University of Gloucestershire School of Business,

Computing and Social Sciences

MSc Cyber Security

CT7080 Artificial Intelligence 2023/2024

Research Area Machine learning Heuristic-Based malware detection approach

Student:

Darlington Chigozie Okeke

S4318329

June 10, 2024

TABLE OF CONTENTS

1. Machine Learning (ML)

Machine learning is a contemporary approach that allows computers to intelligently adjust to patterns and commands, requiring less programming or supervision (Mahesh, 2020, p. 1). The application of ML in analyzing network traffic and detecting anomalies is on the rise because ML provides high accuracy in classifying and handling large datasets efficiently (Mahesh, 2020, p. 1). Mahesh commented that the most popular ML can be categorized into 7 types: Supervised Learning, Unsupervised Learning, Semi-supervised Learning, Reinforcement Learning, Multi-task Learning, Ensemble Learning (EL), Neural Network (NN), Instance Based Learning (Mahesh, 2020, p. 1). The Artificial Intelligence Machine Learning model employed in this research is categorized into Supervised Learning, Instance-Based Learning and Ensemble Learning Technique which are: Decision Tree (DT), K Nearest Neighbors (KNN), Random Forest(RF), Naïve Bayes (NB) and XGBoost as shown in figure 1.

1.1 ML Algorithm for Intrusion Detection

According to Saranya *et al*., (2020, p. 3), machine learning (ML) can be broadly classed into three categories: supervised, unsupervised, and semi-supervised algorithms. ML has been shown to be effective in accurately identifying threats from massive datasets.

1.2 Supervised ML Algorithm

Supervised learning works efficiently with entire classed labelled data and finds relationship between the trained dataset and the tested dataset (Saranya *et al*., 2020, p. 3). The training and validation dataset is provided for analysis and therefore this class of ML Algorithm is suitable for analyzing network traffic and network intrusion. The classification selected for the analysis falls under supervised ML Algorithm and is Naïve Bayes, K Nearest Neighbour and Neural Network. For regression, the suitable classification for the analysis is Decision Tree and Random Forest (Saranya *et al*., 2020, p. 3).

1.3 Unsupervised ML Algorithm

As there won't be any training for the algorithm, it will classify intrusion detection using unsupervised learning and work independently to find the underlying structure in unlabeled data (Saranya *et al*., 2020, p. 4). This can be accomplished by utilizing algorithms for dimensionality reduction such as Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) or clustering or association analysis using K-means, K-mediods, and C-means (Saranya *et al*., 2020, p. 4).

1.4 Semi-supervised ML Algorithm

This ML technique is a combination of unsupervised and supervised learning. This technique makes use of both labelled and unlabeled data to train and test datasets for classification and regression tasks (Saranya *et al*., 2020, p. 5). A significant amount of unlabeled data and a small amount of labelled data are used by the semi-supervised machine learning technique. This kind of machine learning offers a flexible dataset that enhances intrusion detection systems' precision and forecasting (Saranya *et al*., 2020, p. 5).

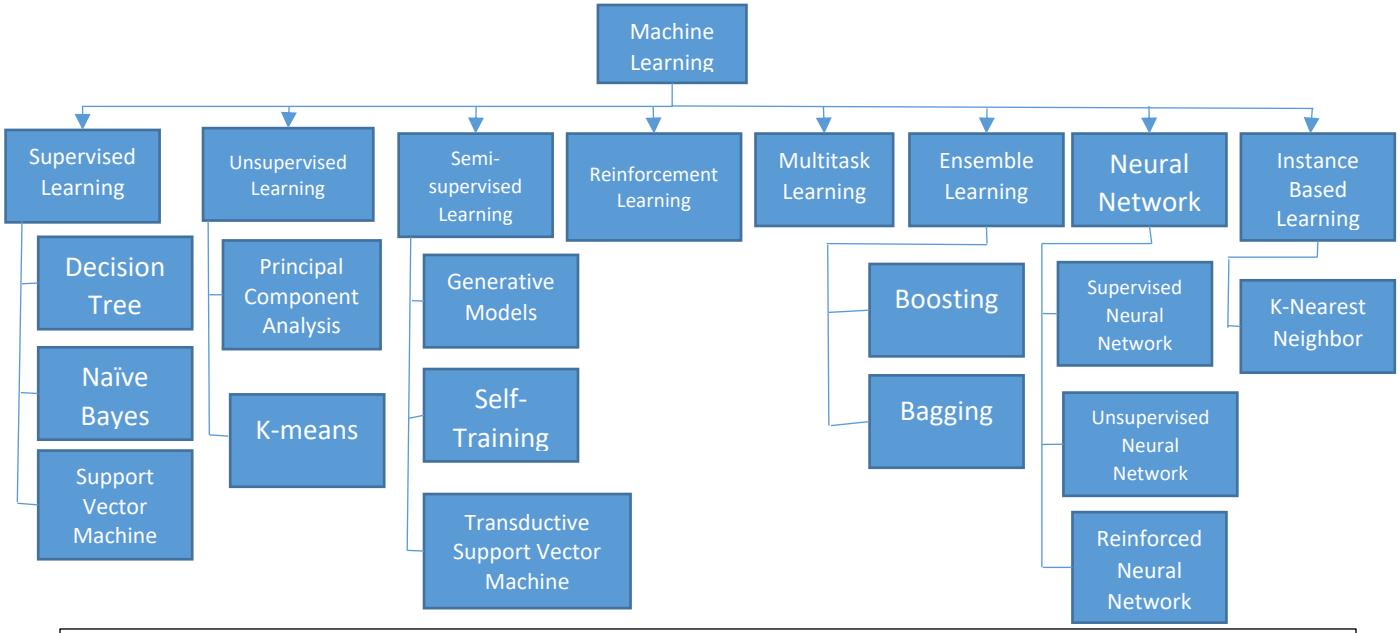

Figure 1. Machine Learning Algorithm
Source: Mahesh (2020 p. 1)

1.5 Random Forest (RF)

'RF is a classification model and ensemble learning approach that makes an intrusion detection prediction based on the majority of votes of randomly selected training sub-set of data' (Gezer *et al*., 2019, p. 13). When a new data (malicious/Benign) is input into the classification, the existing trees participates and make a prediction that determines the final result of the dataset (Gezer *et al*., 2019, p. 7). Among four classifier datasets, RF is a well-liked machine learning method that provides the best classification (Gezer *et al*., 2019, p. 12). The more variables or datasets included, the more accurate the model will be and it won't "over-fit" (Saranya *et al*., 2020, p. 4).

1.6 Table 1: Comparative Analysis of machine learning algorithm. Source: (Elmrabit *et al*., 2020, p. 3).

Algorithm	**Advantages**	**Disadvantages**
Logistic Regression (LR)	Fast speed and low computational cost. One can forecast its result as a likelihood.	Might experience under fitting. When the feature space is big, the performance is disappointing.
Gaussian Naïve Bayes (GNB)	Quick training on both large and small datasets. Less susceptible	It must determine the previous probability. If the characteristics of the sample are similar, it does

	to data gaps. Work on a large dataset effectively	not perform well. Zero frequency/probability issues
K-Nearest Neighbours (KNN)	Regression and classification are two uses for it. Simple to comprehend and apply	Inadequate analysis of unbalanced samples. High computational complexity when dealing with big datasets
Decision Tree (DT)	Quick predictions. Tackling very non-linear data. Capable of working with both categorical and numerical data.	Sufferer from an excessive fit. The model needs more time to be trained.
Adaptive Boosting (AdaB)	Sub-classifiers can be constructed using a variety of algorithms. Does not directly result in over-fitting	The weak classifiers that are chosen will determine the performance. Prone to anomalies. Sensitive to data noise. less rapid than XGBoost
Random Forest (RF)	Able to manage outliers with resilience. In comparison, noise has less of an impact on it.	Longer training period because of the increased tree generation. Much greater resources and processing power are required.
Convolutional Neural Network (CNN)	Effectively manage high-dimensional data using shared convolution kernels	Using gradient descent, the training results converge to the local minimum rather than the global minimum with ease.
Long Short-Term Memory (LSTM)	Gradient disappearance is significantly reduced by gate mechanisms, which also make parameter modifying much easier.	High computing cost
Gated Recurrent Units (GRU)	The gradient vanishing problem is mitigated by gate mechanisms.	High computational cost
Simple Recurrent Neural Network (RNN)	Sequence prediction can make explicit use of contextual information that it has learned.	The gradient vanishing problem arises easily.
Deep Neural Network (DNN)	When compared to conventional ML techniques, it is capable of performing feature engineering independently.	Numerous parameters require adjustment

