Engineering Resilient Systems (Part 2)

Jonah McElfatrick
1700463

ABSTRACT

Provided with network data for analysis, this paper serves to provide the reader with a recommendation into a possible machine learning classifier that can be used in order to categorize the provided packet data into either an attack type or to classify it as a normal packet. This will be portrayed by discussion of multiple machine learning algorithms alongside their own strengths and weaknesses that would be applied to this scenario. Also outlined in this paper will be the phases of building an appropriate model for this scenario including the different stages of the data pipeline and different evaluation metrices that can be used to test the recommended classifier.

Notation

Included in appendix A is a table including the terminology and notation used throughout this paper.

1 Introduction

Machine learning is the process in which applications are able to learn over time and hence improve their output without the need for additional programming. Machine learning has many real-world applications including image or facial recognition, self-driving cars, stock market predictions and many more. There are many different types of machine learning methods, each of which have their own problem to solve. Some of the main types of machine learning include:

- Supervised Learning
  - Trains to match the inputted data to a desired output based on training input data. Includes labelled data as training data.
- Unsupervised Learning
  - Works independently and discovers patterns in its inputted data by itself. Uses unlabelled data in its input.
- Semi-Supervised Learning
  - Uses a mixture of labelled and unlabelled data together in order to train the model.
- Reinforcement Learning
  - Minimizes the cost of an action to maximize the cumulative reward by using the output of the previous cycle on the next in order to improve results.

Each of these types of machine learning can help to solve different issues. One of these issues being classification of data. Classification is the process of taking an input and applying the correct label to it. An example of this would be to classify emails as ‘spam’ or ‘not spam’. Classification itself has multiple different types including:

- Binary
- Multi-Class
- Multi-Label
- Imbalanced

Each of these different types of classification algorithms provide different outcomes when in use. Binary algorithms class data into one of two classes, multi-class algorithms are able to classify data into one of many output classifications, multi-label algorithms allow for output data to be classified as multiple classifications and imbalanced classification algorithms are where the class distribution of data is not balanced.

2 Background

Machine learning can have many applications, some of these include roles in both the health and security of a computer network. Machine learning can be used for detecting possible stages of failure in a network and when they are likely to occur in order to perform certain actions to help prevent or solve the issue. Other security-based applications include acting as a web application firewall or packet classifier. Web-based machine learning firewalls when compared to standard web-based firewalls offer fewer false positives and an increase in true positives due to the ability to ‘learn’ from the sample data (McCall, 2020). With machine learning becoming more available to the masses through the development of different libraries available for programming languages such as R and Python, more and more applications of machine learning are being developed and tested.

3 Recommendations

The data provided for this scenario is labelled data. Labelled data is sample data with corresponding labels for the each of the data’s attributes. For the data set provided,

to be able to classify the network packet data into the type of attack, a supervised multi-class machine learning algorithm can be used. This allows for the packets to be classified as one of the many attack classes or the 'normal' class. Using this type of algorithms would allow for classification of incoming attacks, which in turn, using a rule-based system, could prevent the incoming attack packets from reaching their destination. Using this form of algorithm would also allow for the inputted data to be classified as one of many classes instead of one of two classes like it would in a binary classification algorithm.
Following are examples of machine learning algorithms that could be used to classify the given network data packets.

K-Nearest Neighbour

KNN works by taking the (K) number of closest neighbouring data points to the query and comparing them to figure out what the most frequent or average label is that should classify the query.
Advantages of K-Nearest Neighbour included:

- Relatively, easy to understand and implement
- Can be used for both classification and regression solutions

Applications of the K-Nearest Neighbour algorithm include facial recognition and recommendation systems such that are used in popular applications such as Netflix and YouTube.

Decision Tree

Decision Tree algorithms work by breaking down the data into branches repeatedly in relation of rules that can be used to classify the data. During this process a top down tree structure is developed with leaf nodes lying out the outside of the branches indicating classes of data. Decision trees have multiple different applications including assessing marketing growth, medical diagnosis assistance, fraud detection and much more.
An example of the structure of a decision tree can be seen in the figure below.

