AI Based IDS

20/09/2020

Acknowledgement

"I would like to thank my family and supervisor for supporting me with my project. Also thank Middlesex University of Degrees that gave invaluable experience and knowledge while studying."

Abstract

Using AI based IDS an rising approach to tackle new attack vectors similar to other attacks or when through changes that makes a newer threat. This research consists of investigating AI based IDS is better on classification than a conventional IDS to identify threats in real time. This consists of using 2 similar Multilayer Perceptrons and one ensemble to compare not only time and accuracy but precision, recall, f1-score and balanced accuracy score. It is possible and it can be used for real time detection for anomaly activities in network traffic, in this case it is an example with KDD_CUP99. The results have provided sufficient balanced accuracy scores that gave insights on much in depth for future work as the next possible step to make AI based IDS stand alone software.

Keywords

Artificial Intelligence (AI), Intrusion Detection System (IDS), Confusion Matrix (CM), Neural Network (NN), Ensemble, Early Stopping and Random Forest Classifier

Chapter 1

1 Introduction

1.1 Introduction to Thesis

In this chapter, a brief overview of development in AI and IDS, AI methods that can be introduced in IDS for the hypothesis and determining the best possible solution. Also discussing Methodological approach, research aims and dissertation outlines.

1.2 Background information on AI and IDS

Pioneered back in 1950s the era of [1]*(Russel and Norvig, 2009)* AI, developed concepts of how AI algorithms could be to perceive the environment like humans, thus came the Turing test (a blind test to have individuator to speak to a person and a computer, and predict which is more 'human' than the other candidate). The Turing test has not been passed by AI based Computers[1]*(Russel and Norvig, 2009)*, but this created more interest around 1960-1980s where probability and statistical reasoning became a full research. AI is a separate branch from Computer science or Physics or Mathematics. The nature of AI is having a combination of all and thus the application it is being used can be different. IDS was introduced back in *(Team, 1980)[2]* 1980s (1984-1986) for financial systems for behaviour analysis to identify the difference between legitimate users and fake users, thus creating patterns for these types of behaviours. The focus was more on identifying abnormal activities. More on Chapter 3, section 3.3.6 Confusion Matrix.

1.3 Hypothesis

The main hypothesis for this project is to find AI based IDS is better than conventional IDS (behaviour analysis via statistical approach). Also, this leads to a secondary hypothesis to find "Good" AI based IDS system to perform better than existing IDS and adapt to unseen attack vectors. Types of attack vectors have risen to bypass firewall and IDS scans via stenography-based attacks (to hide malware in media files).

1.4 Methodological approach

Requirement: Understanding Ensembles method and single classifier methods (Naïve Bayes, Neural network, Decision Tree and etc), find results that can provide AI based IDS can outperform a conventional IDS.

Design a new or evolved ensemble or single classifier method: Neural Network requires more focus on best hypothesis which consists additionally verifying feature selection, generalisation error, error rate, entropy and information gain. They can make changes to the existing methods of NN and apply pre/post corrections from other methods to NN, if possible.

Implement and Testing: Neural Network with other features for better logical reasoning when compared with other algorithms for satisfactory results.

1.6 Research aims

Neural Network: Researching on reducing complexity in high models, applying Bayes statistical learning methods (such as Bayes law), applying decision trees such as chi squared. Methods from best hypothesis which consists of feature selection, generalisation error, error rate, entropy and information gain. This will give space to increase several changes to provide better results. So, analysis on changing existing algorithms from related work [1].

1.7 Contributions

Three redevelopments were involved, Stage 1, Numpy array solution of Neural Network(unsuccessful), Stage 2 Pandas Dataframe (Successful on feed forward), Stage 3 uses sci-learn libraries to support the ideology of all previous stages also flexible enough for investigating feature selection, generalisation error, error rate, entropy (cross entropy and softmax), information gain.

Chapter 2

2 Literature review

2.1 Introduction to Literature Review

AI based IDS will provide better predictions and classification on newer threats and attack vectors. IDS was based on a Rule based Expert system replicated to the human decision system.

2.2 Overview of Neural Network

A Neural Network consists of perceptions, which has an Input, activation function and output. The concept of Summing and activation function [2]. The summing function is chain rule, which it takes the derivatives from the previous layers in order to construct a cost function which determines if change (to weights and biases) occurs near to the input side (near hidden layers or activation function layer), the effect could shift to the likeness of classifying the input correctly. This method is the main function of Neural Networks success and the other part is activation function that finds the best path to reach the local minima (then reach the global minima). The gradient descent is the sigmoid function, ReLU and etc, this helps to reach the local minima to help to classify the feature apart.

2.3 Related work

Ensemble for Feature selection [3]*(Bolón-Canedo and Alonso-Betanzos, 2019)*: A review and future trends covers ensemble as a set of hypothesis from classification under single hypothesis since 2000s and relates to feature selection which implies dimensional reduction to remove unnecessary inputs that has little to no effect to accuracy or errors. The issues discussed is when using feature selection and reduction in dimensionality, the requirement to more data set is necessary else it will have high error. It requires unique feature selection algorithms in order to prevent high errors, such as median and mean calculations with all data results to provide an average result instead of over-fitted error.

Ensemble-learning approaches for network security and anomaly detection[4]*(Vanerio and Casas, 2017)*: Use of super learners can increase better classification, First level learners which also consists of MLP as an option. The use of MLP or similar algorithms provides good first level learning, especially in anomaly detections the results yielded better. Ad Detection in network anomalies is also considered as debatable on which classification provides much accuracy and better classification. Super learners which provided better results but this was calculated under semi-synthetic dataset. Relies on Weighted-majority-voting similar classifiers (weighted sum of the individual single classifier with satisfactory level in identifying anomalies).

Intrusion Detection System using bagging ensemble selection[5]*(Sreenath and Udhayan, 2015)*: Assessing performance based on ensemble type bagging with selection provided a better performance even on k-fold validation (a performance evaluation method). This gives an insight that even in some cases the accuracy can be pinned for bias classification but performance can be absorbed as a successful result. Although using NSL_KDD or KDD99 does not yield much difference.

This allowed for this project to understand some changes that do not affect as much on results and data cleaning, prepossessing methods can help to achieve much more realistic results then high results that proves the classification incorrect.

2.4 Review of Dataset: KDDCUP99

The AI based IDS evaluation core dataset is ‘KDDCUP99 – 10% corrected’. The [6]*(KDD Cup 1999 Data, 1999)* dataset must be read properly via data reading methods, such as removing Nulls and duplicates, and helps mitigate dataset classification errors (inaccurate predictions). The KDD dataset contains ‘23’ labels of intrusion-type, 4 which are strings and other ‘38’ are integers and floats. Categorical variables and features (38 ints and floats) helps to build the data-frame for confusion matrix, which provides scores of accuracy, precision, recall and f1. Confusion Matrix helps to replace the traditional dataset reading by changing all into 2d arrays and observing if they are linearly separable, which is not possible for ‘no-relationship’ such as KDDcup99.

Some debate is still going amongst many ML enthusiasts that NSL_KDD is better than KDD_CUP99[7]*(Ravipati and Abualkibash, 2019)*. AS a matter of fact is that KDD_CUP99 has several errors which can be done with preprocessing and data cleaning to achieve similar results in NSL_KDD. The many issues were duplicates but other than that it has been modified, KDD_CUP99 has been corrected but accuracy will have dips which cannot be avoided. This helps to build the idea of unstable data sets as inputs in real time IDS systems, which enables it to be cleaned and used when training segments are performed.

2.4.1 Anomalies (attack vectors)

These are the reasons the research has taken place. In order to prevent anomalies in a network, it is required to find the new type of attack types that are taking place. In some extent to the project research, it is required for a thought process. What if normal and anomalies have been separated. It results in a state of identifying the 'differences' from normal traffic. This is where the dataset is flooded with many of the same input samples in the dataset which leads to duplicates. This shows that normal is dominated more than attack vectors

2.5 Research

Building a Neural Network without using pre-built libraries will provide an invaluable lesson on the core functionality with in-depth knowledge of calculus (derivatives, propagation, gradient descent and cost function), but final results contained with build libraries which had better performance and were much more flexible. Neural Network is one of the AI methods that contains complex functions to provide unique and black-box operations in the hidden layer, which is programmed but learnt from iterations (or cycles) of running and updating the weights to increase the strength. This is then kept for every connection in-between input to output.

2.6 Dissertation outline

In the following sessions there will be more developments of this work, Section 3 will provide methodology, experiments and research undertaken for this project. Section 4 consists of Results (design to results). Section 5 provides discussion of the overall status of the project. Finally Section 6 concludes with conclusion and further work that could be implemented to make it better.

Chapter 3

3 AI methods and Background Application

3.1 Introduction to AI methods and Background Application

3.2 Design
Tools and Resources; -

Hardware: Laptop: Intel i5 8265u with Intel iGPU UHD 620, 8GB Ram

Software: OS-native: Pop! os (Ubuntu), PyCharm community, python 3.8.2, External python libraries: scikit-learn, scikitplot, matlibplot numpy, pandas

Building a neural network ground up provides much versatility on what can be added to expand the strength. But also considering the case of computation may take time due to CPU intensive tasks taking on iterations and loops while feed forward and back-propagation occurs until reaching 50-200 iterations or reaching desired error rates.

3.2.1 Graphical Representation of Fully Connected Neural Network

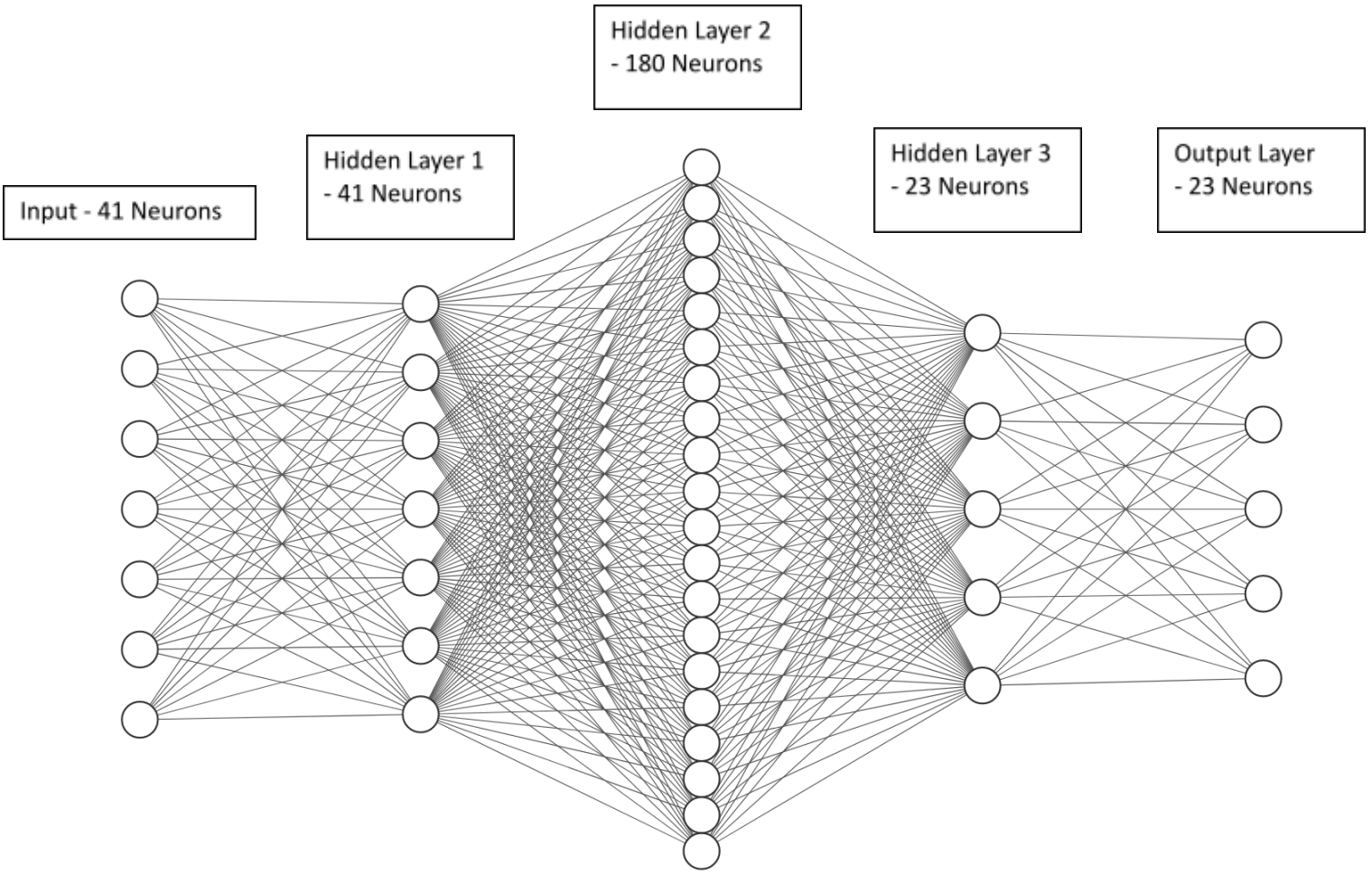


Figure 1 - Designed from [13] of MLP fully connect Neural Network

The Neural Network structure is mostly hidden but focuses on processing all MLP classifiers via a full connected network. This representation is close to the exact final results which has been to show the classification takes place via 41 neurons to match the inputs in the first hidden layer, then cross process and find similarities in a non-linear data, on a 180 neuron 2nd hidden layer to give over quadruple of 1st hidden layer for recursive analysis to in-depth findings in relationships, then pushed on 23 neurons, 3rd hidden layer which to reduce dimensions from 41 to 23 neuron in output layer, which will random outputs but cost function in output layer should be to sort the random outputs from hidden layer 3 as concise to one class identification for output layer.

