“SMART IDS – IOT Senor Network using ML”

ISM PROJECT REVIEW

TEAM MEMBERS:

- Siddharth 20BDS0400
- Vishesh Bhargava 20BCE2506
- Seshank 20BDS0136

Under Guidance of:
DR. Prakash G

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

1. ABSTRACT

In this growing world, IOT came out and emerged as a technology to connect various heterogeneous nodes, objects and devices such as people, infrastructure, thus making an end-to-end communication for data transfer and hence making a large network of these heterogeneous devices. Due to its large network, it lacked resources and thus became the weakest and easiest link for cyber attackers. To manage it, some classical encryption techniques have been implemented for securing data in the IOT network, because high level encryption techniques cannot be implemented on such a large network due to lack of resources. Following this, node security and data breaches are two bigger challenges which we identified by going through most of the related research papers. Thus, it becomes very necessary to explore a solution for the security of nodes of an IOT based devices' network to avoid data breaches. In this project, we are managing to employ some of the rule-based approaches and algorithms of Artificial Intelligence to strengthen the security of the network along with the existing security protocols. This project provides a comprehensive and deep level understanding of security approaches for IOT based sensor devices using AI based approaches and algorithms.

2. INTRODUCTION

The Internet of Things (IoT) infrastructure and artificial intelligence (AI) technologies are combined to create artificial intelligence of things (AIoT). The objective of AIoT is to improve human-machine interactions, IoT operations, and data management and analytics. AIoT is revolutionary and advantageous for both kinds of technology since AI enhances IoT through connectivity, signalling, and data exchange, while IoT enhances AI through capabilities for machine learning and improved decision-making processes. IoT network components like individual devices frequently have weak security or weren't created to connect to public networks. Considering the constant growth of IOT in the real world, Data transfer with full integrity, confidentiality and availability is very crucial as it decides the response created by the actuator. Attacks such as SQL injection, Spoofing, Cross-site Scripting, MITM attack finds weak points in the network and exploits it to extract or manipulate data and cause problems.

2.1. OUR CONTRIBUTION:

In this paper, we suggest a highly effective IDS model that makes use of cutting-edge deep learning (DL) and metaheuristic optimization techniques. Initially, the characteristics were quickly and effectively retrieved using a convolutional neural network (CNN) model. To retrieve the relevant characteristics, there are numerous successive convolution units. Only the characteristic extrication stage of the process, which facilitates the recovery of valuable characteristics that can reflect the original information in a smaller space.

2.1.1. DOMAIN:

Artificial Intelligence refers to the capacity of a system and its act of copying, implementing human intelligence by training itself using experienced based learning. AI helps in solving real world problems using algorithms in order to generate a response.

TECHNICAL ASPECTS OF AI:

- It uses pattern detection that helps in identifying patterns with the aid of a machine learning algorithm. The process of categorizing information based on previously acquired knowledge or on statistical data extrapolated from patterns and/or their depiction is known as pattern recognition.
- AI can recognize and prioritize risk and form a plan of action to discard the vulnerabilities.
- It performs Intrusion Detection, it monitors the flow of data packets and their activities, finds an anomaly and alerts the system to make a quick decision.
- AI systems are trained on huge datasets to provide highly accurate responses.

Machine Learning is an area of study of AI, which allows computer systems to self-educate from the training dataset and enhance its accuracy over time without being specifically programmed. We train the models by feeding them samples of training data.

TECHNICAL ASPECTS OF ML:

- Deep learning is a machine learning method that educates computers to perform tasks that individuals accomplish without thinking about it.
- CNNs (Convolutional neural network) are especially helpful for recognising objects, classes, and categories in photos by looking for patterns in the images. Additionally, they can be quite successful at categorizing audio, time-series, and signal data.
- CNN can also be used for feature extraction of a particular skeleton dataset.
- KNN (K nearest neighbour) is based on the supervised learning method. It assumes that the new situation and the existing circumstances are comparable, and it places the new case in the class that is most consistent with the original classes.

2.1. PROPOSED SYSTEM:

In the world of IoT sensor networks, detecting anomalies in network traffic is of utmost importance to ensure the integrity and security of the network. One of the most popular techniques for this task is the use of Intrusion Detection Systems (IDS). Random Forest is a popular machine learning algorithm that has been proven effective for IDS in previous research. In this proposal, we suggest using Random Forest for IDS on an IoT sensor network. Moreover, we propose the use of Chi-Square as a feature selection method to improve the performance of the Random Forest model. This technique will enable us to identify the most relevant features from the input data and eliminate the redundant ones, leading to improved accuracy and reduced training time. Our approach aims to provide a robust and efficient IDS solution for IoT sensor networks that can adapt to the changing nature of IoT traffic patterns.

3. LITERATURE SURVEY

S NO	AUTHOR AND YEAR	TECHNIQUES USED	LIMITATIONS
1	L. M. Rasdi Rere, 2016 CITE: Rere, L. M., Mohamad Ivan Fanany, and Aniati Murni Arymurthy. "Metaheuristic algorithms for convolution neural network." *Computational intelligence and neuroscience* 2016 (2016).	1. The fundamental concept of Simulated Annealing algorithm is the use of heuristic search, which not only permits modifications that enhance the optimization technique but also preserves certain modifications that are not optimal. 2. Harmony Search algorithm imitates how a musician would act to create a perfect harmony. During a search, it might strike an equilibrium among discovery and exploitation.	• There are numerous adjustable variables in Simulated Annealing algorithm. • Long scheduling of tasks can result in slow operation. • In Harmony Search, preterm convergence is present.
2	Zenab Elgamal, 2022 CITE: Z. Elgamal, A. Q. M. Sabri, M. Tubishat, D. Tbaishat, S. N. Makhadmeh and O. A. Alomari, "Improved Reptile Search Optimization Algorithm Using Chaotic Map and Simulated Annealing for Feature Selection in Medical Field," in IEEE Access, vol. 10, pp. 51428-51446, 2022, doi: 10.1109/ACCESS.2022.3174854.	1. Reptile Search Algorithm is a nature-inspired metaheuristic system that incentivizes the hunting and surround behaviour of crocodiles. When contrasted to other metaheuristics, the RSA algorithm's singular search yields encouraging results. 2. Chaotic map algorithm helps in initialization of answers also known as search agents. The startup stage of RSA employs chaotic maps to increase the variety of its results. It enhances the population variety.	• Starting circumstances are highly influential and delicate in the chaotic map algorithm. • 100% accuracy is not possible using chaotic map

3	Qihang Yuan, 2022 CITE: Yuan, Qihang, Yongde Zhang, Xuesong Dai, and Shu Zhang. "A Modified Reptile Search Algorithm for Numerical Optimization Problems." *Computational Intelligence and Neuroscience* 2022 (2022).	1. Elite Alternative Pool Strategy, in this the most ideal person is followed by RSA to update positions. It speeds up the method's resolution. It is a good substitute pooling technique used to sustain equilibrium between the method's utilization and discovery.	• It reduces the variety in community of data. • Isolated optimums are most likely to persist.
4	Shakir Zaman, 2020 CITE: Zaman, Shakir, Haseeb Tauqeer, Wakeel Ahmad, Syed M. Adnan Shah, and Muhammad Ilyas. "Implementation of intrusion detection system in the Internet of Things: A survey." In *2020 IEEE 23rd International Multitopic Conference (INMIC)*, pp. 1-6. IEEE, 2020.	1. Random Forest helps us to construct decision trees and also to merge them to obtain the best results or trends. Classification and regression are both achieved by this algorithm. 2. Support Vector Machine works on the idea of decision plane. A decision plane serves as a divider between representatives from various classes. They serve as supervised algorithms in the heuristic analysis model. 3. Association mining categorises groups of objects that frequently occur concurrently in your dataset. It seeks to identify useful guidelines that aid in the creation of new information. Relations among the objects define the association rules.	• In Random forest algo, Using a huge amount of decision trees can lead to inadequate and slow results. • SVM is not effective with huge datasets as it takes a lot of time to train. • In association mining, the results to be produced are dependent on the result of another object.
5	Fathima Hussain, 2020 CITE: Hussain, Fatima, Rasheed Hussain, Syed Ali Hassan, and Ekram Hossain. "Machine learning in IoT security: Current solutions and future challenges." *IEEE Communications Surveys & Tutorials* 22, no. 3 (2020): 1686-1721.	1. Dimensionality reduction is used here to transform a larger dataset into one with less dimensions while maintaining similar results. These methods are frequently used in machine learning to solve classification and regression issues while producing a more accurate predictive model. Helps in Malware analysis. 2. In this paper, Clustering is used to organise the sets of data into various clusters that are made up of related data points. The items with potential similarity continue to be in a group that shares little to no similarity with some other group. It helps in detection of anomalies in the network/ system.	• Out of the various available clustering techniques, every technique outputs varied results. • In dimensionality reduction, there is a high possibility of losing relevant and important information from the dataset. • In dimensionality reduction, the result does not convey its nature (i.e., vector or scalar).

