','rmsprop']}
    grid_search = GridSearchCV(estimator=classifier,
                         param_grid=grid_param,
                         scoring='accuracy',
                         cv=5)
    grid_search = grid_search.fit(X_trainn,y_trainn)
    best_parameters=grid_search.best_params_
    best_accuracy = grid_search.best_score_
```

4.2. Results

After running the code successfully, we have finally obtained the results in this section. In the intermediate result, we have noticed that the ensemble bagging and boosting methods that get trained on a dataset learns parallelly and sequentially respectively, giving us overfitting models with higher accuracy, precision and recall. The machine learning model voting classifier in the next level gets highly influenced by the previous level overfitting models. Due to this, the voting classifier also shows overfitting in the evaluation result. Hence, we choose a deep learning model, Artificial neural network that gets trained with numerical processing of input data using relu and sigmoid functions over the voting classifier that simply predicts using majority count of overfitting model results. The result of ANN model which works independent of previous level ensemble models results is 99.83% which is best.

The following figures Fig 6 to Fig 13 shows the Principal component Analysis(PCA) of the ensemble methods which is the visualization of ensemble model prediction boundaries and actual result points. It helps to understand how well a model predicts over others just with visualization. The green shades and red shades of a model's PCA tells that any instance falling on that area is anomaly and normal. The red and green points are the actual outcome of the instance. Hence, a good model is expected to have a good count of green dots on green shade and red dots on red shade. Alter color of graph shade and point color shows that the model is predicting wrong output for the result. From the result below, we notice that different models have their own way of understanding the dataset to predict results. More instances ensures a final graph for the model which won't change after any addition of the dataset.

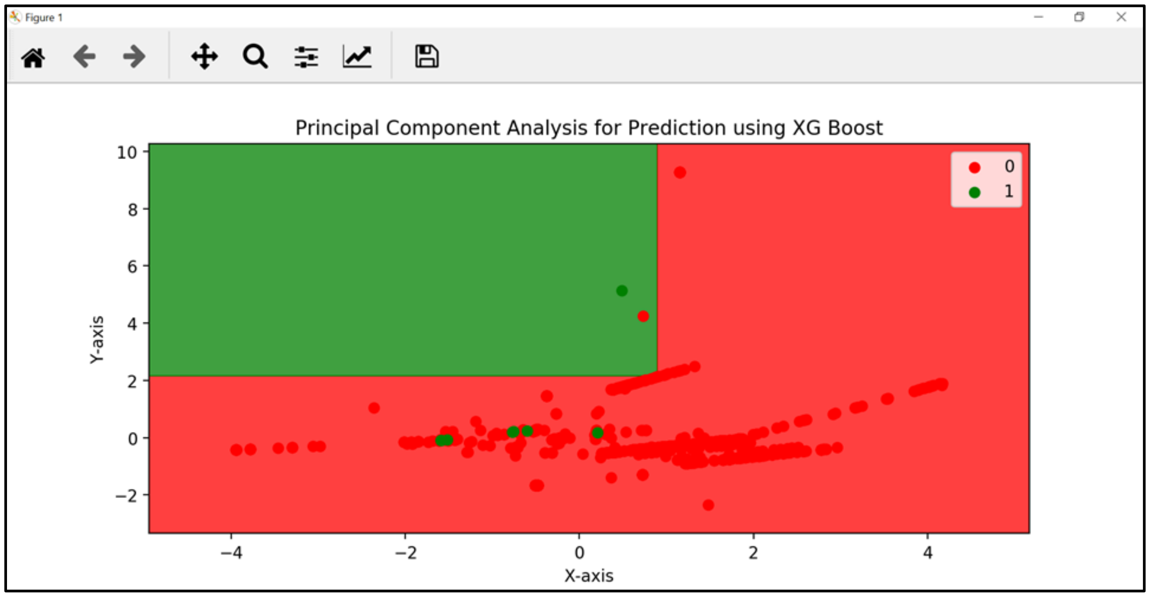


Fig 6: PCA for XGBoost

The XGBoost method gives an accuracy of 99.88%. The precision and recall that were obtained are 99.88% and 100%. This method can handle missing data and provides cross validation.

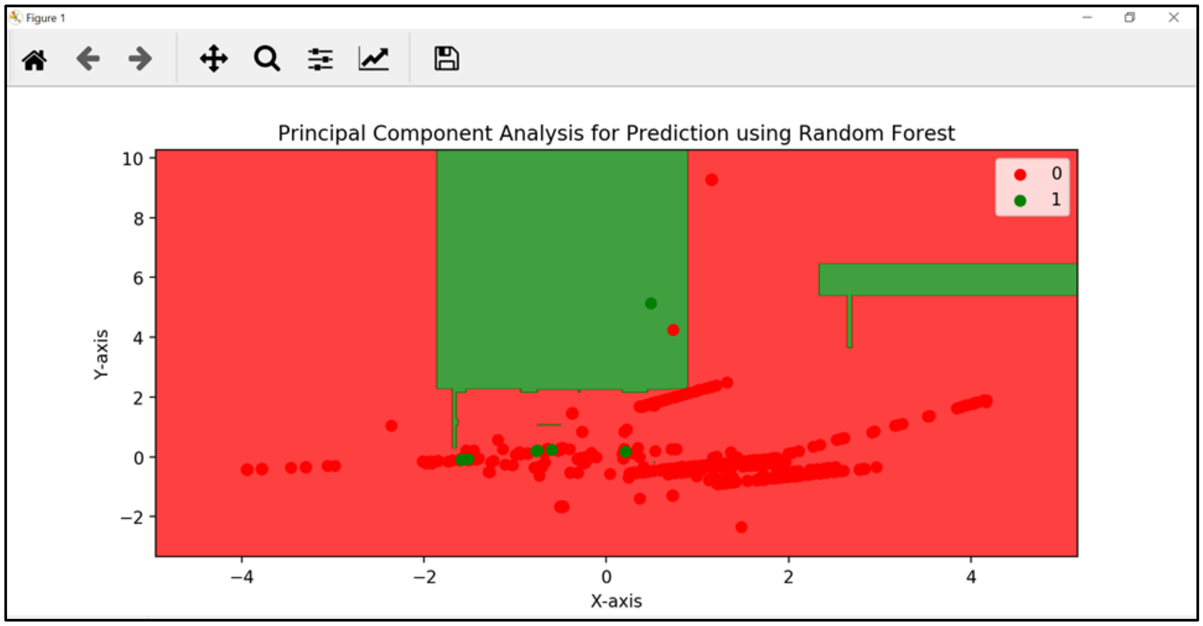


Fig 7: PCA for Random Forest

The Random Forest method gives an accuracy of 99.88%. The precision and recall that were obtained are 99.88% and 100%. This method prevents overfitting while handling missing data simultaneously.