3.2.2 Data Cleaning

Data cleaning, removing NaN values and removing duplicates is a dataset preparation practice for removing redundant data samples, without data cleaning, it can cause generalisation error due to running the same duplicate data samples which can further reduce accuracy. Providing clean data for classifiers to return in lower samples (inputs) which removes an average of 60% in samples for higher generalised classification(got 40% to 60% from cleaning when investigating this approach).

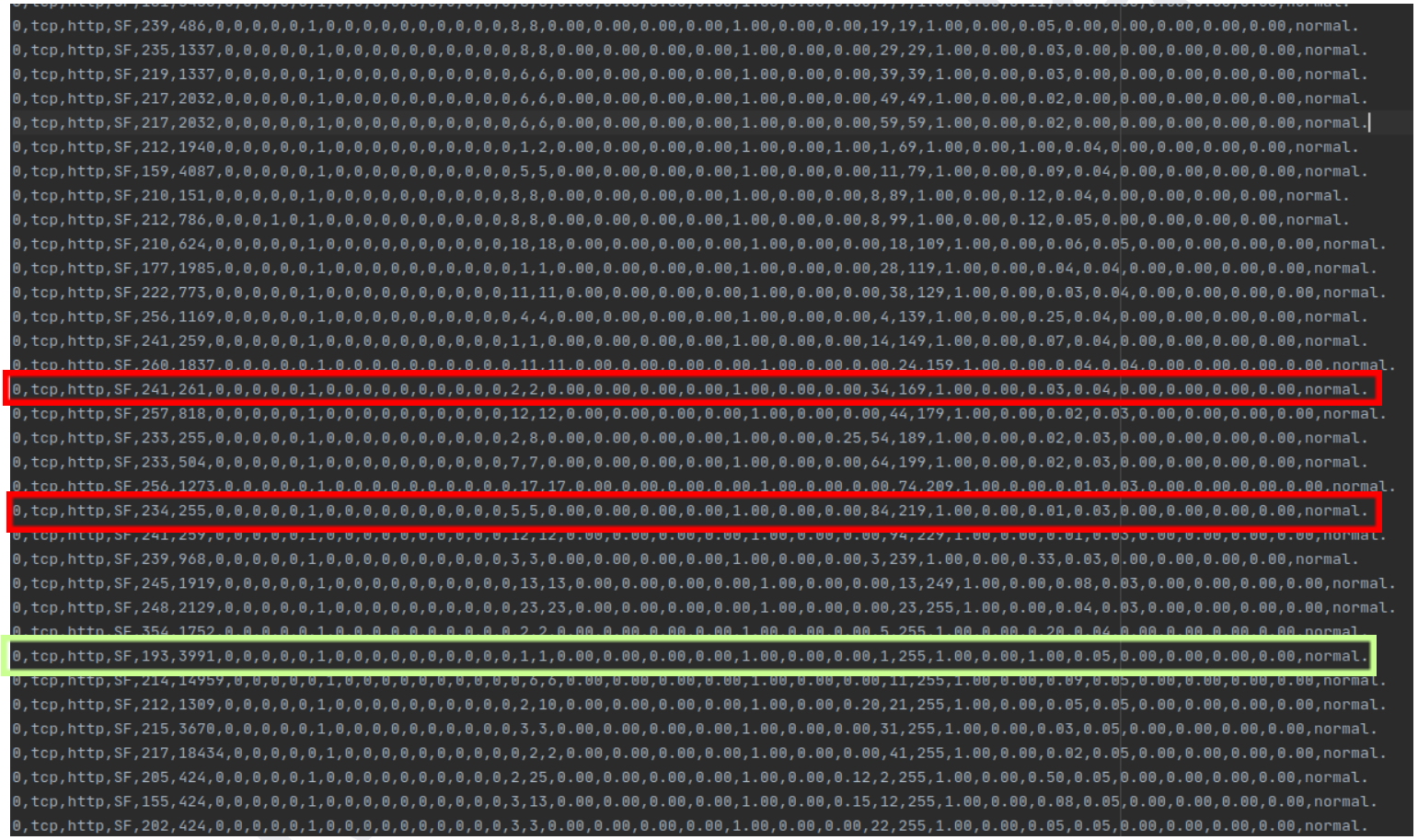


Image 1: Dataset (training and validation set) (Red) selected duplicates and (lime-green) majority of nulls

```
DaSt.dropna()
DaSt.drop_duplicates(keep='first', inplace=True)
```

Image 2: Code for cleaning redundant data

3.2.2 Feed Forward

Feed Forward, Input -> N number of hidden Layer -> output with Weights, biases all over the network connection (input*weights + bias) and finally True labels to compare input samples with input prediction is either correct or error is in terms of cost function.

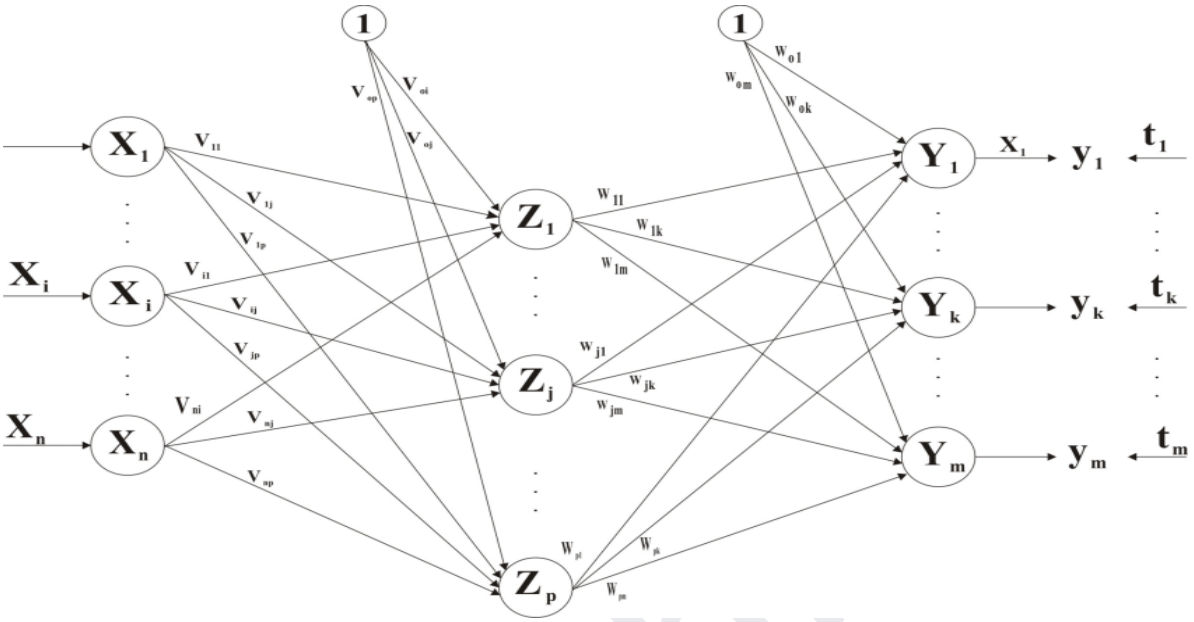


Image 3: [8]Feed forward with inputs * weights and bias for each Layer

This is performed with True labels and previous layer predictions, to calculate cost via Cross-entropy or standard (y - yhat)squared, section 3.3.2. In this dataset multiple inputs - single class classification means multiple outputs must be filtered to find one class for each sample. Done via one-hot encoding to vectorise for labels, example of 3 attack vectors, {nmap, buffer_overflow, ip_sweep}, which is separated for each number of values in an array into a matrices of (number of elements * number of input data), {[1,0,0],[0,1,0],[0,0,1]}, is a 3 by n matrix that is used for storing attack vectors from input data as predicted with a 1 on designated part of the array. These arrays are formed as columns that can be located based on label's identification.

```
0,tcp,http,SF,239,486,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,19,19,1.00,0.00,0.05,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,235,1337,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,29,29,1.00,0.00,0.03,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,219,1337,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,6,6,0.00,0.00,0.00,0.00,1.00,0.00,0.00,39,39,1.00,0.00,0.03,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,217,2032,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,6,6,0.00,0.00,0.00,0.00,1.00,0.00,0.00,49,49,1.00,0.00,0.02,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,217,2032,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,6,6,0.00,0.00,0.00,0.00,1.00,0.00,0.00,59,59,1.00,0.00,0.02,0.00,0.00,0.00,0.00,0.00,normal.
```

Image 4: (RED)41 inputs and (Blue) labels which are true predictions for the input labels for every input sample

The creation of this array is because there is only one true label which makes this method possible

```
# Start of Output
def OutP_cst(labels, last_label_ve):
    label_valAP = []

    for m in range(labels.shape[0]):
        if str(last_label_ve) == str(labels[m] + '.'):
            label_valAP.append(1)
        else:
            label_valAP.append(0)
    qui = pd.DataFrame({'c': label_valAP})
    return qui
```

Image 5: mapping on each run, which is alphabetical order, if true 1 else 0.

Under KDD_CUP99 Names(.txt), it contains the 23 output classes, in which the mapping takes place as illustrated below.

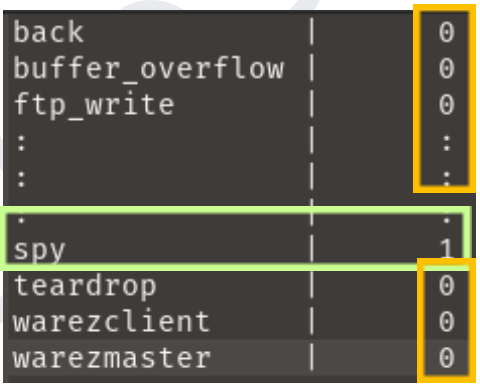


Image 6: (lime-green)Spy is the true value in this input sample, this means all will be zero (orange).

But Stage 1 and stage 2 uses no formalise or one-hot encoding but uses an if else statement to either labels from inputs (separate process) matches with y labels order(ascending alphabetically in file 'kddcup_names.txt'), then add 1 in array if Input_label == y_labels, else 0. So the array translates to [1,0,0] when nmap is first, if ip_sweep then [0,0,1], this method

is bit on the computation side to get values on spot then track a particular array in one hot encoding.

Stage 3 uses formalisation which consists of individualised mapping for output, that results in identification as value instead of string.

3.2.3 Back-Propagation and its applications

Back-propagation, from Calculated cost from above can be used as input and propagate backwards by calculus and derivatives (which is the inverse of all functions used between the layers from feed forward), covered in Section 3.3.

Calculus, the core of Neural Network (using a method called chain rule, also derivatives) on identifying small changes in a slope(for neural Network). This can help to identify if reaching local maxima instead of local minima, which in turn requires to reach global minima after many inputs have been Back-propagated, an local minima would gradually be reachable but still measuring very small steps (learning rate) to reduce overstepping the local minima. so at the start with high errors and nearly to top of the maxima(which can be far from global minima) requires back-prop to go down the slope with derivatives and reach local minima, in this case it can be referred as the cost function or error rate to be low for better results in classification. This will be required to reduce error close to 0.01. This shows the accuracy of the classifier at the end with mean value from all costs of other samples.

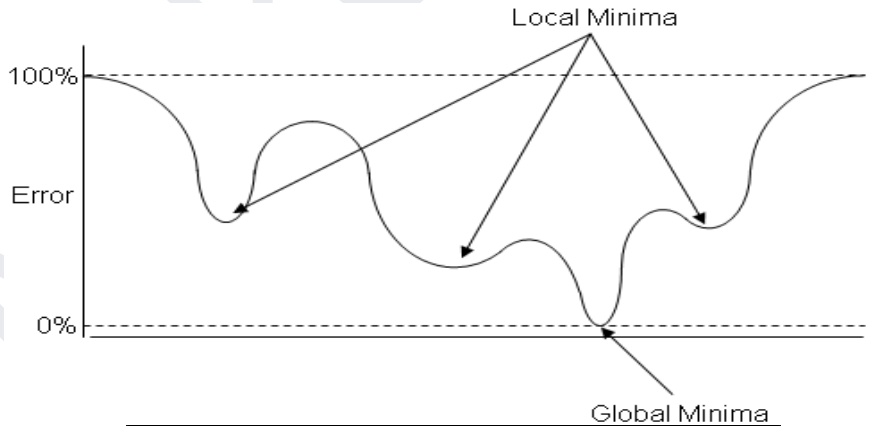


Image 7: [9]Slope of Reaching Local and global minima

The term derivatives means a very small value between a point, and a small value either positive or negative to understand the slope's current ground, which helps to reach the local minima. Positive means going towards maxima and negative towards minima. The derivatives are calculated for every step in-between the Output layer to input layer. This requires a derivative of the Output to hidden layer with respect weights and bias. This tends

to identify the correct nudge to a weight back to feed-forward after cycling from back-prop (with positive or negative numbers in floating points), and can make an impact to some degree. For example a small nudge to the previous layers values can have change to the weights that inputs with weights and activation could be reacting differently, section 3.3.3.