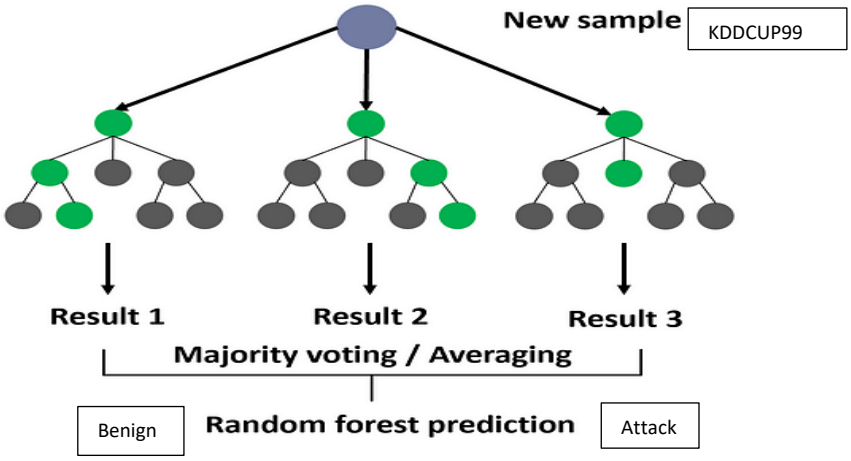

Fig 2. The Random Forest Model.
Source: Roi Medium (2023)

1.7 Deep Learning (DL) Algorithm for Intrusion Detection

DL is a subclass of ML that extracts deep network parameters by utilizing several hidden layers. What sets these methods apart from machine learning is their deep structure and their ability to extract the important features from the information on their own and generate an output. According to Ahmed *et al*., (2020, p.10), there are several types of deep learning algorithms, such as recurrent neural networks, auto encoders, deep neural networks, deep belief networks, and convolutional neural networks.

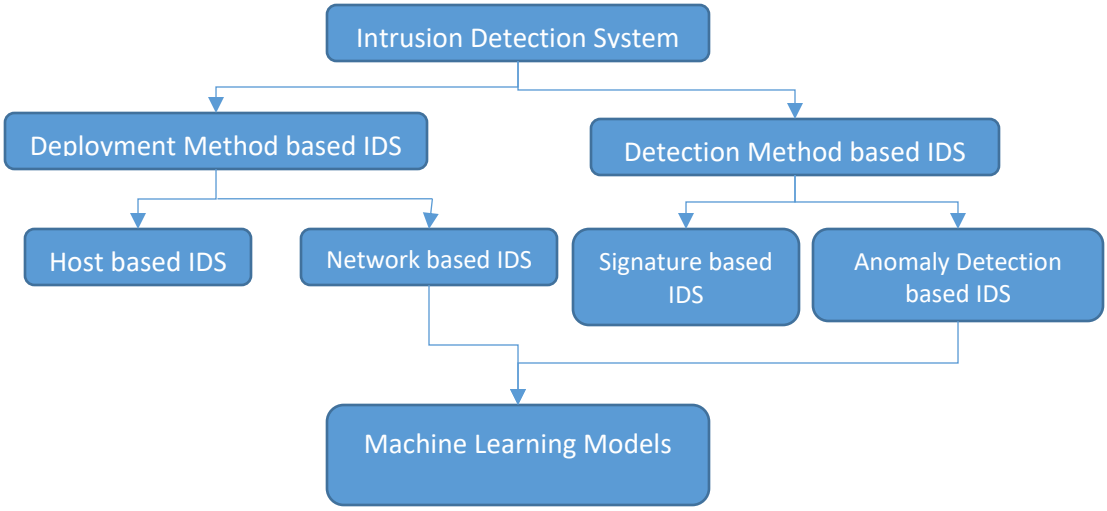

Fig 3. Intrusion Detection System Taxonomy Source: Ahmad *et al*., (2020, p.5)

1.8 Fast Forward Neural Network

A neural network is a set of algorithms that simulates how the human brain functions in order to identify underlying relationships in a batch of data (Mahesh 2020, p. 5). Deep neural networks are so good at classifying inputs as benign or malicious, they are becoming more and more common in security applications like network intrusion detection systems. The different types of DL includes: Artificial Neural Network, Convolutional Neural Network and Recurrent Neural Network (Abou Khamis, and, Matrawy, 2020, p. 1). Neural Network can be classified into three types which are: Supervised Neural Network, Unsupervised Neural Network and Reinforcement Neural Network. The DL technique employed in this analysis is Supervised Neural Network using Feed forward neural network. Because the output of the supervised neural network is known, it is possible to compare the expected and actual outputs (Mahesh, 2020, p. 5). A few exemplary models in the DL area include CNN (Convolution Neural Network) and RNN (Recurrent Neural Network). Since the CNN model gathers low-level characteristics from low layers and convolutional assembles them into high-level features, it is extensively utilized in the image classification space (Jha *et al*., 2020 p. 5). Because recurrent neurons, also known as cells, have feedback connections between the past and present, recurrent neural networks (RNNs) are frequently utilized with sequence data (Jha *et al*., 2020 p. 5).

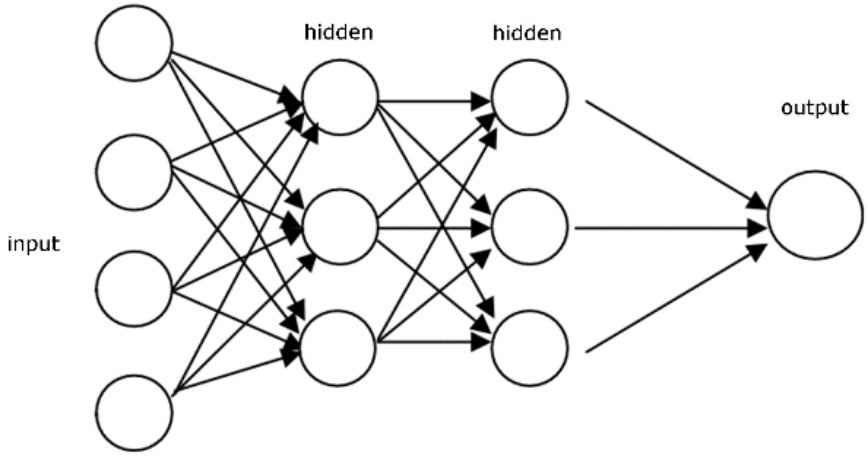

Fig. 4 Galaxy Morphology Classification using Automated Machine Learning
Source: Reza (2020)

The architecture of this design used in my Deep Learning model is of four Layers (Abou Khamis and Matrawy 2020):

- The input layer which is specifically defined as the number of features in the input shape is the first layer of the hidden layer.

- The Hidden Layer which in this case is two hidden layer defined with 128 and 64 units respectively. The two hidden layer uses what is called ReLu (Rectified Linear Units) activation function that engages non-linearity into model. At this stage, dropout function is used to prevent over fitting by selecting 0.5 fraction of the data to zero during the training.

- The number of classes employed in the classification tasks is reflected in the output layer, which has 23 units. The classification report is put together using accuracy evaluation metrics.

- The training dataset is used to train the model, and evaluation is done which provides clarity on the test loss and accuracy. This provides accuracy on the trained datasets and predicated class label.

