INTRUSION DETECTION SYSTEM USING DEEP LEARNING

Bachelor Thesis

by

SOMESH KUMAR

PURVANSH SONTHALIA

VEMANA JOSHUA IMMANUEL

A thesis submitted to

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, KALYANI

for the partial fulfillment of the degree of

Bachelor of Technology

in

Computer Science and Engineering

May,2023

Certificate

This is to certify that this report entitled ***"Intrusion detection system using Deep Learning"*** *is a bonafied record presented by* ***Purvansh Sonthalia (Roll No. CSE/20066), Somesh Kumar (Roll No. CSE/20082), Vemana Joshua Immanuel (Roll No. CSE/20100)*** *ungraduate students in the Department of* ***Computer Science and Engineering, Indian Institute of Information Technology Kalyani, India****, for the award of Bachelor of Technology in Computer Science and Engineering, is an original research work carried by him under my supervision and guidance. The thesis has fulfilled all the requirements as par the regulation of IIIT Kalyani and in my opinion, has reached the standards needed for submission. The works, techniques and the results presented have not been submitted to any other university or Institute for the award of any other degree or diploma.*

SK HAFIZUL ISLAM,Ph.D

Assistant Professor

Dept. of Computer Science and Engineering

Indian Institute of Information Technology, Kalyani

Kalyani, W.B. - 741235, India.

Acknowledgment

First of all, we would like to take this opportunity to thank our supervisor ***Dr. SK Hafizul Islam*** *without whose effort this thesis would not have been possible. We are so grateful to him for working tirelessly on us, answering our doubts whenever and wherever possible. We are most grateful to Department of Computer Science and Engineering, IIIT Kalyani, India, for providing us this wonderful opportunity to complete our bachelor thesis.*

And last but the biggest of all, we want to thank our parents, for always believing in us and letting us do what we wanted, but keeping a continuous check that we never wandered off the track from our goal.

Purvansh Sonthalia (CSE/20066)

Somesh Kumar (CSE/20082)

Vemana Joshua Immanuel (CSE/20100)

Dept. of Computer Science and Engineering

Indian Institute of Information Technology, Kalyani

Kalyani, W.B. - 741235, India.

Abstract

This report provides a comprehensive study on Intrusion Detection Systems (IDS) in Networks, starting with an introduction to the topic, a recap of previous work, and a literature survey. The content then delves into various models for intrusion detection, including Logistic Regression, Naive Bayes, Support Vector Machine, Decision Tree, Random Forest, and Artificial Neural Network.

The report analyzes the models' tolerance to noise at different levels and proposes a new model based on Random Forest using Principal Component Analysis (PCA). The proposed model's performance is evaluated in both noiseless and noisy data scenarios, and the results show that the Random Forest with PCA outperforms the other models, especially in the presence of noise.

Finally, the report concludes with a summary of the findings and their implications, emphasizing that the proposed model can be an effective way to detect anomalies in noisy data. Overall, this study highlights the importance of choosing the right model for IDS and considering the impact of noise on its performance.

Contents

Chapter 1

Introduction

Intrusion Detection is a set of techniques and methods that are used to detect suspicious activity both at the network and host level. An intrusion detection system (IDS) inspects all inbound and outbound network activity and identifies suspicious patterns that may indicate a network or system attack from someone attempting to break into or compromise a system/ network.

An IDS installed on a network provides much the same purpose as a burglar alarm system installed in a house, it detect when an intruder/attacker/burglar is present in the system, and subsequently issue some type of warning or alert.

The wide spreading usages of the internet and increases in access to online contents, cybercrime is also happening at an increasing rate. Intrusion detection is the first step to prevent security attacks. IDS detects attacks from a variety of systems and network sources by collecting information and then

analyzes the information for possible security breaches. As the number of IoT devices and other smart devices which are connected to the network continue to multiply wildly, so do the security issues associated with it. Once an IDS system detects a security threat, it can respond in different ways. It can generate an alert to notify security personnel, block the offending traffic, or even automatically shut down the system to prevent further damage. The response depends on the severity of the threat and the level of risk to the network.

In conclusion, IDS is an important component of network security and plays a vital role in maintaining the integrity and confidentiality of network data. However, IDS is not a complete solution on its own and should be used in conjunction with other security measures, such as firewalls, antivirus software, and access controls, to create a comprehensive security strategy.

Chapter 2

Recap of Previous Work

2.1 IDS in smart meters

2.1.1 Drawbacks of traditional electricity meters

Traditional meters require manual readings, where a person comes to the customer's home to record the meter's reading. The data is then taken to the utility office to generate bills based on the current electricity tariff/charges. This process is time-consuming and can lead to errors since the readings are taken manually. Additionally, consumers may not be aware of the current tariff/charges for each unit of electricity, leading to confusion when receiving their bills. Moreover, the bill distribution process can also be delayed, taking several days to get completed, which can cause inconvenience to the customers. These problems have led to a search for a more efficient and accurate way of measuring electricity consumption, which can overcome these limitations.

2.1.2 How does a smart meter resolve these issues

A smart meter is a modern solution to the issues encountered in traditional metering systems. Smart meters utilize radio frequency (RF) communication to send data to a data concentrator unit (DCU), which then relays the information to Utility Control Centers (UCC) via the General Packet Radio Services (GPRS) network. The UCC generates bills based on the current tariff/charges of electricity, and this whole process takes only a few minutes to complete. The smart meter system also allows for two-way communication, which means that consumers can receive data from the UCC on their electricity usage and tariff rates in real-time. The smart meter also comes equipped with an in-home display device that provides hourly, daily, and monthly updates on electricity consumption.

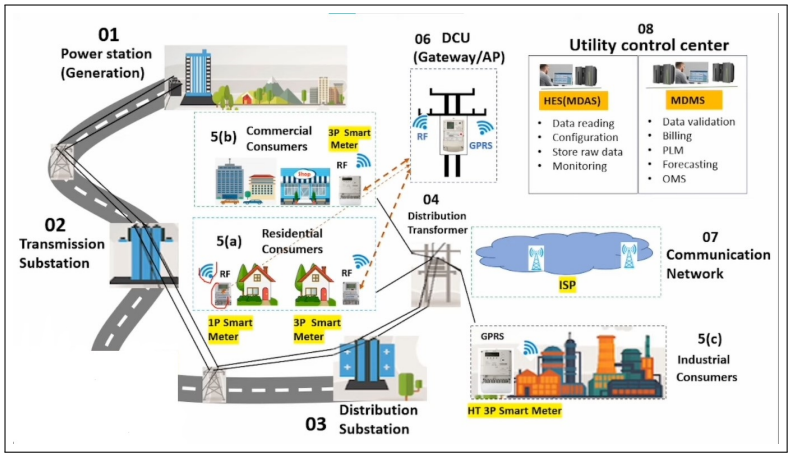

Figure 2.1:

Smart meters also provide the option for both prepaid and postpaid services. However, the 2-way communication system of smart meters makes

the grid susceptible to security threats and intrusion. Attackers can compromise the data that is transmitted between the smart meter and UCC, and consumers can also manipulate the system by under-reporting their actual electricity consumption. Utility companies may also manipulate the system to charge consumers for more electricity usage than what was actually consumed. Therefore, it is important to implement effective security measures to protect against these potential threats.

2.1.3 Training a Neural Network on CICIDS 2017 Dataset

SVM (Support Vector Machine) is a linear model used for classification and regression problems in machine learning. The algorithm creates a line or hyperplane that separates the data into different classes. In ML-based IDS, SVM is commonly used as a classifier. However, the complexity of SVM is directly proportional to the number of inputs, which can affect the model's efficiency. Furthermore, for large datasets, more time and memory are required, which can be a bottleneck in IDS performance.

To address these issues, deep learning algorithms are increasingly being used as multiclass classifier IDS. These models involve a large number of hidden layers arranged in a hierarchical order. Each classifier is binary in nature and detects a specific type of attack in the network. The output of one layer is fed as input to the next layer until the final output is generated.

In line with previous research, we trained a deep learning model on the key-logging section of the CICIDS 2017 dataset. We trained the neural network 1000 times on the entire dataset and plotted the loss function (y-axis) against the number of iterations (x-axis) to evaluate the model's performance.