6	Pynbianglut Hadem, 2021 CITE: Hadem, Pynbianglut, Dilip Kumar Saikia, and Soumen Moulik. "An SDN-based intrusion detection system using SVM with selective logging for IP traceback." *Computer Networks* 191 (2021): 108015.	1. Software defined networking is used which contains a centralized structure to perform findings and analysis. 2. Selective logging is practised in the in-memory structure at the controller, which in turn increases the accuracy. It also helps in reduction of memory consumption. 3. Support vector machine here helps to calculate accuracy with low overhead.	• Using these techniques can result in occurrence of false alarms. • Selective logging will reduce the number of packets being analysed out of the pool of packets.
7	Arun Das, 2022 CITE: A. Das, M. Roopaei, M. Jamshidi and P. Najafirad, "Distributed AI-Driven Search Engine on Visual Internet-of-Things for Event Discovery in the Cloud," 2022 17th Annual System of Systems Engineering Conference (SOSE), Rochester, NY, USA, 2022, pp. 514-521, doi: 10.1109/SOSE55472.2022.9812698.	1. Utilizing state-of-the-art distributed deep learning and BigQuery search algorithms. 2. Distributed deep neural networks (DNNs) over edge visual Internet of Things (VIoT) devices	• Rich source of information is mostly left untapped • Connected cameras and streaming videos not function properly
8	Pooja Chaudhary, 2021 CITE: P. Chaudhary, B. B. Gupta, K. T. Chui and S. Yamaguchi, "Shielding Smart Home IoT Devices against Adverse Effects of XSS using AI model," 2021 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2021, pp. 1-5, doi: 10.1109/ICCE50685.2021.9427591.	1. Employs OS-ELM classifier to detect the XSS attack 2. Employed machine learning (ML) algorithms	• Security concerns such as cross site scripting (XSS) vulnerability that can cause severe damage to the victim. • Data breaches are likely to occur

9	Rajesh Gupta, 2022 CITE: R. Gupta, N. K. Jadav, A. Nair, S. Tanwar and H. Shahinzadeh, "Blockchain and AI-based Secure Onion Routing Framework for Data Dissemination in IoT Environment Underlying 6G Networks," 2022 Sixth International Conference on Smart Cities, Internet of Things and Applications (SCIoT), Mashhad, Iran, Islamic Republic of, 2022, pp. 1-6, doi: 10.1109/SCIoT56583.2022.9953671.	1. Employed machine learning (ML) algorithms such as Random Forest (RF) classifier 2. Distributed deep neural networks (DNNs) are used	• Data dissemination through energy-constrained low-power • IoT sensors suffers from issues such as security and privacy of data.
10	Prabhat Kumar, 2022 CITE: P. Kumar, R. Kumar, S. Garg, K. Kaur, Y. Zhang and M. Guizani, "A Secure Data Dissemination Scheme for IoT-Based e-Health Systems using AI and Blockchain," GLOBECOM 2022 - 2022 IEEE Global Communications Conference, Rio de Janeiro, Brazil, 2022, pp. 1397-1403, doi: 10.1109/GLOBECOM48099.2022.10000801.	1. Used InterPlanetary File System (IPFS) of cloud for transaction. 2. Uses SVM Machine learning algorithm	• Wireless unsecured public communication channel which can exploit the vulnerability of the system • Data dissemination more likely to occur
11	Shaibal Chakrabarty, 2020 CITE: S. Chakrabarty and D. W. Engels, "Secure Smart Cities Framework Using IoT and AI," 2020 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT), Dubai, United Arab Emirates, 2020, pp. 1-6, doi: 10.1109/GCAIoT51063.2020.9345912.	1. Utilizes Black Network protocols 2. Utilizes key management	• The broad reliance upon IoT in smart cities significantly increases the attack surface of the smart system • Exploiting the vulnerability of the system is more common

12	Shreya Aggarwal, 2022 CITE: S. Aggarwal and S. Sharma, "Voice Based Secured Smart Lock Design for Internet of Medical Things: An Artificial Intelligence Approach," 2022 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET), Chennai, India, 2022, pp. 1-7, doi: 10.1109/WiSPNET54241.2022.9767113.	1. Used AI-assisted acoustic feature extraction 2. Used Natural language processing (NLP) to anticipate voice	• Device's security is degraded due to increased network on solid lock system • Huge traffic on network causes poor network segmentation
13	Vasileios A. Memos, 2020 CITE: Memos, Vasileios A., and Kostas E. Psannis. "AI-powered honeypots for enhanced IoT botnet detection." In 2020 3rd World Symposium on Communication Engineering (WSCE), pp. 64-68. IEEE, 2020.	1. It is based on honeynet based detection method and using it with Artificial intelligence for optical botnet detection rate. 2. Their analysis is based on classification and clustering algorithm of data mining techniques and machine learning algorithms.	• Security concerns such as for evaluation of its robustness and effectiveness. • It does not provide large field for detection activities because of its narrow range
14	Md. Yousuf Hossain, 2018 CITE: Nayak, Maya, and Prasannajit Dash. "Smart surveillance monitoring system using Raspberry Pi and PIR sensor." *Statistics* (2014).	1. It is based on detection of motion and identification of face. 2. they used Raspberry Pi and Pi camera module, PIR motion sensor and LCD display.	• In this detection range is minimum • Security concerns as of to make it more secure and safe
15	Heetae Yang, 2018 CITE: Yang, Heetae, Wonji Lee, and Hwansoo Lee. "IoT smart home adoption: the importance of proper level automation." *Journal of Sensors* 2018 (2018).	1. It is based on research and hypothesis development. 2. In which they used PLS-SEM analysis method and smart PLS2.0 tool to verify the research hypothesis.	• Interconnectivity and reliability are required with the right level of automation.
16	Puji siswipraptini, 2021 CITE: Sondakh, Grace. "IoT for smart home system." *Indonesian Journal of Electrical Engineering and Computer Science* 23, no. 2 (2021): 733-739.	1. Utilize the ANN machine learning algorithm in which they used ADALINE neuron network. 2. The testing algorithm for ADALINE works as obtain weight from learning process then set activation of the input unit and calculate the network value.	• Require lots of computational power. • It needs lots of data to train the algorithm.

17	Tagyaldeen Mohamed, 2018 CITE: Mohamed, TagyAldeen, Takanobu Otsuka, and Takayuki Ito. "Towards machine learning based IoT intrusion detection service." In *Recent Trends and Future Technology in Applied Intelligence: 31st International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2018, Montreal, QC, Canada, June 25-28, 2018*.	1. Utilizes the artificial neural network with another machine learning algorithm called random forest with network features. 2. In which they used traffic gathering module and detection module to simulates the results.	• Requires more accuracy because it does not provide 100% accuracy • IoT traffic feature selection can be improved.
18	Jiaqi Li, 2019 CITE: Li, Jiaqi, Zhifeng Zhao, Rongpeng Li, and Honggang Zhang. "Ai-based two-stage intrusion detection for software defined iot networks." *IEEE Internet of Things Journal* 6, no. 2 (2018): 2093-2102.	1. Used the combination of SDN-based IoT framework managed with network functionality virtualization and evaluating the dataset with RF algorithm. 2. The advantage of SDN is to captures the network packets and collects status information with centralized control.	• Need of high security and reliability • performance in a real network for traffic classification
19	Shaftab Ahmed +2 2022	1. The proposed technique uses tokens based dynamic exchange between participating IoT nodes to generate security keys after plug and play engagement of sensor devices. 2. The proposed solution protects IoT system from malicious attacks around exposed sensor nodes.	1. The data traffic form numerous IoT devices affects speed and efficiency of the backbone network. 2. The proposed security mechanisms, difficulty is to guess and introduce barriers at all (gateway) hierarchical levels to handle malicious activities. 3. Requires light weight algorithms. Use of embedded security at security nodes. Use of available tools at Fog and Edge computing levels.
20	Kah Phooi Seng +2 2022	Hierarchical training framework termed as HierTrain. Deploys DNN training task on the mobile edge-cloud computing MECC model.	1. DNN's computationally expensive. 2. The amount of dataset require are at least in thousands to millions. 3. Deployment is unnecessarily more

			complicated and takes much longer.
21	Kazi Istiaque Ahmed +4 2021	1. ML based Authentication schemes 2. Iot nodes uses supervised ML techniques - KNN. 2. Q- learning for anti jamming approach	1. Computational time with large data sets might be slow. (KNN) 2. High memory requires. All the training data s=is store in memory. (KNN) 3. Computationally expensive
22	Liang Xiao +4 2018	Array of Techniques 1. NN – DoS 2. Q- learning and DQN - Jamming 3. SVM, Q – learning, DQN, DNN – Spoofing 4. SVM, Naïve Bayes – Intrusion 5. Q/Dyno-Q/PDS, Random forest - Malware	1. Computational power and cost high due to multiple techniques. 2. Fast to train but slow and ineffective for real time protection - Random Forest 3. Double convergence time – DQN
24	Sohaib A. Latif +6 2022	Blockchain and Cluster structure of SDN – Management of distributed IoT network.	1. Latency issues 2. Complexity of SDM is hight compare to other methods. 3. Configuration of SDN network is not easy and is expensive.
25	Zhihan LV +2 2021	1. Elliptic Curve Cryptography (ECC) 2. Random Forest (RF), Support Vector machine (SVM), ANN	1. ECC- Complicated to Implement 2. ECC is a new algorithm contains risk of unexplored weakness. 3. RF – formations of multiple trees lead to slow and less effective solution
26	Ankit Attkn + 1 2022	1. Use of Symmetric cryptosystem-based protocols. TDES or 3DES 2. Asymmetric cryptosystem-based protocols – RSA and El Gamal algorithm 3. Blockchain SHA- 256 based on KECCAK- sponge functions 4. KNN, random forest and logistic regression	1. 3DES is limited by its small block size 2. RSA slows down the whole operation due to involvement of large numbers 3. high processing required due to decryption process at user end

3.1. LIMITATIONS OF ML-AI IN IOT NETWORKS:

Most classic ML algorithms need to be significantly modified to process IoT data because they are typically not scalable and effective. Some of the limitations are as follows:

- Processing power and energy limitations: IoT devices are often tiny and have limited processing power due to energy limitations. Therefore, straight use of typical ML algorithms is inappropriate in these situations with limited resources.