Another addition to derivatives is of 3 main reversed formulas (Section 3.3.4) used to get a derivative from the input. There are 3(or 4) for each layer, which are weights, inputs that was from the previous layer and z is weights inputs and bias calculated before applying activation function. A simple example to illustrate the weighted inputs + bias, then appended to ReLU, which results as input for next layer, if propagated it will be finding Layer 2 weights, inputs and bias from sum, and ReLU derivatives to find sum of Layer 1 and finding individual inputs derivatives and finally calculates best nudge to provide reduced Log error (cost function rate or score).

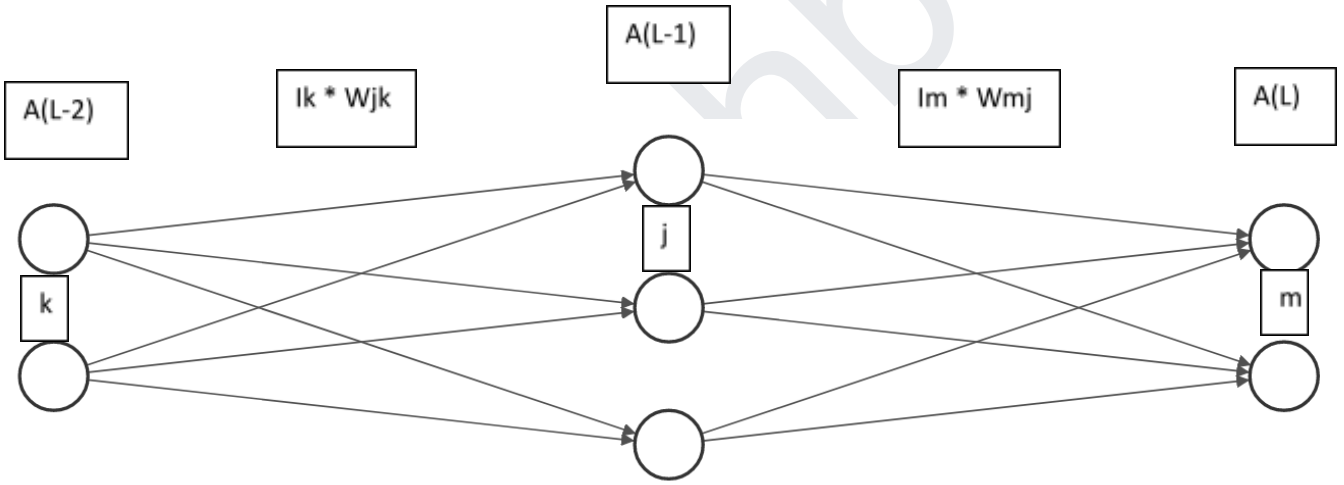


Image 8: Calculas of each Layers' weights and inputs and bias.

One more individually for each activation used (if sigmoid then prime sigmoid and if ReLU which has a separate derivative of x > 0 if 1 and x <= 0 if 0)

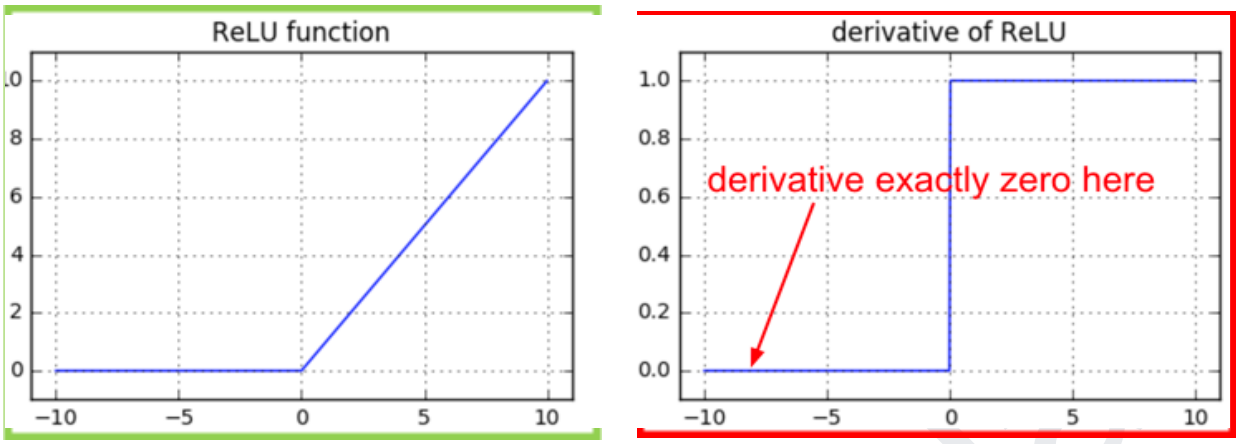


Image 9: [10] (green) Relu function and (Red) derivative of ReLU

3.3 Review of Neural Network design

Development stages: Stage, 1 Stage 2 and finally the fully functional stage 3. More is covered on Section 3.3.5

3.3.1 Neural Network

Has Inputs, weights, input functions (multiple each input and weights and sum all results with bias), Activation function for determining the strength of the connection, for Sigmoid is 1 or 0 and ReLU results to 0 to infinity from previous layers connections strength.

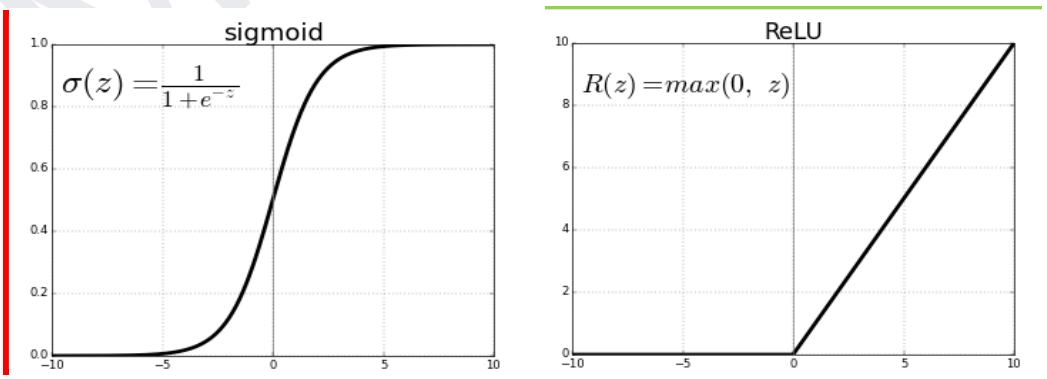


Image 9: [11] (Red) sigmoid function and (Green) ReLU

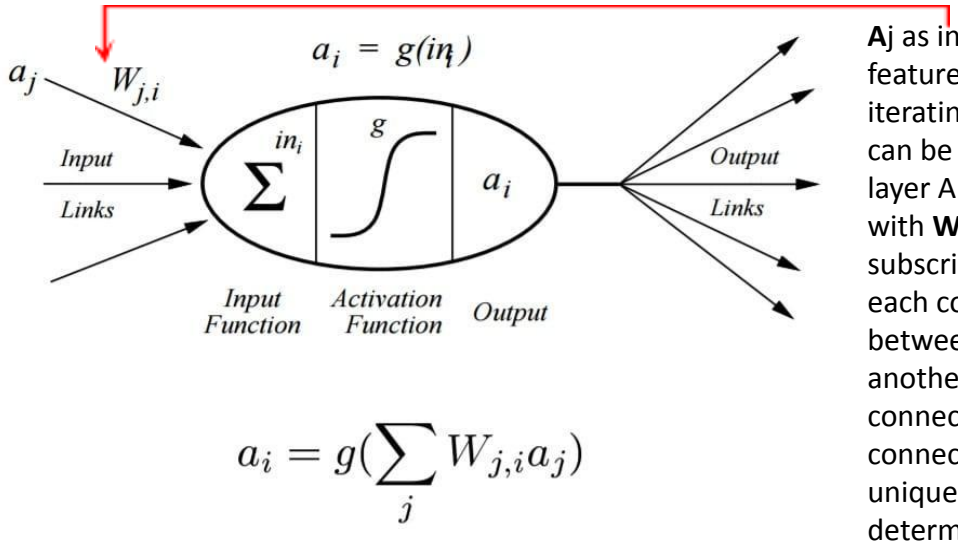


Image 10: [12] The structure of Feed forward

Aj as input from 1 feature with j as iterating subscript. Also can be labelled as first layer Applies the same with **W**ji with 2 subscripts to provide each connection between an input and another neural connection, each connection having unique weights determines strength of two neurons.

Input Function presents as sum function over all input links plus bias, then as input for **G,** a activation function to determine the slopes direction (positive or negative to get towards local minima)

Ai can be labelled as output from g = (**A**j) and input for next layer links with independent weights. and **A**i = Activation Function with a Sum of all values of (**A**j multiplied by **W**ji)

3.3.2 Softmax and Cross-Entropy (Log loss)

The softmax and cross entropy both are used for the function probability distribution which can lead results to fit within 0.0 - 1.0. **Softmax** uses an exponent with number of inputs as powers and gets a squashed value between 0-1 in floating points, which is correlated with discrete probability distribution on predicated over log(predicted) and actual over log(actual).

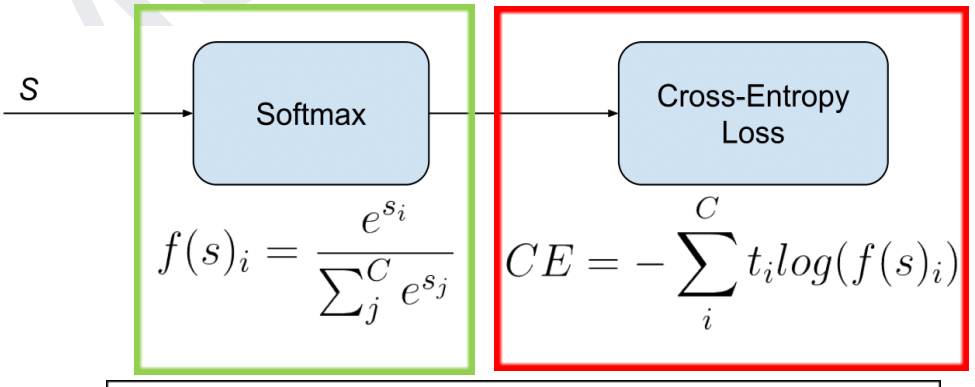
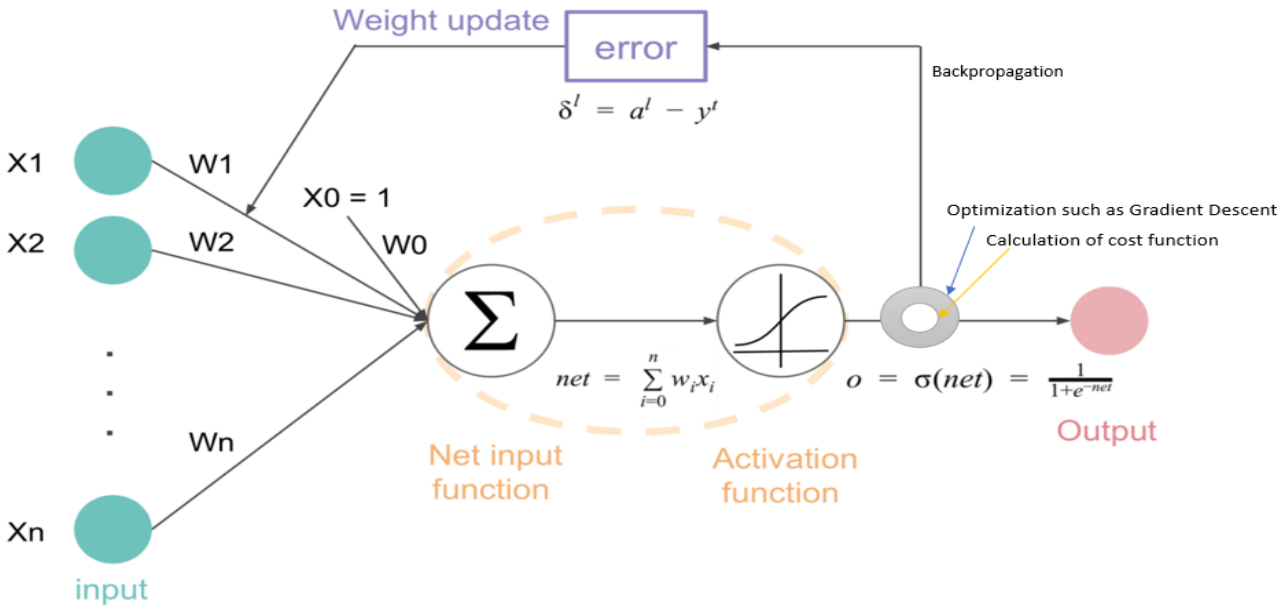


Image 11: [13] (Green) softmax function and (Red) Cross-Entropy Loss

Cross-Entropy Loss uses log on all inputs in sum of all final features (most output is based on label amount), Then calculates a Cost of all y_predicated over y_actual data. For back propagation, Cross-Entropy method helps to find derivatives using exponent, this is due to natural log value can be derived from exponent and vice versa for softmax.

