

# **A Performance Comparison Between Machine Learning Models on Zero-Day Attack Detection**

Ahmed Musa Awon

17101201

Afid Odree

17101183

Samia Islam

17101002

Afia Yeasmin

17101182

Bivasha Bashir Biva

17101174

A thesis submitted to the Department of Computer Science and Engineering  
in partial fulfilment of the requirements for the degree of  
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering  
BRAC University  
January 2021

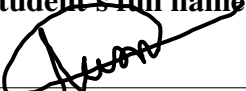
©2021. BRAC University  
All rights reserved.

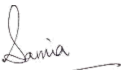
# Declaration


It is hereby declared that


1. The thesis submitted is our own original work while completing degree at BRAC University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.


Student's full name and signature:

  
\_\_\_\_\_  
**Ahmed Musa Awon**  
**17101201**

  
\_\_\_\_\_  
**Samia Islam**  
**17101002**

  
\_\_\_\_\_  
**Afid Odree**  
**17101183**

  
\_\_\_\_\_  
**Afia Yeasmin**  
**17101182**

  
\_\_\_\_\_  
**Bivasha Bashir Biva**  
**17101174**

# Approval

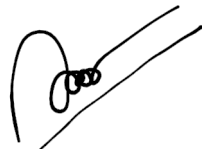
The thesis “A Performance Comparison Between Machine Learning Models On Zero Day Attack Detection” is submitted by:

1. Ahmed Musa Awon (17101201)
2. Afid Odree (17101183)
3. Samia Islam (17101002)
4. Afia Yeasmin (17101182)
5. Bivasha Bashir Biva (17101174)

Of Fall, 2020 has acquired a satisfactory remark in partial fulfilment for the degree of B.Sc. in Computer Science Engineering on January 11th, 2021.

## Examining Committee:

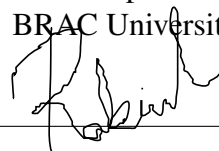
Supervisor:  
(Member)



---

Dr. Muhammad Iqbal Hossain  
Associate Professor  
CSE Department  
BRAC University

Program Coordinator:  
(Member)



---

Dr. Md. Golam Rabiul Alam  
Associate Professor  
CSE Department  
BRAC University

Head of Department:  
(Dean)

---

Mahbubul Alam Majumdar, PhD  
Professor and Dean  
School of Data and Sciences  
BRAC University

# **Ethics Statement**

This is to declare that the following research works have been conducted within all ethical guidelines. All sorts of bias have been avoided involving experimental design, data analysis, interpretation, peer- review process and other facets of research. Most importantly authors pledged to avoid plagiarism at all costs.

# Abstract

Traditional IDS has been shielding against cyber threats for many years but it falls short on detecting zero-day attacks. These are the attacks that are unique with unknown attack patterns and mutating attack signatures making them difficult to detect. Machine learning approaches have been extensively used in Intrusion Detection Systems (IDS) to detect both known and unknown attacks. However, the widespread and rapid growth of zero-day attack forces researchers to continuously seek to increase the performances of models to better detect these attacks. In this paper, we used supervised machine learning approaches to detect zero-day attacks. The dataset used for demonstration and evaluation was the latest CSE-CIC-IDS2018 dataset with 80 features and 14 different types of attacks. All the attacks' labels were represented as a single label called 'Attack'. The main aim behind this proposal was to compare between the performances of the mainstream Machine Learning models in detecting Zero Day attacks. The proposed model of Artificial Neural Network (ANN), Random Forest (RF) and K-Nearest Neighbor (KNN) all achieved high accuracies with optimal parameter settings. With RF having an accuracy of 98.90% , ANN with 98.3% and KNN with an accuracy of 98.53%.A better estimate of the performance of the models can be seen by the false-negative rates of each model.

**Keywords:** *IDS; Zero-Day Attacks; Supervised Machine Learning; Artificial Neural Network; Random Forest; K-Nearest Neighbour; False Negative Rate*

# Acknowledgement

All praise be to The Almighty God, the Most Merciful. We are grateful to all our faculty members who have guided us incessantly throughout our undergraduate program. We would especially like to thank our supervisor for providing us with helpful feedback on our work and ensuring the best output from us. Last but not the least, our parents and peers for always believing in us and supporting us.

# Table of Contents

<b>Declaration</b>	<b>i</b>
<b>Declaration</b>	<b>ii</b>
<b>Ethics Statement</b>	<b>ii</b>
<b>Abstract</b>	<b>iv</b>
<b>Acknowledgement</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Problem Statement . . . . .	2
1.3 Objective and Contribution . . . . .	2
1.4 Thesis Structure . . . . .	3
<b>2 Background</b>	<b>4</b>
2.1 Literature Review . . . . .	4
2.2 Algorithms/Models . . . . .	6
2.2.1 Artificial Neural Networks . . . . .	6
2.2.2 K-Nearest Neighbour . . . . .	7
2.2.3 Random Forest . . . . .	8
<b>3 Proposed Model</b>	<b>9</b>
3.1 Dataset Description . . . . .	9
3.1.1 Data Pre-processing . . . . .	9
3.2 Model Description . . . . .	10
3.2.1 Preparing ANN Model . . . . .	10
3.2.2 Preparing KNN Model . . . . .	11
3.2.3 Preparing Random Forest Model . . . . .	11
<b>4 Experimentation</b>	<b>13</b>
<b>5 Result Analysis</b>	<b>17</b>
<b>References</b>	<b>22</b>

# List of Tables

4.1	Best four parameters found for ANN from RandomSearch Algorithm . . . . .	13
4.2	5 Best selected models and the effect of n_estimator and max_depth on RF . . .	14
4.3	Effect of max feature on the best tuned model of RF . . . . .	15
4.4	3 Best selected models found for KNN from RandomSearch algorithm . . . . .	16
5.1	False Negative Rate of different ML Models . . . . .	19



# List of Figures

- 5.1 Performance analysis of different ML models . . . . . 18
- 5.2 Comparison among the false negative rates of different ML models . . . . . 20

# Chapter 1

## Introduction

### 1.1 Motivation

The use of computer networks has been accelerating and gaining importance in economic and social life like the internet. Companies or businesses have adopted network technologies for solving their purposes faster and in an efficient manner. With growing computer networks being prevalent throughout the world, caution arises on securing the networks from cyber threats. A rise in the computer networks also give rise to more unwanted cyber threats. With so many threats infiltrating the process the caution arises in increasing cyber security. Zero-day attack is considered as an ideal obstacle in the field of cyber security.