- The development of ML-based networks is predicated on the assumption that the complete data set will be processed during the training phase. This might not apply to IoT data, though. Additionally, as the dimensionality of the data increases, an algorithm's ability to anticipate outcomes declines.

- With a variety of types, forms, and semantics, the data produced in IoT networks exhibits syntactic and semantic heterogeneity.

IoT theme of applications comprising of sensor devices help in collection of large amounts of data from multiple resources. This makes IoT systems vaster and vulnerable for attackers to penetrate into the network which results in loss of efficiency. Thus, to resolve this problem, AI comes into effect.

AI along with IoT manages the seamless flow of data through a secure network in IoT devices. IoT where manages gadgets collaborating utilizing the web, AI causes the gadgets to gain from their information and experience. While, IoT gives information, man-made reasoning procures the ability to open reactions, offering both imagination and setting to drive smart activities. As the information conveyed from the sensor can be broke down with AI, organizations can settle on smart choices. The man-made brainpower IoT prevails with regards to accomplishing the smart sensing systems.

4. PROPOSED SYSTEM

A. Random Forest

Random Forest is a powerful ensemble learning algorithm widely used for classification and regression tasks in various fields, including machine learning, statistics, and bioinformatics. In the context of Intrusion Detection System (IDS) in a IoT sensor network, Random Forest can be an effective tool to classify network traffic and detect potential security threats. The algorithm works by building multiple decision trees based on random samples of the dataset and selecting the most relevant features at each node. The individual trees are then combined to form a random forest, where each tree's prediction is weighted to produce the final classification result. The ability of Random Forest to handle large datasets with complex and non-linear relationships between features makes it an attractive option for IDS in IoT sensor networks, where the data is often high-dimensional and noisy. In this article, we will explore the use of Random Forest for IDS in an IoT sensor network and demonstrate how feature selection techniques can be applied to improve its performance.

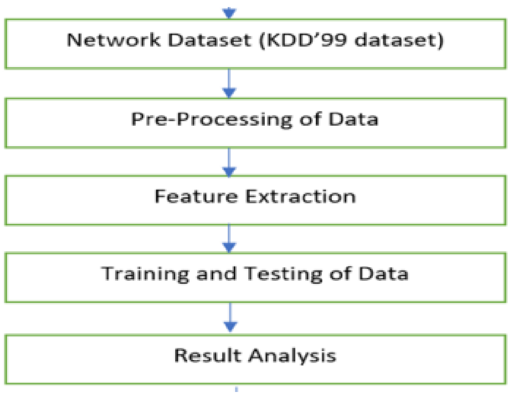

Network Dataset (KDD'99 dataset)
Pre-Processing of Data
Feature Extraction
Training and Testing of Data
Result Analysis

Fig 4.1 Basic overview of the model

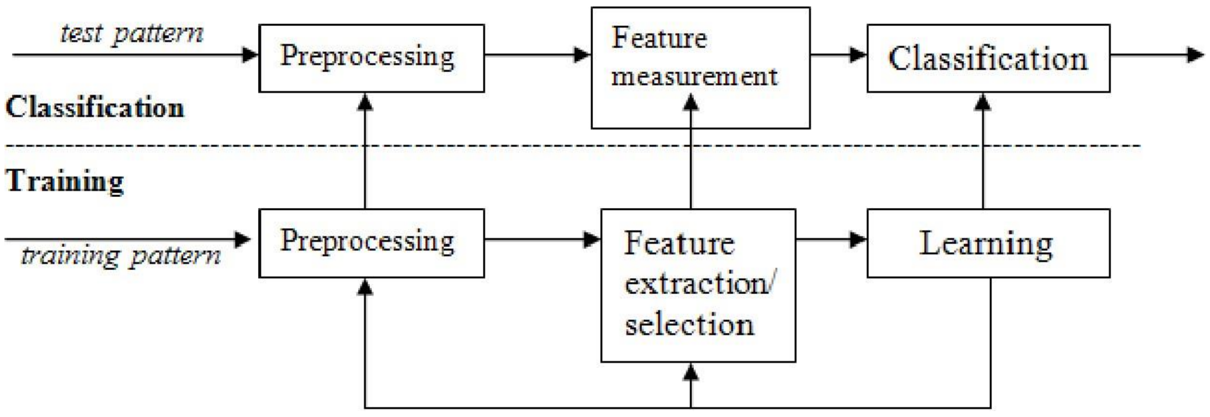


Fig 4.2 working of a classifier

B. EXPERIMENTAL SETUP:

DATASET DESCRIPTION:

Based on KDDCup-99, the improved version without duplicate network traffic samples, the NSL-KDD (Table 4.1) dataset was created. The NSL-KDD was developed as part of the intrusion detection challenge that DARPA (military advanced research projects agency) launched in 1998. The NSL-KDD data were acquired via studies conducted at the MIT Lincon laboratory, where network traffic data was gathered over a ten-week period. 100 individuals and about 1000 UNIX machines made up the experimental setup. Over 5 million records in a raw Transmission Control Protocol/Internet Protocol (TCP/IP) dump format made up the network traffic data that was gathered. Owing to the enormity of the dataset, the data gatherers only released a minor version that included 41 features for each connection record and the types of attacks denial-of-service (DoS), probing, remote-to-user (R2L), and user-to-root (U2R). The Bot-IoT dataset is made up of over 72 million connection records that were acquired from numerous IoT devices.

Statistics of redundant records in the KDD train set

Original records | Distinct records | Reduction rate

Attacks: 3,925,650 | 262,178 | 93.32%

Normal: 972,781 | 812,814 | 16.44%

Total: 4,898,431 | 1,074,992 | 78.05%

Statistics of redundant records in the KDD test set

Original records | Distinct records | Reduction rate

Attacks: 250,436 | 29,378 | 88.26%

Normal: 60,591 | 47,911 | 20.92%

Total: 311,027 | 77,289 | 75.15%

Sr. no	Features	Feature Type
1.	duration	numeric
2.	protocol_type	factor
3.	service	factor
4.	flag	factor
5.	src_bytes	numeric
6.	dst_bytes	numeric
7.	land	factor
8.	wrong_fragment	numeric
9.	urgent	numeric
10.	hot	numeric
11.	num_failed_logins	numeric
12.	logged_in	factor
13.	num_compromised	numeric
14.	root_shell	numeric
15.	su_attempted	numeric
16.	num_root	numeric
17.	num_file_creations	numeric
18.	num_shells	numeric
19.	num_access_files	numeric
20.	num_outbound_cmds	numeric
21.	is_host_login	factor
22.	is_guest_login	factor
23.	count	numeric
24.	srv_count	numeric
25.	serror_rate	numeric
26.	srv_serror_rate	numeric
27.	rerror_rate	numeric
28.	srv_rerror_rate	numeric
29.	same_srv_rate	numeric
30.	diff_srv_rate	numeric
31.	srv_diff_host_rate	numeric
32.	dst_host_count	numeric
33.	dst_host_srv_count	numeric
34.	dst_host_same_srv_rate	numeric
35.	dst_host_diff_srv_rate	numeric
36.	dst_host_same_src_port_rate	numeric
37.	dst_host_srv_diff_host_rate	numeric
38.	dst_host_serror_rate	numeric
39.	dst_host_srv_serror_rate	numeri
40.	dst_host_rerror_rate	numeric
41.	dst_host_srv_rerror_rate	numeric
42.	class	factor

Table 4.1 Dataset

C. Random Forest

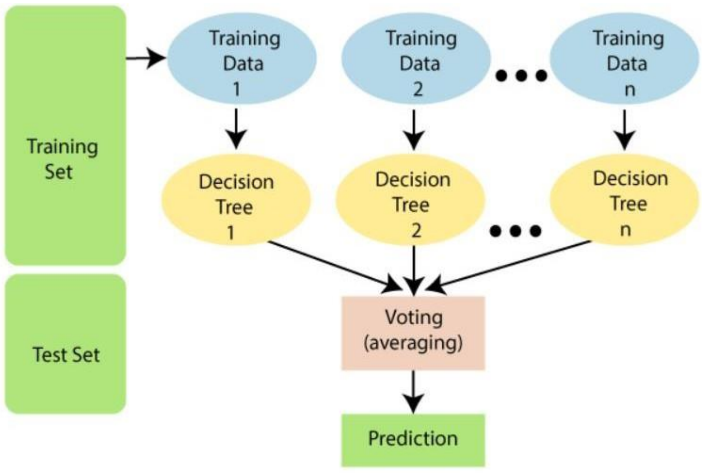


Fig 4.3 Random Forest

Instead of relying on one decision tree, the random forest (fig 4.3) takes the prediction from each tree and bases its prediction of the final output on the majority votes of predictions. Random Forest is a classifier that contains several decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. Higher accuracy and overfitting are prevented by the larger number of trees in the forest.

Algorithm for Random Forest:

Choose the number of trees to grow (n_estimators) and the number of features to consider when looking for the best split (max_features).

For each tree:

a. Draw a bootstrap sample of the training data.

b. For each node in the tree:

i. Select a random subset of features to consider.

ii. Find the best feature and threshold to split the node on, based on a criterion such as the Gini impurity or information gain.

iii. Create two child nodes based on the split, and repeat until a stopping criterion is reached (such as a minimum number of samples in a leaf node).