3.3.3 Back Propagation

Image 12: [14] The backprop method (chain rule)



Back propagation is a method of taking past information to understand the slope better to determine best derivatives from the current positions which can help to descend the slope to the minima. More iterations are done, which leads to global minima.

Propagating backwards from cost function requires to reverse the way cost function was calculated, the derivative is small value from the previous information which leads to current slope direction, this can help to update the weights based on the derivative of cost function/last layer with respect to weights and bias. If the cost function result is still high after derivatives on x1w1 are calculated it can take another input or propagate further back to the previous layer (if applicable) and find an effective cost reducing weighted input.

This can be done repetitively for every variable from previous layer weights, inputs, bias and before activation function. This helps to further investigate on a nudge to weights on for example w1 is higher then w5, this given an insight of: w5 has stronger connection already, w5 is not required as much w1 or even more interesting is more iterations required to assign new weight based on new input values. This can change the priority of the neuron's weight depending on the derivatives with respect to weights and biases for a particular layer.

3.3.4 Derivatives

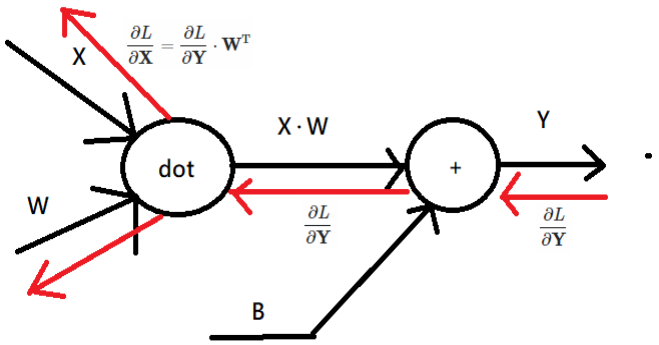
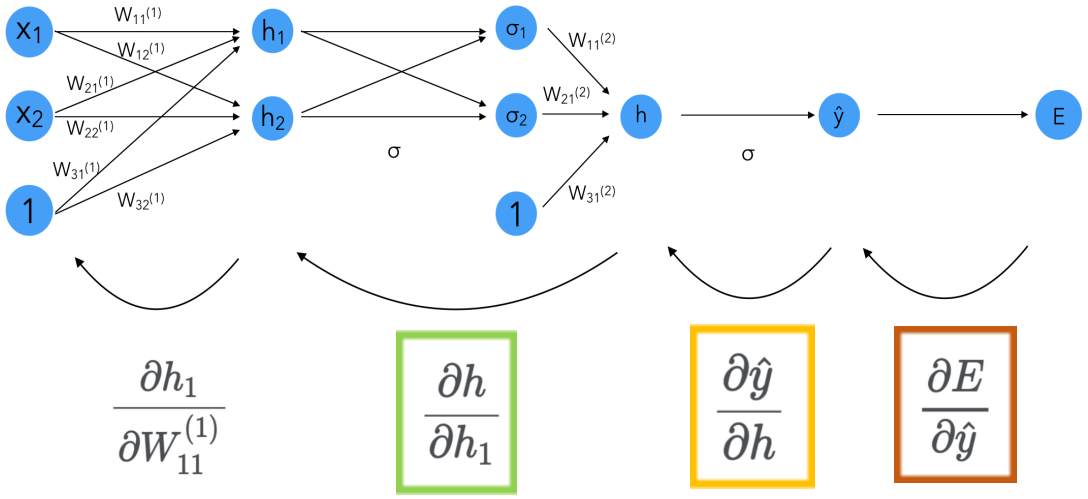


Image 12: [15] The backprop method (chain rule)

Derivatives help to propagate backwards which gets a very small value that can be positive or negative, in this case it is used to get negative in order for reaching the local minima. So knowing what next will be showing either path is much steeper. Having the derivatives is really helpful, but when compared with many samples it will be time taking. Using Scholastic Gradient Descent will return from a random batch of samples, these can be summed to result much faster descents instead of slow descents from individual samples.

Image 13: [16] (Red) derivative h1 respect to weight, (Green) derivative of hypothesis with respect to



3.3.5 Additional features for Neural Network

The method that will be used in this project is a Neural network, it requires 42 inputs in the input layer, 1 hidden layer and 1 output layer with a final layer. As described in 2.2, the

summing function and activation function do play the key role of 'firing the neuron' (after weights and bias have been correctly placed), section 3.2.1.

This also leads to a bottleneck in performance, through in identification but Hidden layers are black boxed functions (unknown actions happens which is the core of all Neural Network, in this case use of ReLU as the activation function), if it was to run in real network then the convention network will be better due to real time analysis. As Explained in chapter 1 1.6 (Methods from best hypothesis which consists of feature selection, generalisation error, error rate, entropy and information gain). These are the key to improve the neural network after or before hidden layers, this can possibly provide better classification or reduced loss some due to the methods followed above. Currently run along with signature and activity based IDS.

3.3.5.1 Feature selection

Feature selection, used to classify by reducing dimensional complexity to remove features that are unnecessary or contain 0s as inputs. Which makes no change when processing the data in a Neural network. This is done via block boxed activation functions. These can help to improve the classification, especially in specific single classification. In this case is it links to Image 6, Section 3.2.1

3.3.5.2 Generalisation error

Generalisation error used to help the network to not optimise overtime with one dataset but give a reasonable output based on epoch, iterations and batch size (determines the weights that may be used).

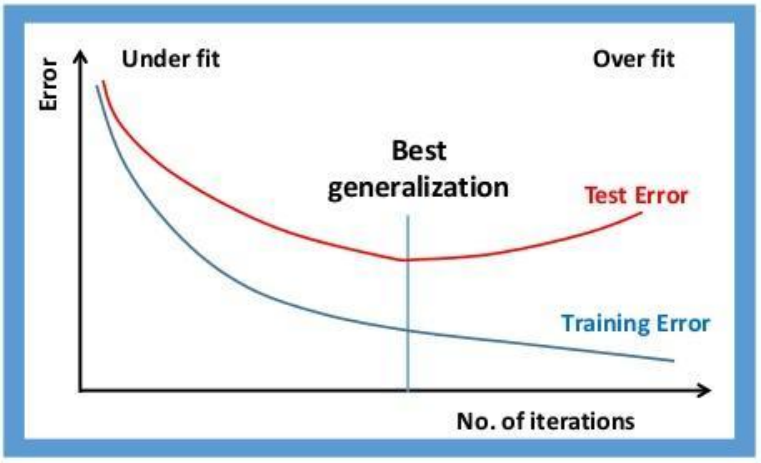


Image 14: [17] A visual representation of generalization

3.3.5.3 Error Rate

Error Rate is used for understanding the total error rate, which is used to determine if the current method used has affected the cost function to increase or decrease. This helps to understand that change can scale due to learning rate and iteration (cycle per input

samples). Error rate also helps in some misclassification in accuracy. Error rate shows issues which are naturally targeted to minimise in Neural networks cost function and back propagation. Image 14 shows test error (as error rate)

3.3.5.4 Entropy

Entropy, measuring uncertainty of random variables. Thus leads to what information can be gained and if the information is any use to use on the next run. In KDD_CUP99 datasets, the requirement of a probability is higher, due to what will be the 1 input (under classification of 23 labelled outputs only one to be selected), which leads to see all the outputs from the hidden layer to see the 'information gains'. If there is high error, then derivatives respect to weights and bias which has slight or major change based on the strength of the connection from a(l-1) to a(l) with weight(jk)L. Image **X** below will given an example of this core functionality and how it can lead to various changes in learning new information, limiting number of batch-size and learning rate can bring an good learning to reach local minima before over fitting or get more errors.

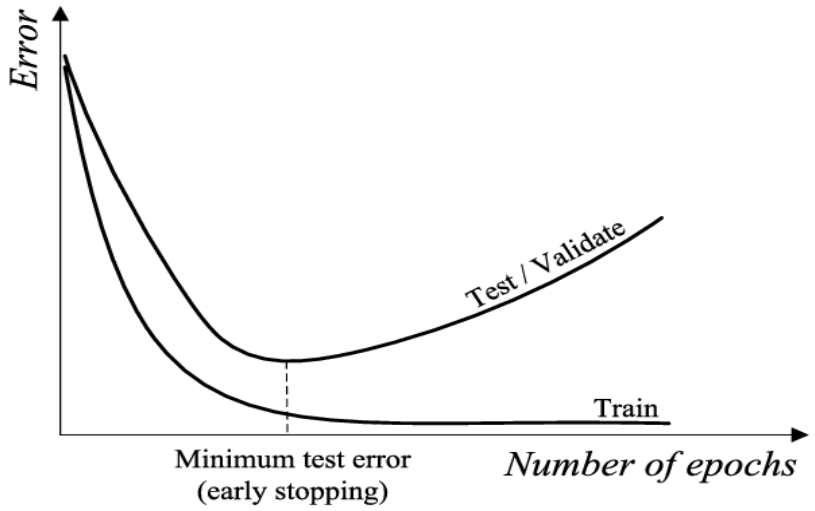


Image 15: [18] A visual representation of Early Stopping

3.3.5.5 Reason for implementation for extra parameters

The errors are generated based on larger inputs which leads to not reaching the local minima by overstepping. This case can also be applied to larger learning steps which can also prevent from reaching local minima.

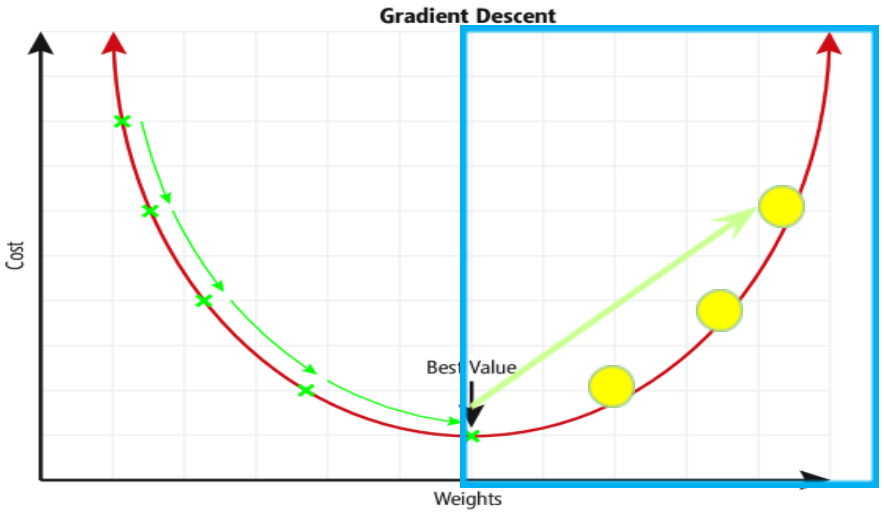


Image 16: [19] A visual representation of overstepping when reaching the yellow circle within the blue square.

For errors to be minimised there is also a desired error to prevent from reaching towards the maximum iteration value. This means the more the network is trained, the likelihood of error is higher due to optimisation to the one dataset (training set). Then under test set (validation set) it will have a large impact in performance even if inputs have been scaled to a certain degree for better output or learning rate have been reduced when using large inputs, to compensate for scaler used for inputs.