Parameter	**Value**
No. of hidden layer	3
Layer 1	128 neurons
Layer 2	96 neurons
Layer 3	64 neurons
Dropout	0.5
Optimizer	ADAM
Activation function	ReLU and Softmax
Learning rate	0.01
Epoch	10
Batch Size	32

Table 2: Training, Parameters, and Architecture for IDS Models in Feed Forward Neural Network
Source: The Author (Adapted from: Abou Khamis and Matrawy 2020 p. 3)

2. Application of Random Forest and Deep Neural Network in detecting bad and good Network Traffic from KDDCUP99

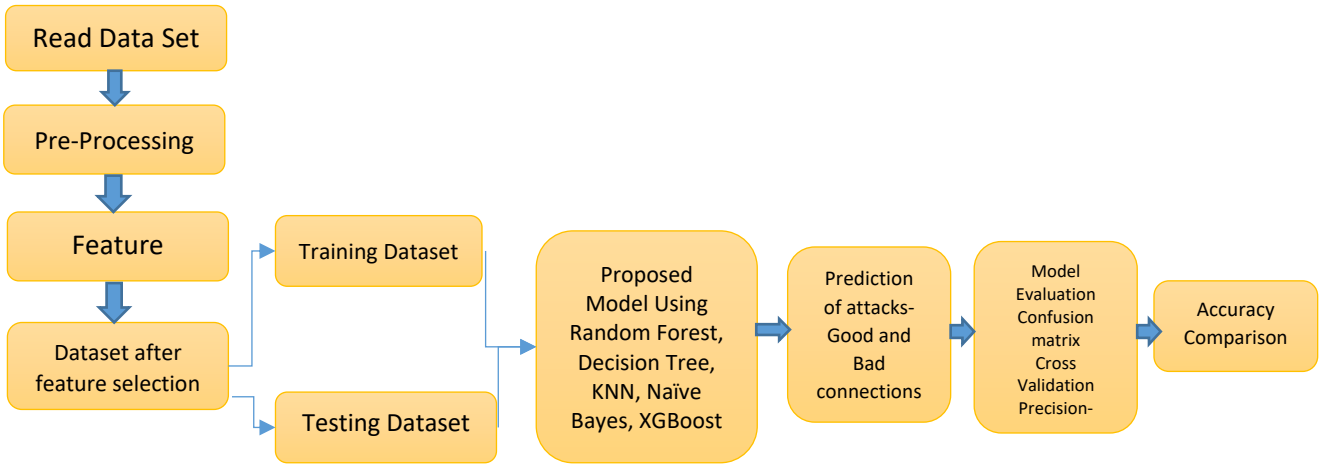

Fig. 5 Intrusion Detection Model Using ML
Source: The Author

2.1 Data sets and Pre-processing

One of the most prominent and extensively used datasets for IDS is KDD Cup'99. For training and testing, it has about two million and five million records, respectively. Each record is classified as either normal or attack and contains 41 distinct traits or attributes. The KDD 99 benchmark, which assesses anomaly detection, includes an explorable data set with a wide variety of simulated intrusions. There are 41 features in the KDD training dataset. Attacks are classified as either normal or malicious depending on how they are assessed (Dhikhi and Saravanan, 2020, p. 3).

The first stage is reading the CSV data file into the Dataframe and reading the dataset's columns and rows as well understanding feature selection. The second involves calculating the frequency of every distinct value in the dataset's "label" column. Subsequently it produces a KDDCUP99 Dataset descriptive statistics. Preprocessing is carried out in order to remove the symbolic and non-numeric elements that are not necessary for the detection procedure (Dhikhi, and Saravanan, 2020, p.5). In the experiment, preprocessing was done by categorical data into numeric ‘convert_dummy_variable’. To determine the value count of the "label" column, the rows and labels must next be categorized. The categorical variables were thereafter converted to dummy variables. Changes are applied to the Dataframe, renaming the final column "target." I currently have "label" and "target." There are 23 attacks in the testing dataset and 21 attack types in the training set.

Two labels—benign (0) and attack (1)—are used to classify objects. There were roughly 100,778 flows in the training set and 25,195 in the testing set. The variables that are 0 are then removed from the Dataframe and changed to features. Next, I scale each feature in the Dataframe to a range of 0 to 1 using min-max scaling to improve the convergence of gradient-based optimization algorithm as seen in Table 1. Disadvantages of Random Forest. At this point, I perform a transformation on the target variable to classify occurrences that are classified as anything other than "normal" as "attacks," and I then count the number of instances in each category. To provide a diagnostic representation of the normal and attacks, I employed a scatter plot and a box plot as seen in Figure 6.

2.2 Feature Extraction

Feature extraction refers to the process of converting unprocessed data into features that are more suitable for machine learning algorithm to work with. Dimension reduction technique is used in this model, Principal Component Analysis (PCA) and the results were converted to scatter plot and box plot to get a good understanding of the numerical relationship. The component used is broken down into two and then the PCA object is fit to the features and then creates a Dataframe for the analysis.

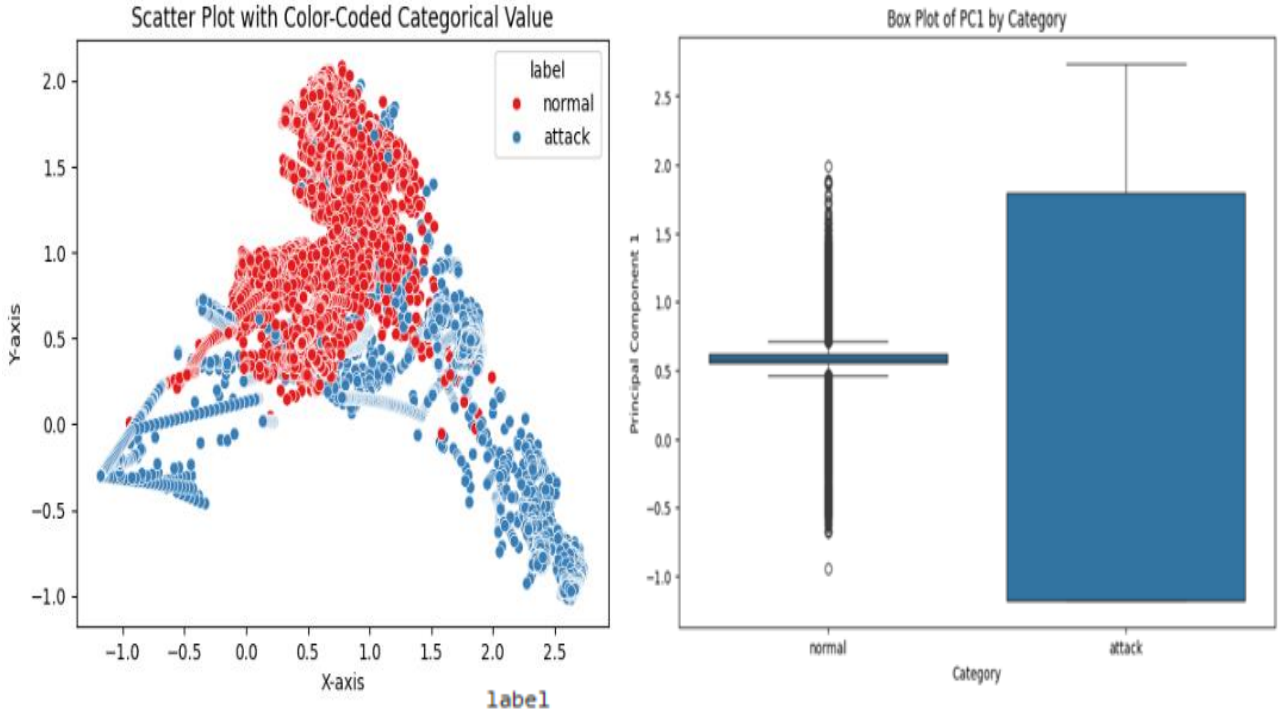

Fig 6. Scatter plot and Box plot
Source: The Author

2.3 Training and Classification

The next step is training the Dataset using the Machine Learning algorithm selected for classification. The final step is classification results derived from the trained and tested data. In this stage, the classification will determine whether the given input is benign or malicious (Agarkar, and Ghosh 2020 p. 5). Training converts the categorical variable to a format suitable for machine learning algorithm. The four steps that were carried out is drop non-numeric column, one-hot encode categorical variables, and convert data type to 'int8' to help with the memory usage and extracting the target variable form the original dataframe. Finally, exclusion of rows where the 'label' column has values corresponding with 'ipsweep' and 'land'.