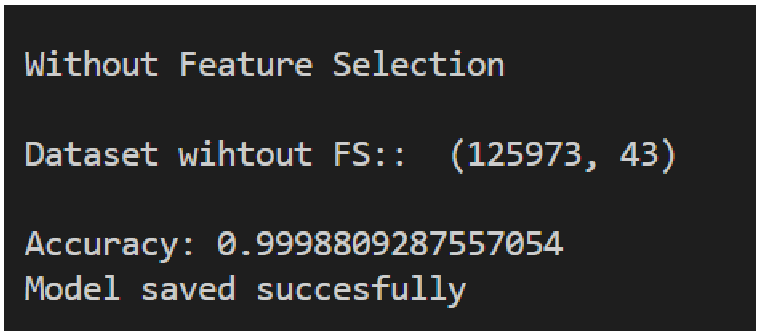
To make a prediction on a new sample, run it through each tree in the forest and aggregate the results (for classification, take the mode of the class predictions, and for regression, take the mean of the target variable predictions).

Code:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
import pickle


print("\nWithout Feature Selection\n")

def factorize_string_columns(df):
    for column in df.select_dtypes(include=['object']):
        df[column] = pd.factorize(df[column])[0]
    return df

# Load NSL-KDD dataset
data = pd.read_csv('NSL-KDD/KDDTrain+.txt', header=None)

# Convert categorical columns to numerical data
data = factorize_string_columns(data)
print("Dataset wihtout FS:: ", data.shape)


# Preprocessing
X = pd.concat([data.iloc[:, :40], data.iloc[:, 42:]], axis=1)
y = data.iloc[:, -2] # labels
encoder = LabelEncoder()
y = encoder.fit_transform(y)
scaler = StandardScaler()
X = scaler.fit_transform(X)


# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12)


# Train model
model = RandomForestClassifier(n_estimators=50, criterion='gini' ,random_state=123)
model.fit(X, y)


# Test model
y_pred = model.predict(X_test)

# Evaluate model
acc = accuracy_score(y_test, y_pred)
print('\nAccuracy:', acc)
```

```
# Save model to file
with open('model.pkl', 'wb') as file:
    pickle.dump(model, file)

file.close()
print("Model saved succesfully\n")
```

Output:



```
Without Feature Selection

Dataset wihtout FS::  (125973, 43)

Accuracy: 0.9998809287557054
Model saved succesfully
```

D. CHI-SQUARE PROCEDURE:

We are using chi-squared test for Feature Extraction, we use the SelectKBest algorithm is a feature selection method in machine learning that selects the K best features from a dataset based on their scores. The scores are computed using a specific statistical test, such as chi-squared or mutual information, that measures the relationship between the features and the target variable.

Goodness of Fit:

$$\chi^2 = \sum \frac{(O - E)^2}{E}$$

Df = number of categories – 1

Test for Independence (Contingency Table)

$$X^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}}.$$

Df = (rows-1)(columns-1)

EQ 1

The experiments used to evaluate the proposed IoT security methodology are presented in this section, along with the assessment process itself. These tests included a variety of evaluation criteria, real-world datasets, and in-depth comparisons to other approaches to choose features.

I. Measures of Evaluation: To evaluate the effectiveness of the suggested methodology and all comparable approaches, a number of assessment indicators are employed. We define those markers in accordance with the confusion matrix notion.

II. Average Accuracy: It refers to the rate of correct detection of intrusion. It can be calculated as

$$\mathrm{AV}_{\mathrm{Acc}} = \frac{1}{N_r} \sum_{k=1}^{N_r} \mathrm{Acc}_{\mathrm{Best}}^{k},$$

$$\mathrm{Acc}_{\mathrm{Best}} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FN} + \mathrm{FP} + \mathrm{TN}},$$

EQ 2

III. Average Recall: The percentage of incursion that was correctly predicted is referred to here as a true-positive rate (TPR). It is determined as

$$\mathrm{AV}_{\mathrm{Sens}} = \frac{1}{N_r}\sum_{k=1}^{N_r}\mathrm{Sens}_{\mathrm{Best}}^{k},$$

$$\mathrm{Sens}_{\mathrm{Best}} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}.$$

EQ 3

IV. Average Accuracy: It shows what proportion of all positive cases are TP instances. It is calculable as

$$\mathrm{AV}_{\mathrm{Prec}} = \frac{1}{N_r}\sum_{k=1}^{N_r}\mathrm{Prec}_{\mathrm{Best}}^{k},$$

$$\mathrm{Prec}_{\mathrm{Best}} = \frac{\mathrm{TP}}{\mathrm{FP}+\mathrm{TP}}.$$

EQ 4

V. Rate of Performance Improvement (PIR): It is defined as being used to calculate the rate of improvement obtained by the developed method.

$$\mathrm{PIR} = \frac{M_{\mathrm{RSA}} - M_{\mathrm{Alg}}}{M_{\mathrm{RSA}}} \times 100,$$

EQ 5

First phase 1:-

Prepare the Dataset :- First we collect a dataset of labelled instances where each instance is labelled as either normal or anomalous. When collecting the dataset, it's important to ensure that it represents a diverse range of scenarios and environments. This can help to improve the accuracy and robustness of the IDS system by capturing a wide range of possible anomalies. Additionally, it's important to label the instances accurately to ensure that the machine learning model can learn to distinguish between normal and anomalous instances.

Data Pre-processing :- Pre-process the dataset by performing tasks such as removing duplicates, handling missing values, and converting categorical data into numerical data. Data pre-processing is a critical step in building an effective IDS system. It helps to remove noise and inconsistencies from the dataset, which can improve the accuracy and efficiency of the machine learning model. Additionally, it helps to ensure that the data is in a format that is suitable for feature extraction and selection.

Second Phase 2:-

Feature Extraction :- In this step we are going to extract features from the pre-processed dataset. This can be done using techniques such as principal component analysis (PCA) or independent component analysis (ICA). Feature extraction is the process of selecting the most relevant features from the dataset. This is typically done to reduce the dimensionality of the dataset and to extract the most important information. PCA and ICA are popular techniques for feature extraction, as they can help to reduce the dimensionality of the dataset while retaining the most important information.

Third Phase 3:-

Feature Selection :- In this step we perform feature selection using the chi-square and random forest algorithm. In chi-square it will calculate the statistics for each feature and selecting the top k feature features with highest chi-square value and in random forest it building a random forest model on the dataset and calculating the feature importance scores for each feature.

Feature selection combination :- In this step we Combine the features selected from chi-square and random forest to get the final set of features for the IDS system. The combination of feature selection with chi-square and random forest can help to create a more robust and accurate feature set for the IDS system. By selecting the most important features from both methods, the feature set can capture a wide range of important information while also reducing the dimensionality of the dataset. Table

Sr. no	Column Names	Column Type
1.	"duration"	"numeric"
2.	"protocol_type"	"factor"
3.	"service"	"factor"
4.	"flag"	"factor"
5.	"src_bytes"	"numeric"
6.	"dst_bytes"	"numeric"
7.	"land"	"factor"
8.	"urgent"	"numeric"
9.	"hot"	"numeric"

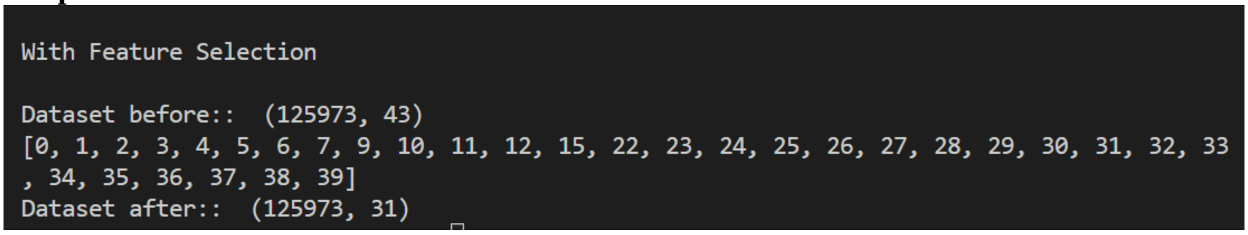
10.	"num_failed_logins"	"numeric"
11.	"logged_in"	"factor"
12.	"su_attempted"	"numeric"
13.	"is_guest_login"	"factor"
14.	"count"	"numeric"
15.	"srv_count"	"numeric"
16.	"serror_rate"	"numeric"
17.	"srv_serror_rate"	"numeric"
18.	"rerror_rate"	"numeric"
19.	"srv_rerror_rate"	"numeric"
20.	"same_srv_rate"	"numeric"
21.	"diff_srv_rate"	"numeric"
22.	"srv_diff_host_rate"	"numeric"
23.	"dst_host_count"	"numeric"
24.	"dst_host_srv_count"	"numeric"
25.	"dst_host_same_srv_rate"	"numeric"
26.	"dst_host_diff_srv_rate"	"numeric"
27.	"dst_host_same_src_port_rate"	"numeric"
28.	"dst_host_srv_diff_host_rate"	"numeric"
29.	"dst_host_serror_rate"	"numeric"
30.	"dst_host_srv_serror_rate"	"numeric"

Code:

```
import pandas as pd
from sklearn.feature_selection import SelectKBest, chi2

print("\nWith Feature Selection\n")

def factorize_string_columns(df):
    for column in df.select_dtypes(include=['object']):
        df[column] = pd.factorize(df[column])[0]
    return df

# Load NSL-KDD dataset
data = pd.read_csv('NSL-KDD/KDDTrain+.txt', header=None)

# Convert categorical columns to numerical data
data = factorize_string_columns(data)
print("Dataset before:: ",data.shape)