3.4 Implementation process of Neural Network and other developments

3.4.1 Redevelopments of Approaching the AI-IDS

3.4.1.1 Stage 1

Named as 'NeuNetModule_Stage1.py'

Stage 1 was preparation for ground up approach on Neural Network(Data parsing to Feed forward to back propagation(calculus, derivatives and Scholastic Gradient Descent)), section 3.3 contains neural network.The amount of ground up work was based on multiple resources which lead to entire Neural network and most important functions embedded (referring to file called 'NeuNetModule.py'); -

```
# print(DaSt.isnull().sum())
NeuNetModule_Stage1.main_NN(DaSt, labels, list2narray)
# clf = MLPClassifier(solver='sgd', alpha=1e-5, hidden_layer_sizes=(5, 2), random_state=1)

inp_tra, inp_test, oup_tra, oup_test = train_test_split(DaSt, labels, test_size=0.20)
```

Ai_IDS_Stage3

```
/usr/bin/python3.8 /home/mukesh/PycharmProjects/AI-IDS/Ai_IDS_Stage3.py
Initialing Weights

Test the array
Traceback (most recent call last):
  File "/home/mukesh/PycharmProjects/AI-IDS/Ai_IDS_Stage3.py", line 62, in <module>
    NeuNetModule_Stage1.main_NN(DaSt, labels, list2narray)
  File "/home/mukesh/PycharmProjects/AI-IDS/NeuNetModule_Stage1.py", line 444, in main_NN
    start(Input, WeiArr_IntP_Hid_fwd, initial_Layer, BIAS, WeiArr_Hid_OutP_fwd, Labels_pred_y, Labels_in_Data,
  File "/home/mukesh/PycharmProjects/AI-IDS/NeuNetModule_Stage1.py", line 393, in start
    for se in range(len(Input[:, 0])):
  File "/home/mukesh/.local/lib/python3.8/site-packages/pandas/core/frame.py", line 2902, in __getitem__
    indexer = self.columns.get_loc(key)
  File "/home/mukesh/.local/lib/python3.8/site-packages/pandas/core/indexes/base.py", line 2891, in get_loc
    return self._engine.get_loc(casted_key)
  File "pandas/_libs/index.pyx", line 70, in pandas._libs.index.IndexEngine.get_loc
  File "pandas/_libs/index.pyx", line 75, in pandas._libs.index.IndexEngine.get_loc
TypeError: '(slice(None, None, None), 0)' is an invalid key
```

Image 17: various runtime errors on casting floating points

In this Stage, the most important introduction was the piping or moving arrays to functions with for loops seems workable but due to run time errors on data types it is an unforeseen strategy which required new methods and approaches.

3.4.1.2 Stage 2

Named as ‘NuNeModFeFo_Stage2.py’

Stage 2 was directly influenced from Stage 1 but underwent many changes (referring to file called 'NuNeModFeFo_Stage2.py'), the main point of change from Stage 1 to 2 is numpy arrays are changed to pandas DataFrame.

Numpy arrays have raw processing power and manipulation of given data array, whereas Panda DataFrame has higher accuracy on locating data from a large collection data types, Helps to address by rows and columns index (string or integer), also cross compatible with numpy array to use sum,

```
num = numpy.array(Rl_ot['col'].values)
To = Inp_Wei(num, Weights_hid, BIAS)
RLu_Inp = pd.DataFrame({'In': To})
```

Image 18: Numpy array and pandas Dataframe

```
# Start of Input*Weight

def Inp_Wei(Inp, Wei, bia):
    test = Inp.dot(Wei)
    dft = test + bia
    return dft
```

Image 18: dot product(to multiple matrices)

```
def softMax_act(Pred):
    x = Pred.values - (numpy.max(Pred.values))
    expon = numpy.exp(x)
    arr = []
    for c in range(expon.shape[0]):
        tep = float(expon[c])

        expon2 = tep / numpy.sum(expon)
        arr.append(float(expon2))

    softMax_activ = pd.DataFrame({'res': arr})

    return softMax_activ
```

Image 19: natural log and exponents

natural log and exponents (log and exponents were useful but lead to runtime error with NaN (unstable softmax issue*).

The newer features were introduced for better classifications on non-linear classification (multiple outputs - single classification issues) softmax,

probability distribution and Cross-Entropy(log loss).

Softmax applies all inputs with ‘soft’ max all over 23 outputs which should sum up to 1.0, changing to probability values for identifying attack vectors, probability distribution to determine the likelihood of a variable may be within a specific range.

This can help to set Cross-entropy with natural log for multiple outputs - single classification from the branch of non-linear regression. As one of the main example is to address entropy (measure of uncertainty of assigning to random variables), is getting discrete probability distribution on sum for each cost function calculation. (1.0 - true labels, true labels) which results for each calculation between softmax and cross-entropy.

This helps to make the hypothesis fit the data set, stationary remaining data in between inputs, this is sampled with distribution to make connection from past to future, else the probability of getting an attack vector can be overlooked which results in a biassed IDS system.

Also defining error rate of that previous method's mistakes and making changes in labelled data, in this case a training and test set is derived from the training set. This is where ‘holdout cross-validation’ or ‘k-Fold cross-validation’

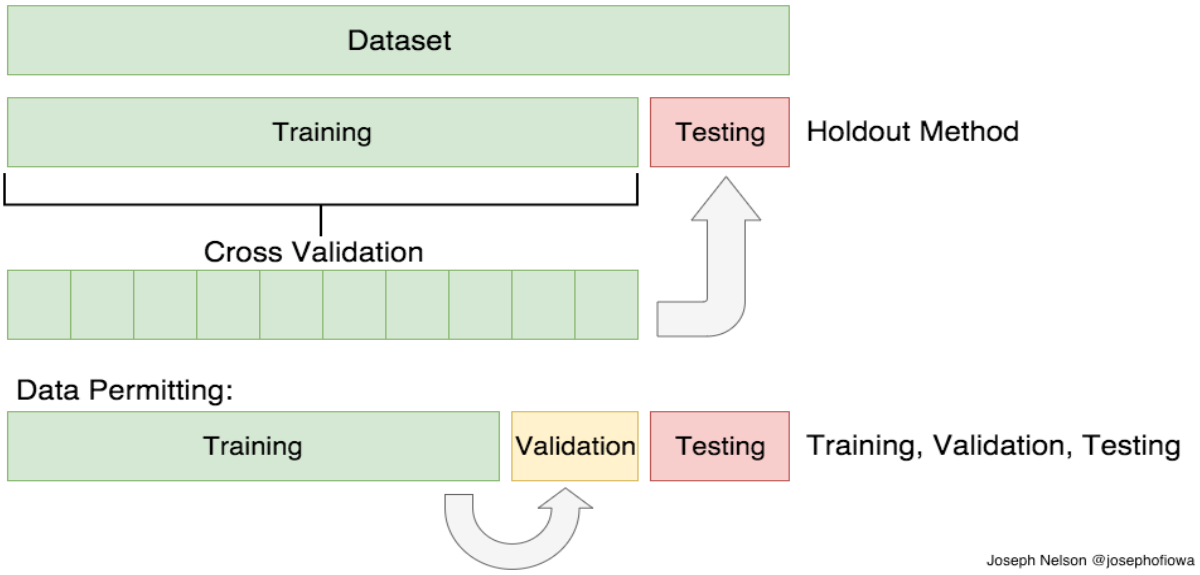


Image 20: [20] natural log and exponents

Both concepts involve using a divided training set and test set. Using one random set as evaluation set and in-between will consist of a validation set for using hyper parameters tuning and classification adjustments. This also results in cross validation which leads to a longer duration data being permitted for taking stationary in account before testing with classifiers.

The other issues to be addressed is peeking when running training and test classifications. Separating these methods is best advised due to peeking on a test set which alters the error rate for the best hypothesis. This seems biassed which results unfit to current hypothesis in real time IDS scanning for attack vectors. Which in turn cannot fix and optimise but run with learned configuration from previous data.

Results are not visible due to run time errors, another piping error similar to Stage 1.

```
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
arr2 (230,)
arr3 (230,)
Traceback (most recent call last):
  File "/home/mukesh/PycharmProjects/AI-IDS/Ai_IDS_Stage3.py", line 63, in <module>
    NuNeModFeFo_Stage2.start(DaSt, list2narray, labels)
  File "/home/mukesh/PycharmProjects/AI-IDS/NuNeModFeFo_Stage2.py", line 178, in start
    DelWeig, DelAl1, DelBias = BackPro_DevrAlay(fin_Cst, Rl_ot, Ot_Lbl, Weights_hid.T, BIAS)
  File "/home/mukesh/PycharmProjects/AI-IDS/NuNeModFeFo_Stage2.py", line 241, in BackPro_DevrAlay
    BpArr02 = numpy.dot(arr1['a'], arr2['b'], arr4['d'])
IndexError: only integers, slices (`:`), ellipsis (`...`), numpy.newaxis (`None`) and integer or boolean arrays are valid indices
```

Image 21: Run time error in Stage 2

The generated runtime error are after many finished functions but due to piping numpy or pandas within for loop or function, they have changed array and data types that does not allow back prop

3.4.1.3 Final: Stage 3

Stage 3 Neural Networks, steps to provide cleaner dataset that reduces replication of same data samples from classified repeatedly to reduce the effect of over fitting/generalisation error. Example in Section 3.2.2 Data cleaning.

Scikit learn library has been used as main functions of all results scoring, on Neural Network function by meeting several changes that combine with other methods used.

```
import csv
from datetime import datetime
import pandas as pd
import numpy
import matplotlib.pyplot as plt
import NuNeModFeFo_Stage2
from sklearn.neural_network import MLPClassifier
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import balanced_accuracy_score
import scikitplot as skP
```

Image 22: Libraries used by Stage 3

Ensemble learning is not based on algorithms but the back end core methods used. Although they all derived from similar backgrounds it is important to combine and investigate a better combination than the standard variation. Neural Networks in Scikit learn required prerequisites to match the objectives to support the hypothesis.

Currently the Training set ‘kddcup.data_10_percent_corrected.csv’ have been tested by splitting in to training(80%) and test sets (20%)

```
# clf = MLPClassifier(solver='sgd', alpha=1e-5, hidden_layer_sizes=(5, 2), random_state=1
inp_tra, inp_test, oup_tra, oup_test = train_test_split(DaSt, labels, test_size=0.20)
```

Image 21: Stage 3 Data split as training and validation set

3.4.1.4 Summarising and selecting results method

For these 3 Stages, classification report with confusion matrices and recall score have been mainly focused instead of accuracy score, because accuracy scores are an overall average on finding as much true values as possible (which does not correlate with outputs predictions), if matched up with lower iterations and batch numbers and inaccurate scoring focuses on classification to y label.

This method is to test the settings with high error rate and gradually find an optimum spot for better generalisation for the test. The average shape to consider is con-curve(U-shape based reaching towards minima in y axis) (local or global minima), this is the aim for gradient descent but taking small number of steps to descent is computationally expensive, thus SGD (Scholastic Gradient Descent) taking random small batches and calculate Gradient descent on each batch which is finally calculated together, this takes much larger steps to reach global minima much quicker.

```
print('MLP classifier Normal in progress....')
NN_normal = MLPClassifier(hidden_layer_sizes=(41, 180, 23), max_iter=500, solver='sgd')
NN_normal.fit(inp_tra, oup_tra)
```

This occurs when data have very small values (section 2.4, image **x**) as outputs which need support to bring the true data results in-between 0-100%. In the results gathered below, some results have slight changes after some changes and additional features have been added in order to bring the results to higher standards.

Data fitting respect to label data is aligned and ordered for final y_predication to validate the accuracy of each input. With this information can lead to verify the classification and provide a precision, recall, f1-score and support.

```
#ROC curve
NN_EarlyStpTest = NN_EarlyStp
NN_EarlyStpTest.fit(inp_test, oup_test)
NN_EarlyStp.fit(inp_tra, oup_tra)
```

As the other related works which consist of a line graph with number of accuracy over duration of test. It will have a large increase, which can help to prove hypotheses but the missing section is the balanced accuracy score. Which focuses on recall (the correct classifications over true positives, in section 3.3.6). This evaluation can provide ground truth behind the classifiers and aim for generalised classifications is for average results over higher accuracy based results leads to incorrect evaluations

3.4.2 Confusion Matrix

Confusion Matrix is used for finding relationships with correct and incorrect predictions from previous training sets. As an IDS coming can predict correct packets from a network as not, which leads to incorrect predictions. Procedure of AI applications is to train and test with a number of iterations to get accuracy close to the original results. TP and TN are targets for our classifiers to aim for and to avoid FP and FN, this will require the overall accuracy in order to get overview of the current datasets correctness (accuracy can be a correct or incorrect based on dataset).

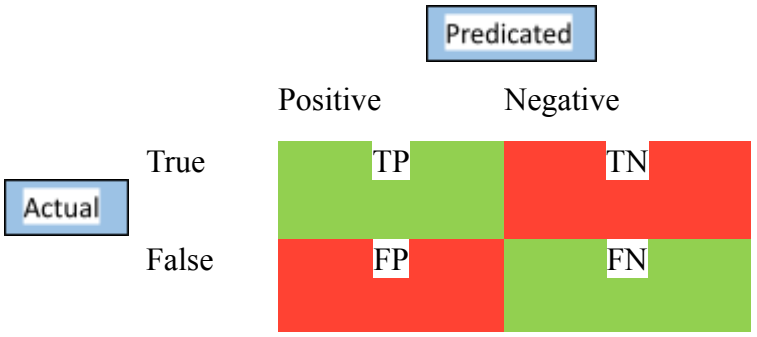


Figure 3: Confusion matrix table for Actual results and predicated

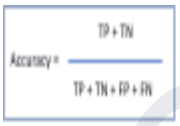


Figure 4: Accuracy metric

Accuracy can help to get the calculation of all instances of actual results and predictions from a classifier and outputs a numeric percentage of accuracy.