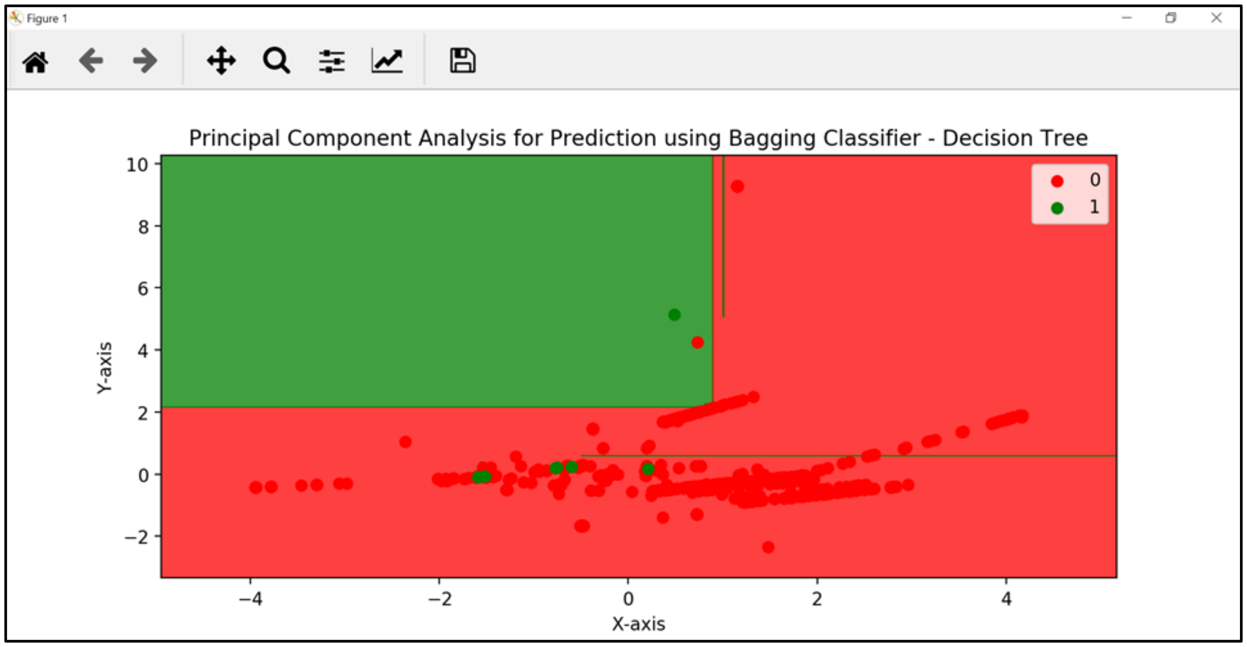


Fig 8: PCA for Bagging Classifier

The Bagging Classifier gives an accuracy of 99.99%. The precision and recall that were obtained are 99.94% and 99.96%. It combines multiple methods to provide better results while also reducing variance.

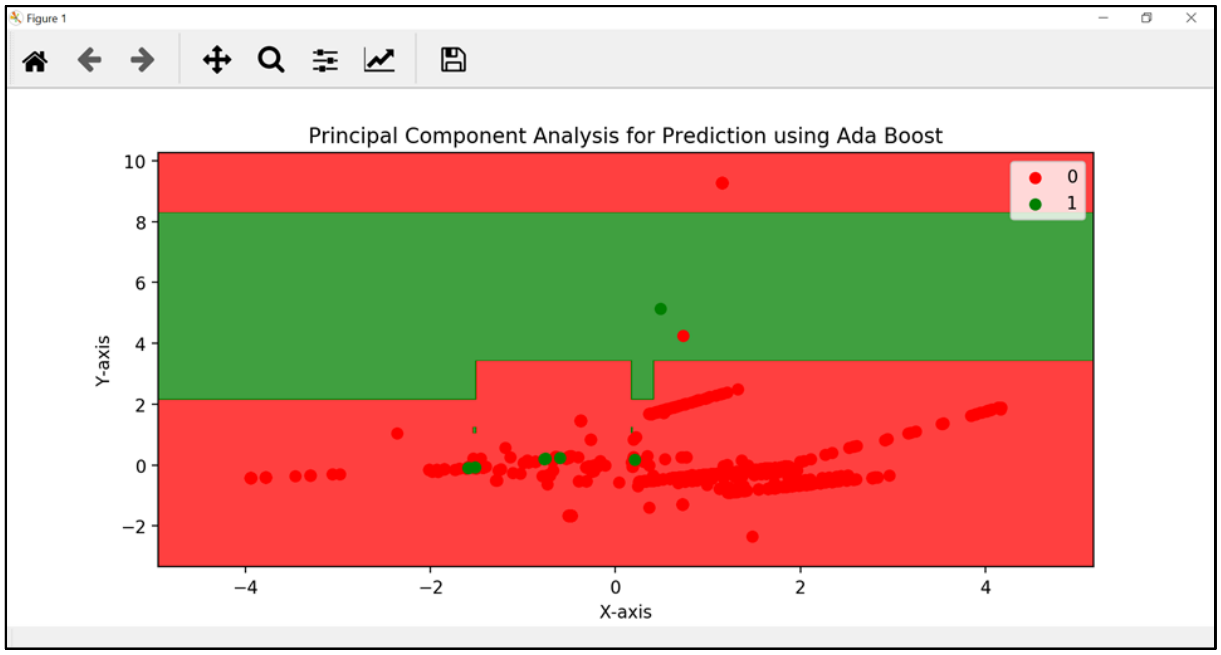


Fig 9: PCA for AdaBoost

The AdaBoost method gives an accuracy of 99.96%. The precision and recall that were obtained are 99.98% and 99.98%. It works with a wide range of classifiers while also having a low generalization error.