# split the data into features and target variable
X = pd.concat([data.iloc[:, :40], data.iloc[:, 42:]], axis=1)
y = data.iloc[:, -2] #labels

# apply feature selection
selector = SelectKBest(chi2, k=31)
X_new = selector.fit_transform(X, y)

# get the selected feature indices
selected_features = selector.get_support(indices=True)
```

```
# print the selected feature names
feature_names = list(X.columns)
selected_feature_names = [feature_names[i] for i in selected_features]

print(selected_feature_names)

selected_features = data.iloc[:, selected_feature_names]
X = selected_features.iloc[:, :]
print("Dataset after:: ",X.shape)
```

Output:



```
With Feature Selection

Dataset before::  (125973, 43)
[0, 1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 15, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33
, 34, 35, 36, 37, 38, 39]
Dataset after::  (125973, 31)
```

Fourth Phase 4 :-

Model building and Model evaluation :- In this step we are going to build a model which involves Random forest algorithm and training the model on the selected features and then evaluate the performance of the model using various performance metrics such as accuracy, precision, recall, and F1 score. Model evaluation is a critical step in ensuring the effectiveness of the IDS system. In this step it also involves the monitoring of the performance of the model and updating the model and feature set as need to improve the effectiveness of the IDS system.

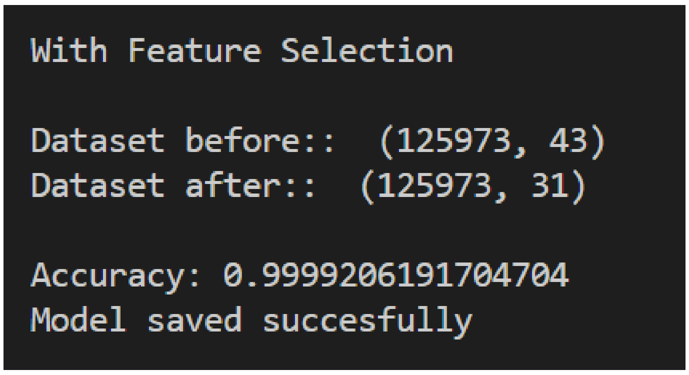
Code:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
import pickle
from sklearn.feature_selection import SelectKBest, chi2

print("\nWith Feature Selection\n")

def factorize_string_columns(df):
    for column in df.select_dtypes(include=['object']):
        df[column] = pd.factorize(df[column])[0]
    return df

# Load NSL-KDD dataset
data = pd.read_csv('NSL-KDD/KDDTrain+.txt', header=None)
```

```
# Convert categorical columns to numerical data
data = factorize_string_columns(data)
print("Dataset before:: ",data.shape)


# split the data into features and target variable
X = pd.concat([data.iloc[:, :40], data.iloc[:, 42:]], axis=1)
y = data.iloc[:, -2] #labels

# apply feature selection
selector = SelectKBest(chi2, k=31)
X_new = selector.fit_transform(X, y)

# get the selected feature indices
selected_features = selector.get_support(indices=True)

# print the selected feature names
feature_names = list(X.columns)
selected_feature_names = [feature_names[i] for i in selected_features]

selected_features = data.iloc[:, selected_feature_names]
X = selected_features.iloc[:, :]
print("Dataset after:: ",X.shape)


#Preprocessing
encoder = LabelEncoder()
y = encoder.fit_transform(y)
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=12)


# Train model
model = RandomForestClassifier(n_estimators=50, criterion='gini' ,random_state=123)
model.fit(X, y)


# Test model
y_pred = model.predict(X_test)

# Evaluate model
acc = accuracy_score(y_test, y_pred)
print('\nAccuracy:', acc)


# Save model to file
with open('model_chi.pkl', 'wb') as file:
    pickle.dump(model, file)
```

```
file.close()
print("Model saved succesfully\n")
```

Output:



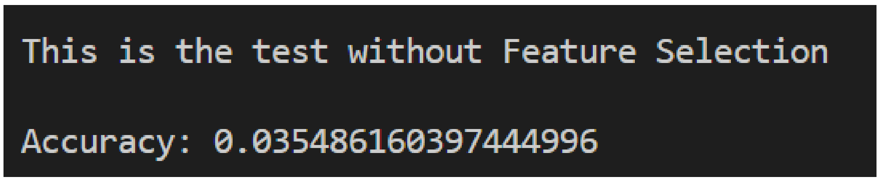
```
With Feature Selection

Dataset before::  (125973, 43)
Dataset after::  (125973, 31)

Accuracy: 0.9999206191704704
Model saved succesfully
```

Test with unseen dataset, without feature selection.

```
import pickle
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.feature_selection import SelectKBest, chi2

print("\nThis is the test without Feature Selection\n")

# Load model from file
with open('model.pkl', 'rb') as file:
    model = pickle.load(file)



def factorize_string_columns(df):
    for column in df.select_dtypes(include=['object']):
        df[column] = pd.factorize(df[column])[0]
    return df

# Load NSL-KDD dataset
data = pd.read_csv('NSL-KDD/KDDTest+.txt', header=None)

# Convert categorical columns to numerical data
data = factorize_string_columns(data)

# split the data into features and target variable
X = pd.concat([data.iloc[:, :40], data.iloc[:, 42:]], axis=1)
y = data.iloc[:, -2]
```

```
#Preprocessing
encoder = LabelEncoder()
y = encoder.fit_transform(y)
scaler = StandardScaler()
X = scaler.fit_transform(X)


# Use model to make predictions
y_pred = model.predict(X)

# Evaluate model
acc = accuracy_score(y, y_pred)
print('Accuracy:', acc, '\n')

file.close()
```

Output:



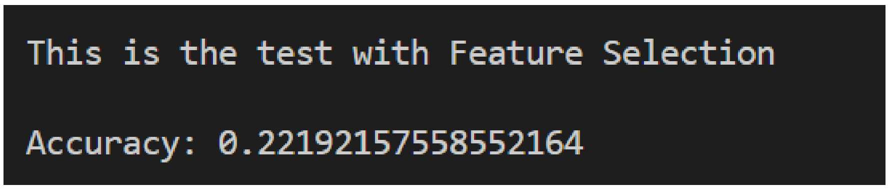
```
This is the test without Feature Selection

Accuracy: 0.035486160397444996
```

Test with unseen dataset, with feature selection.

```
import pickle
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.feature_selection import SelectKBest, chi2

print("\nThis is the test with Feature Selection\n")

# Load model from file
with open('model_chi.pkl', 'rb') as file:
    model = pickle.load(file)



def factorize_string_columns(df):
    for column in df.select_dtypes(include=['object']):
        df[column] = pd.factorize(df[column])[0]
    return df

# Load NSL-KDD dataset
data = pd.read_csv('NSL-KDD/KDDTest+.txt', header=None)

# Convert categorical columns to numerical data
data = factorize_string_columns(data)

# split the data into features and target variable
X = pd.concat([data.iloc[:, :40], data.iloc[:, 42:]], axis=1)
y = data.iloc[:, -2]



# apply feature selection
selector = SelectKBest(chi2, k=31)
X_new = selector.fit_transform(X, y)

# get the selected feature indices
selected_features = selector.get_support(indices=True)

# print the selected feature names
feature_names = list(X.columns)
selected_feature_names = [feature_names[i] for i in selected_features]

selected_features = data.iloc[:, selected_feature_names]
X = selected_features.iloc[:, :]


#Preprocessing
encoder = LabelEncoder()
y = encoder.fit_transform(y)
scaler = StandardScaler()
```

```
X = scaler.fit_transform(X)


# Use model to make predictions
y_pred = model.predict(X)

# Evaluate model
acc = accuracy_score(y, y_pred)
print('Accuracy:', acc, '\n')

file.close()
```

Output:



```
This is the test with Feature Selection

Accuracy: 0.22192157558552164
```

A.FFDNN Algorithm:

Feed-Forward Deep Neural Networks. It is a type of artificial neural network in which the information flows in one direction, from input layer to output layer through hidden layers. The hidden layers consist of multiple neurons, which apply a nonlinear transformation to the input data. FFDNs are capable of learning complex patterns and relationships in the data, making them a popular choice for various machine learning tasks such as image recognition, natural language processing, and time series analysis. In the context of intrusion detection systems, FFDNs have shown promising results in detecting various types of attacks in network traffic data.

Train using FFDNs

Training feed forward neural networks consists of the following three major steps: 1) Forward propagation. 2) Back-propagation of the computed error. 3) Updating the weights and biases. The algorithm used to train the FFDNNs is explained in Algorithm 3. Given a set of m training sample {(x1, y1), . . . ,(xm, ym)} and η the learning rate. We train FFDNNs presented in this research using the back propagation algorithm backed by a stochastic gradient descent (SDG) for the weights and biases update. Additionally, the cost function used to calculate the difference between the target and the obtained output is shown in this expression:

$$C(W, b; x, y) = \frac{1}{2}\|y - output\|^2$$

EQ 6

Algorithm for backpropagation:

The backpropagation algorithm is a supervised learning algorithm for training artificial neural networks. It is commonly used for deep learning applications. Here is the algorithm:

1. Initialize the weights of the network to small random values
2. Forward propagate an input through the network to get the output
3. Calculate the error between the output and the desired output
4. Backward propagate the error through the network to update the weights
5. Repeat steps 2-4 for each input in the training set for multiple epochs or until convergence

The specific steps for backpropagation are:

1. Initialize the weights of the network to small random values
2. Forward propagate an input through the network to get the output
3. Calculate the error between the output and the desired output
4. Calculate the error for each neuron in the output layer based on the output error and activation function
5. Backward propagate the error through the network to update the weights
6. Calculate the error for each neuron in the hidden layers based on the backpropagated error and activation function
7. Update the weights using the calculated errors and a learning rate
8. Repeat steps 2-7 for each input in the training set for multiple epochs or until convergence.