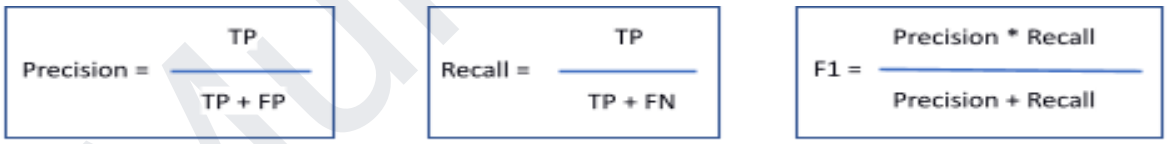


Figure 5: Precision, Recall and F1 score metric

Using accuracy can give many incorrect predictions if data has defects (nulls, duplicates and multiple data types), using precision, recall and f1 can help to get higher TPs (True in observation and correct prediction) and lower FPs (False in observation and incorrect prediction). F1 score gives an average value between precision and recall, which can be used for classification data's 'true positives'.

3.4.2 Example of Confusion matrix and importance

An example of real classification of Confusion matrix in python, figure 6 has 23 results which is for each intrusion type and the recurrences label ‘normal’ and ‘neptune’ which are populated and sorted in ascending order to compare with confusion matrix in figure 7. Figure 6 uses Random Forest Classifier from Scikit learn. When comparing figure 6 on figure 2 (t-SNE) that relationship can be seen when separating along 23 arrays and using precision, recall and f1-score metric to get the average which is around 77%.

Report for Classification

	precision	recall	f1-score	support
0	1.00	1.00	1.00	17594
1	0.71	0.83	0.77	6
2	1.00	0.00	0.00	2
3	1.00	0.00	0.00	2
4	1.00	1.00	1.00	10315
5	1.00	1.00	1.00	137
6	1.00	1.00	1.00	8
7	0.98	1.00	0.99	40
8	1.00	1.00	1.00	181
9	1.00	1.00	1.00	83
10	0.99	1.00	1.00	135
11	1.00	1.00	1.00	4
13	1.00	1.00	1.00	189
14	1.00	0.67	0.80	3
15	1.00	0.97	0.98	187
17	0.97	0.97	0.97	39
18	1.00	0.00	0.00	1
19	1.00	0.67	0.80	3
20	0.99	1.00	0.99	185
22	1.00	0.00	0.00	4
accuracy			1.00	29118
macro avg	0.98	0.76	0.77	29118
weighted avg	1.00	1.00	1.00	29118

Figure 6: Example of Confusion matrix classification (RandomForestClassifier)

As described Figure 6 uses Random Forest as one of the best classifications in ensemble methods for Supervised learning methods. It uses the Voting majority ensemble method to determine the most voted classification which is appended as results for the last classification. More will be discussed.

The results from the Confusion matrix are transposed and used for testing F1-score. The training set has been split to training and test sets for an evaluation of the learning algorithm. Training sets provide information hypothesis h and test sets to measure accuracy (split (0.8|0.2) 80% for training sets and 20% for test sets). These can provide the actual correct predictions to make sure the algorithm implemented is correct. If issues arise then it can be understood that there are measures to be taken such as generalisation error can reduce the better prediction of a specific, lower-in-count attack vector.

Error and loss. Error can be minimised but loss can be increased as if False Positive, True Negatives have classified as the number of iteration go higher, this can be considered as loss.

This can result from noise, outliers or arbitrarily large iterations on same dataset samples can result in overfitting which can raise loss and fall in accuracy.

3.4.3 Review of Implementation of Stage 3

As a preparation of final Results, it is necessary to go through the selected settings for making sure the test is unbiased. The following of 3 tests with hyper-parameters, which are;

MLP normal Hyper-parameters	Values for hyper-parameter	Description
Hidden layer sizes	(41, 180, 23)	3 Layers in total, 41 layers to match 41 inputs, then 180 neurons for black boxed for calculation (activation function) and 23 layers before finally output
maximum iteration	500	Maximum iteration also referred to epoch as limited number of iteration per input for reducing cost via back-propagation
solver	: (SGD) Scholastic Gradient Descent	SGD, randomly batched data samples selected during every SGD has been calculated for an gradient descent which is finally calculated after finishing all random batches and results in an array of gradient descent across all batches.summed and added to reach global minima. Difference between local and global minima, local minima is for a particular set of samples whereas global minima is across all samples (data inputs)

All hyper-parameters are same as MLP normal, to keep the results as true as possible and additional one extra hyper-parameter

MLP with Early Stopping Hyper-parameters	Values for hyper-parameter	Description
Early Stopping	True	Early stopping has a significant reduction in time usage, that results in faster results with 1-5% difference in classification

Some are default which are Min tree spawn, max tree spawn which are

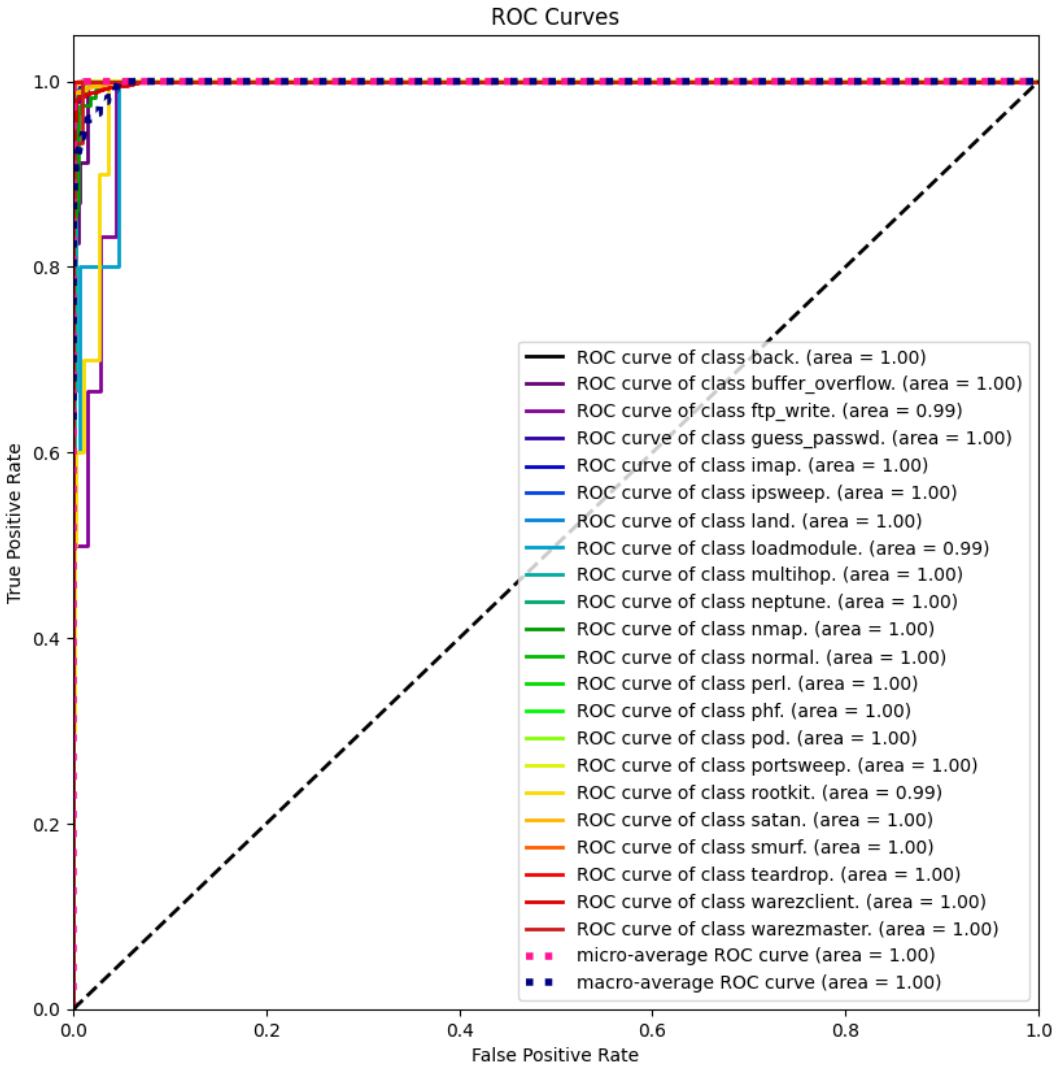
Random Forest Classifier Hyper-parameters	Values for hyper-parameter	Description
Criterion	gini	gini impurity: best picks from random nodes for trees, random features for better classification on overall results. Based on Higher Gini gain = Better split. It helps to evaluate on probability of incorrect classifying inputs from datasets
Maximum Features	default 0 default 1	based on number of features from input dataset as automated

Chapter 4

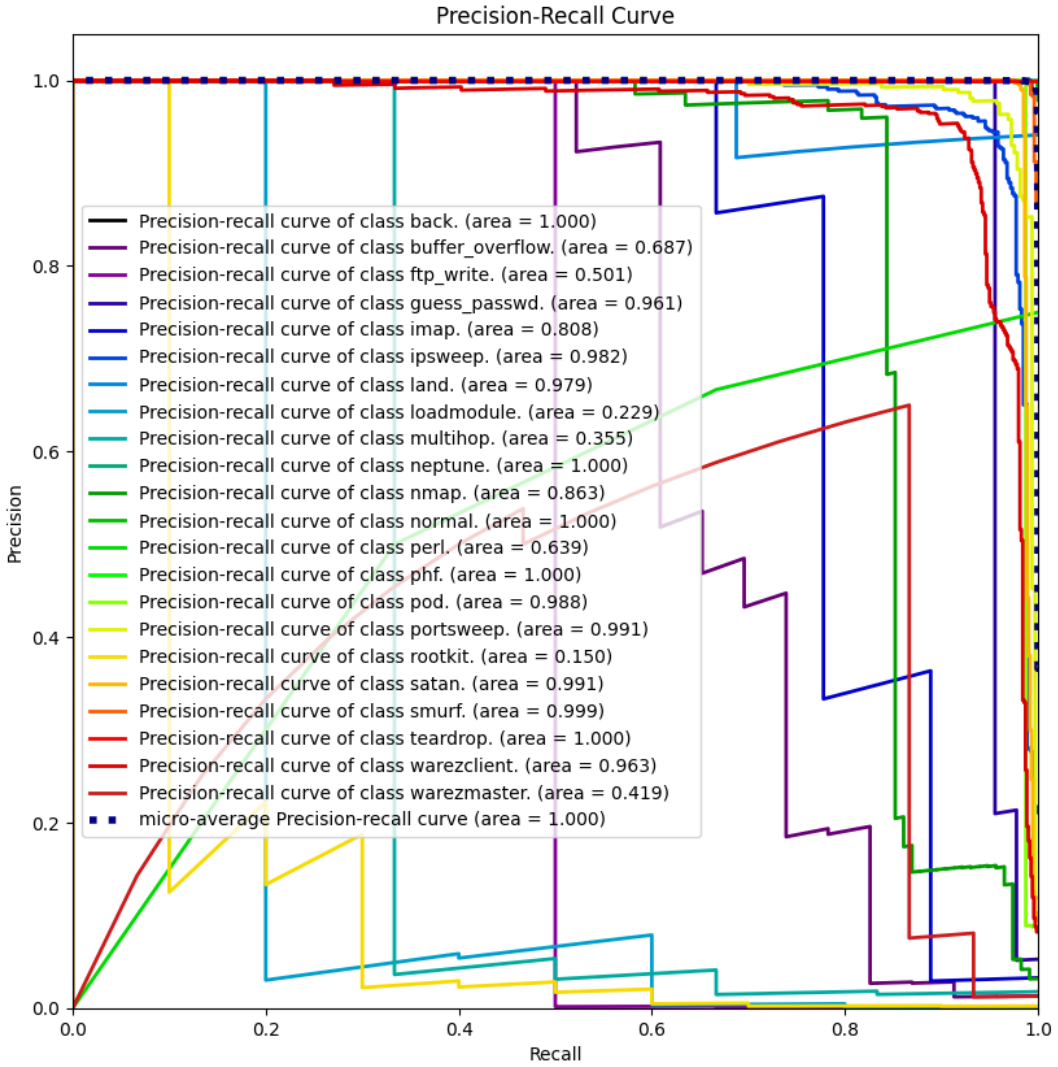
4 Testing Results and Analysis

One of the testing classifiers in an ensemble method is Random Forest, it has been built with many classifiers which results with an appropriate accuracy and possibly provide better true predictions even with some data in corrections included.

For this getting results results, this only displays for MLP with early stopping



The ROC curve shows the cut that precision-recall graph missed, this shows that ROC shows the stationary result of false positive rate in the graph above



The precision recall shows the various classes have ‘stepped’ to reach recall, and precision has always been high which in this case shows that it was a incorrect classification, so when basing together with precision, it shows the 22 classes have be randomly batched and ran for a long time (close to 3-6 minutes)

Firstly the Overall classification results are based on separate classifications that have not passed through each other (unlike an ensemble).