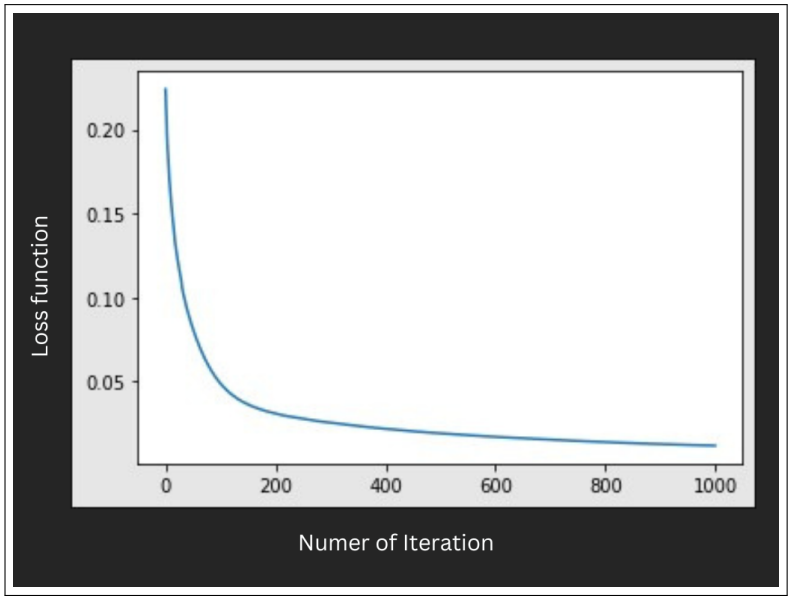

Figure 2.2:

2.2 IDS in CAN BUS

The figure shows that the engine control unit communicates with other controllers such as the odometer and brakes over the Controller Area Network (CAN), among other bus systems. Normal and safety critical communications occur in this manner. How is a car connected to Network? (CAN-Control Area Network) Several Intelligent Transportation Systems (ITSs) applications have been proposed and implemented in recent years to improve drivers experiences and road safety.This design was not a problem when automobiles were standalone entities, however, today, ITS applications communicate with external entities, such as other vehicles in the case of CACC. This makes it possible for the attackers to exploit vulnerabilities. From [9] the community has proposed several prevention and detection solu-

tions to address cyber security threats to connected vehicles. The preventive solutions typically require the modification of the CAN protocol, an impracticality for used and new automobiles . The detection based solutions (or IDS), often use artificial data, data collected from simulators , or data related to researchers devices to detect malicious behaviours which limits the practicality of the results.

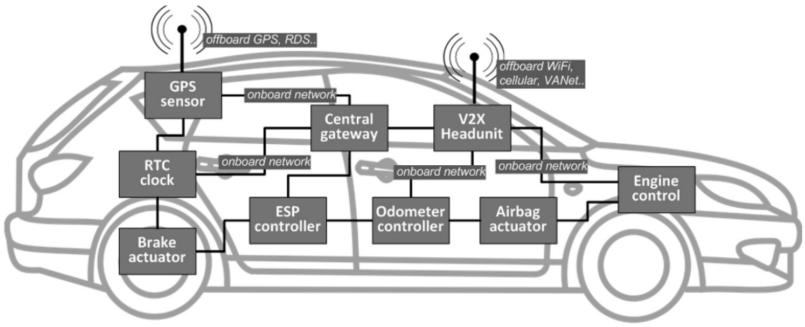

Figure 2.3: Electronic Control Units communicate using in-vehicle networks

2.3 Network Intrusion Detection System

This[8] cutting-edge approach introduces a Fast Hierarchical Deep Convolutional Network for detecting different types of intrusions in a server network. A key advantage of this technique is its remarkable reduction in execution time, achieving up to a 36% decrease compared to previous methods. This improvement is primarily attributed to the utilization of the SRS (Soft Root Sign) activation function between network layers. The model demonstrates

an impressive average accuracy of 98% in accurately classifying connections as either intrusions or non-intrusions, covering a wide range of intrusion types.

$$SRS(p) = \frac{p}{\frac{p}{\alpha} + e^{-\frac{p}{\beta}}} \qquad SRS'(p) = \frac{(1 + \frac{p}{\beta})e^{-\frac{p}{\beta}}}{(\frac{p}{\alpha} + e^{-\frac{p}{\beta}})^2}$$

Figure 2.4:

2.3.1 Model Structure

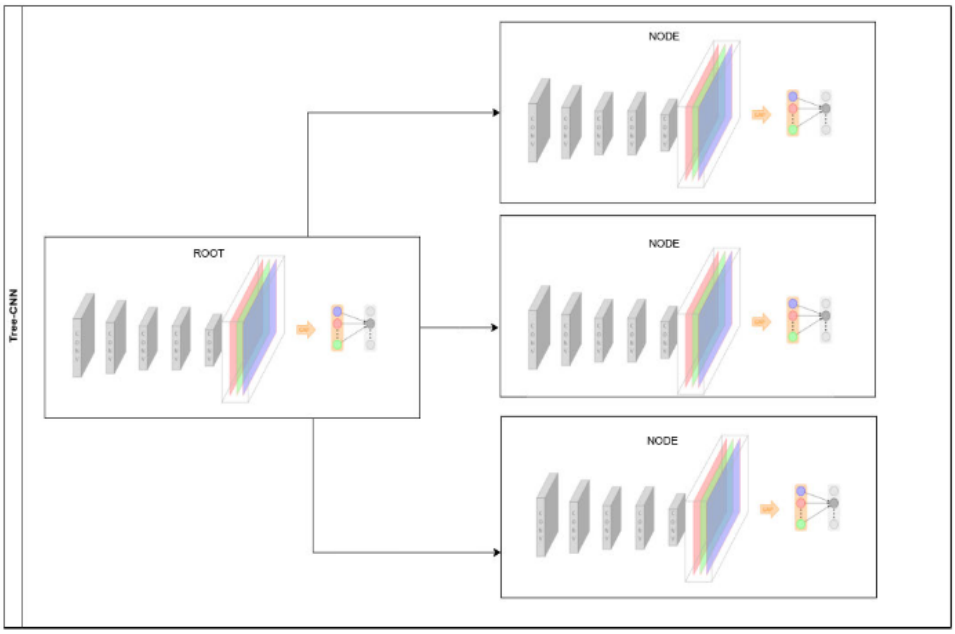

Figure 2.5: Simplified structure of Model

The model follows a tree structure, with a root node that branches out into multiple child nodes. Each node represents an individual Deep Convolutional network, and the output of each network is divided and used as input for its corresponding child nodes. This hierarchical architecture enables the model to capture complex patterns and features at different levels, leading to effective intrusion detection in the server network.

2.3.2 Advantages

- Fast processing time: The technique offers significant improvements in execution speed, enabling efficient real-time intrusion detection in server networks.
- High accuracy: The model demonstrates a remarkable accuracy of 98%, effectively distinguishing between intrusions and non-intrusions across various types.
- Continuous improvement: Being a deep learning technique, the model has the capability to enhance its detection abilities over time. It can learn from existing intrusions while also adapting to new intrusion patterns.

2.3.3 Disadvantages

- Large training data requirement: The technique necessitates a substantial amount of training data to effectively learn and generalize patterns. Sufficient labeled data is needed for model training, which may require human effort in providing accurate intrusion labels.

- Lengthy training time: Due to its deep learning nature, the model may require a significant amount of time for training, especially when working with complex datasets.

- Limited capability against new malware: Since the model relies on labeled data, it may not be initially equipped to handle new or previously unseen malware. Human-provided labels are required for the model to learn and adapt to new intrusion patterns.

2.3.4 Experimental results

The trained model, using datasets such as DARPA98, KDD99, ISCX2012, ADFA13, and CICIDS2017, achieved stability during training. When tested on a medium-sized company's internal network and a University hosted Database, the model exhibited an accuracy of 98%. The 2% failure rate primarily consisted of false positives rather than misclassifying true negatives.

Chapter 3

Literature Survey

3.1 Research Papers

We have conducted an extensive study on various research papers that explore the different types of attacks that can occur in a network. Through this, we have gained insight into the classification algorithms used in Intrusion Detection Systems (IDS) and their corresponding accuracy rates. In [2], the authors worked on wireless NIDS and utilized various classification algorithms to detect network intrusions. They found that the Random Forest algorithm, with 32 features, achieved the highest accuracy rate of 99.64%. On the other hand, in [1], the authors worked on the UNSW-NB 15 dataset and employed different classification algorithms such as SVM, NB, DT, and RF. They reported that the Random Forest algorithm attained the highest accuracy rate of 97.49%.

Additionally, in [3], the authors used a combination of regression trees and random forest on the UNSW-NB 15 dataset, which resulted in an accuracy

rate of 87.76%. In [10] and [4], the authors used Naive Bayes and Support Vector Machine algorithms on the NSL KDD and CICIDS 2017 datasets and achieved accuracy rates of 93.36% and 92.56%, respectively.

We reviewed various research papers that utilized different classification algorithms on different datasets for intrusion detection. Through this review, we gained knowledge about different types of attacks that can occur in a network, various models that can be used for intrusion detection, and the importance of feature selection, feature extraction, and feature engineering.

To further advance our project, we added noise to the NSL-KDD dataset and evaluated the accuracy of different models on the noisy data. This step allowed us to observe the robustness of the models against noise, which is a common occurrence in real-world network traffic. The noisy dataset can be generated by introducing random or systematic errors in the data, such as altering numerical values, flipping bits, or adding irrelevant features.

In conclusion, by applying different classification algorithms to various datasets and evaluating their accuracy on noisy data, we have gained valuable insights into the strengths and limitations of different models for intrusion detection. These findings can help in designing more robust and efficient intrusion detection systems.