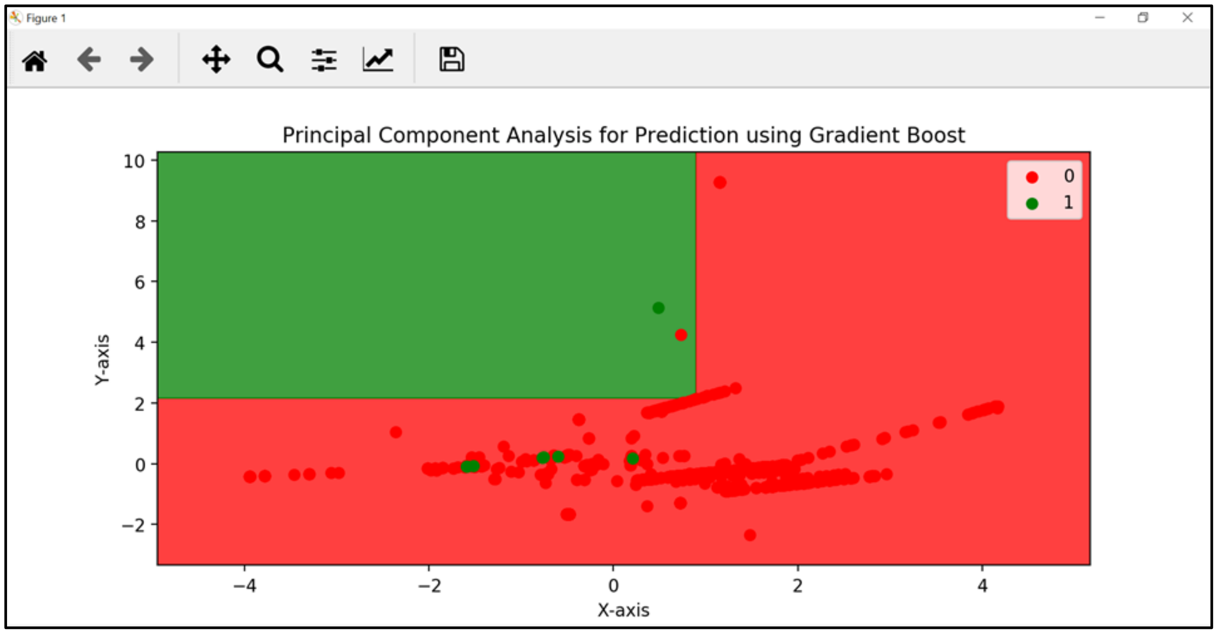


Fig 10: PCA for Gradient Boost

Gradient Boost gives an accuracy of 99.90%. The precision and recall that were obtained are 99.90% and 100%. It provides good accuracy and it even works without preprocessing.

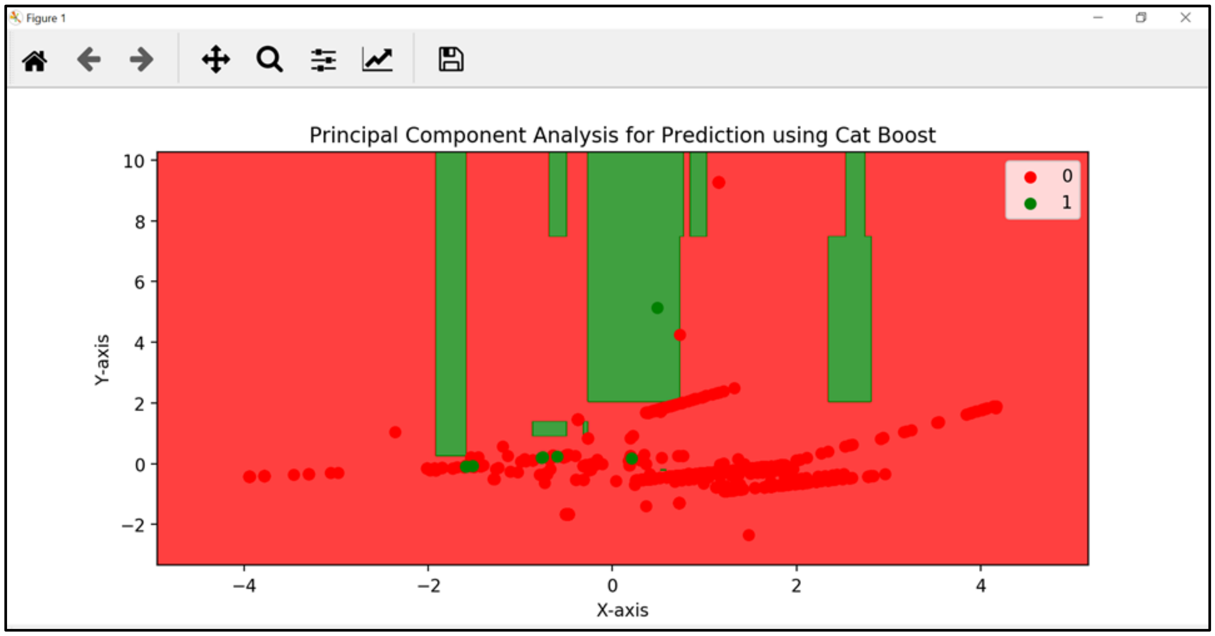


Fig 11: PCA for CatBoost

CatBoost gives an accuracy of 99.90%. The precision and recall that were obtained are 99.90% and 100%. It provides really high accuracy and lowers overfitting.