Zero-day attacks are vulnerabilities or weaknesses in software, hardware, microcode that are unknown to the party who are responsible for fixing the flaw. The straightforwardness of the name 'Zero-day' suggests its nature of giving no time between its discovery and first attack. Detecting such attacks are very difficult for its unknown attacking pattern or mutating attack signatures. The inability of detecting such vulnerabilities pose a serious threat to companies or organizations. These attacks have existed since many years and are rapidly growing in number. According to Ponemon Institute's 2018 State of Endpoint Security Risk report, 76% of all successful attacks on organizations were zero-day attacks [14]. In the third quarter of 2019 malware attacks significantly increased and zero day malwares accounted for 50% of all detected malwares [15]. These reports suggest how zero day attacks keep on growing and shows how widespread these attacks are. The unprecedented behavioral pattern of Zero-day attacks have gained interest of many cyber security specialists.

The substantial use of Machine Learning in the Intrusion Detection System(IDS) has been prevalent for quite some time in order to detect known, as well as, unknown attacks. Even so, the expeditious development of zero day attacks enables researchers to devise models that can detect zero day attacks better.

## 1.2 Problem Statement

The acceleration in use of computer networks has been proved efficient to solve our problem and other purposes. Cyber threats are equally growing with the increased dependability of computer networks. Caution arises in securing our networks from the undesirable network threats. While mitigating these constant threats we have to deal with the zero day attacks that are the most challenging to alleviate. For this reason, we used the promising algorithms of machine learning, namely Artificial Neural Networks; Random Forest; and K-Nearest Neighbours; that detects the zero day attack patterns and compare the results obtained from each of these to determine which works best. While most researchers focus on the accuracy percentage which is somewhat very similar in such cases, we have further analysed our data based on the false negative rates as well.

## 1.3 Objective and Contribution

The prime objectives of our research is as follows:

- Collect the dataset
- Comprehend the features and how they relate to the overall dataset
- Eliminate useless features to reduce dimensionality
- Clean the data by removing all NaN, infinity and nonnumerical values
- Run various machine learning algorithms on the dataset
- Finetune the algorithms for better accuracy
- Gain accuracy scores and confusion matrices
- Compare and use the best algorithm

We found the correlation between the attacks on which our model was trained and use it to help us detect any future attacks which are similar to the given attacks. We used Artificial Neural Network (ANN), Random Forest (RF) and K-Nearest Neighbour (KNN) algorithm to classify whether the connection is benign or an attack. Our models were under continuous scrutiny and evaluation to prevent over-fitting or under-fitting issues. Furthermore, the aforementioned algorithms were fine-tuned using appropriate parameters to achieve better accuracy. Our model keeps learning from different interactions it has made and is able to detect any attacks that it has never experienced before. Finally we compare the results obtained from each of the different algorithms we have implemented and see which gives the best outcome overall.

## 1.4 Thesis Structure

The main aim of this report is to compare and contrast the performance of ML based models on Zero Day attack detection. It uses different supervised machine learning algorithms to accomplish this goal and finally determine which algorithm is the most suitable and provides the finest accuracy. All in all, this report describes the procedure carried out by the authors to achieve their results.

Chapter 1, Introduction, enunciates what motivated the authors to research on the given problem statement. The objective of our research and the methodology is explored in brief. A summary of the paper is given in this chapter.

In Chapter 2, Background, the Literature Review gives an overview of the papers that have experimented on similar works regarding this area of concern. This chapter also includes the algorithms that have been used in our system.

The Proposed Model, Chapter 3, describes in detail how we chose the dataset, pre-processed it, selected the appropriate features. Furthermore, this chapter elaborately portrays the models or algorithms that have been used by the authors. In addition, Chapter 4, Experimentation, demonstrates and explains the meticulous procedures that have been executed for our models.

Finally, Results and Analysis is shown in Chapter 5. It includes the outcomes from the different models that have been applied and their comparison is given. The results are analysed critically and the confusion matrix, accuracy and false negative percentages are also attached.

# Chapter 2

## Background

### 2.1 Literature Review

With the intention of developing an effective and efficient Network Intrusion Detection system Niyaz,Q et al.(2016) have designed self-taught learning, a deep learning based approach on the NSL-KDD dataset. The authors in [1] have evaluated the performance of STL on 2-class and 5-class test data and have successfully shown the performance of their model is better compared to the softmax regression. The comparison metrics that they included are accuracy, precision, recall, and f-measure values. They achieved an accuracy rate of 88.39% for 2-class and 79.10% for 5-class classification. The authors are also intended to implement a real-time NIDS using deep learning techniques in future.

Hindy,H et al.(2020) have implemented an auto-encoder based implementation to detect zero-day attack. They planned to create an intelligent Intrusion Detection System with high recall value and acceptable minimum false-negatives rate. One class support vector machine was used to compare the performance of the model and the proposed model demonstrated outperformed accuracy. The encoding-decoding capability of the autoencoder was a plus point for the authors to build the system. They used two different datasets NSL-KDD and CICIDS2017 and got 89-99% accuracy for NSL-KDD and 75-98% for CICIDS2017 dataset. Their future purpose is to evaluate the model with datasets that cover specialized network IDS like IoT and Critical Infrastructure networks as discussed in [2].