3.2 Dataset Selection

We chose the NSL-KDD dataset for its popularity in the field of intrusion detection and its preprocessed state. The data has undergone normalization, which is crucial in improving the accuracy of intrusion detection models. The

dataset comprises of approximately 125,000 network connection records with 41 features. It covers four different types of attacks: Denial of Service (DoS), User to Root (U2R), Remote to Local (R2L), and Probing attacks, making it a comparatively balanced dataset. The dataset also includes the data of commonly used protocols such as Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and Internet Control Message Protocol (ICMP).

In summary, the NSL-KDD dataset is an enhanced version of the KDD Cup '99 dataset and is ideal for training and evaluating network intrusion detection systems.

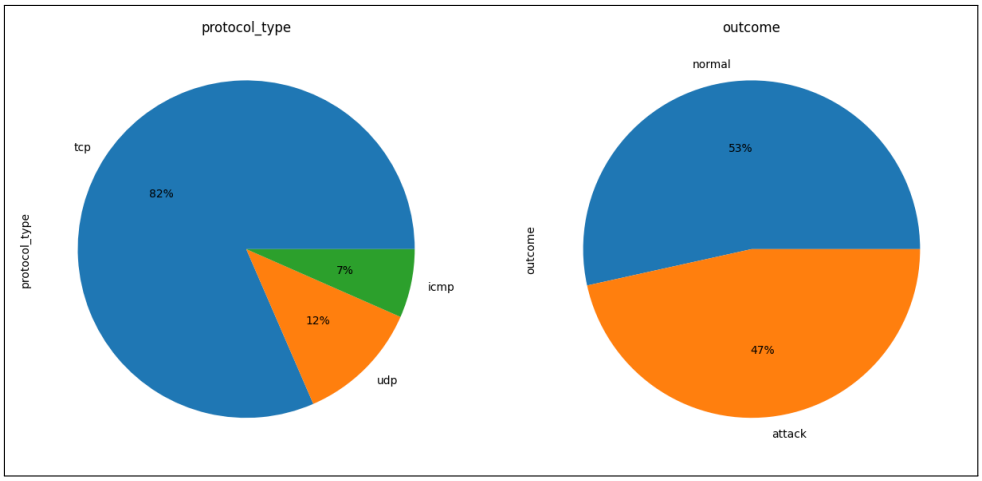

Figure 3.1: NSL KDD Dataset

Chapter 4

Model Comparison

4.1 Logistic Regression

Logistic regression is a statistical technique used to analyse the relationship between a dependent variable and one or more independent variables.The goal of logistic regression is to model the probability of the dependent variable taking on a particular value, given the values of the independent variables.We have used this model as a baseline model. Our goal is trying to build a better model compared to this.Keeping this model we observe the change in performance of models due to noise. The output of a logistic regression model is a probability value between 0 and 1, which can be interpreted as the predicted probability of the dependent variable being equal to 1, given the values of the independent variables. The model achieves this by fitting a logistic function (also known as a sigmoid function).

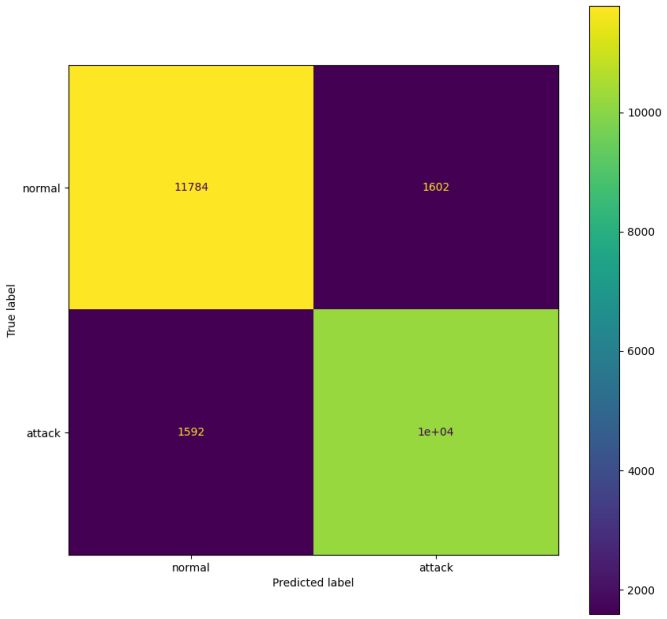

Figure 4.1: Confusion matrix of Logistic Regression

4.2 Naive Bayes

Gaussian Naive Bayes is a variant of the Naive Bayes algorithm, which is a probabilistic classification algorithm based on Bayes' theorem. It is called "naive" because it assumes that the features are independent of each other, which is often not the case in practice.In Gaussian Naive Bayes, the assumption is made that the continuous numerical features in the data follow a Gaussian (or normal) distribution. This means that the probability density function of each feature is a bell-shaped curve that can be characterised by its mean and standard deviation.According to [3] the Gaussian Naive Bayes algorithm is fast and efficient, making it a popular choice for many classification problems, especially those with large datasets. However, it

may not perform well if the assumption of normal distribution is not met, or if there are strong dependencies between the features.According to [5] despite these limitations, Gaussian Naive Bayes is widely used in real-world applications such as spam filtering, text classification, and medical diagnosis. It is also often used as a baseline algorithm for comparison with more complex models.

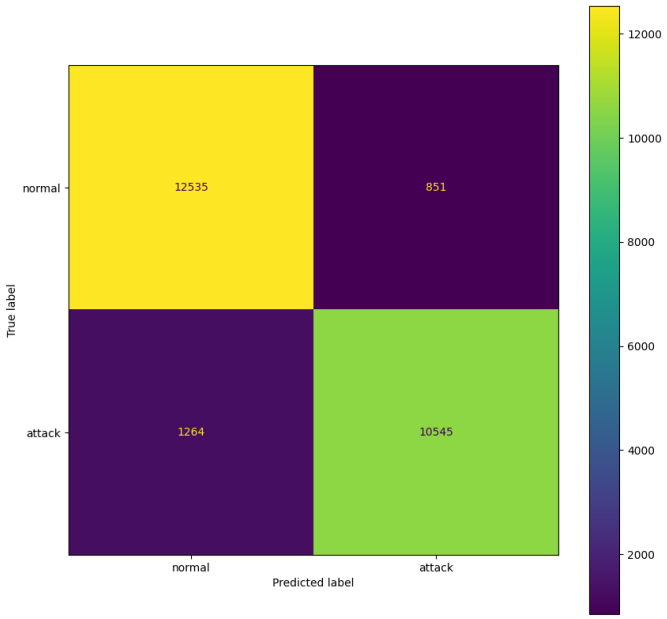

Figure 4.2: Confusion matrix of Naive Bayes

4.3 Support vector machine

SVM stands for Support Vector Machine, which is a powerful machine learning algorithm used for classification and regression analysis. In SVM, the

main goal is to find a hyperplane that separates the different classes of data points as best as possible. A hyperplane is a linear decision boundary that divides the feature space into two regions, where each region corresponds to a different class. Inspired from [1] we came to now that SVMs are known for their ability to handle high-dimensional data, which makes them well-suited for image classification, text classification, and other applications with large numbers of features. They are also widely used in many other fields such as finance, bioinformatics, and engineering. Even [2] and [3] proposed that if the data is not linearly separable, the SVM algorithm can still find a hyperplane by allowing some data points to be misclassified. This is done by introducing a slack variable that relaxes the constraint of perfect separation.

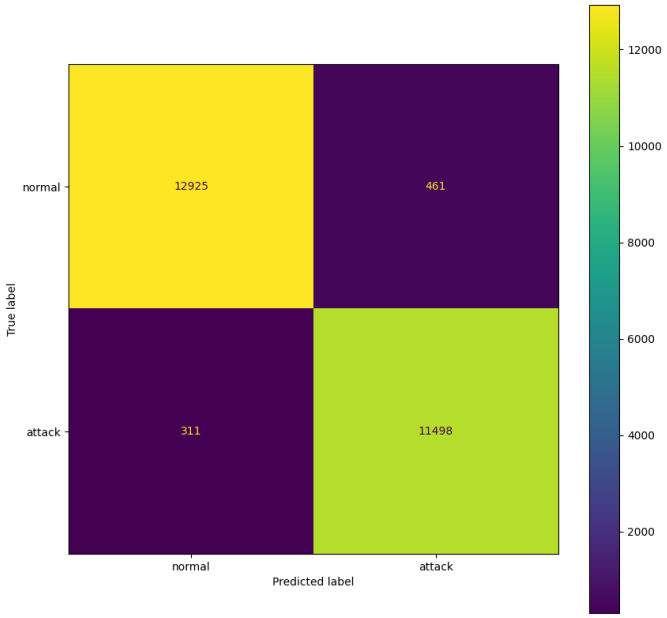

Figure 4.3: Confusion matrix of Support Vector Machine

4.4 Decision Tree

A decision tree is a supervised machine learning algorithm used for classification and regression analysis. It works by recursively partitioning the input data into smaller subsets based on the values of different features, in a hierarchical structure resembling a tree. [10] used it because the goal of the decision tree algorithm is to create a tree that can accurately predict the target variable for new, unseen data. This is done by selecting the best features and splitting criteria at each node, in order to maximize the information gain or minimize the impurity of the resulting subsets.Decision trees have several advantages, such as their interpretability, their ability to handle both categorical and numerical data, and their ability to handle missing values. They are also relatively fast and scalable, and can be used for both classification and regression tasks. Going through [3] we observe that decision trees can suffer from overfitting and may not always generalize well to unseen data, which can be mitigated by using pruning or ensemble methods. There are different types of decision trees, including binary trees, where each internal node has two branches, and multiway trees, where each internal node has more than two branches. There are also different algorithms for building decision trees, such as ID3, C4.5, CART.