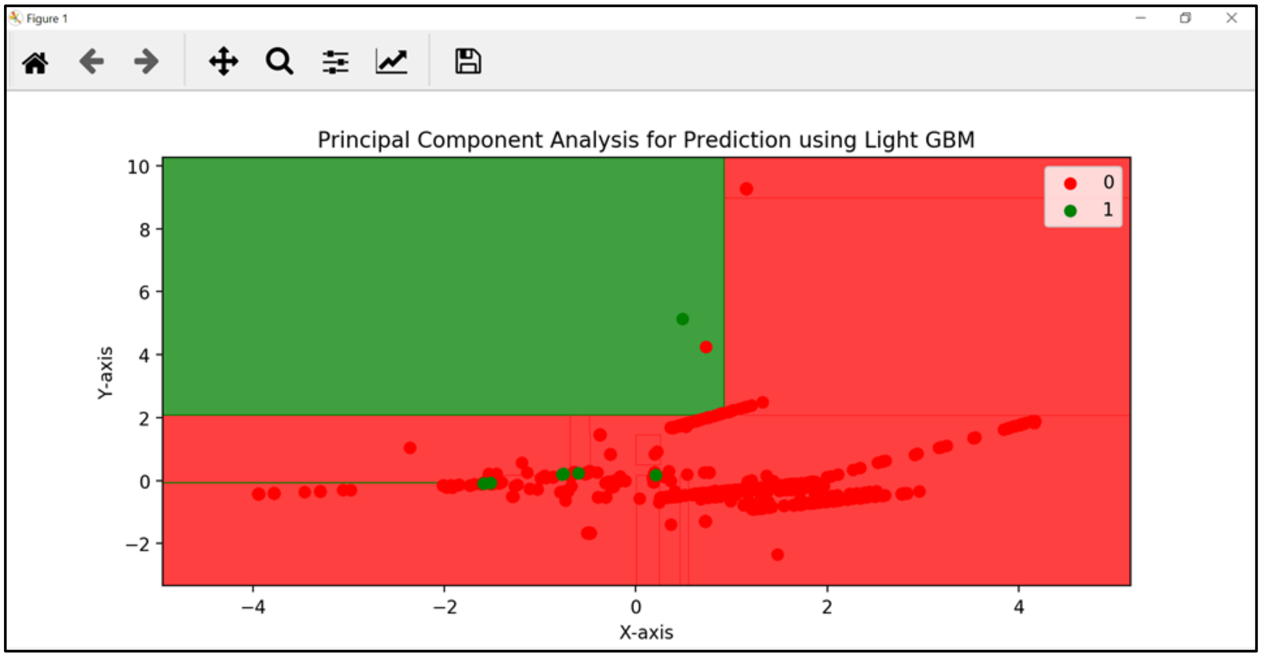


Fig 12: PCA for LightGBM

The LightGBM method gives an accuracy of 99.84%. The precision and recall that were obtained are 99.84% and 100%. It uses low memory while simultaneously providing faster speed and high efficiency.