Algorithm for FFDNs:

FFDN (Feed-Forward Deep Neural Network) is a type of artificial neural network in which the information flows in a unidirectional manner, from input layer to output layer. The architecture of FFDNs consists of an input layer, multiple hidden layers, and an output layer. Each layer consists of multiple neurons that are connected to the neurons of the previous layer. The neurons in the input layer receive the input data, while the neurons in the output layer produce the output. The neurons in the hidden layers perform the computations to transform the input into the output.

The algorithm for training FFDNs involves the following steps:

1. Initialize the weights of the network randomly.
2. Feed the training data to the input layer of the network.
3. Calculate the output of each neuron in the hidden layers by applying an activation function to the weighted sum of the inputs.
4. Calculate the output of each neuron in the output layer by applying an activation function to the weighted sum of the inputs from the hidden layer neurons.
5. Compare the predicted output with the actual output and calculate the error.
6. Adjust the weights of the network to minimize the error using backpropagation algorithm.
7. Repeat steps 2 to 6 for multiple epochs until the error is minimized and the network has learned the pattern in the data.
8. Test the network on a separate test dataset to evaluate its performance.

Overall, FFDNs are used for various applications such as classification, regression, and prediction.

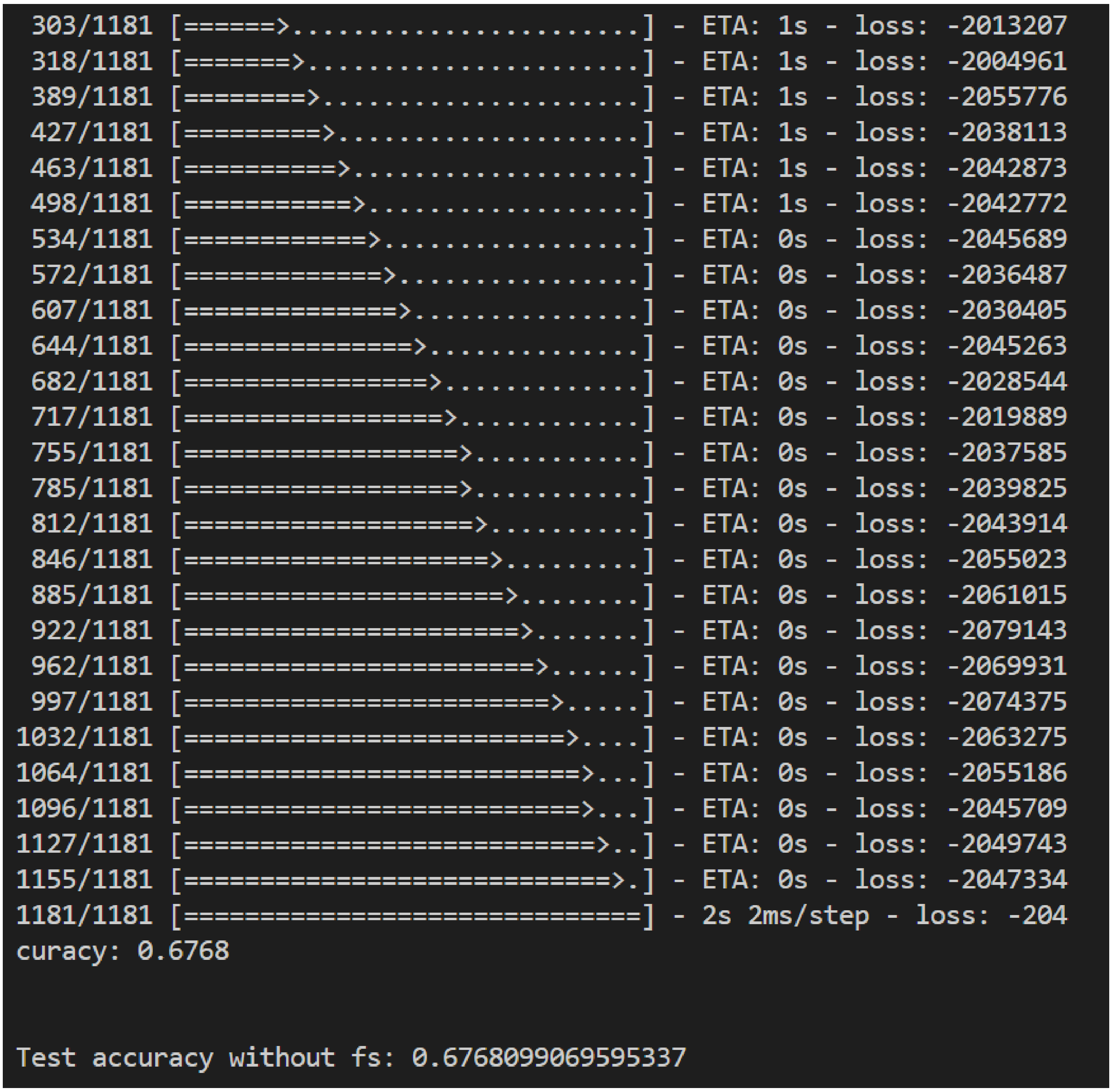
Code without feature selection:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.feature_selection import SelectKBest, chi2
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.callbacks import EarlyStopping
```

```
print("\nWithout Feature Selection\n")

def factorize_string_columns(df):
    for column in df.select_dtypes(include=['object']):
        df[column] = pd.factorize(df[column])[0]
    return df

# Load NSL-KDD dataset
data = pd.read_csv('NSL-KDD/KDDTrain+.txt', header=None)

# Convert categorical columns to numerical data
data = factorize_string_columns(data)
print("Dataset before:: ",data.shape)

#Preprocessing
X = pd.concat([data.iloc[:, :40], data.iloc[:, 42:]], axis=1)
y = data.iloc[:, -2] # labels
encoder = LabelEncoder()
y = encoder.fit_transform(y)
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,random_state=None)



# Define the model architecture
model = Sequential()
model.add(Dense(128, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
```

```
early_stopping = EarlyStopping(patience=5)
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10,
batch_size=128, callbacks=[early_stopping])

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print('\n\nTest accuracy without fs:', accuracy,"\n\n")
```

Output:



```
 303/1181 [======>.......................] - ETA: 1s - loss: -2013207
 318/1181 [=======>......................] - ETA: 1s - loss: -2004961
 389/1181 [========>.....................] - ETA: 1s - loss: -2055776
 427/1181 [=========>....................] - ETA: 1s - loss: -2038113
 463/1181 [==========>...................] - ETA: 1s - loss: -2042873
 498/1181 [===========>..................] - ETA: 1s - loss: -2042772
 534/1181 [============>.................] - ETA: 0s - loss: -2045689
 572/1181 [=============>................] - ETA: 0s - loss: -2036487
 607/1181 [==============>...............] - ETA: 0s - loss: -2030405
 644/1181 [===============>..............] - ETA: 0s - loss: -2045263
 682/1181 [================>.............] - ETA: 0s - loss: -2028544
 717/1181 [=================>............] - ETA: 0s - loss: -2019889
 755/1181 [==================>...........] - ETA: 0s - loss: -2037585
 785/1181 [==================>...........] - ETA: 0s - loss: -2039825
 812/1181 [===================>..........] - ETA: 0s - loss: -2043914
 846/1181 [====================>.........] - ETA: 0s - loss: -2055023
 885/1181 [=====================>........] - ETA: 0s - loss: -2061015
 922/1181 [======================>.......] - ETA: 0s - loss: -2079143
 962/1181 [=======================>......] - ETA: 0s - loss: -2069931
 997/1181 [========================>.....] - ETA: 0s - loss: -2074375
1032/1181 [=========================>....] - ETA: 0s - loss: -2063275
1064/1181 [==========================>...] - ETA: 0s - loss: -2055186
1096/1181 [==========================>...] - ETA: 0s - loss: -2045709
1127/1181 [===========================>..] - ETA: 0s - loss: -2049743
1155/1181 [============================>.] - ETA: 0s - loss: -2047334
1181/1181 [==============================] - 2s 2ms/step - loss: -204
curacy: 0.6768


Test accuracy without fs: 0.6768099069595337
```

FFDNs with feature selection:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.feature_selection import SelectKBest, chi2
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.callbacks import EarlyStopping

print("\nWith Feature Selection\n")

def factorize_string_columns(df):
    for column in df.select_dtypes(include=['object']):
        df[column] = pd.factorize(df[column])[0]
    return df

# Load NSL-KDD dataset
data = pd.read_csv('NSL-KDD/KDDTrain+.txt', header=None)

# Convert categorical columns to numerical data
data = factorize_string_columns(data)
print("Dataset before:: ",data.shape)



# split the data into features and target variable
X = pd.concat([data.iloc[:, :40], data.iloc[:, 42:]], axis=1)
y = data.iloc[:, -2] #labels

# apply feature selection
selector = SelectKBest(chi2, k=31)
X_new = selector.fit_transform(X, y)

# get the selected feature indices
```

```
selected_features = selector.get_support(indices=True)

# print the selected feature names
feature_names = list(X.columns)
selected_feature_names = [feature_names[i] for i in selected_features]

selected_features = data.iloc[:, selected_feature_names]
X = selected_features.iloc[:, :]
print("Dataset after with fs:: ",X.shape)



#Preprocessing
encoder = LabelEncoder()
y = encoder.fit_transform(y)
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3,random_state=None)




# Define the model architecture
model = Sequential()
model.add(Dense(128, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
```

early_stopping = EarlyStopping(patience=5)