To protect the ICS network an Intrusion detection system using deep learning is proposed by Hijazi, A et al.(2018). They have used multi-layer perceptrons with binary classification to build the model. The authors in [3] have trained high-dimensional Modbus packet data and labeled them as normal and attack to train the NN to detect anomaly and normal behaviour of the network. Their proposed model showed higher performance when compared with Self-taught learning and Soft-max regression. The proposed system got an accuracy of 99.9%. The authors mentioned that the proposed system can be enhanced by using time stamps and adding ability to identify Denial of Service attacks.

Malware identification and detection being the priority of the authors (Makandar et al.,2015),they

have proposed a model to detect the malwares based on texture features. They tried to determine malicious behavior of data using Gabor wavelet transform and GIST. The authors in [4] have utilized the feed forward Artificial Neural Network (ANN) method to classify attack binaries. The binary strings were reshaped as matrix and represented as grayscale images and the texture features were extracted by Gabor wavelet with 8 orientations and 4 scales. They were able to get 96.35% accuracy using the feed forward artificial neural network. In future, they are intended to get more accuracy in identifying the malware variants. Similarly, Liu et al. [5] has used K-Nearest Neighbor (KNN) to process grayscale images generated from malware. 98.2% accuracy was reached by the malware classification approaches.

Alazab et al., 2011 have proposed a model that uses numerous data mining techniques to detect and classify zero day attacks based on the window API call. Authors have analyzed the performance of kNearest Neighbor, Sequential Minimal Optimization, Naive Bayes, Backpropagation Neural Networks Algorithm and J4 8 decision tree on the dataset. After evaluating all the algorithms their implemented SVM Normalized PolyKernel has outperformed all other algorithms and NN backpropagation has performed the worst. The system in [6] could achieve 98.5% of true positive rate and 0.025% of false positive rate.

Abri et al., 2019 [4] has evaluated machine learning algorithms and deep learning classifiers to detect which one can detect zero day attack patterns more accurately. Based on four clustering techniques 1. Conventional machine learning 2.Simple Neural Network Learner 3.Deep learning using multiple layers and 4. Deep(er) learning with multiple hidden layers the authors have executed several experiments. Random Forest Classifier offered the best accuracy to identify Zero day attacks based on the experiments by the authors.They achieved an accuracy of 99.51% and validated it by applying 10-fold cross validation. National Science Foundation (NSF) supported their work in [7] under grants 1821560 and 1723765.

With the motivation to detect different malware along with unknown or zero day attacks To-biyama et al. were the first to extract features from a trained recurrent neural network (RNN) and generate feature images. Then, feature images with labels were assembled to form the input to the conventional neural network (CNN) for classification. The system in [8] classifies 26 types of malware from 11 families and this method can reach a classification accuracy of 96%.

Zhou and Pezaros (2019), realised how the dataset chosen plays an important role in determining the performance of an IDS model. So, in their paper [9], they trained their model on the CICAWS2018 dataset which consists of statistical data. Furthermore, they show how simple decision tree classifiers can also detect ZeroDay attacks well, among other five commonly used machine learning models which include Random Forest, Naive Bayes, Neural Network, Quadratic Discriminant and KNearest Neighbours classifier.

The authors, Gupta and Rani (2018), make use of scalable MLlib to detect zero-day malwares by building a big data framework on top of Apache Spark. The proposed architecture in [10]

is able to classify different types of attacks. They implemented Random Forest, Naive Bayes and SVM algorithms for this purpose. Among these, Random Forest proved to give the most accurate result which is scored at 98.88%.

Jionza in the paper Random-Forest-Based Network Intrusion Detection System proposed a Random Forest model in anomaly, misuse and hybrid network IDSs. The model automatically builds the pattern of intrusion for misuse detection by training on the provided data and matches the unseen data to the learnt patterns. The authors in [11] used sampling techniques and feature selection to improve the performance of the misuse detection. For the anomaly detection, the Random Forest model detects new attacks by an unsupervised outlier detection of the Random Forest. The hybrid approach for detection basically combines the advantages of both the misuse and anomaly detection systems to increase predictive performance. The model achieved an accuracy of 92.93%.

## 2.2 Algorithms/Models

### 2.2.1 Artificial Neural Networks

We have used the Artificial Neural Networks (ANN) algorithm which was designed to simulate the way human brains function. Here, the basic building block is a neuron which takes input, processes the input and finally generates an output. The inputs are weighted and the weights are initialized to small randomized values. The sum of the weighted inputs are later passed directly through an activation function which is present in every neuron. These activation functions map the input to output. They are called activation functions because they decide if a neuron is going to be ignited or not based on a threshold. For training, the Stochastic Gradient Descent is still the favored algorithm in Neural Networks. Here, each row in the training dataset is exposed to the network one at a time. The network processes the upward input of activating neurons as it is eventually going to create an output value which is known as Forward Pass. This pass will eventually generate an output. Based on the difference between the real output and the estimated output, the error is calculated. This error is then propagated back across the network and the weights are updated accordingly to minimize the error. This algorithm is called the Back Propagation.

We have decided to use an optimized version of the Gradient Descent known as Adaptive Moment Estimation (Adam) that computes adaptive learning rates for each parameter. The decaying averages of past and previous squared gradients  $m_t$  and  $v_t$  are calculated as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (2.1)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (2.2)$$

The first moment (mean) and the second moment (uncentered variance) of the gradients are estimates of  $m_t$  and  $v_t$  respectively, hence the method name. By calculating bias-corrected first and second moment figures, they combat these biases:

$$\hat{m}_t = \frac{m_t}{(1 - \beta_1^t)} \quad (2.3)$$

$$\hat{v}_t = \frac{v_t}{(1 - \beta_2^t)} \quad (2.4)$$

They then use these to change the parameters that the Adam update rule produces:

$$\theta_{t+1} = \theta_t - \hat{m}_t \frac{\eta}{(\sqrt{\hat{v}_t} + \epsilon)} \quad (2.5)$$

As soon as the model has been trained, it can be used to make predictions. The ability of the model's power can also be tested by making predictions on test or validation data. The network can also be operationally deployed and used repeatedly to make predictions.