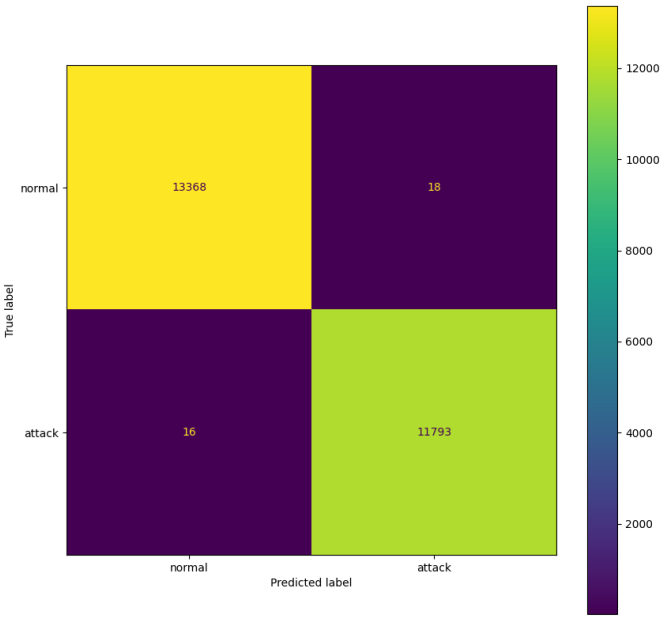

Figure 4.4: Confusion matrix of Decision Tree

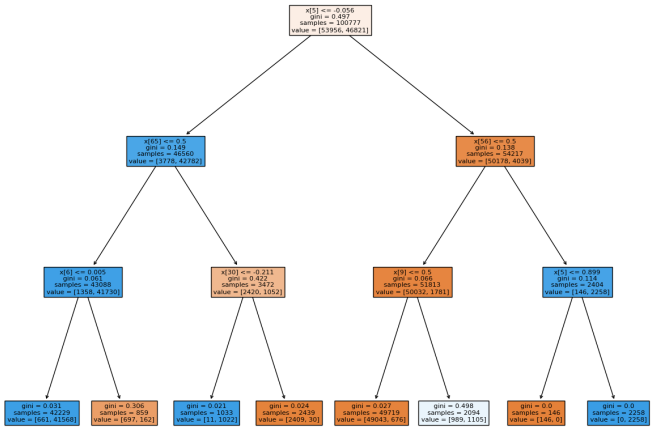

Figure 4.5: Decision Tree

4.5 Random Forest

Random Forest is a type of supervised learning algorithm used in machine learning for classification and regression tasks. The name "Random Forest" comes from the idea of creating a forest of decision trees, where each tree is trained on a random subset of the data and features. The algorithm is easy to use, has a low risk of overfitting, and is highly scalable. It is widely used in real-world applications such as credit scoring, medical diagnosis, and spam filtering.According to [6] in classification tasks, Random Forest uses the majority vote of the trees to predict the class of a new data point. In regression tasks, it uses the average prediction of the trees to predict the value of a new data point.

Through [6] we concluded random Forest can handle a wide variety of data types, including both categorical and numerical data, and can also handle missing data. It is also robust to noise and outliers in the data.

Overall, Random Forest is a powerful and versatile algorithm that can be used for a wide range of supervised learning tasks, making it a popular choice among machine learning practitioners.

Random forests have high accuracy, are robust to outliers, and provide feature importance measures to help understand the importance of each feature in making predictions.

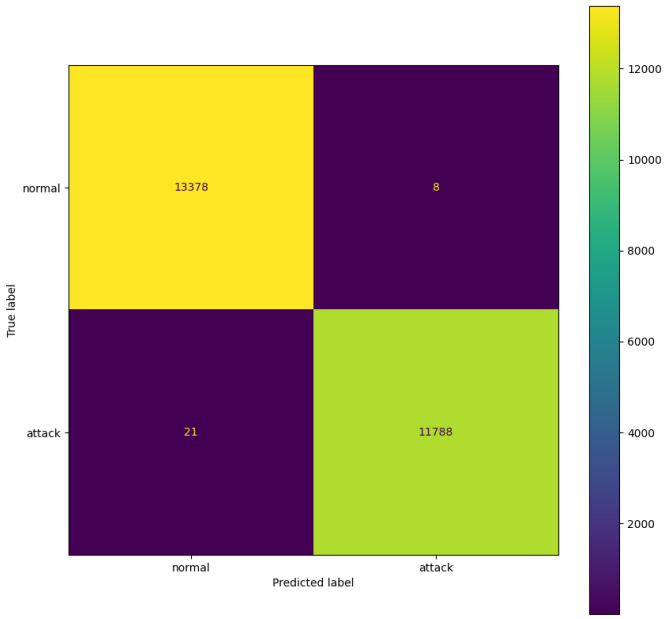

Figure 4.6: Confusion matrix of Random Forest

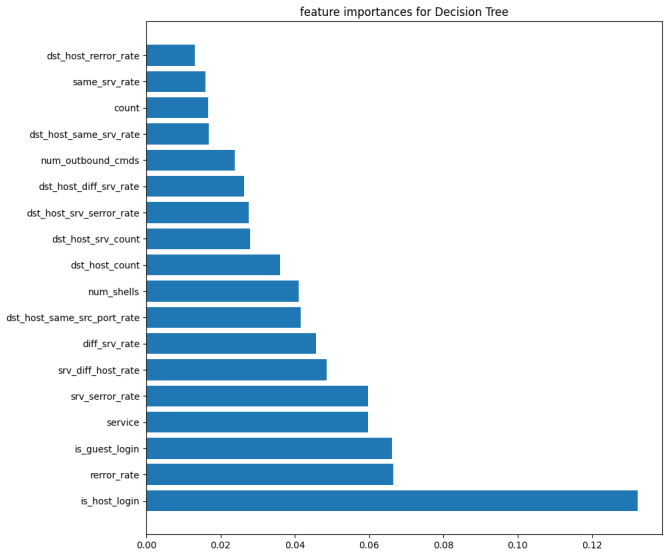

Figure 4.7: Feature Importance of Random Forest

4.6 Artificial Neural Network

Neural networks can be used in Intrusion Detection Systems (IDS) to classify network traffic as normal or malicious. IDS are systems that monitor network traffic for signs of intrusions, such as unauthorised access attempts or malicious activity. Neural networks can be trained on large amounts of network traffic data to learn patterns of normal behaviour and detect anomalies that may indicate an intrusion. The network can be designed to take in various types of input features, such as packet size, protocol, and source/destination IP address, and output a prediction of whether the traffic is normal or malicious. According to [7] one common approach is to use a feedforward neural network with one or more hidden layers. The input layer would receive the network traffic data, and the hidden layers would perform non-linear transformations on the data to extract relevant features. The output layer would then produce a binary classification of normal or malicious traffic.Neural networks can also be used in combination with other machine learning techniques, such as feature selection and dimensionality reduction, to improve their performance on IDS tasks from [11].