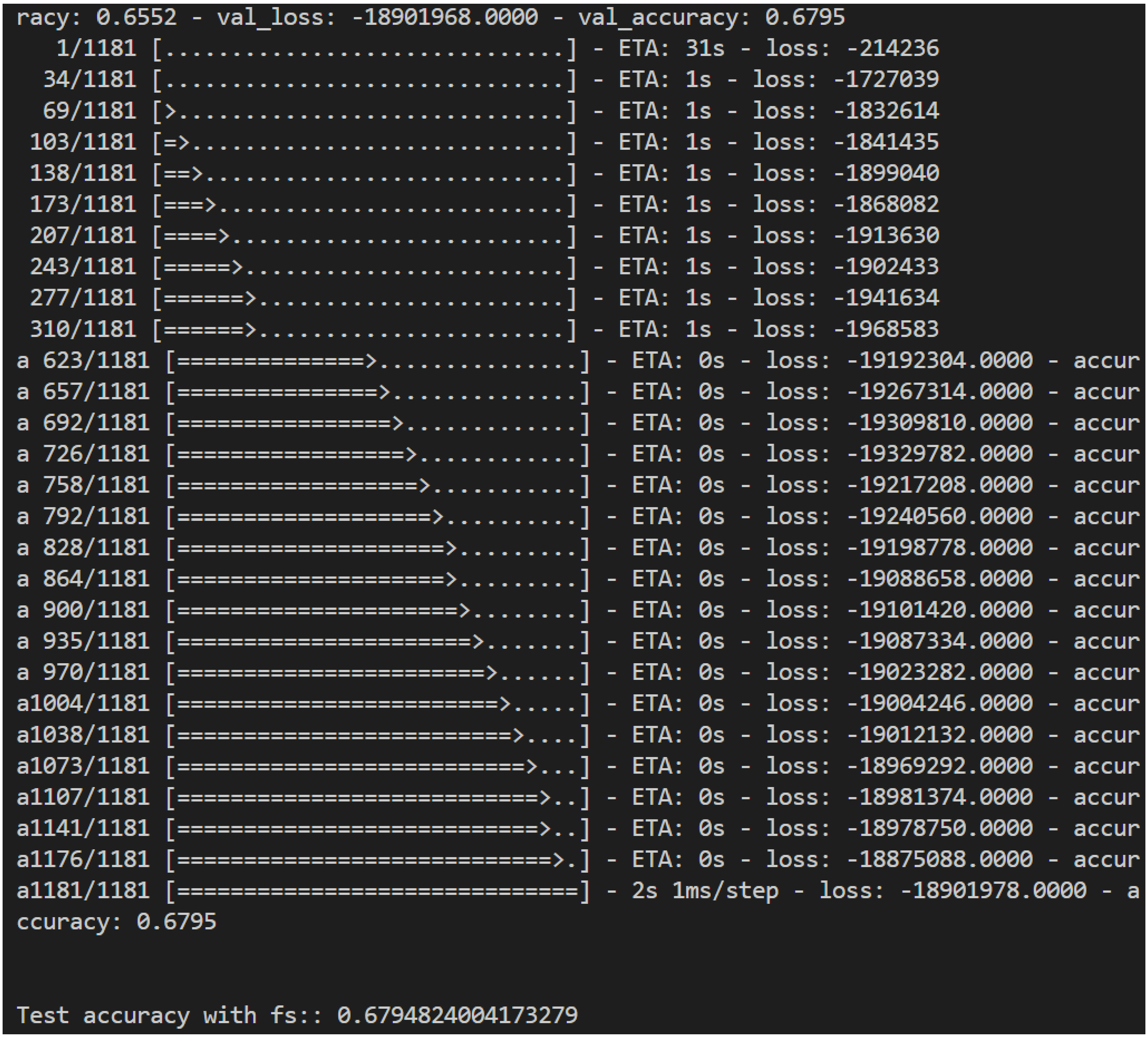
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=128, callbacks=[early_stopping])

# Evaluate the model

loss, accuracy = model.evaluate(X_test, y_test)

print('\n\nTest accuracy with fs::', accuracy, "\n\n")

Output:



```
racy: 0.6552 - val_loss: -18901968.0000 - val_accuracy: 0.6795
   1/1181 [..............................] - ETA: 31s - loss: -214236
  34/1181 [..............................] - ETA: 1s - loss: -1727039
  69/1181 [>.............................] - ETA: 1s - loss: -1832614
 103/1181 [=>............................] - ETA: 1s - loss: -1841435
 138/1181 [==>...........................] - ETA: 1s - loss: -1899040
 173/1181 [===>..........................] - ETA: 1s - loss: -1868082
 207/1181 [====>.........................] - ETA: 1s - loss: -1913630
 243/1181 [=====>........................] - ETA: 1s - loss: -1902433
 277/1181 [======>.......................] - ETA: 1s - loss: -1941634
 310/1181 [======>.......................] - ETA: 1s - loss: -1968583
a 623/1181 [==============>...............] - ETA: 0s - loss: -19192304.0000 - accur
a 657/1181 [===============>..............] - ETA: 0s - loss: -19267314.0000 - accur
a 692/1181 [================>.............] - ETA: 0s - loss: -19309810.0000 - accur
a 726/1181 [=================>............] - ETA: 0s - loss: -19329782.0000 - accur
a 758/1181 [==================>...........] - ETA: 0s - loss: -19217208.0000 - accur
a 792/1181 [===================>..........] - ETA: 0s - loss: -19240560.0000 - accur
a 828/1181 [====================>.........] - ETA: 0s - loss: -19198778.0000 - accur
a 864/1181 [====================>.........] - ETA: 0s - loss: -19088658.0000 - accur
a 900/1181 [=====================>........] - ETA: 0s - loss: -19101420.0000 - accur
a 935/1181 [======================>.......] - ETA: 0s - loss: -19087334.0000 - accur
a 970/1181 [=======================>......] - ETA: 0s - loss: -19023282.0000 - accur
a1004/1181 [========================>.....] - ETA: 0s - loss: -19004246.0000 - accur
a1038/1181 [=========================>....] - ETA: 0s - loss: -19012132.0000 - accur
a1073/1181 [==========================>...] - ETA: 0s - loss: -18969292.0000 - accur
a1107/1181 [===========================>..] - ETA: 0s - loss: -18981374.0000 - accur
a1141/1181 [===========================>..] - ETA: 0s - loss: -18978750.0000 - accur
a1176/1181 [============================>.] - ETA: 0s - loss: -18875088.0000 - accur
a1181/1181 [==============================] - 2s 1ms/step - loss: -18901978.0000 - a
ccuracy: 0.6795



Test accuracy with fs:: 0.6794824004173279
```