## 2.2.2 K-Nearest Neighbour

The KNN or nearest neighbours is a simple, supervised machine learning algorithm that is used in regression and classification problems. KNN is known as a non-parametric and lazy learning algorithm. It is termed non-parametric as it doesn't surmise anything about the data. Lazy learning as it doesn't have a specialised training phase. Testing phase is where all the training data is used. Resulting in the training phase being faster and testing much slower and expensive.

KNN in classification relies on a voting process from the object's 'k' nearest neighbours, the number 'k' being the key deciding factor. The object is classed based on the majority votes received from specified k nearest neighbours.

We then calculate the distance between each test data to each train data using the Euclidean distance formula.

$$dis(x, x_i^j) = \sqrt{\sum_{i=1}^d (x(i) - x_i^j(i))^2} \quad (2.6)$$

We find the weighted distance of the test point from each neighbour.

$$w = 1 - \frac{dis(x, x_i^j)}{\sum_{i=0}^k dis(x, x_i^j)} \quad (2.7)$$

Using the weighted distance value, neighbours are sorted in ascending order. The first K nearest neighbours are then chosen. The test point is designated a class based on the most recurrent



class of  $k$  neighbours.

### 2.2.3 Random Forest

Random Forest is a supervised ensemble algorithm widely used for classification and regression problems for its high effectiveness. The forest is an ensemble of many individual decision trees and the ability to control the number of trees to be present in the forest generally gives the model high robustness. The Random Forest firstly chooses a random subset of the dataset with replacement (bootstrapping) and then randomly selects ' $k$ ' features from the ' $m$ ' features of the dataset. This adds randomness to the model and makes each decision tree uncorrelated to each other. The ' $k$ ' features are used to locate the root node depending on the criterion used. In our paper the criterion used was 'gini' for the gini impurity

$$Gini = 1 - \sum_{i=1}^C (P_i)^2 \quad (2.8)$$

The nodes are then split using the criterion to form daughter nodes, using different random sets of  $k$  features each time, until the tree reaches its maximum depth with only a single sample at the leaf node or is stopped prior to this. This process is continued until ' $n$ ' trees are built. Each tree provides a classification for the test samples. Votes are taken for all the classifications of the target label of each tree and the majority vote (The maximum count of classified labels) gives the final prediction of the Random Forest model.

Random Forests usually provide high accuracy, seldom suffering from over fitting. This is due to the fact that random forest creates a number of decision trees working in ensemble. Here each individual decision tree may over fit and have high variance (if the tree is too deep) or under fit and become inflexible with high bias (if the model is too simple) but because of the aggregation of the results of a large number of decision trees by majority voting we get a model with low bias and low variance allowing Random Forest to have larger trees. Thus, increasing the number of trees in a Random Forest not only results in more accurate predictions but also avoids overfitting. Random Forest classifiers are also adept in handling missing data and can also be used to select features based on feature importance.

# Chapter 3

## Proposed Model

### 3.1 Dataset Description

For the dataset we have used the CSE-CIC-IDS2018 dataset[12], which was developed jointly by Communications Security Establishment(CSE) and the Canadian Institute for Cybersecurity (CIC) which was collected over a period of 10 days. It was prepared by implementing an attack infrastructure of 50 machines and a target organization consisting of 420 machines and 30 servers over 5 departments. It involves the capture of machine network traffic and device logs, along with 80 features extracted using CICFlowMeter-V3[13] from the collected traffic. This dataset includes a wide range of attack scenarios such as Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks and internal infiltration of the network. All in all, with the implementation of the attack scenarios and the infrastructure, the final dataset consists of 80 features and 14 different types of attacks. The dataset contains more than 16 million connections. The connections are either labelled ‘Benign’ (non-attacks) or the name of the attack type they represent.

Although this dataset is the newest dataset currently available for IDS and is designed to address the deficiencies of static and one-time datasets that do not provide adequate data for ever-changing activities and trends of network and intrusion evolution, it is highly biased. Almost 70% of its records are benign, making any model of Machine Learning vulnerable to errors.

#### 3.1.1 Data Pre-processing

First of all, we merged all of the datasets into one. Secondly, since our primary objective is to identify ”Zero Day Attacks” that we believe have parallels with the attacks we already know, we are not interested in categorizing any attack as one of the known attacks. Hence, all the labels fall into one of two categories: either it’s an attack or it isn’t. Then we transformed the labels into numerical values consisting of only 1s and 0s, where 0 means benign and 1 means an attack. After that, all NaN, infinite, and non-numerical values were removed from the dataset in order for our ML algorithms to operate on it. Next, we split the dataset into 75% for training and 25% for testing/validation. In addition, based on their similarity with other features, we dropped

some features to reduce dimensionality. In this respect, a threshold of 0.9 is used. Correlation features greater than this threshold were dropped and were not used in training. Finally, We scaled both the training and testing datasets using Standard Scalar.

## **3.2 Model Description**

### **3.2.1 Preparing ANN Model**

As of the sequence of layers, we will construct a sequential model and add layers until our network topology is satisfied. The first and foremost thing is the number of inputs. In our case, after feature selection, the input layer consists of 47 neurons each referring to a specific feature in our dataset. Our model will have one neuron in the output layer that will have the desired result.