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_10 (Dense)            (None, 64)                7872

 dropout_8 (Dropout)         (None, 64)                0

 dense_11 (Dense)            (None, 128)               8320

 dropout_9 (Dropout)         (None, 128)               0

 dense_12 (Dense)            (None, 512)               66048

 dropout_10 (Dropout)        (None, 512)               0

 dense_13 (Dense)            (None, 128)               65664

 dropout_11 (Dropout)        (None, 128)               0

 dense_14 (Dense)            (None, 1)                 129

=================================================================
Total params: 148,033
Trainable params: 148,033
Non-trainable params: 0
```

Figure 4.8: Artificial Neural Network Model

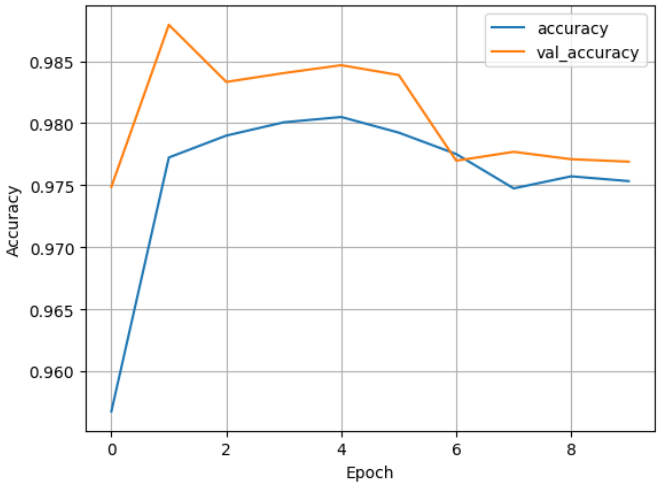

Figure 4.9: Accuracy v/s Epochs6

Chapter 5

Model Tolerance to Noise Level-1 and Level-2

5.1 Logistic Regression

Logistic regression can be sensitive to noisy data, meaning that the presence of outliers or irrelevant features can significantly affect its performance. In the presence of noisy data, logistic regression may produce suboptimal models that have poor generalization performance and high variance. To improve the performance of logistic regression, several approaches can be used. One common approach is to preprocess the data by removing outliers or irrelevant features before training the model. This can be done using techniques such as z-score normalization, which scales the data so that each feature has zero mean and unit variance, or feature selection, which selects only the most relevant features for the task at hand. We can see how logistic regression struggles for noise level-2 and even for noise level-1. Since we used

it as a base line model we try to build a better model from this.

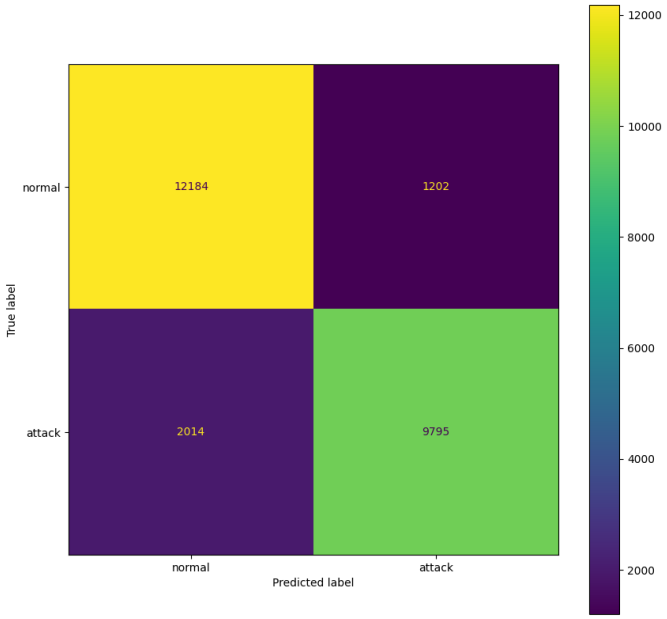

Figure 5.1: Confusion matrix of Logistic Regression in Noise Level-1

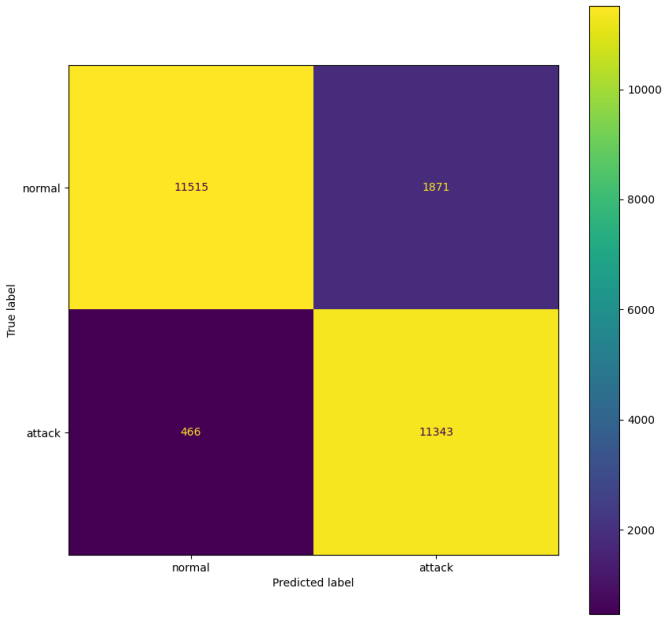

Figure 5.2: Confusion matrix of Logistic Regression in Noise Level-2

5.2 Naive Bayes

When dealing with noisy data, Gaussian Naive Bayes can still be used, but it is important to preprocess the data appropriately. One common approach is to apply some form of noise reduction or feature selection to reduce the impact of the noise.

In addition, it is important to tune the parameters of the algorithm appropriately. One such parameter is the smoothing factor, which can be adjusted to trade off between overfitting and underfitting. If the data is highly noisy, a higher smoothing factor may be needed to reduce the impact of the noise. Overall, Gaussian Naive Bayes can still be effective for classification tasks even when dealing with noisy data, as long as appropriate preprocessing

and parameter tuning is performed. However, it is always recommended to compare the performance of different algorithms on the specific dataset to determine which one is most suitable for the task. We can observe the decline in performance without tuning which is not as we intended.

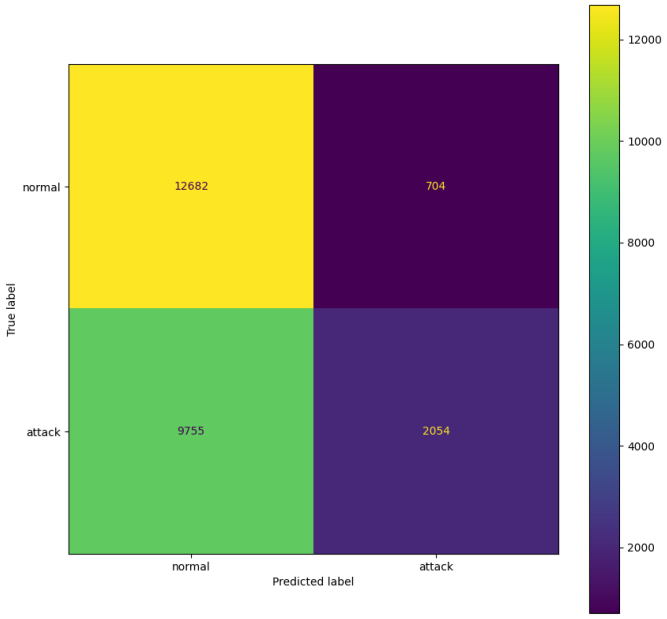

Figure 5.3: Confusion matrix of Naive Bayes in Noise Level-1

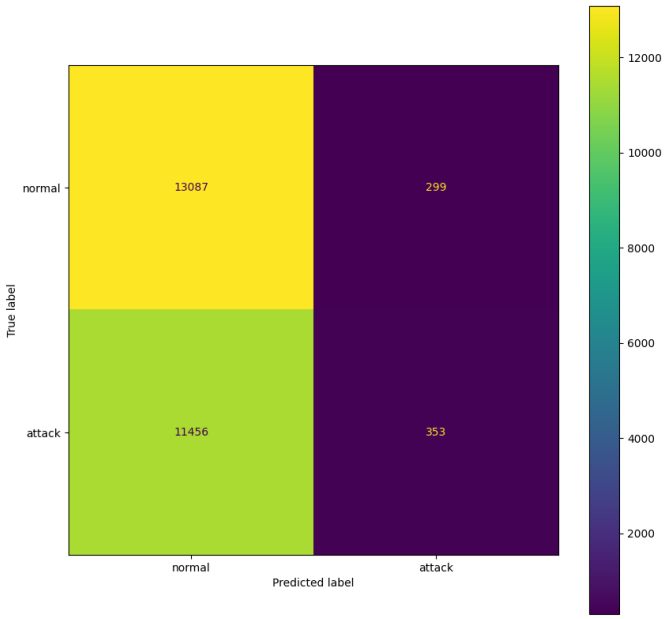

Figure 5.4: Confusion matrix of Naive Bayes in Noise Level-2

5.3 Support vector machine

One of the key preprocessing steps when dealing with noisy data is to apply feature selection or feature engineering techniques to reduce the impact of the noise on the classification task. This can involve removing irrelevant features, transforming the data into a more suitable form, or extracting new features from the existing ones.

In addition, SVMs require tuning of the hyperparameters to achieve good performance on noisy data. The regularization parameter, C, can be adjusted to trade off between overfitting and underfitting. A smaller value of C can help to reduce the impact of noisy data, while a larger value of C may be needed to handle more complex classification tasks.

Another important parameter in SVMs is the choice of kernel function. The choice of kernel can affect the performance of the algorithm on noisy data. For example, the linear kernel may be more robust to noise than other kernel functions such as the radial basis function (RBF) kernel.

Finally, it is recommended to evaluate the performance of SVMs on the specific dataset to determine whether it is suitable for handling noisy data. In some cases, other algorithms such as Random Forest or Gradient Boosting may provide better performance on noisy data than SVMs. We can see that without tuning SVM's can't even match the performance of baseline models mentioned above.

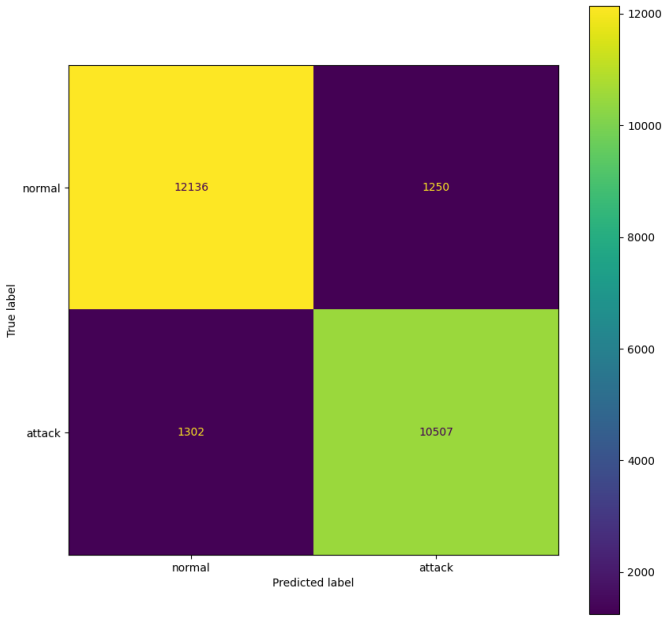

Figure 5.5: Confusion matrix of Support Vector Machine in Noise Level-1

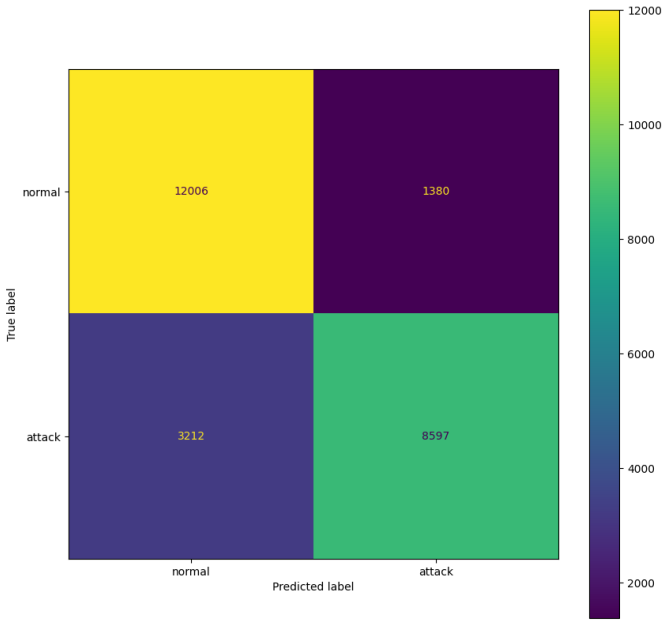

Figure 5.6: Confusion matrix of Support Vector Machine in Noise Level-2