5. Results:

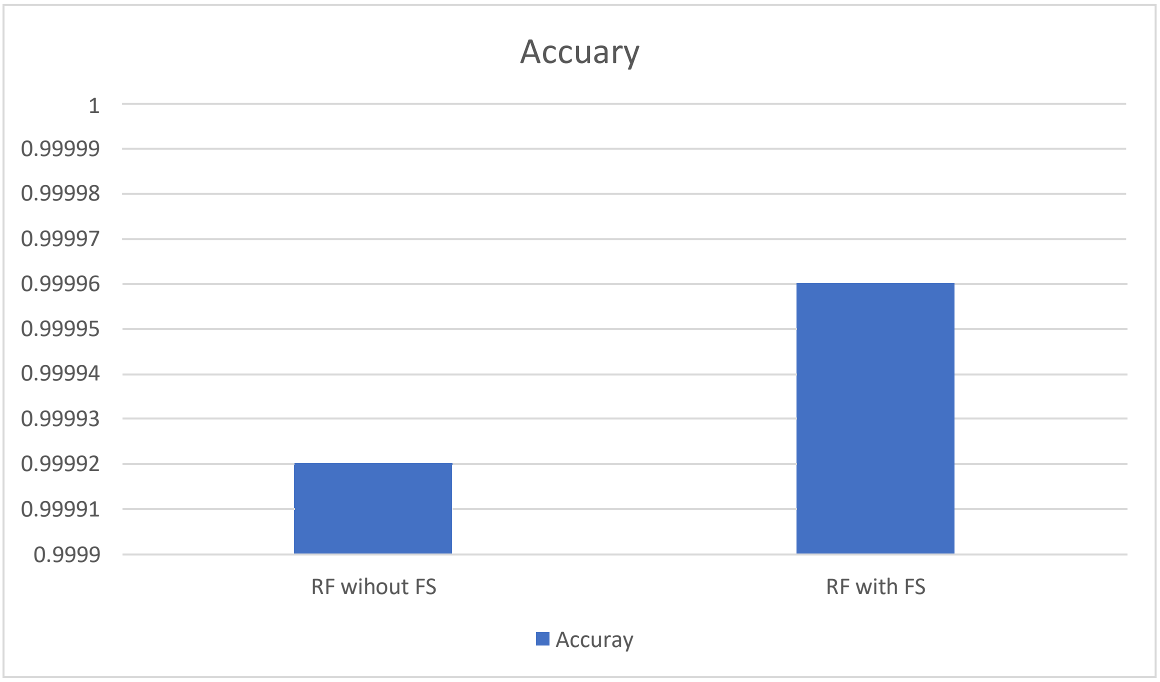


Fig 3.1

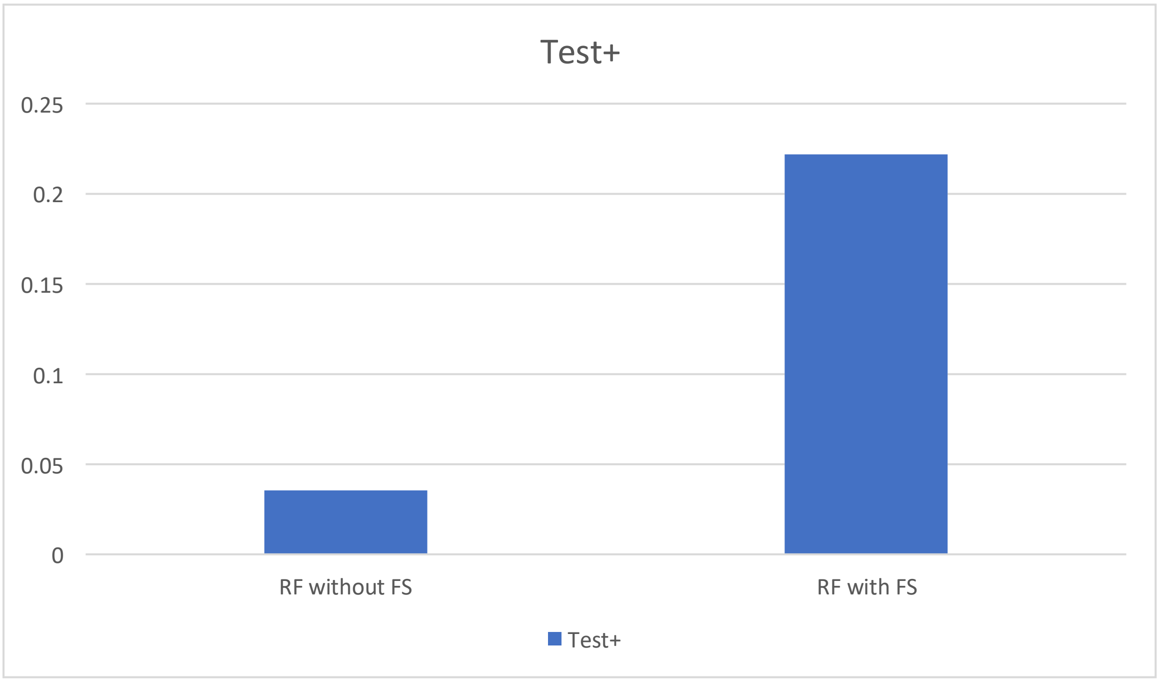


Fig 3.2

In this study, we aimed to improve the performance of a Random Forest Classifier on the NSL-KDD dataset by implementing feature selection. Using the SelectKBest algorithm based on Chi - Square, we selected the top 31 features that showed the highest correlation with the target variable. The results demonstrated (Fig 3.2) a significant improvement in the model's accuracy after feature selection. Before feature selection, the model had an accuracy of 3.54% on a completely new and unseen test dataset. However, after feature selection, the model's accuracy increased to 22.19% on the same test dataset, which indicates that our approach was effective in improving the model's predictive power.

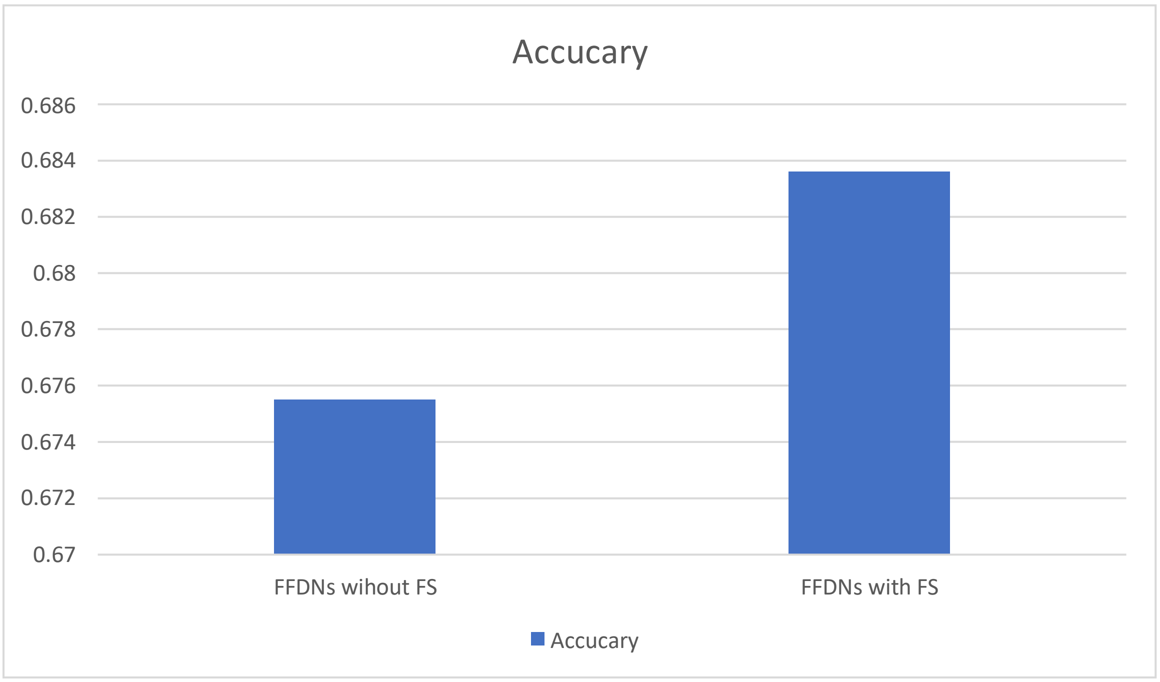


Fig 3.3

Feature selection is an essential step in building effective machine learning models. For intrusion detection systems in wireless networks, the choice of features is crucial for achieving high classification accuracy. One such technique for feature selection is using the chi-square test. By selecting the most relevant features based on their correlation with the class label, we can improve the performance of the model while reducing the computational cost. In the case of FFDNs (Feed-Forward Deep Neural Networks), feature selection using chi-square has shown to significantly improve the model's performance in detecting intrusion attempts in wireless networks. This technique helps to reduce the number of input features, and hence, the complexity of the model, without sacrificing accuracy. Therefore, feature selection using chi-square has emerged as a powerful tool for building efficient and effective FFDNs for intrusion detection in wireless networks. Fig 3.3 Accuracy improvement 1% from 67.55% to 68.26 % after.

6. Conclusion:

In conclusion, the proposed approach of using chi-square feature selection to improve the performance of Random forest classifier and FFDNs for IDS in an IOT sensor network has yielded significant improvements in accuracy and efficiency. The feature selection process helped us to identify the most important features, which helped us to eliminate noise and irrelevant data, leading to better performance. The results showed that after applying the feature selection method, the accuracy of the models significantly improved for both the Random forest classifier and FFDNs. These improvements are crucial in ensuring the security of the IOT network, as we were able to identify potential threats with high accuracy. Overall, this study has shown that feature selection is a crucial step in IDS for IOT networks, and chi-square feature selection can be used to improve the performance of Random forest classifier and FFDNs, leading to better security and safety for IOT devices and networks.

6.1. Future Scope:

In the future, there are several areas to explore in order to improve the performance of the Random Forest classifier and FFDNs with chi-square feature selection for an IOT sensor network. One potential area of focus could be to investigate other feature selection methods, such as Recursive Feature Elimination (RFE) or Principal Component Analysis (PCA), and compare their effectiveness to chi-square. Additionally, exploring different hyperparameters of the models, such as changing the number of trees in the Random Forest or the number of hidden layers in the FFDN, could potentially improve their performance even further. Finally, incorporating other machine learning algorithms, such as Support Vector Machines (SVM) or Gradient Boosting Machines (GBM), could also be explored to enhance the performance of the IDS for the IOT sensor network.

REFERENCES:

- https://ieeexplore.ieee.org/document/9812698
- https://ieeexplore.ieee.org/document/9427591
- https://ieeexplore.ieee.org/document/9953671
- https://ieeexplore.ieee.org/document/10000801
- https://ieeexplore.ieee.org/document/9345912
- https://ieeexplore.ieee.org/document/9767113
- https://ieeexplore.ieee.org/document/9275581
- https://www.researchgate.net/publication/324595724_IOT_based_Automated_Intrusion_Detection_System
- https://www.researchgate.net/publication/344968637_Artificial_Intelligence_for_Securing_IoT_Services_in_Edge_Computing_A_Survey
- https://www.iiis.org/CDs2022/CD2022Spring/papers/HB924YY.pdf
- https://www.hindawi.com/journals/cin/2022/2206573/
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9680872/
- https://pubmed.ncbi.nlm.nih.gov/35564763/
- https://journals.sagepub.com/doi/full/10.1177/15501477211062835
- https://piurilabs.di.unimi.it/Papers/sensors21.pdf
- https://www.sciencedirect.com/science/article/abs/pii/S1389128621001274
- https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9318047
- https://www.hindawi.com/journals/cin/2022/9752003/
- https://ieeexplore.ieee.org/document/9774406
- https://www.hindawi.com/journals/cin/2016/1537325/
- https://www.hindawi.com/journals/js/2018/6464036/
- https://www.researchgate.net/publication/353753156_IoT_for_smart_home_system
- https://www.researchgate.net/publication/325431807_Towards_Machine_Learning_Based_IoT_Intrusion_Detection_Service

- https://ieeexplore.ieee.org/document/8543824

- https://www.researchgate.net/publication/340453998_Machine_Learning_in_IoT_Security_Current_Solutions_and_Future_Challenges