For the number of hidden layers, we experimented with different combinations through trial and error and finally found 3 hidden layers each consisting of 24 neurons to be producing the best accuracy.

Since, Multi Layer Perceptron which is entirely connected performs well in Binary Classification, our Neural Network will also be fully connected, that is each node in every layer is connected to all the nodes in the layer next to it.

Now that our Network is constructed, we need activation functions. For the activation functions, we have used ReLU for all the layers except the output layer which uses the Sigmoid activation function.

For hyper-parameter optimization, we have used RandomSearch. Another alternative option that could be used is Grid Search but RandomSearch converges much faster comparatively. The optimal value for batch size, epochs, k-fold cross validation and optimizer are found to be 1024, 500, 10 and Adam respectively.

Moreover, to prevent overfitting, we have used the Dropout technique. It is a technique where, during preparation, randomly selected neurons are ignored. They are "dropped out" which means that their contributions in the forward pass are temporarily terminated and any weight change in the backward pass are not added. The Dropout rate of 0.1 generates the best accuracy which is also with the help of the RandomSearch algorithm.

### 3.2.2 Preparing KNN Model

Since the value of  $k$  greatly affects the performance of KNN we have to choose a value that best satisfies our model both in terms of performance and speed. Higher values of 'k' will make the algorithm computationally expensive to the point where the model becomes unusable in real life applications and smaller values of 'k' tend to overfit the model by picking up noise.

So we made a parameter grid and set different values of 'k' (5 - 25) to see which value makes the KNN perform the best.

After setting the range of 'k' for the model to be evaluated upon, we also have to set the other parameter values in our parameter grid. This parameter grid was fed to the RandomSearch algorithm to optimize the model parameters. This method of hyper-parameter tuning is very effective as it provides the best parameter settings for the model faster than the grid search algorithm. The results were validated by using 10-fold cross validation of the RandomSearch algorithm.

To avoid overfitting and underfitting of the model we have evaluated the test accuracy and train accuracy for different values of  $k$ . The resulting 'k' value thus does not overfit or underfit the model.

### 3.2.3 Preparing Random Forest Model

As we have seen that having a large number of trees in the forest avoids the model from being over fitted but may in turn be computationally expensive to train. So first we have to find a suitable value for the `n_estimators` parameter (number of decision trees) of the Random Forest. We make parameter grid containing a range of values for `n_estimators` that are neither too less to cause underfitting nor too high to become computationally expensive.

Random Forest with deep trees may result in the model having high variance and becoming too flexible causing it to memorize the training data and become less generalized. So to avoid overfitting, a shallow depth of trees can be set. We make a range of values for the `max_depth` parameter that are not too deep and add it to our parameter grid.

Once the parameter grid is set up we run the RandomSearch algorithm for hyper-parameter turning of our model. This gives us a parameter setting that makes our model perform the best. The number of trees is large enough to avoid overfitting and give higher accuracy. The depth of the individual trees are shallow enough to avoid overfitting. These can be further optimized to increase performance in terms of speed and computational expense.

The `max_features` parameter of the random forest which is the number of features to be randomly chosen to split a node is set to  $\log_2 'm'$  (the number of features in the dataset). The default value for the `max_features` parameter is  $\sqrt{'m'}$  but the difference in accuracy attained

for both is quite insignificant but setting `max_features` as `log2 'm'` makes training faster.

`Bootstrap` value is set to `true` (`bootstrap=True`), this enables bootstrapping of samples which adds randomness to the model. A sample of the dataset is to be randomly selected with replacement ensuring that the model does not have high variance and does not overfit.

The `oob_score` value is set to `true` to use out-of-bag samples for evaluating the models performance.

After all the parameters have been set, we evaluate the model using `stratifiedKfold` cross validation where value of `k` is 10. This gives us an estimate of how well our model should be performing.

# Chapter 4

## Experimentation

This chapter describes our computations and outcomes and the visual representation of our output is also given.

RandomSearch algorithm was run to find the best parameters for the ANN model. Among all the combinations, the best 4 of them are shown below in Table 4.1. The best result for ANN was achieved in Model 1 of Table 4.1, where the parameters are batch size= 1024, epochs= 500 and optimizer= Adam.

*Table 4.1: Best four parameters found for ANN from RandomSearch Algorithm*

Model	Parameters	Precision	Recall	F-Measure	Accuracy
1	<b>Batch size= 1024</b> <b>Epochs = 500</b> <b>Optimizer = Adam</b> <b>(Best)</b>	<b>98.71 %</b>	<b>95.11 %</b>	<b>96.88 %</b>	<b>98.30 %</b>
2	Batch size= 2048 Epochs = 250 Optimizer = Adam	97.36%	94.18%	95.74%	97.63%
3	Batch size= 1024 Epochs = 500 Optimizer = Adam	97.93%	93.56%	95.69%	97.61%
4	Batch size= 2048 Epochs = 500 Optimizer = Adam	98.03%	93.63%	95.78%	97.66%

The Random Forest model was run on Google colab cloud platform. Upon running the Random-Search algorithm the best parameter setting found for the model was where  $n\_estimator=110$   $max\_depth=20$  and  $max\_features= \log_2$  (Model 1 of Table 4.2). This provided high accuracy with a satisfactory training time. On increasing the number of trees and depth of the trees (Model 3 and 4 of Table 4.2) we can see that scores only differ very slightly but take longer to train. If we decrease the number of trees and depth (Model 2 and 5 of Table 4.2), we see that the training time does decrease a bit but the scores also drop slightly.