5.4 Decision Tree

Decision trees can be used for noisy data by first handling the noise through appropriate preprocessing techniques, such as feature selection, normalization, and imputation. Feature selection helps in identifying the most informative and relevant features while removing irrelevant and redundant ones, thus reducing noise. Normalization can help reduce the impact of outliers on the decision tree model. Imputation can be used to handle missing data. Pruning techniques, such as reduced error pruning, can be used to prevent overfitting to the noise in the data. Ensemble methods, such as random forests and boosting, can also improve performance by aggregating the outputs of multiple decision trees. Regularization techniques, such as L1 and L2 regularization, can reduce the impact of noisy data by adding a penalty term to the objective function of the decision tree. It is important to evaluate the performance of the decision tree on a validation set to avoid overfitting to the noise in the data. Decision trees can handle noisy data by combining appropriate preprocessing techniques, pruning, regularization, and ensemble methods. We can see it is robust for noise level-1 but struggles with noise level-2 although using all the possible methods to extract the best of the decision tree.

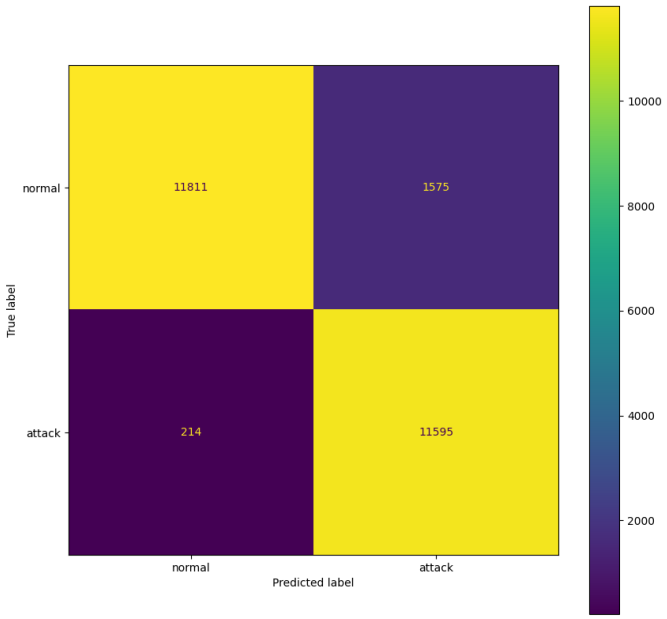

Figure 5.7: Confusion matrix of Decision Tree in Noise Level-1

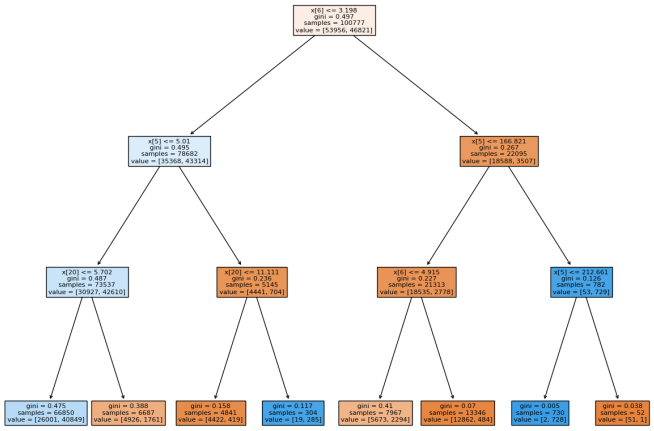

Figure 5.8: Decision Tree of Noise Level-1

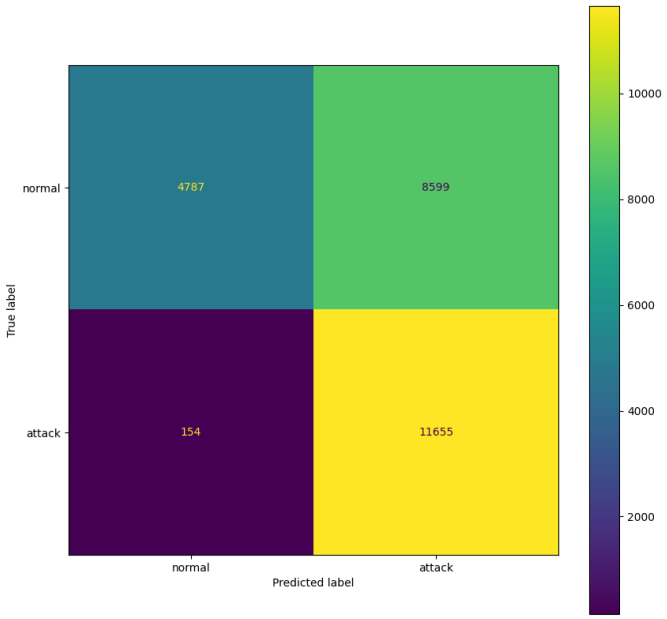

Figure 5.9: Confusion matrix of Decision Tree in Noise Level-2

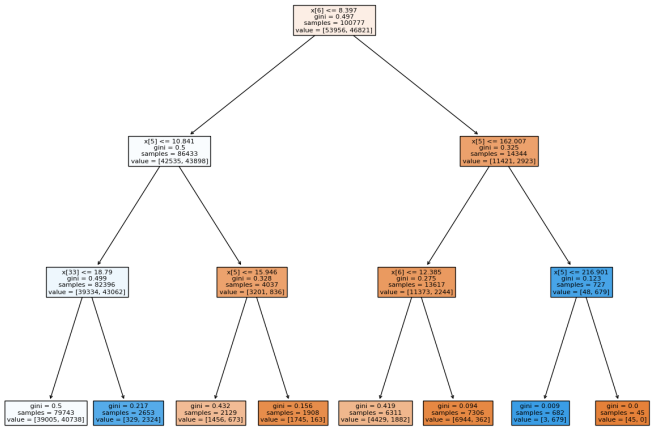

Figure 5.10: Decision Tree of Noise Level-2

5.5 Random Forest

Random Forest is a powerful ensemble learning algorithm that combines multiple decision trees to generate a more robust model. It can be effective for classification and regression tasks, even when the data is noisy. Random Forest addresses the issue of overfitting in decision trees by creating multiple trees using bootstrapping and feature sampling. This creates a set of diverse trees that are combined to form a final model. In the presence of noisy data, Random Forest is still effective because it can handle missing values, irrelevant features, and noisy features. Additionally, the algorithm is not sensitive to the order of the data, making it more robust to noise. Random Forest can be a reliable and effective approach for noisy data classification tasks. We find that it outperforms all models even for noisy data and relatively fast . Through confusion matrix we can observe that it is quite robust to both noise level-1 and noise level-2 further encouraging us to use this model in our proposed model.

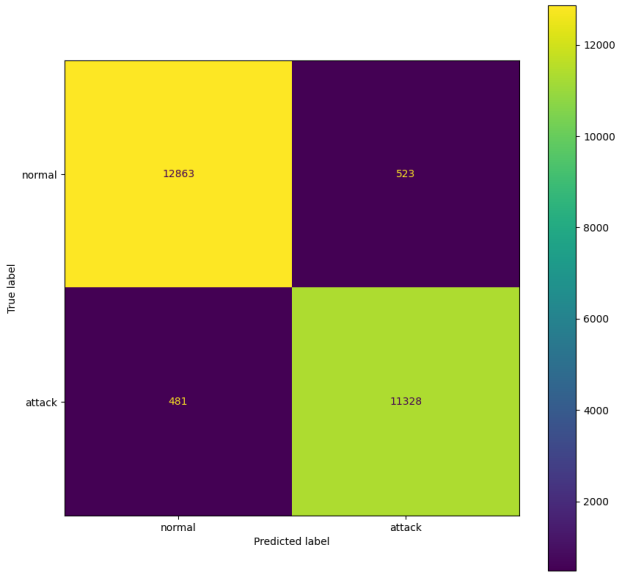

Figure 5.11: Confusion matrix of Random Forest in Noise Level-1

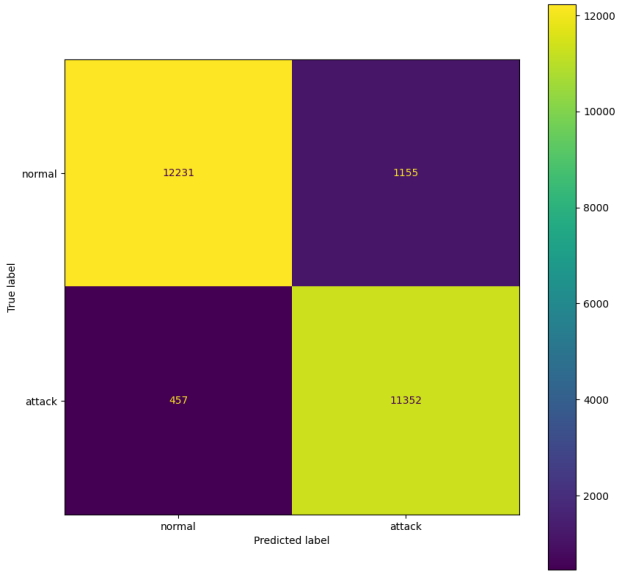

Figure 5.12: Confusion matrix of Random Forest in Noise Level-2

5.6 Artificial Neural Network

Training a neural network on noisy data can be challenging because noise can interfere with the learning process and cause overfitting. One approach to overcome this challenge is to limit the number of epochs during training. By doing this, the network can learn the important features of the data without memorizing the noise. Training a neural network for only 10 epochs on noisy data can help reduce overfitting and improve generalization performance. Additionally, using techniques such as regularization and dropout can further improve the network's robustness to noise. However, training a network for a small number of epochs can also result in underfitting, where the model does not capture all the relevant information in the data. We can see that training for only 10 epochs of Neural Network for noise level-1 and noise level-2 the performance has been reduced by almost 20% each time. However increasing the number of epochs can make the model unbeatable but in the case where a new type of attack has been found we use ml models to train fast and deploy so that most of the attacks are stopped and in the backend we train neural network and deploy for final project.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_10 (Dense)            (None, 64)                7872

 dropout_8 (Dropout)         (None, 64)                0

 dense_11 (Dense)            (None, 128)               8320

 dropout_9 (Dropout)         (None, 128)               0

 dense_12 (Dense)            (None, 512)               66048

 dropout_10 (Dropout)        (None, 512)               0

 dense_13 (Dense)            (None, 128)               65664

 dropout_11 (Dropout)        (None, 128)               0

 dense_14 (Dense)            (None, 1)                 129

=================================================================
Total params: 148,033
Trainable params: 148,033
Non-trainable params: 0
```