4.1 Results

	Precision Score	Recall Score	F1-Score	Accuracy Score	Balanced Accuracy Score	Duration
Multi-Layer Perceptron	98 %	67 %	67 %	100 %	67 %	0:06:56.480846
MLP w/early Stopping	86 %	67 %	66 %	100 %	67 %	0:02:55.679662
Random Forest Classifier	98 %	98 %	98 %	100 %	98 %	0:00:09.512856

Figure 10: Results printed from Pycharm IDE and output.txt file generated on every run of Ai_IDS_Stage3.py

In recall, early stopping and normal MLP has no difference and several precision scores are much lower. This is simply because of prevent overfitting when iterated up to 500 times on a single input sample when back-propagation is used (250 iteration for both feed forward and back propagation)

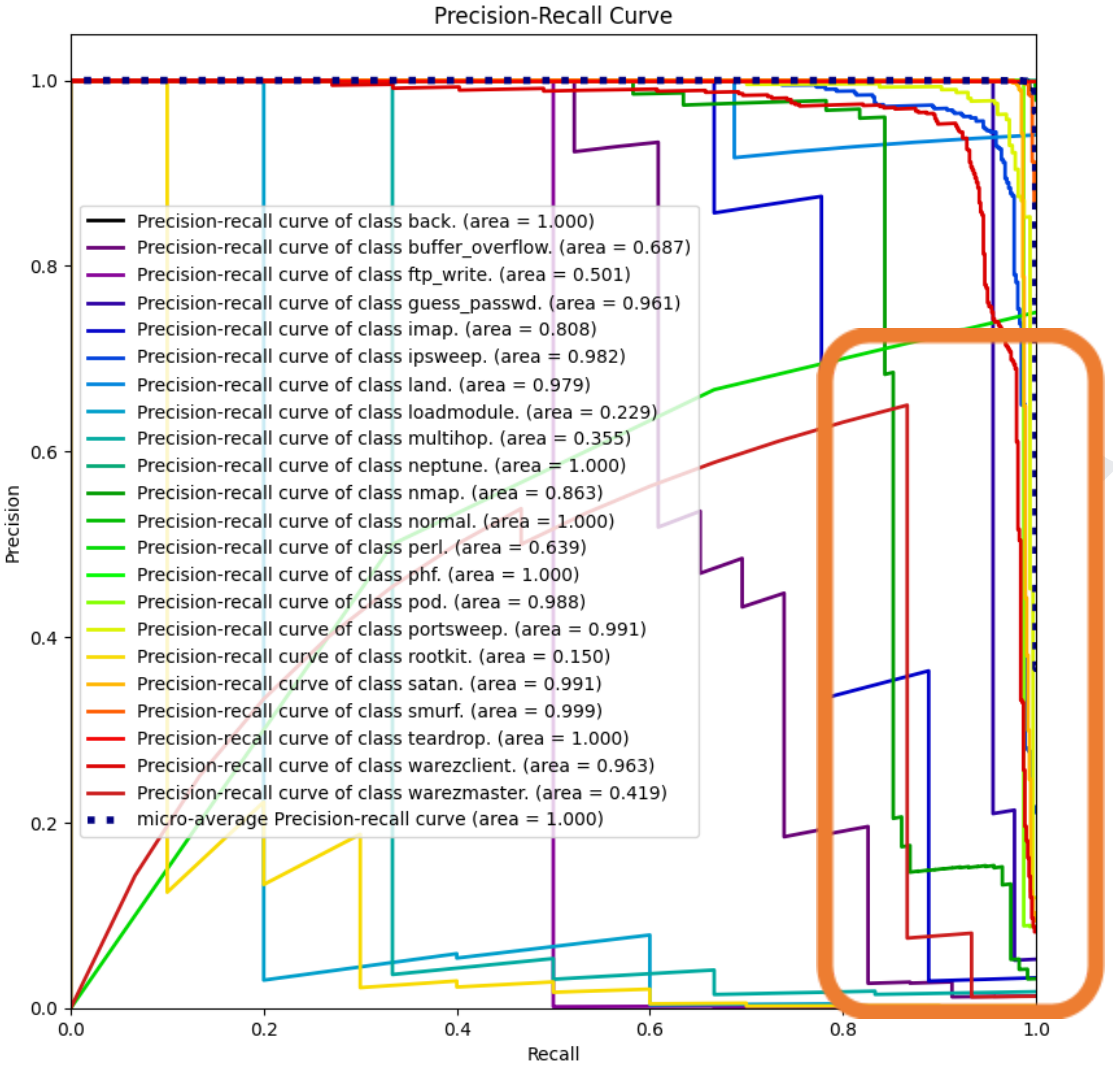
Stage 1 did not generate not get a graph and the values resulted in Null and NaN so these are carried to Stage 2 of developments that contains the features from the previous developments

Stage 2 has one loop which resulted in 23% (0.23) classification rate until backpropagation, which then stopped functioning.

4.2 In-depth analysis of results

MLP normally has several issues that can be addressed based on the results above, Neurons die in-between hidden layers due to ReLU activation function, over-fitting even after scaled inputs and not generalised enough.

- Neurons die in ReLU if they are not activated during training which can result in several neurons not getting values from, which leads to similar data going back and forth.



The brown colour highlighted the dead neurons which dropped and made slight changes from other neurons

If considered all as one variable for MLP as an classifier, it will have a larger impact on the overall classification which has been scaled to compensate for the amount of errors that has been generated. As an example would be scaling inputs that are smaller in value can also result in errors to rise according to the ratio of scale. This can be considered as an non-existing issue in a classification but based on the results it is not possible for a full evaluation.

MLP with Early Stopping has all the critical issues above and it tries to minimise unwanted recursions of1 sample input until maximum iteration. This means to stop when a decried amount of error is reached, at this point it is no longer necessary to continue further. It will cause timing complexity when running in a large training set. This will improve on

generalisation (may have slight change than MLP, moving from generalisation to over fitting). But early stopping provides similar results to MLP with 75% reduced timing to get the results.

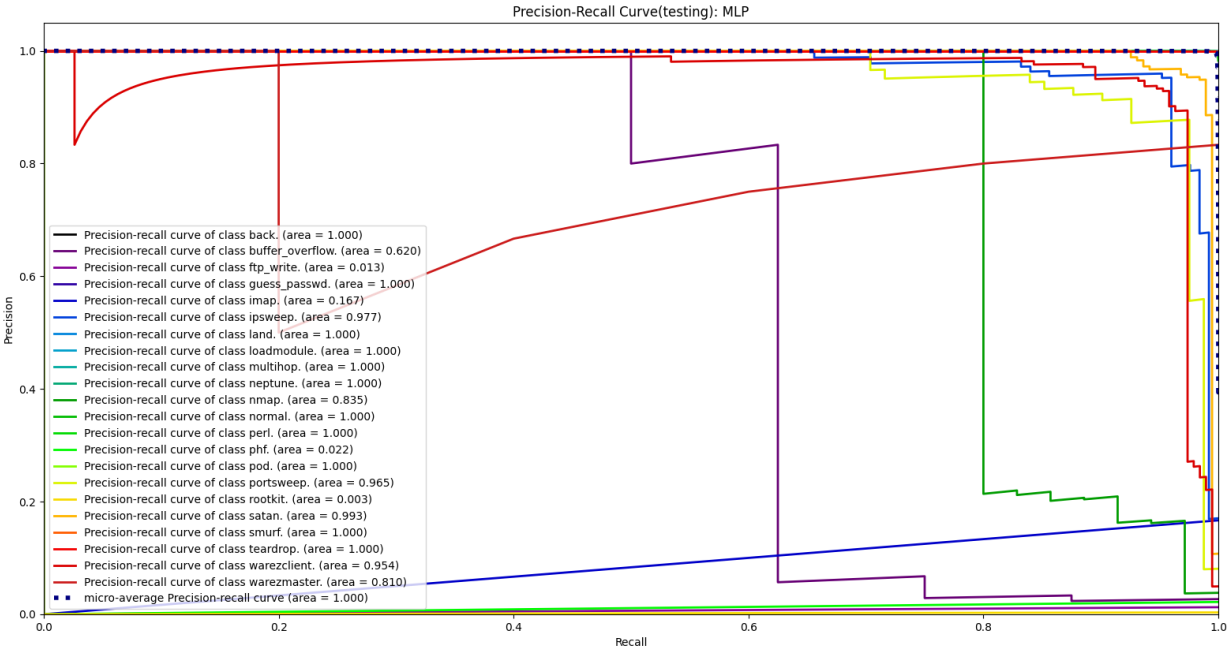
4.2.1 Why do the results vary between each other?

This can be seen as why there is higher classification on Random Forest classifiers and lower on MLP which ran 5 times longer in duration, In theory this ideology should be correlating between the amount of time put can increase in accuracy. This also leads to generalisation error in data sets or under-fitting causes generational error, in this case both types of error are not good which will result in poor classification either way (during validation sets).

4.2.2 Testing Results

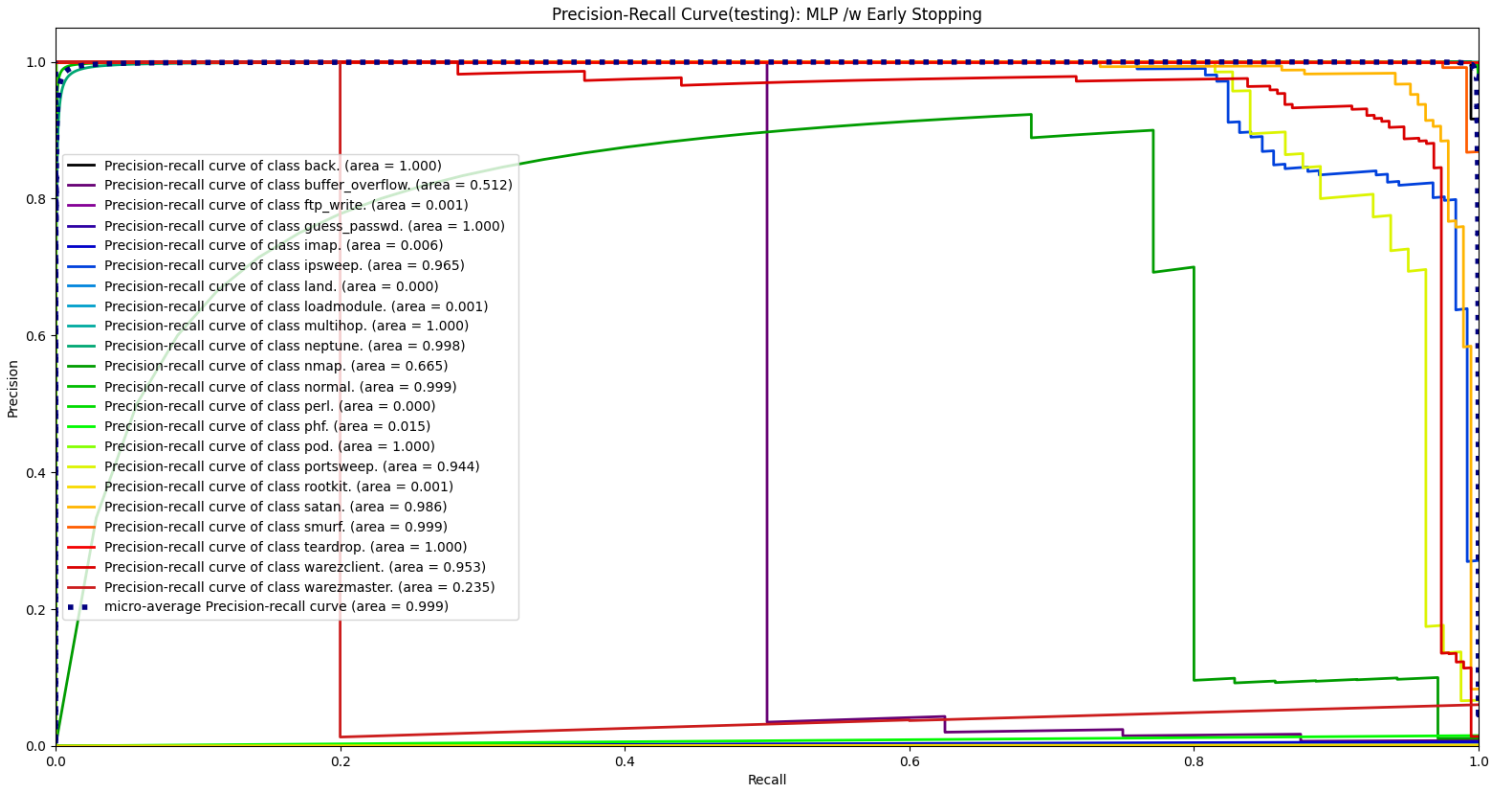
All other Test results with precision recall displayed here