*Table 4.2: 5 Best selected models and the effect of  $n\_estimator$  and  $max\_depth$  on RF*

Model	Parameters	Precision	Recall	F-Measure	Accuracy	Oob Score	Train Time
<b>1</b>	<b><math>n\_estimator = 110</math> <math>max\_depth = 20</math> <math>max\_features = \log_2 m</math> (BEST)</b>	<b>99.90%</b>	<b>96.60%</b>	<b>98.22%</b>	<b>98.90%</b>	<b>98.90%</b>	<b>13.4 min</b>
2	$n\_estimator = 110$ $max\_depth = 10$ $max\_features = \log_2 m$	99.86%	96.56%	97.71%	98.56%	98.59%	10.3 min
3	$n\_estimator = 110$ $max\_depth = 50$ $max\_features = \log_2 m$	99.81%	96.60%	98.20%	98.89%	98.87%	12.7 min
4	$n\_estimator = 150$ $max\_depth = 20$ $max\_features = \log_2 m$	99.90%	96.59%	98.21%	98.90%	98.88%	17.2 min
5	$n\_estimator = 100$ $max\_depth = 20$ $max\_features = \log_2$	99.88%	96.60%	98.21%	98.90%	98.88%	12.6 min

On the best model found so far we then change the max features to half the total features ( $\text{max\_features} = m/2$ ). We can see that not only did the models' predictive performance increase slightly but the time needed for training increased significantly as well. The bump in accuracy is not too much thus trading the training time for such a little increase in accuracy will not be wise. Now we again increase the max feature to be  $m$  ( $\text{max\_features} = m$ ). Now all the features are used to split the nodes. We can see that the training time has considerably increased however the predictive performance roughly remained the same.

*Table 4.3: Effect of max feature on the best tuned model of RF*

Model	Parameters	Precision	Recall	F-Measure	Accuracy	Oob Score	Train Time
1	<b>n_estimator =110</b> <b>max_depth =20</b> <b>max_feature =</b> <b><math>\log_2 m</math></b> <b>(BEST)</b>	<b>99.90%</b>	<b>96.60%</b>	<b>98.22%</b>	<b>98.90%</b>	<b>98.90%</b>	<b>13.4 min</b>
2	n_estimator =110 max_depth =10 max_feature = $m/2$	99.86%	96.76%	98.29%	98.94%	98.92%	51 min
3	n_estimator =110 max_depth =50 max_feature = $m$	99.70%	96.79%	98.23%	98.90%	98.90%	82 min



In the case of KNN, from the RandomSearch algorithm we got the best 3 models with different k values as shown in Table 4.4. We can see that the scores are almost similar but model 1 with k=5 gives the best accuracy as well as makes the model less complex.

*Table 4.4: 3 Best selected models found for KNN from RandomSearch algorithm*

Model	Parameters	Precision	Recall	F-Measure	Accuracy
<b>1</b>	<b>K=5 (BEST)</b>	<b>98.65%</b>	<b>93.39%</b>	<b>95.95%</b>	<b>98.53%</b>
2	K=10	99.04%	92.94%	95.89%	98.51%
3	K=15	98.78%	93.09%	95.85%	98.50%

# Chapter 5

## Result Analysis

The performance of our model was determined by how accurately it can detect attacks, known and unknown alike. The four parameters of the confusion matrix which are: TP (True Positive), FP (False Positive), TN (True Negative) and FN (False Negative) were used to evaluate the performance. The Accuracy verifies the ratio of the samples that were predicted accurately to the samples that were provided in total.

$$Accuracy = (TP + TN)/(TP + TN + FP + FN) \quad (5.1)$$

Additionally, Precision specifies the ratio of positive observations that were predicted correctly to the total positive observations of the test set.

$$Precision = TP/(TP + FP) \quad (5.2)$$

Recall indicates the ratio of positive observations that were predicted correctly to the sum of true positives and false negatives in the test set.

$$Recall = TP/(TP + FN) \quad (5.3)$$

F-measure determines the weighted average of the precision and recall.

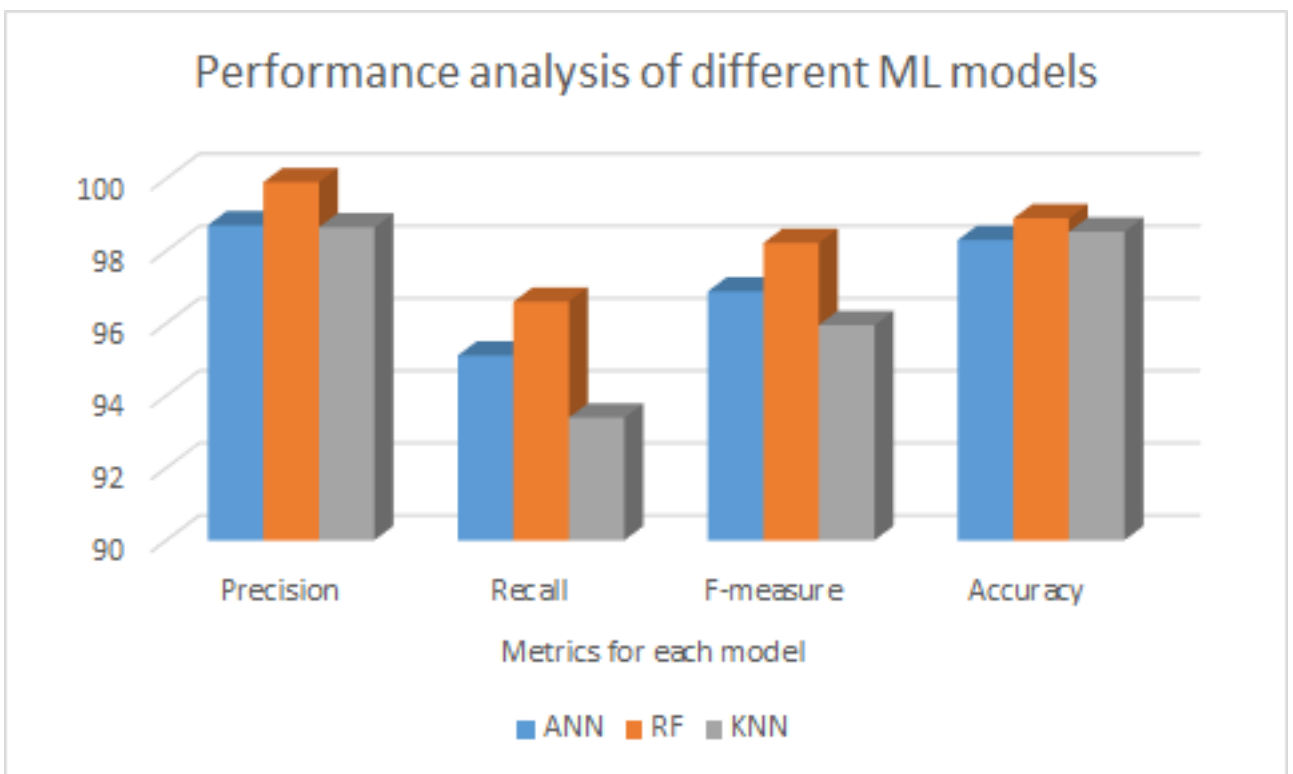
$$F - measure = (2 * Precision * Recall)/(Precision + Recall) \quad (5.4)$$