Figure 5.13: Artificial Neural Network Model in Noise Level-1

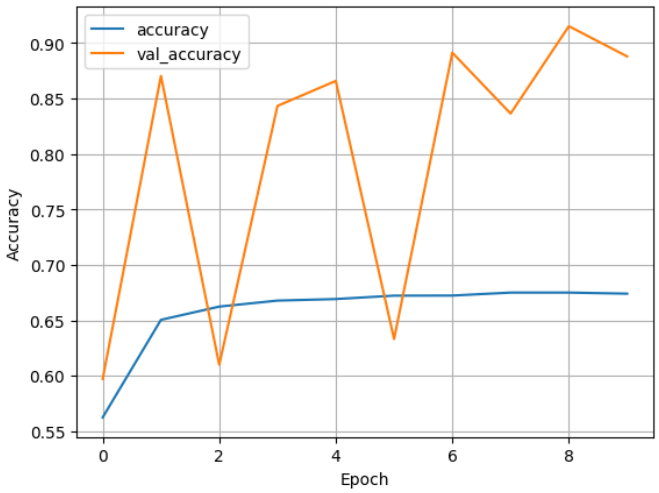

Figure 5.14: Accuracy v/s Epochs in Noise Level-1

Chapter 6

Proposed Model

6.0.1 Random Forest using PCA

Principal Component Analysis (PCA) is a statistical technique used to reduce the dimensionality of large datasets by identifying the underlying structure or patterns in the data. It involves transforming the original variables of a dataset into a new set of uncorrelated variables known as principal components. These principal components are ordered in decreasing variance and can be used to summarize the dataset, with the first few components typically containing most of the information in the original data. PCA is often used for data compression, feature extraction, and visualization, and it can also help to remove noise and redundancy from the data. Using PCA we reduced the number of features from 120 to 20 in our original dataset.
Combining Random Forest with PCA (Principal Component Analysis) can be an effective technique for reducing the dimensionality of the dataset and improving the performance of the classifier. In the case of Random Forest,

reducing the dimensionality of the dataset can help improve the performance of the classifier by reducing the number of irrelevant features and minimizing the effects of noise and overfitting. By using PCA to reduce the dimensionality of the dataset, we can obtain a smaller set of features that are more relevant for the classification task.

6.0.2 Performance in Noiseless Data

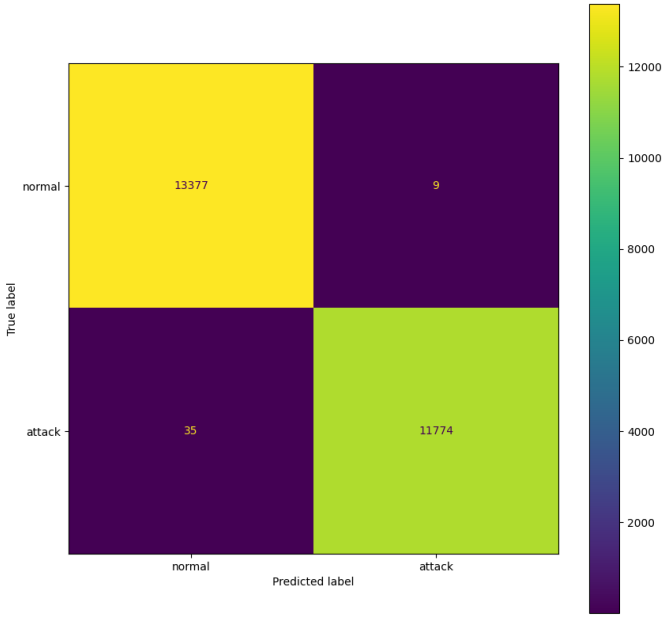

Figure 6.1: Confusion matrix after applying PCA

The combination of Random Forest and PCA involves first applying PCA to the dataset to obtain a reduced set of principal components, and then using

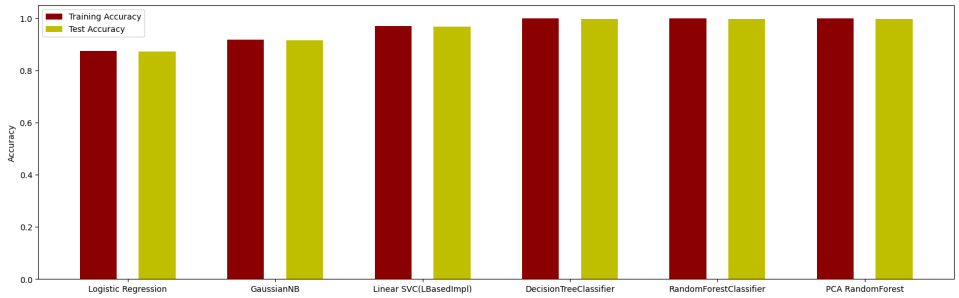

Figure 6.2: Training and Test Accuracy

Random Forest to classify the data based on the reduced set of features. This approach can be particularly useful for datasets with a large number of features, where overfitting and computational complexity can be a problem. However, it is important to note that the choice of the number of principal components to retain can affect the performance of the classifier, and it may require some trial and error to determine the optimal number. We observe significant changes in training time and performance for both noisy and noiseless dataset. It even outperforms neural networks and random forest for noisy level-1 and noisy level-2. For noisy data PCA is required because we remove unwanted features with noise which would have caused the model to perform worse. The idea of using PCA was taken from [11] which gave information about various cons of using PCA in IDS for noisy data.