2.4 Deep Learning Using KDDCUP99 Dataset

Neural Network model using TensorFlow/Keras on classification tasks is used in this experiment. The first stage of the experiment is importing libraries, including Tensorflow, Label Encoder and evaluation metrics from scikit-learn. The second step is to train and test the data by performing fit and transform on label 'y' and split the data into training and testing data. The third step is defining the neural network, add dense layers with ReLU activation function and Dropout layers for regularization. Then I compiled using Adam and loss function (appropriate optimizers). The fourth is to specify the accuracy as the metrics to be performed during the training of the data and then specify the number of epoch to be 20, the number of batch size to be 32, and validation split to be 0.2. Then I evaluated the trained model on the test data using 'model.evaluate' and print the accuracy. The fifth step is I made the prediction using 'model.predict' and converted the predicted model to predicted class using 'tf.argmax'. Lastly I calculated the accuracy, classification report, and confusion matrix using scikit-learn metrics.

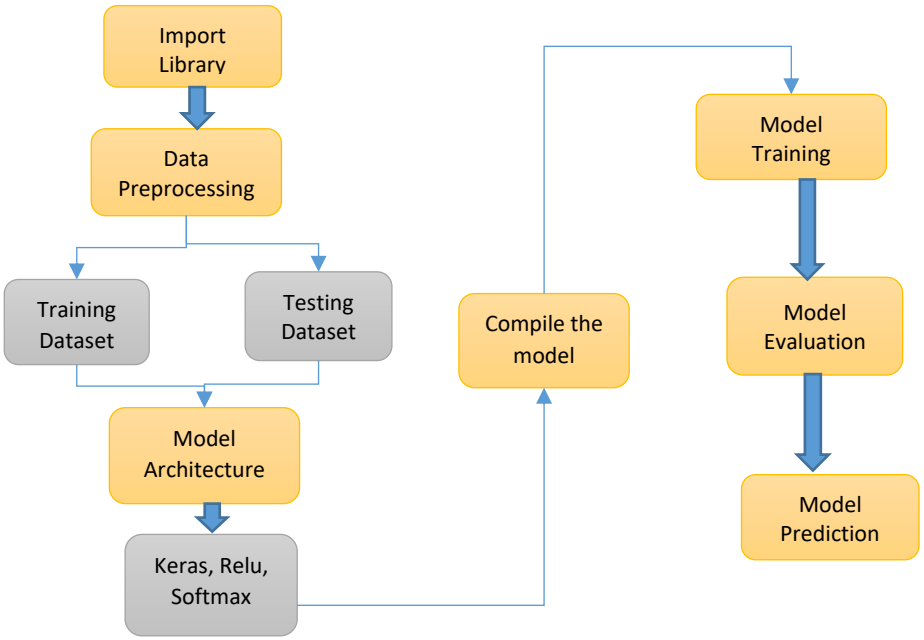

Fig. 7 Neural Network Model Adopted in the Experiment.
Source: The Author

3. Experimental Results

3.1 Performance Evaluation of Deep Learning Neural Network model

Models	Accuracy (%)	Precision	Recall	F-Score	Support	Time in Seconds
Fast Forward Neural Network	0.9996	1.00	1.00	1.00	98804	657.94

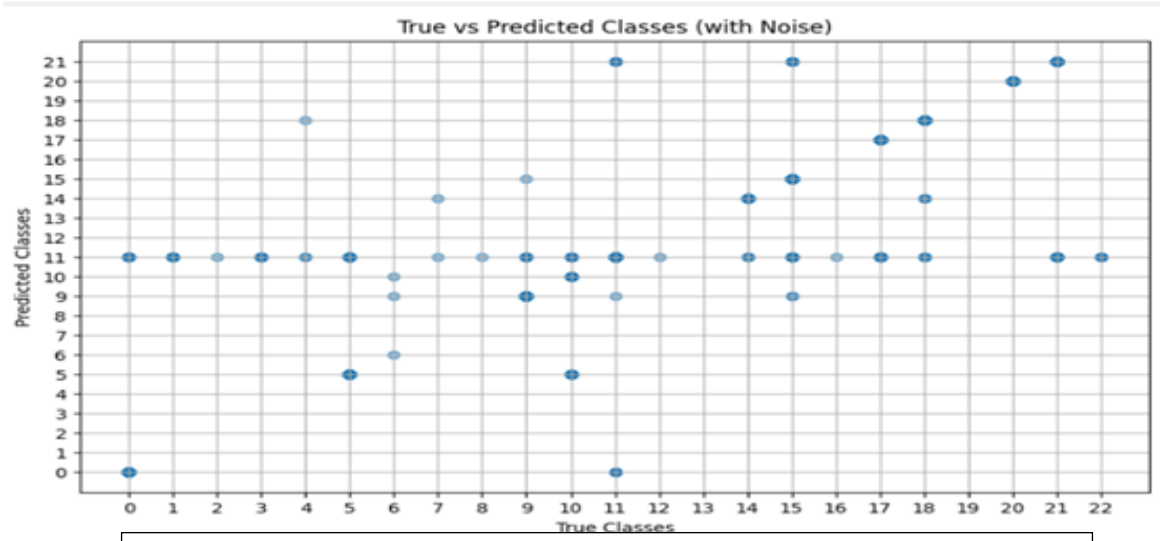

Fig 8. Feed Forward DL artificial neural network scatter plot Accuracy Classification For 23 Class. Source: The Author

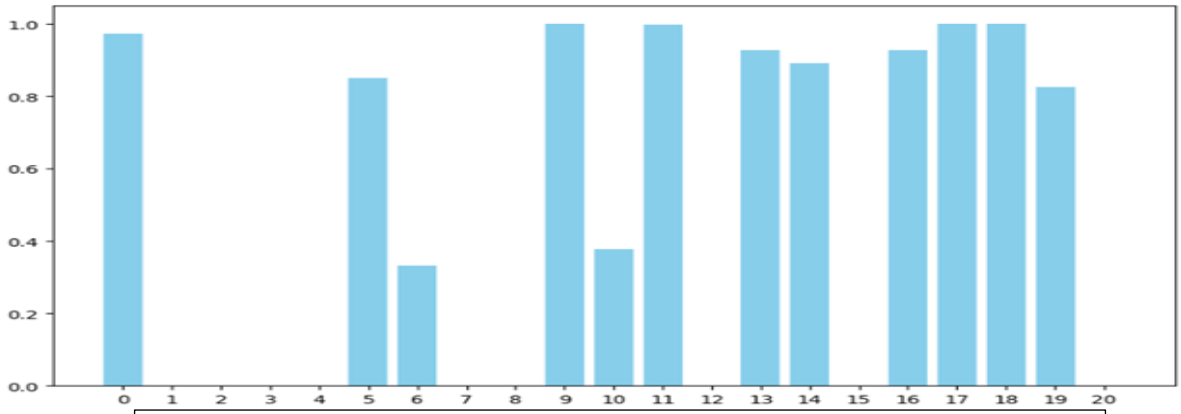

Fig 9. Feed Forward DL bar chart Accuracy Classification for 23 Class Source: The Author

I experimented with the batch size and epoch. One of the decision model's problems was how to handle continuous and multiclass targets at the same time. Figure 10 illustrates this point: higher batch size and epoch counts translate into higher performance at 0.9996 accuracy. A bar graph and scatter plot were used to investigate the true and expected values because a confusion matrix was unable to match the data due to its imbalance as shown in figure 8 and 9