The OOB scores are used to validate the Random Forest algorithm. It gives a better predictive model whose variance is low and also prevents leakage of data.  
(Equation)

The proposed Random Forest model has high accuracy along with a similar cross validation score. Our model performs better than the model used in paper[10]. Where the Random Forest had an accuracy of 98.88%, slightly lower than our 98.90%. Our model also outperformed the Random Forest model of the paper[11] where it had an accuracy of 92.93% on the IDS. In the paper [9] the proposed model was run for individual attacks. For this, the Random Forest did poorly in some cases with low accuracies and high false positive rates. Our model on the other hand was trained on a dataset where all attacks were combined and represented as 1. For this our model had better accuracy and low false positive rate. Similarly, our KNN model, as

well as, ANN model outshined the corresponding models described in paper [6]. The precision and F-measure of our KNN model is 98.65% and 95.95% whereas these scores in paper [6] is 94.80% and 94.80%, respectively. Moreover, their worst model is NN which scores 88.1% in precision, 86.4% in Recall and 85.9% in F-measure. Comparing these values with our ANN model, we obtain 98.71%, 95.11% and 96.88% in these criteria, which is better.

The results obtained from the RF model are the best among the models that we implemented. It has the highest scores in precision, recall, F-measure and accuracy. ANN comes a close second with the aforementioned scores being slightly lower than that of RF model but higher than the KNN model.



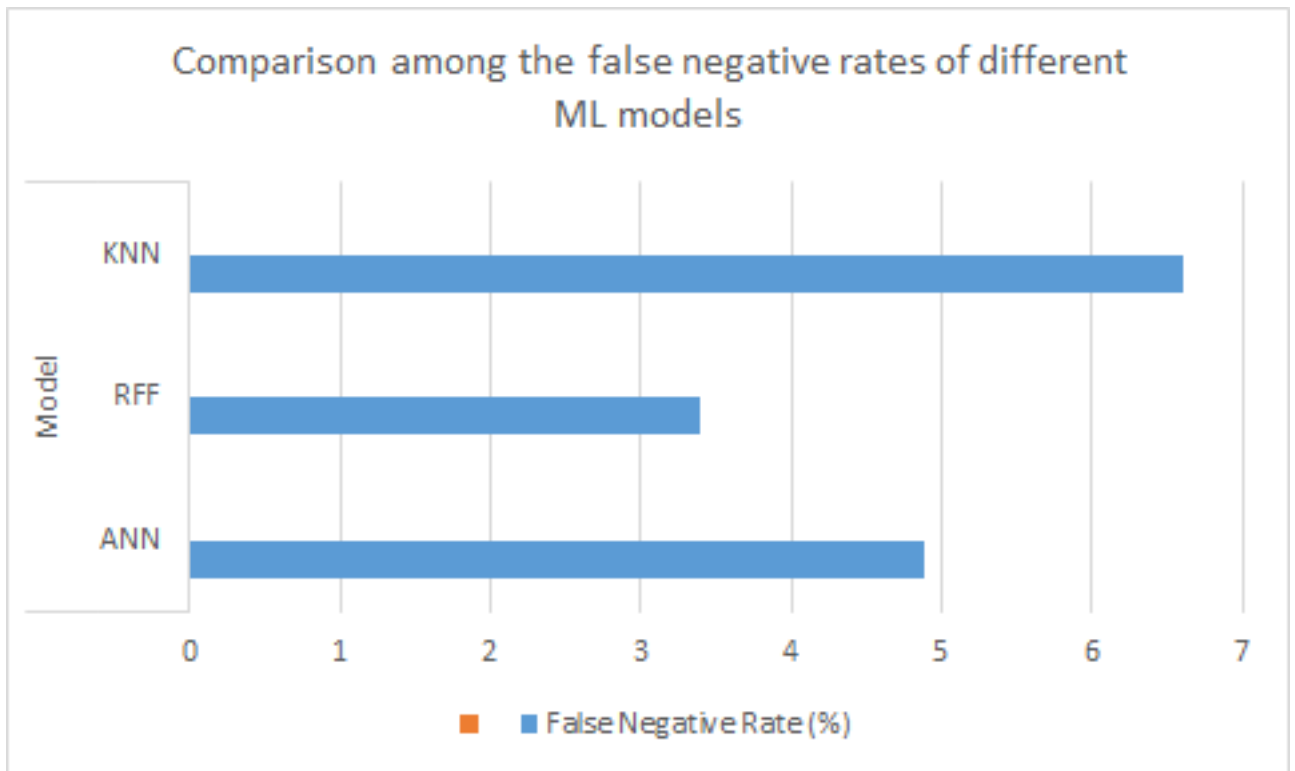
**Figure 5.1: Performance analysis of different ML models**

However, almost all the models show high accuracy in their results. This is due to the dataset being biased. Hence, we are focusing on False Negatives. It is defined as the outcome that estimates the negative class inaccurately. A user/ company may endure a huge loss if even a single attack does not get detected.