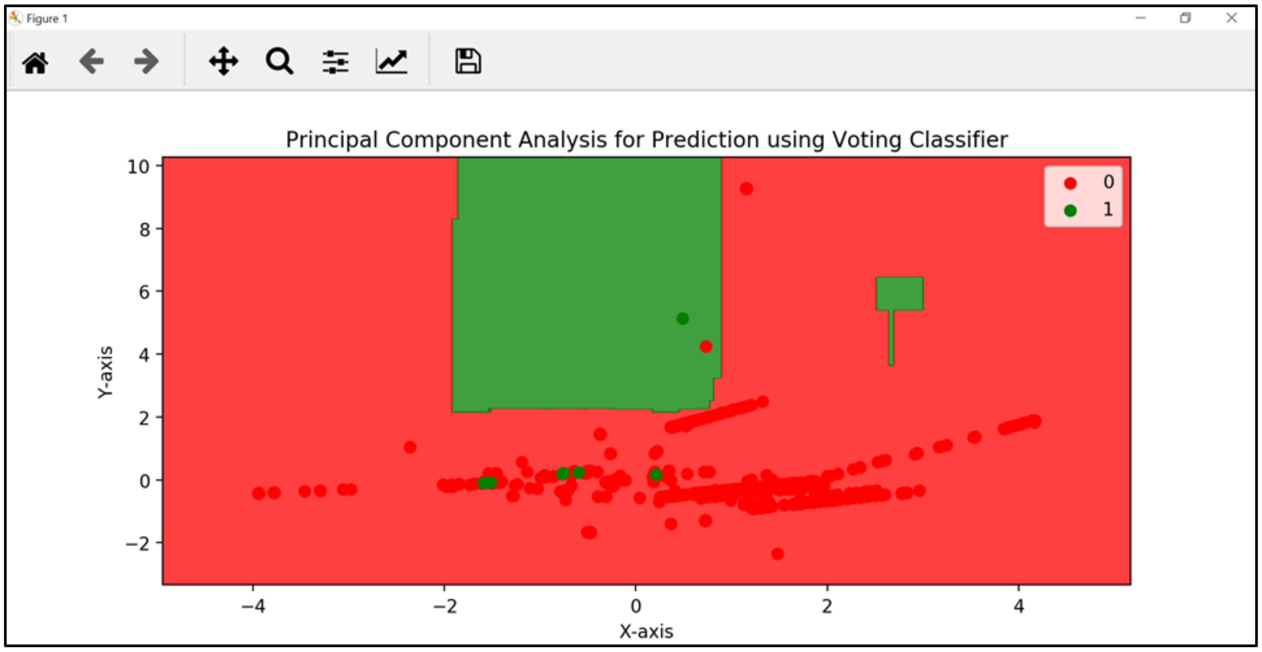
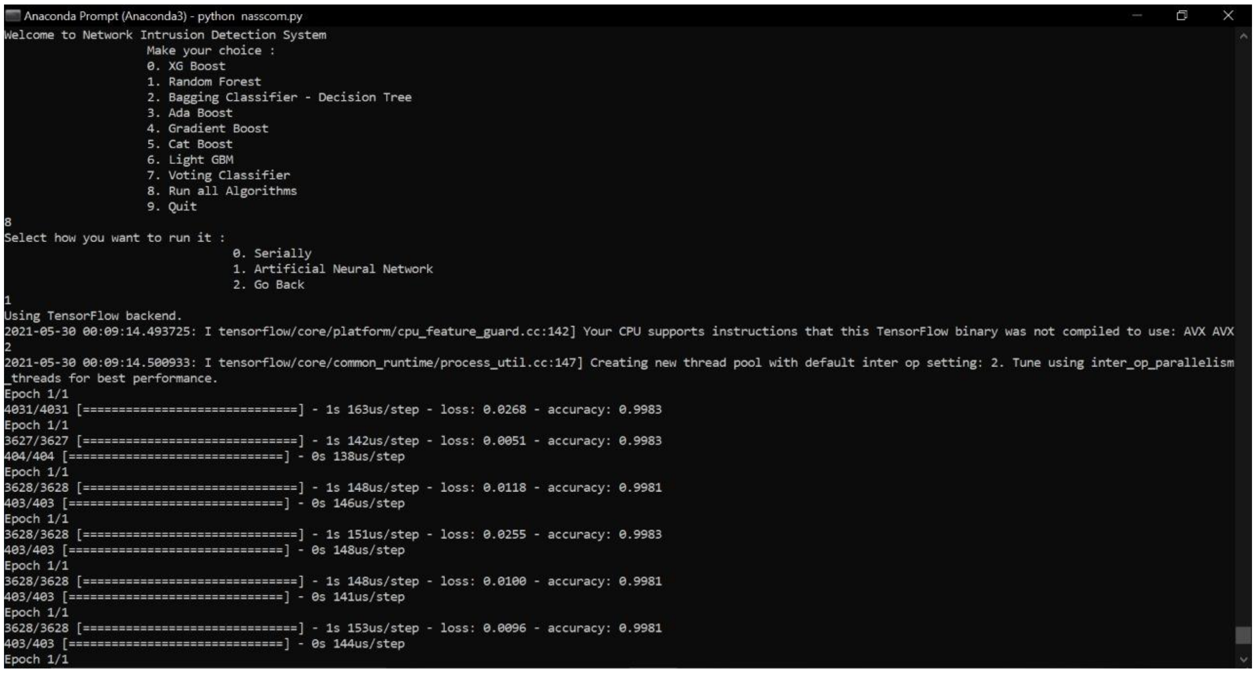


Fig 13: PCA for Voting Classifier

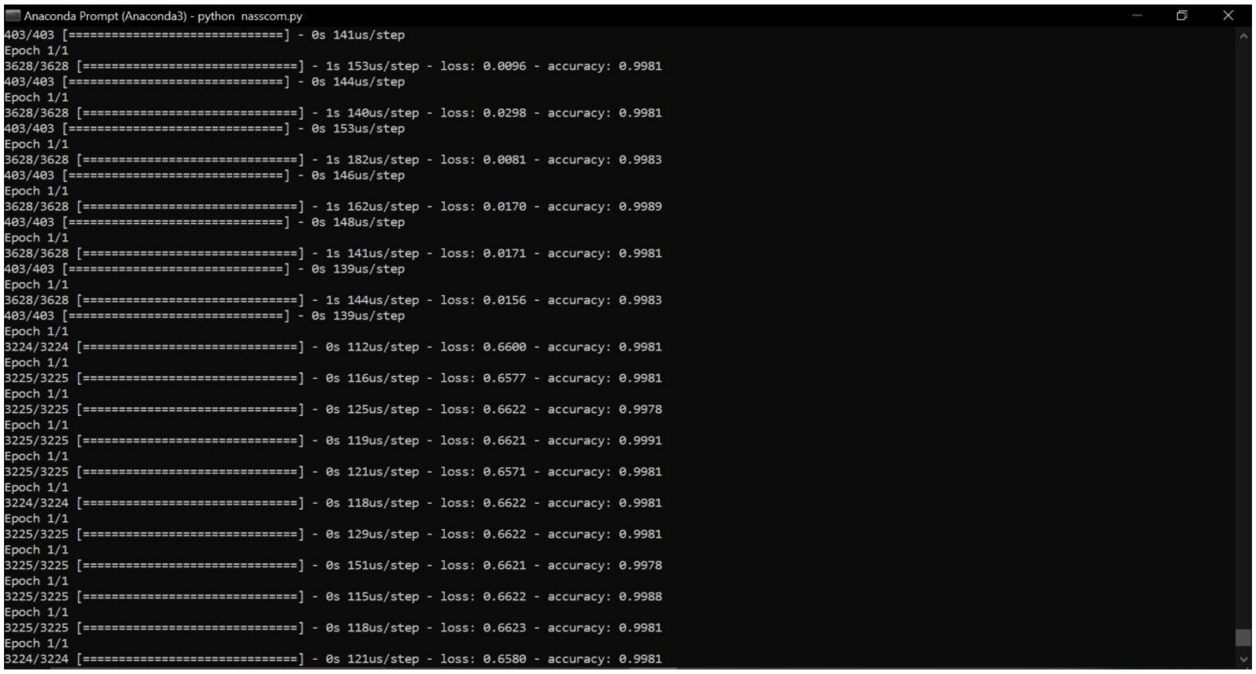
The Voting Classifier gives an accuracy of 99.90%. The precision and recall that were obtained are 99.90% and 100%. It helps in preventing incorrect prediction from a single model and provides generalized fit.