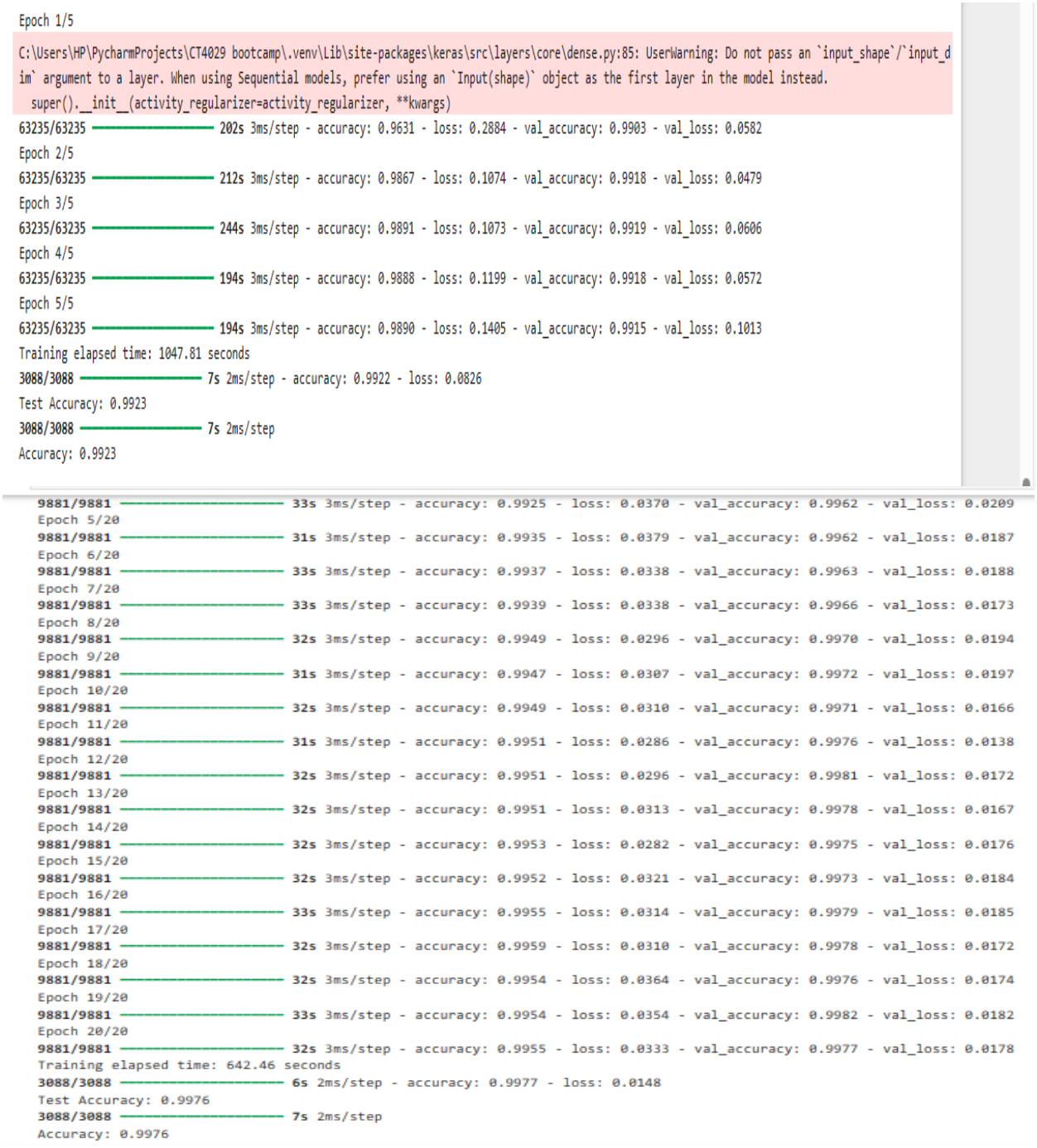

Fig 10. Adjusting the epoch and batch size to ascertain the point of over fitting
Source: The Author

Table 3. Comparative Analysis of Machine Learning of other research
Source: The Author

Author	Machine Learning Techniques	Best Technique Found	Dataset	Accuracy (%)
Shhadat *et al*., (2020)	K Nearest Neighbor, Support Vector Machine (SVM), Bernouli, Naïve Bayes, Random Forest, Decision Tree, Logistic Regression, Hard Voting	Decision Tree (DT), **Random Forest (RF)**	306 features (contains 984 malicious & 172 Benign)	98.2% 95.8%
Irshad *et al*., (2019)	Support Vector, Naives Bayes, RF	**RF**	JSON report	86.8%
Kumar and Geetha (2020)	XGBoost, KNN, DT, AdaBoost, RF, Extra Trees, GB, NB, DT, RF	XGBoost	Open data source	98.5%
Damaševičius *et al*., (2020)	KNN, SVM, DT, RF, Neural Network, AdaBoost, Extra Trees, LDA, Logistic,	Extra Trees	Windows PE Samples (ClamMP datasets)	98.8%

	Passive, Ridge, SGDC			
Narayanan and Davuluru (2020)	SVM, RNN, CNN, LR	LSTM	Microsoft Malware classification challenge (Kaggle)	99.8%
Jha *et al*., (2020)	RNN: One hot encoding, Random feature vector & Word2Vec	RNN	Kaggle	95%
Agarkar and Ghosh (2020)	SVM, Bayesian network, DT, KNN, Neural Network, NB, LR, Light GBM,	Light GBM, **RF**	Open data source	99.50% 99.47%
Kambo *et al*., (2023)	RF, NBT, LMT, J48, Graft, REPT, Gradient boosting, Adaboost, ensemble models	**RF**, XGBoost	Malaca Project for malware samples	99.99% 99.86%
Euh *et al*., (2020)	AdaBoost, XGBoost, RF, Extra Trees, Rotation Trees	XGBoost, **RF**	Web databases (binary classifications)	93%
Kumar et al., (2020)	DT, RF, Gradient Boosting, LR,	**RF**, Decision Tree	Brazilian Malware Dataset	99.7%

	XGBoost, AdaBoost	and XGBoost	(Windows P file)	
Wang, D., *et al*., 2019	KNN, SVM, XGBoost, DT, Gradient Boosting DT, CNN, LSTM	**RF**	Power System	Acc: 93.91% Det: 93.6%
Tuan, T.A *et al*., 2020p. 9	SVM, DT, NB, ANN, USML	USML USML	UNBS-NB 15 KDDCUP99	94.78 98.08

PERFORMANCE EVALUATION

Models	Accuracy (%)	Precision	Recall	F-Score	Time in Seconds
Decision Tree	0.9993	1.00	1.00	1.00	3.75
KNN	0.9974	1.00	1.00	1.00	1379.42
Naïve Bayes	0.8912	0.99	0.89	0.92	4.40
Random Forest	0.9995	1.00	1.00	1.00	48.41

Table 4. Performance of Random Forest with other classification algorithm in this experiment analyzed with weighted average
Source: The Author

3.2 Comparison with other Machine Learning (ML) classification algorithm

The dataset KDDCup99 is analyzed with five ML Algorithm as seen in Table 4. The Dataset comprises of a sub dataset of “494K” Training and “500K” test datasets with a selected feature of “23”. The performance of the classification is to predict Benign and Malicious network traffic. The ML algorithm include DT, KNN, NB and RF. The accuracy comparison of the analysis resulted to Random Forest with the highest accuracy of 0.995 and Naïve Bayes with the lowest accuracy of 0.8912 as seen in in Figure 13. The metrics that were compared in the performance evaluation includes: Precision, Recall, F-Score, and Time in Seconds.

3.3 Predictive Models using Confusion Metrics

A common structure for assessing accuracy is the confusion matrix, sometimes referred to as the error matrix. Its primary function is to make a comparison between the measured values and the classification findings. It has the ability to present the categorization results' accuracy as a confusion matrix (Li *et al*., 2020, p.10). The confusion matrix may show four conditions (Tuan, T.A *et al*., 2020p. 9):

True Positive (TP): The attack was detected in the class characteristic that the classifier correctly identified.
True Negative (TN): The class feature's value is negative, corresponding to normal traffic.
False Positive (FP): When a regular traffic flow is mistakenly classified as an attack by the classifier.
False Negative (FN): An attack record is mistakenly classified as regular traffic by the classifier.

Based on these parameters, we can create seven metrics to assess the performance of the classifiers: accuracy, area under the curve (AUC), false alarm rate (FAR), sensitivity, specificity, false positive rate (FPR), and Matthews's correlation coefficient (MCC) (Tuan, T.A *et al*., 2020 p. 9).