6.0.3 Performance in Noisy Data

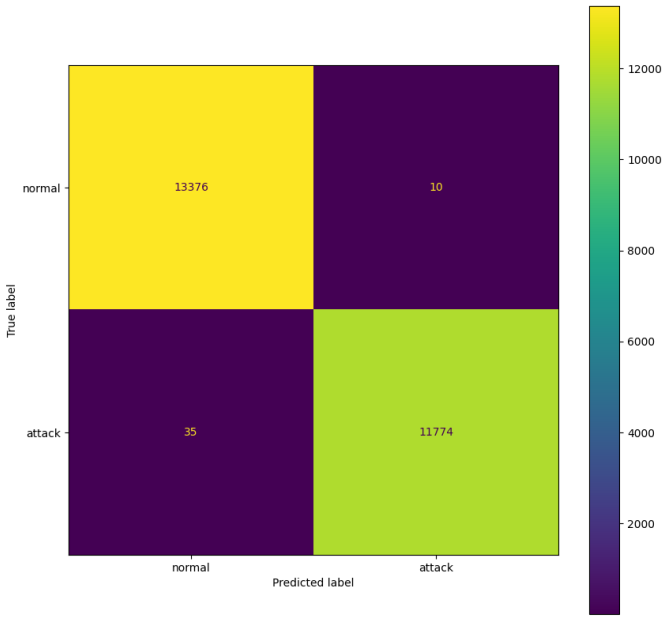

Figure 6.3: Confusion matrix after applying PCA Noise Level-1

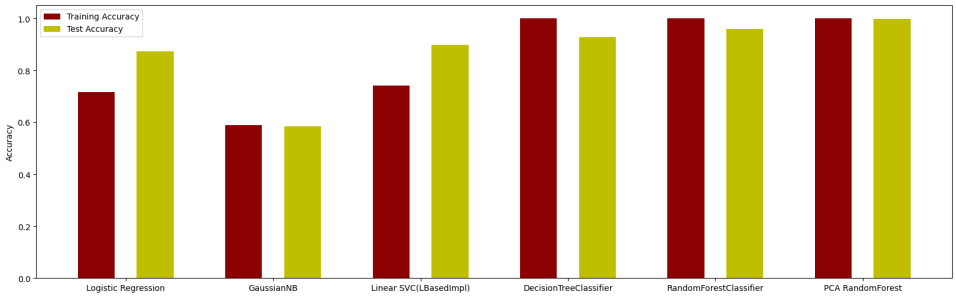

Figure 6.4: Training and Test Accuracy

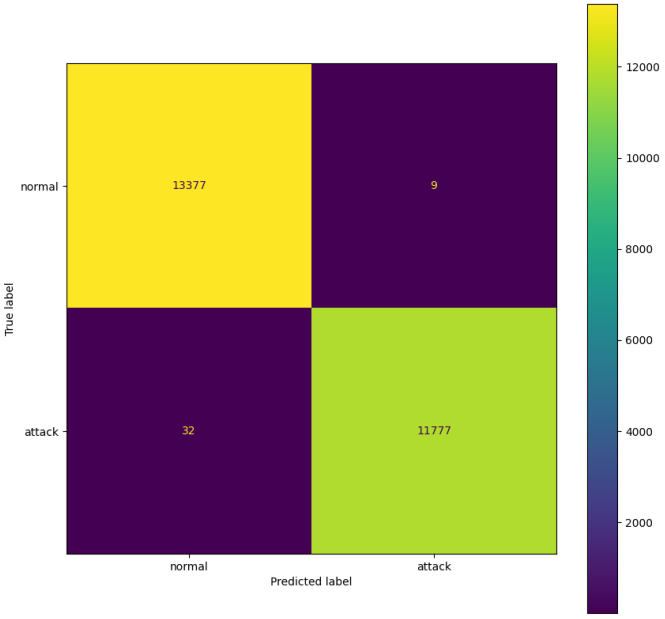

Figure 6.5: Confusion matrix after applying PCA Noise Level-2

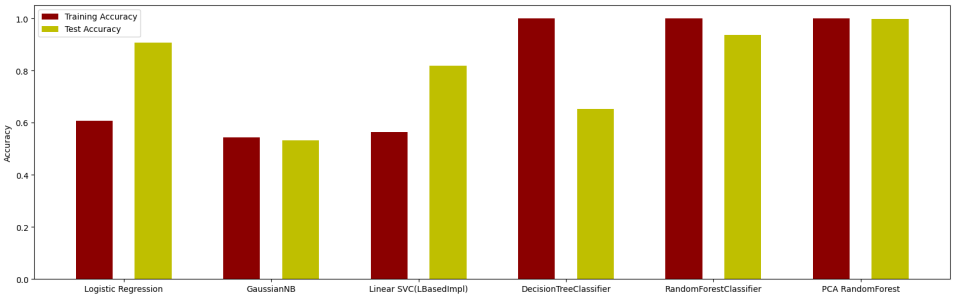

Figure 6.6: Training and Test Accuracy

Chapter 7

Conclusion

In conclusion, we have gained insights into the performance of various machine learning models and their robustness to artifacts and noisy data. Logistic Regression served as the baseline model for our project. We observed that SVM was sensitive to noise but performed well for noiseless data, which is consistent with the findings of [3]. We explored the use of Random Forest [5] and concluded through experimentation that using Random Forest with PCA produced the best results. One reason for this is that the noise was concentrated on only 20 features, which was resolved by applying PCA. While it is true that using Artificial Neural Networks (ANN) for a greater number of epochs could outperform other models, in practice, any new intrusion must be promptly trained and deployed in the environment. Therefore, we conclude that Random Forest with PCA is an effective way to detect anomalies in noisy data.

Bibliography

[1] Razan Abdulhammed et al. "Effective features selection and machine learning classifiers for improved wireless intrusion detection". In: *2018 International symposium on networks, computers and communications (ISNCC)*. IEEE. 2018, pp. 1–6.

[2] Mustapha Belouch, Salah El Hadaj, and Mohamed Idhammad. "Performance evaluation of intrusion detection based on machine learning using Apache Spark". In: *Procedia Computer Science* 127 (2018), pp. 1–6.

[3] Karuna S Bhosale, Maria Nenova, and Georgi Iliev. "Data Mining Based Advanced Algorithm for Intrusion Detections in Communication Networks". In: *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*. IEEE. 2018, pp. 297–300.

[4] Zina Chkirbene et al. "Hybrid machine learning for network anomaly intrusion detection". In: *2020 IEEE international conference on informatics, IoT, and enabling technologies (ICIoT)*. IEEE. 2020, pp. 163–170.

[5] Kishor Kumar Gulla et al. “Machine learning based intrusion detection techniques”. In: *Handbook of Computer Networks and Cyber Security: Principles and Paradigms* (2020), pp. 873–888.

[6] Alif Nur Iman and Tohari Ahmad. “Improving intrusion detection system by estimating parameters of random forest in Boruta”. In: *2020 International Conference on Smart Technology and Applications (ICoSTA)*. IEEE. 2020, pp. 1–6.

[7] Farrukh Aslam Khan et al. “A novel two-stage deep learning model for efficient network intrusion detection”. In: *IEEE Access* 7 (2019), pp. 30373–30385.

[8] Robson V Mendonça et al. “Intrusion detection system based on fast hierarchical deep convolutional neural network”. In: *IEEE Access* 9 (2021), pp. 61024–61034.

[9] Lotfi ben Othmane et al. “On the performance of detecting injection of fabricated messages into the can bus”. In: *IEEE Transactions on Dependable and Secure Computing* 19.1 (2020), pp. 468–481.

[10] Kazi Abu Taher, Billal Mohammed Yasin Jisan, and Md Mahbubur Rahman. “Network intrusion detection using supervised machine learning technique with feature selection”. In: *2019 International conference on robotics, electrical and signal processing techniques (ICREST)*. IEEE. 2019, pp. 643–646.

[11] Congyuan Xu et al. “An intrusion detection system using a deep neural network with gated recurrent units”. In: *IEEE Access* 6 (2018), pp. 48697–48707.