```
Anaconda Prompt (Anaconda3) - python  nasscom.py
Welcome to Network Intrusion Detection System
                  Make your choice :
                  0. XG Boost
                  1. Random Forest
                  2. Bagging Classifier - Decision Tree
                  3. Ada Boost
                  4. Gradient Boost
                  5. Cat Boost
                  6. Light GBM
                  7. Voting Classifier
                  8. Run all Algorithms
                  9. Quit
8
Select how you want to run it :
                              0. Serially
                              1. Artificial Neural Network
                              2. Go Back
1
Using TensorFlow backend.
2021-05-30 00:09:14.493725: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX AVX
2
2021-05-30 00:09:14.500933: I tensorflow/core/common_runtime/process_util.cc:147] Creating new thread pool with default inter op setting: 2. Tune using inter_op_parallelism
_threads for best performance.
Epoch 1/1
4031/4031 [==============================] - 1s 163us/step - loss: 0.0268 - accuracy: 0.9983
Epoch 1/1
3627/3627 [==============================] - 1s 142us/step - loss: 0.0051 - accuracy: 0.9983
404/404 [==============================] - 0s 138us/step
Epoch 1/1
3628/3628 [==============================] - 1s 148us/step - loss: 0.0118 - accuracy: 0.9981
403/403 [==============================] - 0s 146us/step
Epoch 1/1
3628/3628 [==============================] - 1s 151us/step - loss: 0.0255 - accuracy: 0.9983
403/403 [==============================] - 0s 148us/step
Epoch 1/1
3628/3628 [==============================] - 1s 148us/step - loss: 0.0100 - accuracy: 0.9981
403/403 [==============================] - 0s 141us/step
Epoch 1/1
3628/3628 [==============================] - 1s 153us/step - loss: 0.0096 - accuracy: 0.9981
403/403 [==============================] - 0s 144us/step
Epoch 1/1
```

Fig 14: Output 1 of ANN

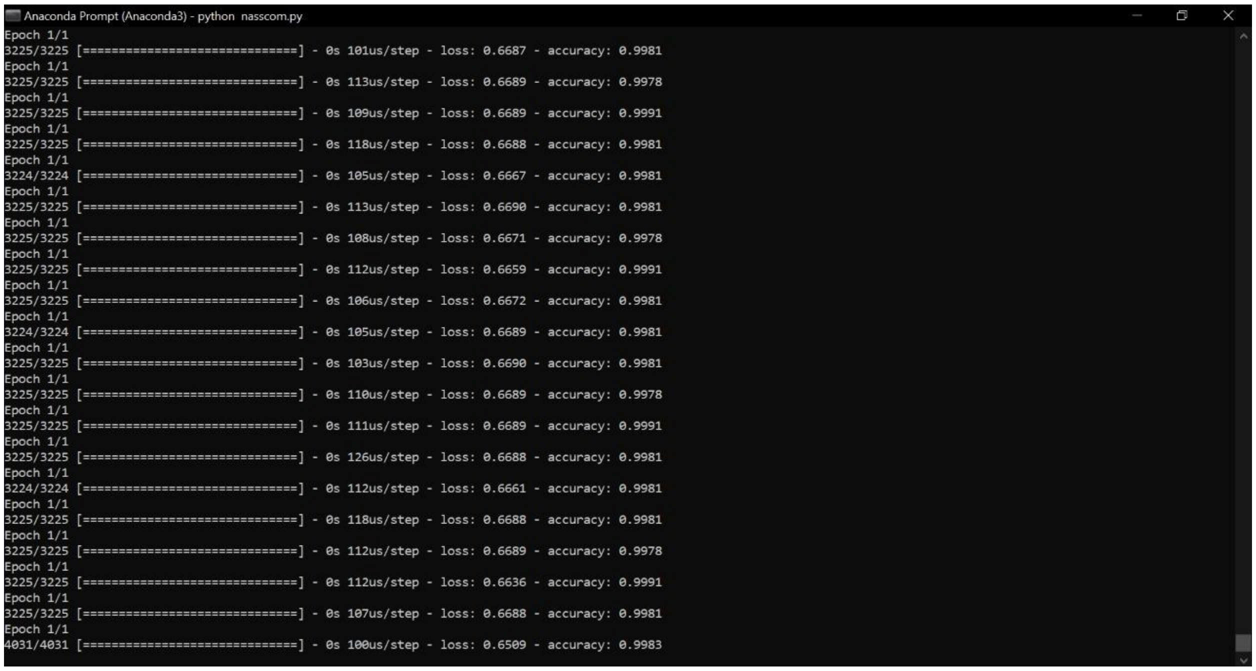
Fig 14 shows initialization of ANN deep learning model in tensorflow backend. Initial loss is 0.0268 and initial accuracy is 0.9983.