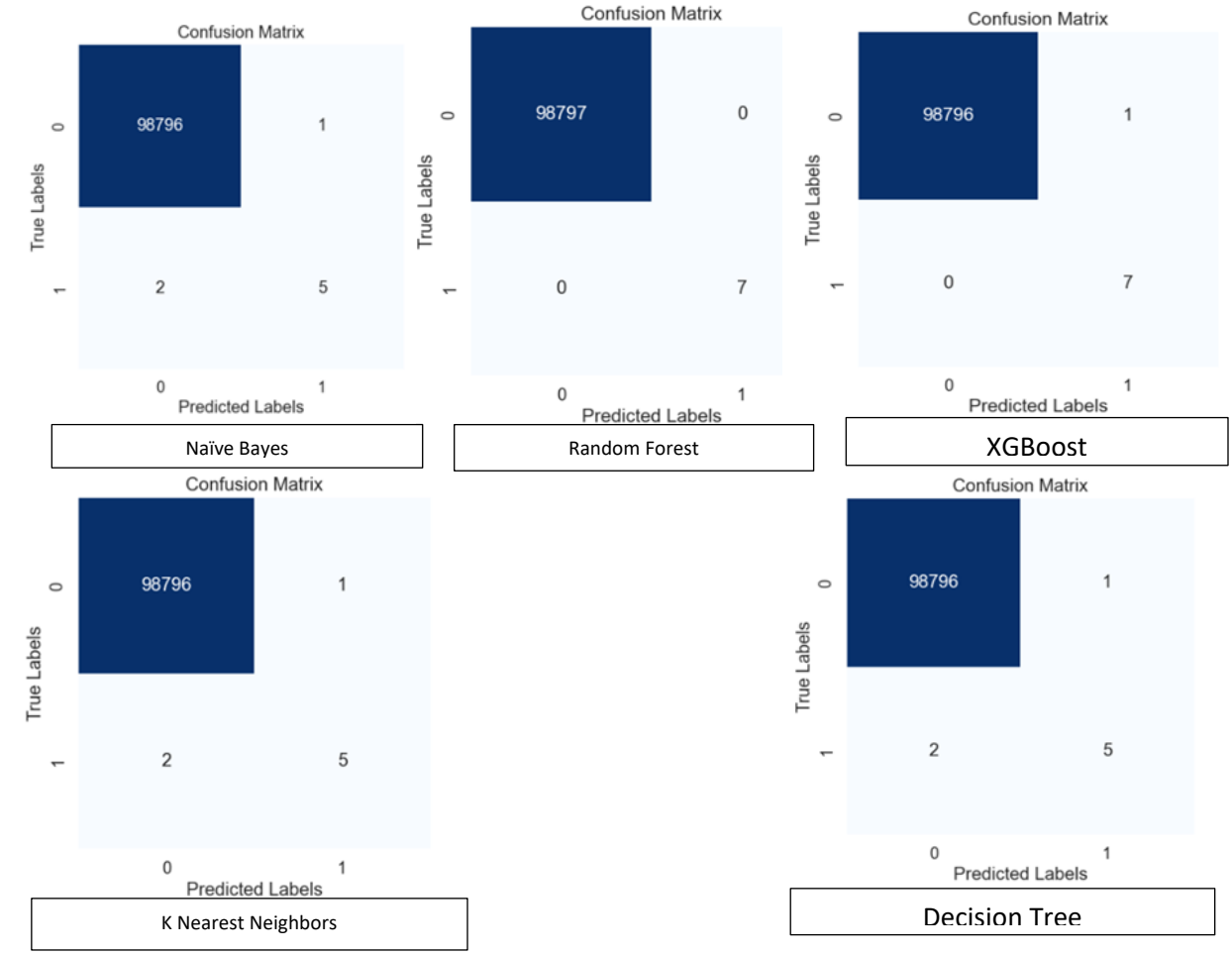

Fig 11. Machine Learning Model Summary of Confusion Matrix: Predictive Labels versus True Labels
Source: The Author

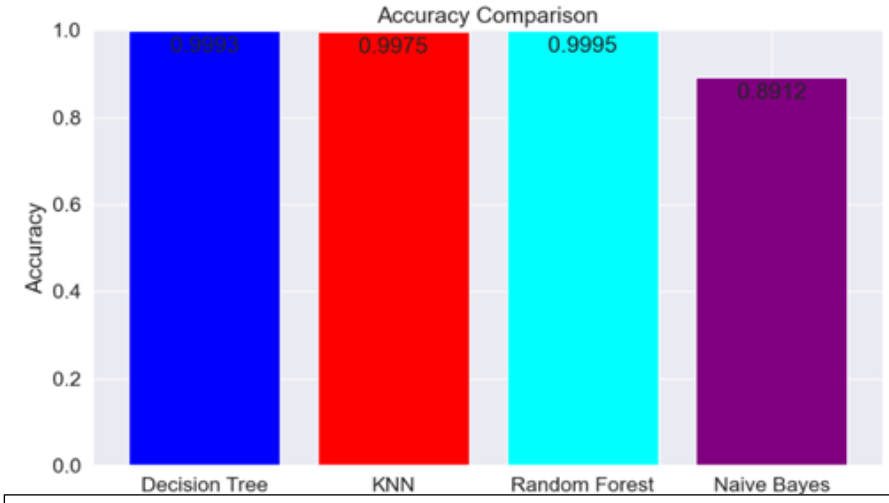

Fig 12. Accuracy Comparison of KDDCup detect network traffic data
Source: The Author

3.4 Performance Metrics

As a result of the dataset been unbalanced as the malware values is more than the normal value. The following evaluation metrics were used to measure the performance of the selected ML algorithm as shown in Table 4: accuracy, precision, True Positive Rate (TPR), also known as Recall, False Positive Rate (FPR), F-1 Score, Receiver Operating Characteristics (ROC) curve and Confusion Matrix (Elmrabit *et al*., 2020, p.5) as shown in Table 4.

- The percentage of all right classifications is known as "accuracy". The proximity of the measurements to one another is referred to as "precision".
- "Recall" quantifies the proportion of true positives that are categorized as attacks.
- "FPR" calculates the proportion of regular traffic that is reported as an assault on regular connection data.
- A measure of test accuracy is the "F1-score".
- "The area under the curve (AUC)", which represents the trade-off between the TPR and FPR using a distinct probability threshold, is a summary of the size of the area under the ROC curve.
- Based on the ability to correctly categorize network traffic into the appropriate attack type, the "confusion matrix" is used to evaluate performance.

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

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

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

$$\text{FPR} = \frac{FP}{FP+FN}$$

$$\text{F1 score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

3.5 AUC

The confusion matrix, which is used to assess the model's capacity for prediction, forms the basis of the ROC curve. Categorization imbalance is a common problem in the real data set. More negative than positive samples are present (or vice versa) (Li *et al*., 2020, p.13). Over time, the testing dataset's positive and negative sample distribution may shift. The ROC curve can stay the same in this situation.

The AUC (Area under Curve) is the area bounded by the coordinate axis under the ROC curve and its maximum value is 1. The detection results are reliable if the AUC is near to 1.0 (Li *et al*., 2020, p.13). The following factors are used to assess the classifier's (prediction model's) quality based on AUC:

AUC = 1, perfect classification.
AUC = [0.85, 0.95], good effect.
AUC = [0.7, 0.85], general effect.
AUC= [0.5, 0.7], less effective.
AUC = 0.5 indicates that the model has no predictive value, similar to a random guess.
AUC <0.5, which is inferior to a random estimate.