MLP - precision recall



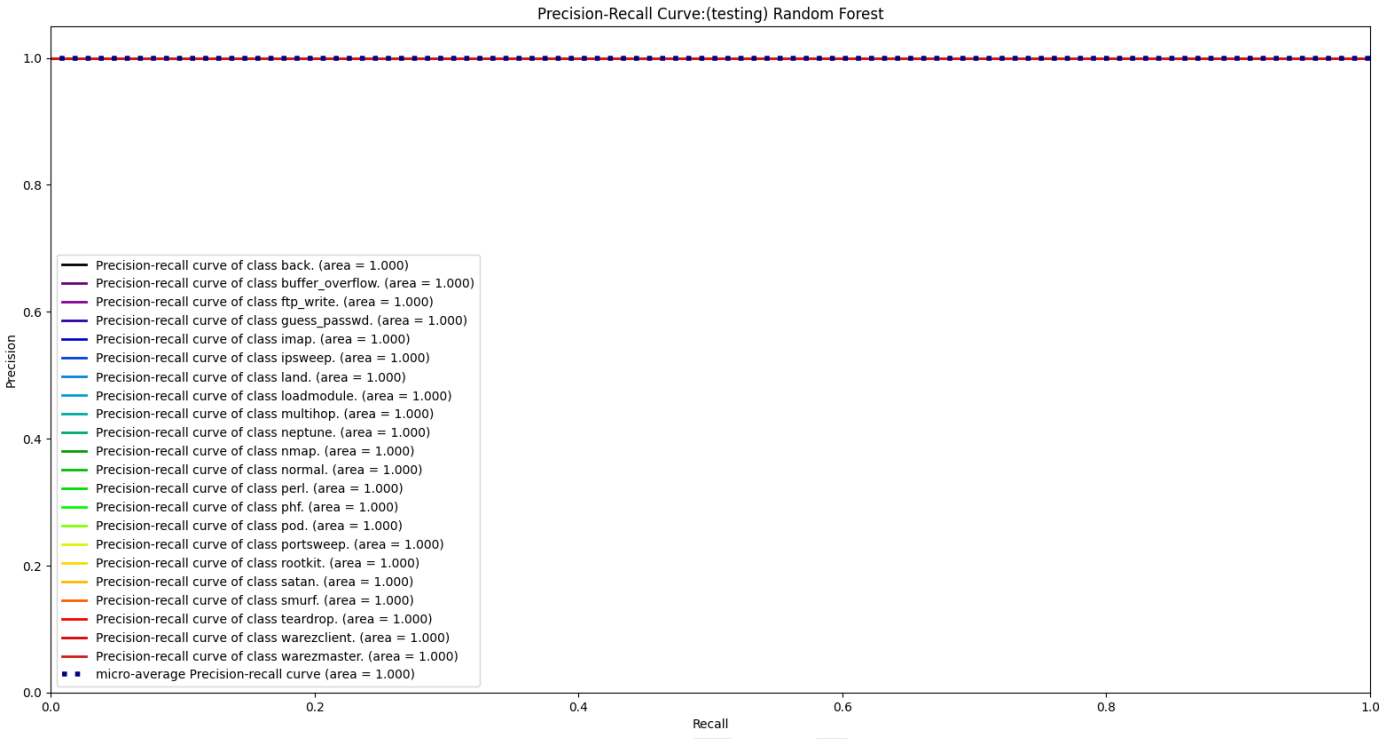
This has the same hyper parameters as MLP with early stopping down below. The results above state that testing has proven difficult had precision reduced in-between the validation procedure.

MLP /w early stopping



MLP with early stopping provided much in-depth detail in how the learning progress, due to some redundant data having been removed, has some low results such as rootkits. This indicates that the amount of data after data has been cleaned is not enough, in order to get high recall and precision at a higher rate. This requires more similar data without nulls which can lead to fixation on the training set.

Random Forest



The precision recall of random forest still shows a strong predictor of 1.0 all over the dataset.

This Finally shows the need for a dataset for all classifiers, although 500,000 was cleaned (removal of nulls and duplicates), which led to 150,000. This still is not enough for proving more datasets which requires more computation power and space (leads to AI's infamous time-space complexity issue).

Chapter 5

5 Conclusion and Future Work

5.1 Conclusion

The conclusion from the results above was confirming that AI based IDS are the next solution and it is much better in terms of conventional signature (or anomaly) based IDS. They have the ability to scale into businesses and industries with a certain degree of adaptation. It can also have a sharing ability to provide several new classified attack vectors to be trained and prepared, without much long configurations to patch or prepare for the attack type. For the results it assures the implementation if feature selection, error rate,

generalisation error, entropy and information gain has been implemented as part of additionally rediscovered the existence within Multi Layer Prescription.

There are several checks to be made and take the previous results as a preliminary test for new upcoming developments. Although results satisfy the current hypothesis, it does not fully evaluate as a stand-alone and functionally IDS, that means expecting a recall score of 99% even on test sets (without labels). Minimising close to zero compensation for attack vectors bypass IDS scan

5.1.1 Final Investigation of Varying results

After another set of research and measuring possibilities of error margins within each type of classifiers, the results each have possible errors which can lead to several following predictions.

Firstly confirming that MLP with early stopping proves much better real time training segments on large datasets, reducing unwanted timing complexity when finding attack vectors on a network. Although early stopping does provide around about 1 minute of operation time on 150000 dataset samples, this has to considered as an good solution due to the following reasons, computing power has to play a role in process speed but if under a same device with large computing speed differences may determine the classifiers strength of identifying attack vectors. As the results show the time difference between MLP normal and early stopping varies not only time but performance in generalisation, thus reducing or avoiding generalisation error.

	Precision Score	Recall Score	F1-Score	Accuracy Score	Balanced Accuracy Score	Duration
Multi-Layer Perceptron	98 %	67 %	67 %	100 %	67 %	0:06:56.480846
MLP w/early Stopping	86 %	67 %	66 %	100 %	67 %	0:02:55.679662
Random Forest Classifier	98 %	98 %	98 %	100 %	98 %	0:00:09.512856

Alternatively when compared to the time difference between Random Forest and MLP with early stopping nearly one minute of operation time difference, this gives an insight of outstanding results, but Random forest is a bit tedious. This is due to random state of trees and features generated from default setting may present with large time difference and a close-to-error-less classification but coming back to the problem is that AI-based IDS can solve much better than signature based IDS, yes it will have significantly better increase both in identification and quicker time. Is the method chosen the right classifier to solve the Anomalies problem, not simply due to anomalies having several instances.

This requires many dedicated systems to identify each individual attack vector, this was previously thought of as computationally complex due to time of processing and space of

scanning or processing concurrently. The results only state the scikit-learn libraries used, but it has a very prominent back-end which made half an hour worth of processing between 5-10 minutes by scikit-learn. The developers utilised parallel and concurrent processes with multiple threads for multiple batches (based on SGD) aligned with CPU cores (logical and physical) and can handle and provide with a classification that is close to satisfactory in terms of overall performance.

The precision recall line graph for all classes of classifications proves of not enough dataset inputs to justify the output classes enough, which results in a low result in precision-recall. Then it also requires many complex datasets which have much more than 0s as inputs. KDD_CUP99 has proven that the dataset is not enough for best classification results. Thus, obtaining much better systems and datasets for research are necessary.

5.2 Future Work

Back in Session 2.4.1, one question of separating normal and anomalies was taken as a through process question, what was not added and later rediscovered was, what if Normal inputs have classified as individual activities? This goes much in-depth than labelling 'normal' in traffic but especially the time of normal traffic and when certain normal traffic will be present.

Then this leads to the next chained issue on over fitting is not an option but using multi-class to classify individual 'Normal' activities will provide much better understanding of the network, this can allow all abnormal traffic flagged as 'anomalies'.

This finally leads to one bigger issue of classification for anomalies, how can system administrators block, remove or prevent threats by knowing types of attacks occurring from detecting via IDS. As an example of criticising on the datasets, it can be theoretically thought as using two separate classifiers which can be more specific on working on a particular feature then generalising over multiple features. This creates probability distribution over normal and anomalies, which can then share the same ground via ensemble methods like weighted majority.

Current issues and future issues, False positives and True negatives are one of biggest problems to be dealt with when scanning for attack vectors. If the recorded attack vector was wrong on a AI-IDS (example for MLP with Early Stopping), it will give wrong attack vectors which can lead to incorrect configurations.This will predict normal as well as False Positives (Actual attack vectors classified as Positives), this can lead to not enough generalised AI-IDS. But the real issue relies on tackling the problem from a different point of view such as individual classifiers which can be combined with another set of classifiers.

From the overview of future work to focus on, using over fitting for individual features can result with stronger predictions, which minimises marginal errors compared with current generalised AI based IDS that includes both Normal and Attacks vectors under one dataset.

References

[1] Stuart Russell and Peter Norvig. 2009. Artificial Intelligence: A Modern Approach (3rd. ed.). Prentice Hall Press, USA.

[2]Team, T., 2020. The History Of Intrusion Detection Systems (IDS) - Part 1 | Threat Stack. [online] Threat Stack. Available at: <https://www.threatstack.com/blog/the-history-of-intrusion-detection-systems-ids-part-1> [Accessed 2 October 2020].

[3] Bolón-Canedo, V. and Alonso-Betanzos, A., 2019. Ensembles for feature selection: A review and future trends. Information Fusion, 52, pp.1-12.

[4]Vanerio, J. and Casas, P., 2017. Ensemble-learning Approaches for Network Security and Anomaly Detection. Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks - Big-DAMA '17,.

[5]Sreenath, M. and Udhayan, J., 2015. Intrusion detection system using Bagging Ensemble Selection. 2015 IEEE International Conference on Engineering and Technology (ICETECH),.

[6]Kdd.ics.uci.edu. 2020. KDD Cup 1999 Data. [online] Available at: <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html> [Accessed 2 October 2020].

[7]Ravipati, R. and Abualkibash, M., 2019. Intrusion Detection System Classification Using Different Machine Learning Algorithms on KDD-99 and NSL-KDD Datasets - A Review Paper. SSRN Electronic Journal,.

[8]External-content.duckduckgo.com. 2020. [online] Available at: <https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fwww.researchgate.net%2Fprofile%2FMurugan_Solaiyappan2%2Fpublication%2F324834395%2Ffigure%2Ffig1%2FAS%3A621000098074624%401525069369493%2FFeed-Forward-Neural-Network-Architecture-22-Non-linear-Auto-Regressive-and-eXogenous-NN.png&f=1&nofb=1> [Accessed 2 October 2020].

[9]Mnemstudio.org. 2020. [online] Available at: <http://mnemstudio.org/ai/nn/images/minima1.gif> [Accessed 2 October 2020].

[10]External-content.duckduckgo.com. 2020. [online] Available at: <https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fcdn-images-1.medium.com%2Fmax%2F800%2F1*g0yxlK8kEBw8uA1f82XQdA.png&f=1&nofb=1> [Accessed 2 October 2020].

[11]External-content.duckduckgo.com. 2020. [online] Available at: <https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fcdn-images-1.medium.com%2Fmax%2F1600%2F1*XxxiA0jJvPrHEJHD4z893g.png&f=1&nofb=1> [Accessed 2 October 2020].

[12]Simplilearn.com. 2020. [online] Available at: <https://www.simplilearn.com/ice9/free_resources_article_thumb/perceptron-with-sigmoid-activation-function.jpg> [Accessed 2 October 2020].

[13]Gombru.github.io. 2020. [online] Available at: <https://gombru.github.io/assets/cross_entropy_loss/softmax_CE_pipeline.png> [Accessed 2 October 2020].

[14]I.stack.imgur.com. 2020. [online] Available at: <https://i.stack.imgur.com/7Ui1C.png> [Accessed 2 October 2020].

[15]External-content.duckduckgo.com. 2020. [online] Available at: <https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fi.stack.imgur.com%2F1tb3D.png&f=1&nofb=1> [Accessed 2 October 2020].

[16]External-content.duckduckgo.com. 2020. [online] Available at: <https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fmiro.medium.com%2Fmax%2F1200%2F1*4G__SV580CxFj78o9yUXuQ.png&f=1&nofb=1> [Accessed 2 October 2020].

[17]Image.slidesharecdn.com. 2020. [online] Available at: <https://image.slidesharecdn.com/microsoftventuresmachinelearning-140509183557-phpapp02/95/intro-to-machine-learning-by-microsoft-ventures-20-638.jpg?cb=1399660754> [Accessed 2 October 2020].

[18] External-content.duckduckgo.com. 2020. [online] Available at: <https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fwww.researchgate.net%2Fprofile%2FJorge_Santos30%2Fpublication%2F37655851%2Ffigure%2Fdownload%2Ffig10%2FAS%3A669570926911493%401536649557125%2FTrain-error-versus-test-error-and-the-early-stopping-rule.png&f=1&nofb=1> [Accessed 2 October 2020].

[19]Docs.microsoft.com. 2020. [online] Available at: <https://docs.microsoft.com/en-us/archive/msdn-magazine/2019/april/images/mt833406.0419_lavigne_figure2_hires(en-us,msdn.10).png> [Accessed 2 October 2020].

[20]External-content.duckduckgo.com. 2020. [online] Available at: <https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fmaelfabien.github.io%2Fassets%2Fimages%2Fnn_23.png&f=1&nofb=1> [Accessed 2 October 2020].