```
Anaconda Prompt (Anaconda3) - python  nasscom.py
403/403 [==============================] - 0s 141us/step
Epoch 1/1
3628/3628 [==============================] - 1s 153us/step - loss: 0.0096 - accuracy: 0.9981
403/403 [==============================] - 0s 144us/step
Epoch 1/1
3628/3628 [==============================] - 1s 140us/step - loss: 0.0298 - accuracy: 0.9981
403/403 [==============================] - 0s 153us/step
Epoch 1/1
3628/3628 [==============================] - 1s 182us/step - loss: 0.0081 - accuracy: 0.9983
403/403 [==============================] - 0s 146us/step
Epoch 1/1
3628/3628 [==============================] - 1s 162us/step - loss: 0.0170 - accuracy: 0.9989
403/403 [==============================] - 0s 148us/step
Epoch 1/1
3628/3628 [==============================] - 1s 141us/step - loss: 0.0171 - accuracy: 0.9981
403/403 [==============================] - 0s 139us/step
Epoch 1/1
3628/3628 [==============================] - 1s 144us/step - loss: 0.0156 - accuracy: 0.9983
403/403 [==============================] - 0s 139us/step
Epoch 1/1
3224/3224 [==============================] - 0s 112us/step - loss: 0.6600 - accuracy: 0.9981
Epoch 1/1
3225/3225 [==============================] - 0s 116us/step - loss: 0.6577 - accuracy: 0.9981
Epoch 1/1
3225/3225 [==============================] - 0s 125us/step - loss: 0.6622 - accuracy: 0.9978
Epoch 1/1
3225/3225 [==============================] - 0s 119us/step - loss: 0.6621 - accuracy: 0.9991
Epoch 1/1
3225/3225 [==============================] - 0s 121us/step - loss: 0.6571 - accuracy: 0.9981
Epoch 1/1
3224/3224 [==============================] - 0s 118us/step - loss: 0.6622 - accuracy: 0.9981
Epoch 1/1
3225/3225 [==============================] - 0s 129us/step - loss: 0.6622 - accuracy: 0.9981
Epoch 1/1
3225/3225 [==============================] - 0s 151us/step - loss: 0.6621 - accuracy: 0.9978
Epoch 1/1
3225/3225 [==============================] - 0s 115us/step - loss: 0.6622 - accuracy: 0.9988
Epoch 1/1
3225/3225 [==============================] - 0s 118us/step - loss: 0.6623 - accuracy: 0.9981
Epoch 1/1
3224/3224 [==============================] - 0s 121us/step - loss: 0.6580 - accuracy: 0.9981
```

Fig 15: Output 2 of ANN

Fig 15 shows the intermediate epochs during ANN training phase. The ANN model runs to train itself in multiple epochs. For each epoch, a part of the entire training set is taken to train the model using feed forward neural network algorithm.



```
Anaconda Prompt (Anaconda3) - python  nasscom.py
Epoch 1/1
3225/3225 [==============================] - 0s 101us/step - loss: 0.6687 - accuracy: 0.9981
Epoch 1/1
3225/3225 [==============================] - 0s 113us/step - loss: 0.6689 - accuracy: 0.9978
Epoch 1/1
3225/3225 [==============================] - 0s 109us/step - loss: 0.6689 - accuracy: 0.9991
Epoch 1/1
3225/3225 [==============================] - 0s 118us/step - loss: 0.6688 - accuracy: 0.9981
Epoch 1/1
3224/3224 [==============================] - 0s 105us/step - loss: 0.6667 - accuracy: 0.9981
Epoch 1/1
3225/3225 [==============================] - 0s 113us/step - loss: 0.6690 - accuracy: 0.9981
Epoch 1/1
3225/3225 [==============================] - 0s 108us/step - loss: 0.6671 - accuracy: 0.9978
Epoch 1/1
3225/3225 [==============================] - 0s 112us/step - loss: 0.6659 - accuracy: 0.9991
Epoch 1/1
3225/3225 [==============================] - 0s 106us/step - loss: 0.6672 - accuracy: 0.9981
Epoch 1/1
3224/3224 [==============================] - 0s 105us/step - loss: 0.6689 - accuracy: 0.9981
Epoch 1/1
3225/3225 [==============================] - 0s 103us/step - loss: 0.6690 - accuracy: 0.9981
Epoch 1/1
3225/3225 [==============================] - 0s 110us/step - loss: 0.6689 - accuracy: 0.9978
Epoch 1/1
3225/3225 [==============================] - 0s 111us/step - loss: 0.6689 - accuracy: 0.9991
Epoch 1/1
3225/3225 [==============================] - 0s 126us/step - loss: 0.6688 - accuracy: 0.9981
Epoch 1/1
3224/3224 [==============================] - 0s 112us/step - loss: 0.6661 - accuracy: 0.9981
Epoch 1/1
3225/3225 [==============================] - 0s 118us/step - loss: 0.6688 - accuracy: 0.9981
Epoch 1/1
3225/3225 [==============================] - 0s 112us/step - loss: 0.6689 - accuracy: 0.9978
Epoch 1/1
3225/3225 [==============================] - 0s 112us/step - loss: 0.6636 - accuracy: 0.9991
Epoch 1/1
3225/3225 [==============================] - 0s 107us/step - loss: 0.6688 - accuracy: 0.9981
Epoch 1/1
4031/4031 [==============================] - 0s 100us/step - loss: 0.6509 - accuracy: 0.9983
```

Fig 16: Output 3 of ANN

Fig 16 shows the result obtained after the final epoch. The accuracy of the model is 0.9983 and an increase in loss from 0.0268 to 0.6509 ensures that the ANN model is not influenced by previous level models and hence, it is not overfitting in nature. Below table 1 shows the evaluation of level 1 and level 2 algorithms of the network intrusion detection system.