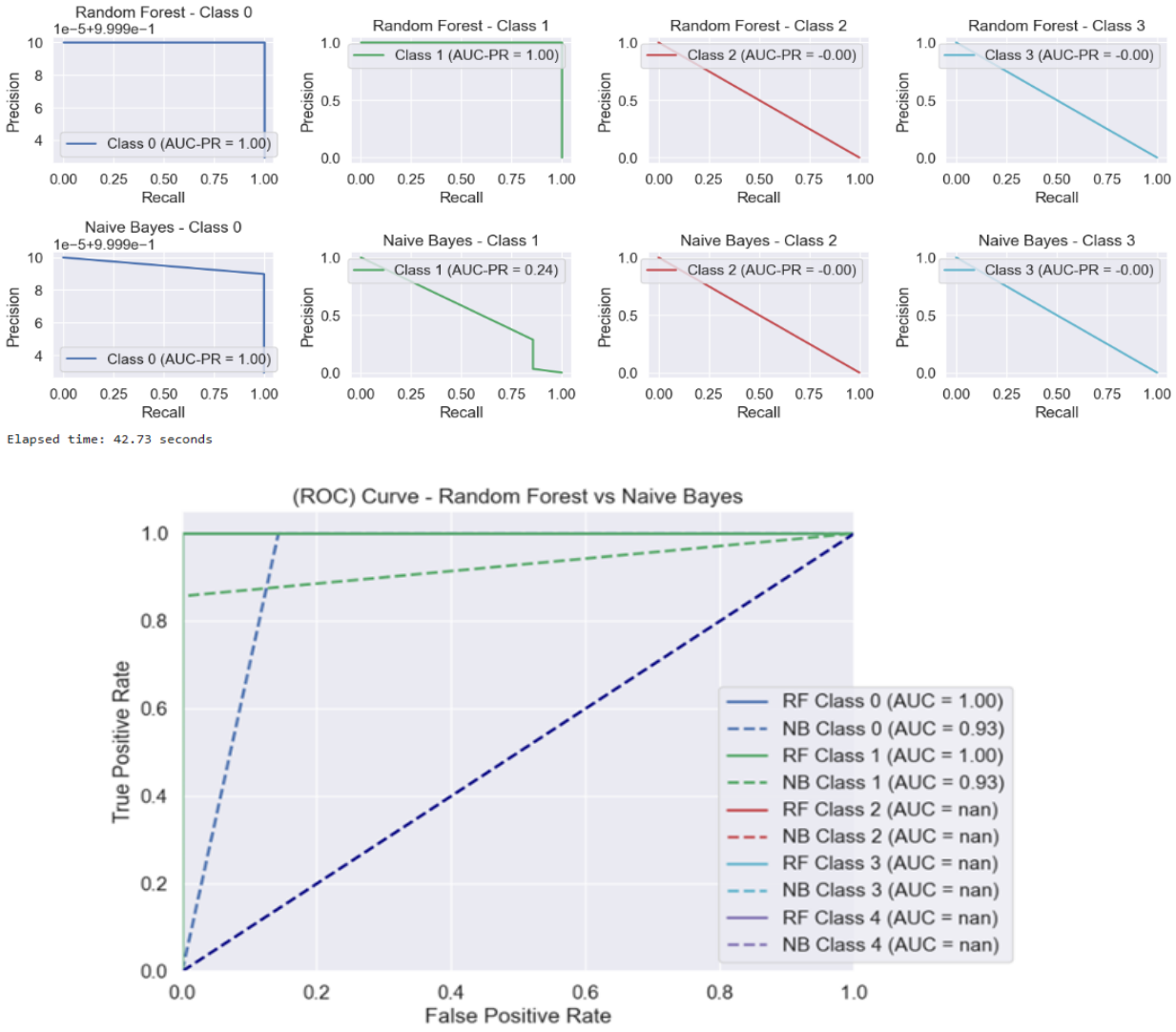

Fig 13. AUC and ROC Curve Metrics of Confusion Metric
Source: The Author

4. Proposed Artificial Intelligence Model

Network attackers have devised adversary means to disrupt the artificial intelligence classification models by inputting false data as input (Roseline, and Geetha, 2021 p. 17). Advancement of network intrusion detection has led to drawbacks which includes: Inability to handle large data and a consistent rate of detection for all types of network intrusion, inefficient adaptability and self-learning ability to suit any situation. Deep Learning breaches the gap and loop holes of these drawbacks due to its strong adaptability and exceptional predicting abilities (Li *et al*., 2020, p.1). Classifiers used to synthesis the results of this study included DT, KNN, RF, NB, XGBoost and Neural Network. Following an analysis of the test and experimental data for all six classifiers, Random Forest achieved 95% accuracy in the test in detecting malware, together with perfect accuracy (1.00), strong recall (1.00), and a strong f1-score (1.00).

Therefore I will propose Random Forest as a reliable, effective, and proactive ML algorithm to identify and prevent network intrusion attacks in the future as shown in Figure 14. The remote employee establishes an internet connection, which flows across the corporate VPN to reach the firewall's RF ML model. Next, train the network traffic to look for IP addresses that seem suspect. A botnet was found in an IP address, which will be blocked and the safe connection will be permitted to access the company's local area network and connect to the machines there.

Cyber security consultants and experts have evolved to ML approaches in malware detection (Muhammad, and Tao, 2023, p.10). Cybersecurity consultants accomplish this by dividing the dataset into training and testing sets for ML algorithms, where the test sets are used to evaluate the algorithm's performance and the training sets are used to teach the algorithm the required function (Muhammad, and Tao, 2023, p.10). In order to accurately mitigate and control security breaches and generate high-quality forecasts, machine learning techniques are used.

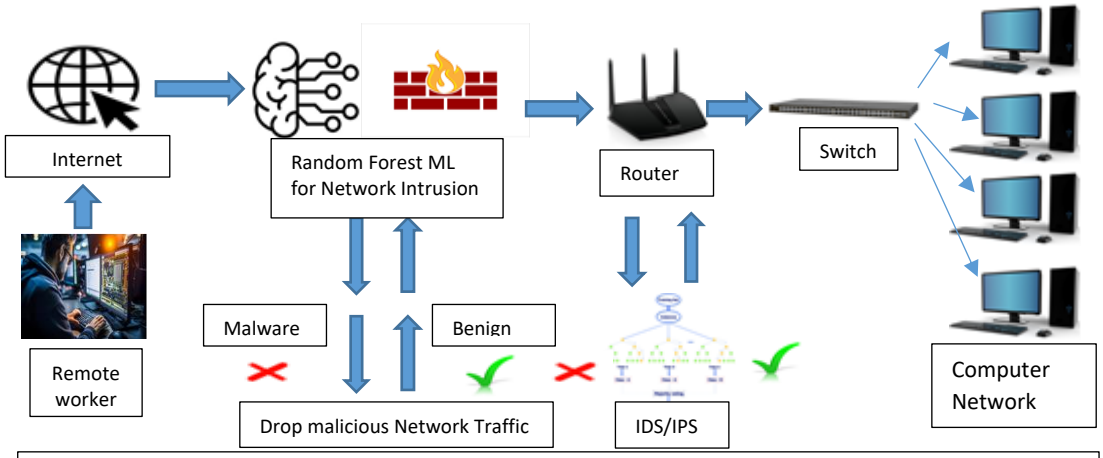

Fig 14. Proposed Random Forest Network Intrusion Detection on internal network traffic.
Source: The Author

5. Appendix A

```
df_filtered = kdd_data_10percent[~kdd_data_10percent['label'].isin([5, 6])]
```

```
# Drop non-numeric column
X = df_filtered.drop(columns=["label"])

# One-hot encode categorical variables
X = pd.get_dummies(X)

# Convert data type if needed
X = X.astype('int8')

# Target variable
y = df_filtered["label"]
```

```
import numpy as np
```

```
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=42)
```

```
train_test_split(y, shuffle=False)
```

```
[0         normal
 1         normal
 2         normal
 3         normal
 4         normal
            ...
 370510    smurf
 370511    smurf
 370512    smurf
 370513    smurf
 370514    smurf
 Name: label, Length: 370515, dtype: object,
 370515    smurf
 370516    smurf
 370517    smurf
 370518    smurf
 370519    smurf
            ...
 494015    normal
 494016    normal
 494017    normal
 494018    normal
 494019    normal
 Name: label, Length: 123505, dtype: object]
```

```
from sklearn.ensemble import RandomForestClassifier
rft = RandomForestClassifier(n_estimators=100, random_state=42)

rft.fit(X_train, y_train)

y_pred = rft.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

cm_rft = confusion_matrix(y_test, y_pred)
```

```
Accuracy: 0.9995
```

Figure A1. Training of the models using Random Forest Machine Learning Model.
Source: The Author

```python
import tensorflow as tf
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Assuming X contains your features as a pandas DataFrame and y contains your original class labels
X_array = X.values

# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Fit and transform labels
y_encoded = label_encoder.fit_transform(y)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_array, y_encoded, test_size=0.2, random_state=42)

# Define the model architecture
model = tf.keras.Sequential([
    tf.keras.layers.Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(23, activation='softmax')  # Output layer with 23 units for 23 classes
])

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',  # Since labels are encoded as integers
              metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.2)

# Evaluate the model on test data
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {test_accuracy:.4f}")
```

```python
# Predictions
y_pred = model.predict(X_test)
y_pred_classes = tf.argmax(y_pred, axis=1)

# Convert y_test to numpy array
y_test = tf.convert_to_tensor(y_test)

# Accuracy
accuracy = accuracy_score(y_test, y_pred_classes)
print(f"Accuracy: {accuracy:.4f}")

# Classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred_classes))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred_classes)
print("\nConfusion Matrix:")
print(cm)
```