In table 5.1 the different values of false negative that were obtained from different machine learning models are shown. The best value was procured from the Random Forest model.

*Table 5.1: False Negative Rate of different ML Models*

Model	False Negative Rates
<b>RF (BEST)</b>	<b>3.39%</b>
ANN	4.88%
KNN	6.60%



**Figure 5.2: Comparison among the false negative rates of different ML models**

From Fig 5.2, we can see that the Random Forest model has the lowest percentage of False Negatives. Thus, it is more likely to detect Zero Day attacks efficiently and precisely.

# Conclusion

The real challenge we face every day is to keep our information secure. With IoT taking over our lives, slowly but surely, our data and well-being is compromised due to lack of secure networks. Our proposed system is to address this problem which is long overdue and needs to be attended to with utmost priority.

Among the 3 machine learning based algorithms that we implemented in our research, we obtained the best result from Random Forest. It gave us an accuracy of 98.9% which outperforms Artificial Neural Networks (98.3%) and K-Nearest Neighbour (98.53%). However, while working on our system, we have realized that the dataset we have used is quite biased even though it is the most recent dataset that is available.. Hence, for further research we plan to reduce the biasness of the available dataset to ensure more accurate results. Since we achieved the best outcome from our Random Forest model, we would like to research more on it so as to achieve even better results. We would also try to reduce the rate of false negatives in this case. Furthermore, we would delve more into Neural Networks and try to obtain a more promising output as Deep learning is gaining increasing popularity in recent times and extensive research on this area can take it to greater heights.

# References

- [1] Niyaz, Q., Sun, W., Javaid, Y. A., Alam, M. (2016). *A Deep Learning Approach for Network Intrusion Detection System..* Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), Vol. 3, no. 9 pp. 1 – 6. doi:10.4108/eai.3-12-2015.2262516
- [2] Hindy H., Atkinson,R., Tachtatzis, C., Colin, J., Bayne, E., Bellekens, X. (2020). *Utilising Deep Learning Techniques for Effective Zero-Day Attack Detection..* Electronics, 9(10), 1684. doi:10.3390/electronics9101684
- [3] Hijazi, A., EL Safadi, E., Flaus, J. M. (2018) *A Deep Learning Approach for Intrusion Detection System in Industry Network.* Lebanese University 5 Décembre, 12:50 · The First International Conference on Big Data and Cybersecurity Intelligence (BDCS Intell' 2018)At: Beyrouth, Lebanon.
- [4] A. Makandar and A. Patrot. (2015) "*Malware analysis and classification using Artificial Neural Network,*" International Conference on Trends in Automation, Communications and Computing Technology (I-TACT-15), Bangalore, 2015, pp. 1-6, doi: 10.1109/ITACT.2015.7492653
- [5] L. Liu, B.-S. Wang, B. Yu, and Q.-X. Zhong, Sep. 2017. "*Automatic malware classification and new malware detection using machine learning*" *Frontiers Inf. Technol. Electron. Eng.*, vol. 18, no. 9, pp. 1336–1347
- [6] Alazab, M., Venkatraman, S., Watters, P., Alazab, M. (2010). *Zero-day Malware Detection based on Supervised Learning Algorithms of API call Signatures..*9th Australasian Data Mining Conference, AusDM 2011, 121, 171-182
- [7] F. Abri, S. Siami-Namini, M. A. Khanghah, F. M. Soltani and A. S. Namin. (2019). "*Can Machine/Deep Learning Classifiers Detect Zero-Day Malware with High Accuracy?,"* IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 2019, pp. 3252-3259, doi: 10.1109/BigData47090.2019.9006514
- [8] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, and T. Yagi. "*Malware detection with deep neural networks using process behavior,*" in Proc. IEEE 40th Annu. Comput. Softw. Appl. Conf., Jun. 2016, pp. 577–582
- [9] Qianru Zhou, Dimitrios Pezaros (2019): "*Evaluation of Machine Learning Classifiers for Zero-Day Intrusion Detection- An Analysis on CIC-AWS-2018 dataset*", <https://arxiv.org/abs/1905.03685>

- [10] Deepak Gupta, Wrinkle Rani (2018): *Big Data Framework for Zero-Day Malware Detection, Cybernetics and Systems*, DOI: 10.1080/01969722.2018.1429835
- [11] J. Zhang, M. Zulkernine and A. Haque (2008). “*Random-Forests-Based Network Intrusion Detection Systems*”, in *IEEE Transactions on Systems, Man, and Cybernetics- Part C: Applications and Reviews*, Vol. 38, No. 5, pp. 649- 659, September 2008
- [12] “CSE-CIC-IDS2018 on AWS” UNB. [Online]. Available:  
<https://www.unb.ca/cic/datasets/ids-2018.html?>
- [13] “CLC Flow Meter (formerly ISCXFlowMeter)” UNB. [Online]. Available:  
<https://www.unb.ca/cic/research/applications.html?>
- [14] “2018: The Four Zero Day Attack Stats and Trends You Need To know” Votiro.[Online]. Available:  
<https://votiro.com/2018-the-four-zero-day-attack-stats-and-trends-you-need-to-know/>
- [15] “As malware and network attacks increase in 2019, zero day malware accounts for 50 of detections” Helpnetsecurity. [Online]. Available:  
<https://www.helpnetsecurity.com/2019/12/13/network-attacks-2019/>