Ensemble Model	Accuracy %	Precision %	Recall %
XGBoost	99.8808	99.8809	100
Random Forest	99.8809	99.8808	100
Bagging Classifier	99.9907	99.9403	99.9602
AdaBoost	99.9603	99.9801	99.9801
Gradient Boost	99.9007	99.9007	100
CatBoost	99.9007	99.9007	100
LightGBM	99.8412	99.8412	100
Voting Classifier	99.9007	99.9007	100
ANN	99.8300	-	-

Table 1: Performance metrics of ensemble models

4.3. Mapping Results with Problem Statement and Existing Works

The proposed system is more reliable and efficient by implementing multiple ensemble machine learning algorithms in a parallel environment, and overcoming the underfitting and overfitting nature of individual algorithms with the help of deep learning artificial neural networks which can tune its nature according to the real time environment in which it is deployed. Unlike multilevel machine learning algorithms which can have influence among its levels, an artificial neural network in level 2 of IDS results in an intrusion detection system which provides more security and reliability. Further development of this system with real time data after deploying will help us to increase the detection rate even higher and lower false negative rate. The system can be used to predict any other attacks as well by classifying them as network intrusion. Thus, the aim of this project is accomplished.

References

[1]. D. Papamartzivanos, F. Gómez Mármol and G. Kambourakis, 2019, “Introducing Deep Learning Self-Adaptive Misuse Network Intrusion Detection Systems”, IEEE Access, vol. 7, pp. 13546-13560 DOI: 10.1109/ACCESS.2019.2893871

[2]. Omar Y. Al-Jarrah, Yousof Al-Hammdi, Paul D. Yoo, Sami Muhaidat, Mahmoud Al-Qutayri, 2018, “Semi-supervised multi-layered clustering model for intrusion detection”, Digital Communications and Networks, Volume 4, Issue 4, pp. 277-286

DOI: 10.1016/j.dcan.2017.09.009

[3]. K.V.V.N.L. Sai Kiran, R.N. Kamakshi Devisetty, N. Pavan Kalyan, K. Mukundini, R. Karthi, 2020, “Building a Intrusion Detection System for IoT Environment using Machine Learning Techniques”, Procedia Computer Science, Volume 171, pp. 2372-2379

DOI: 10.1016/j.procs.2020.04.257

[4]. Mahmoud M. Sakr, Medhat A. Tawfeeq, Ashraf B. El-Sisi, 2019, “Network Intrusion Detection System based PSOSVM for Cloud Computing”, I. J. Computer Network and Information Security, Volume 3, pp. 22-29

DOI: 10.5815/ijcnis.2019.03.04

[5]. T. Saranya, S. Sridevi, C. Deisy, Tran Duc Chung, M.K.A.Ahamed Khan, 2020, “Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review” ; Procedia Computer Science, Volume 171, pp. 1251-1260

DOI: 10.1016/j.procs.2020.04.133

[6] J.S. shanthini, Dr. S. Rajalakshmi, “Data Mining Techniques For Efficient Intrusion Detection System: A Survey”, International Journal On Engineering Technology and Sciences – IJETS™ ISSN(P): 2349-3968, ISSN (O): 2349- 3976 Volume II, Issue XI, November – 2015

[7] Abhaya, Kaushal Kumar, Ranjeeta Jha, Sumaiya Afroz, “Data Mining Techniques for Intrusion Detection: A Review”, International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 6, June 2014.

[8] Larriva-Novo, X., Vega-Barbas, M., Villagra, V. A., Rivera, D., Alvarez-Campana, M., & Berrocal, J. (2020). Efficient Distributed Preprocessing Model for Machine Learning-Based Anomaly Detection over Large-Scale Cybersecurity Datasets. Applied Sciences, 10(10), 3430.

[9] Pawlicki, M., Choraś, M., & Kozik, R. (2020). Defending network intrusion detection systems against adversarial evasion attacks. Future Generation Computer Systems, 110, 148-154.

[10] Elmasry, W., Akbulut, A., & Zaim, A. H. (2020). Evolving deep learning architectures for network intrusion detection using a double PSO metaheuristic. Computer Networks, 168, 107042.

[11] Shahraki, A., Abbasi, M., & Haugen, Ø. (2020). Boosting algorithms for network intrusion detection: A comparative evaluation of Real AdaBoost, Gentle AdaBoost and Modest AdaBoost. Engineering Applications of Artificial Intelligence, 94, 103770.

[12] Zhang, J., Tu, H., Ren, Y., Wan, J., Zhou, L., Li, M., & Wang, J. (2018). An adaptive synchronous parallel strategy for distributed machine learning, IEEE Access, 6, 19222-19230.