Figure A2. Training of the models using Neural Network Deep Learning.
Source: The Author

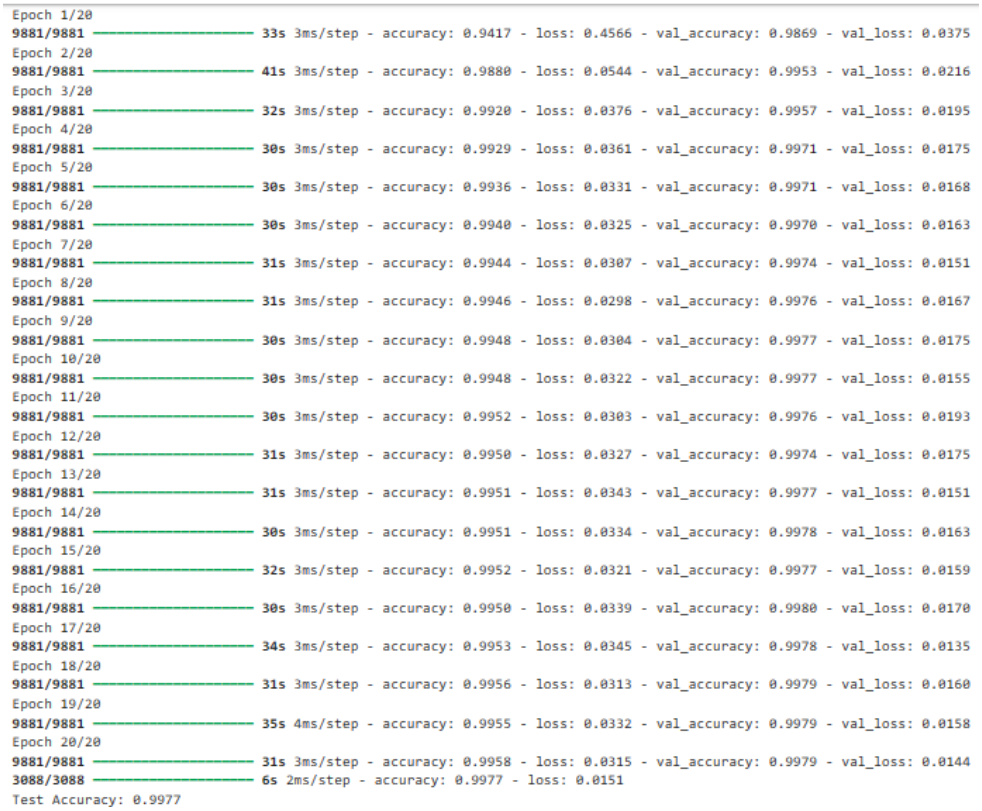

Figure A3. Accuracy of the models using Neural Network Deep Learning.
Source: The Author

6. References

Abou Khamis, R. and Matrawy, A., (2020), October. Evaluation of adversarial training on different types of neural networks in deep learning-based idss. In *2020 international symposium on networks, computers and communications (ISNCC)* (pp. 1-6). IEEE.

Agarkar, S. and Ghosh, S., (2020), December. Malware detection & classification using machine learning. In *2020 IEEE International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC)* (pp. 1-6). IEEE.

Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J. and Ahmad, F., (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, *32*(1), p.e4150.

Akhtar, M.S. and Feng, T., (2023). Evaluation of machine learning algorithms for malware detection. *Sensors*, *23*(2), p.946.

Damaševičius, R., Venčkauskas, A., Toldinas, J. and Grigaliūnas, Š., (2021). Ensemble-based classification using neural networks and machine learning models for windows pe malware detection. *Electronics*, *10*(4), p.485.

Darwish, A., Ezzat, D. and Hassanien, A.E., (2020). An optimized model based on convolutional neural networks and orthogonal learning particle swarm optimization algorithm for plant diseases diagnosis. *Swarm and evolutionary computation*, *52*, p.100616.

Dhikhi, T. and Saravanan, M.S., (2020). An intellectual detection system for intrusions based on collaborative machine learning. *International Journal of Advanced Computer Science and Applications*, *11*(2).

Elmrabit, N., Zhou, F., Li, F. and Zhou, H., (2020), June. Evaluation of machine learning algorithms for anomaly detection. In *2020 international conference on cyber security and protection of digital services (cyber security)* (pp. 1-8). IEEE.

Euh, S., Lee, H., Kim, D. and Hwang, D., (2020). Comparative analysis of low-dimensional features and tree-based ensembles for malware detection systems. *IEEE Access*, *8*, pp.76796-76808.

Gezer, A., Warner, G., Wilson, C. and Shrestha, P., (2019). A flow-based approach for Trickbot banking trojan detection. *Computers & Security*, *84*, pp.179-192.

Irshad, A., Maurya, R., Dutta, M.K., Burget, R. and Uher, V., (2019), July. Feature optimization for run time analysis of malware in windows operating system using machine learning approach. In *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)* (pp. 255-260). IEEE.

Jha, S., Prashar, D., Long, H.V. and Taniar, D., (2020). Recurrent neural network for detecting malware. *computers & security*, *99*, p.102037.

Kamboj, A., Kumar, P., Bairwa, A.K. and Joshi, S., (2023). Detection of malware in downloaded files using various machine learning models. *Egyptian Informatics Journal*, *24*(1), pp.81-94.

Kumar, R. and Geetha, S., (2020). Malware classification using XGboost-Gradient boosted decision tree. *Adv. Sci. Technol. Eng. Syst*, *5*(5), pp.536-549.

Li, X., Chen, W., Zhang, Q. and Wu, L., (2020). Building auto-encoder intrusion detection system based on random forest feature selection. *Computers & Security*, *95*, p.101851.

Mahesh, B., (2020). Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, *9*(1), pp.381-386.

Maniriho, P., Mahmood, A.N. and Chowdhury, M.J.M., (2022). A study on malicious software behaviour analysis and detection techniques: Taxonomy, current trends and challenges. *Future Generation Computer Systems*, *130*, pp.1-18.

Narayanan, B.N. and Davuluru, V.S.P., (2020). Ensemble malware classification system using deep neural networks. *Electronics*, *9*(5), p.721.

Reza, M. (2020) Galaxy Morphology Classification using Automated Machine Learning.

Roi Medium. (2023) https://medium.com/@roiyeho/random-forests-98892261dc49 (Accessed: 4 April 2024)

Roseline, S.A. and Geetha, S., (2021). A comprehensive survey of tools and techniques mitigating computer and mobile malware attacks. *Computers & Electrical Engineering*, *92*, p.107143.

Saranya, T., Sridevi, S., Deisy, C., Chung, T.D. and Khan, M.A., (2020). Performance analysis of machine learning algorithms in intrusion detection system: A review. *Procedia Computer Science*, *171*, pp.1251-1260.

Shhadat, I., Hayajneh, A. and Al-Sharif, Z.A., (2020). The use of machine learning techniques to advance the detection and classification of unknown malware. *Procedia Computer Science*, *170*, pp.917-922.

Wei, P., Li, Y., Zhang, Z., Hu, T., Li, Z. and Liu, D., (2019). An optimization method for intrusion detection classification model based on deep belief network. *Ieee Access*, *7*, pp.87593-87605.

Wang, D., Wang, X., Zhang, Y. and Jin, L., (2019). Detection of power grid disturbances and cyber-attacks based on machine learning. *Journal of information security and applications*, *46*, pp.42-52.

Tuan, T.A., Long, H.V., Son, L.H., Kumar, R., Priyadarshini, I. and Son, N.T.K., (2020). Performance evaluation of Botnet DDoS attack detection using machine learning. *Evolutionary Intelligence*, *13*(2), pp.283-294.