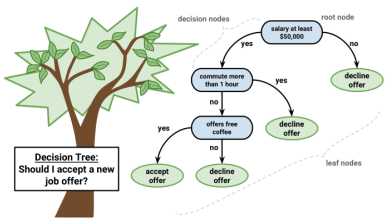


Figure 1: Decision Tree Structure (Saxena, 2017)

Advantages of Decision Trees include:

- Simple decision rules make them easy to understand
- Is robust when encountering edge case data from training data.
- Classification can be quite fast once rules are developed.
- Can be used for both classification and regression solutions.

Random Forest Tree

RFT is a decision tree-based algorithm that creates and implements multiple decision trees concurrently and then merges them together in order to gain a greater accuracy in the classification of the input data. Applications of random forest trees include but are not limited to stock market predictions, customer segmentation and IDS's.
Advantages of Random Forest Trees include:

- Runs efficiently on larger databases
- Can handle a vast amount of input variables
- Can carry out both classification and regression

Artificial Neural Network

ANN's work by simulating the biological network of the human brain. Each network consists of node layers which include an input layer, a number of hidden layers and an output layer. Each node is connected to every node in the next layer, with each node having a specific weight and threshold. This means that if the threshold is surpassed, then the corresponding node is activated. If a neural network has more than one hidden layer in its structure, then it is considered a deep neural network.An example diagram of a neural network can be seen below.

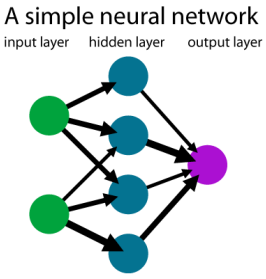


Figure 2: Neural Network (How Do Neural Network Systems Work?, 2020)

Advantages of Neural Networks include:

- Has the ability to store the input inside of the network instead of being stored inside a database.
- Has the ability to parallel process, allowing them to complete multiple tasks at the same time.

ANN's can have a wide variety of real-world applications including facial recognition, object detection, signature identification and autonomous vehicles.

Data Pipeline

In order to build an appropriate machine learning model to classify the given network packet data, all aspects of the data pipeline must be considered. This includes the data pre-processing, training, analysis and communication of results. Following in this paper, is an outline of the main stages in the production of a model.

Data Ingestion/pre-processing

During this step, the incoming data is filtered and cleaned to allow for use in the following stages. This is required due to input data needing to be in a numeric format for the algorithm to be able to process the data. Another reason for filtering and pre-processing of data is that data may possibly be missing before being processed by the algorithm; this has to be resolved before being used as training data.

Analysing the provided training and testing data files, it can be seen that there are 45 different headings for the training and testing data. These headings are included in appendix B. These headers require to be pre-processed and sorted into numerical data. It is required to be converted into numerical data in order for the model to train. This can be carried out by using the factor() function in R. This converts non-numerical data into numerical data by assigning each unique string in the column a number. All other numerical data can be encoded using the as.numeric(as.character()) function in R. Included in appendix C is an example of how the provided input data can be pre-processed in R.

Modelling/Training

Training the model requires inputting training data into the algorithm. The training data in this case is labelled data with the correlating desired output class stored within the training data. During this step, depending on the algorithm chosen, different aspects of the learning cycle can be altered for increased accuracy or speed.

An example, if a neural network was chosen, there are many different hyperparameters that can be altered in order to give better or worse results, faster or slower speeds and more. These attributes of a model are reliant on the type of training input data and how the hyperparameters are configured.

Some of the hyperparameters than can be altered are the hidden layers size, which is the number of neurons withing a single layer, the learning rate and the alpha number, which is the regularisation term. Using a mixture of previous knowledge and trial and error, the accuracy of the model can be increased by tweaking the vast amount of different hyperparameters that are available. Included in appendix D is example code, in Python, of using a neural network to train a model with a set of hyperparameters.

Analysis

For the analysis stage of the pipeline, testing can be carried out in order to verify that the model can categorize the data as required. This can be carried out using the testing data which is in the same format as the training data but has been put aside in the initial steps of the procedure to allow for verification that the model is running correctly once trained. The ratio for splitting the training and testing data is normally an 80/20 split. This allows for the model to use unknown data during testing to verify that it can correctly classify the packet data into their corresponding attack category. Depending on the output of the model, such as mentioned above, hyperparameters may be required to be tweaked in order to improve the accuracy of results.

Communication of Results

In order to communicate the results from the created model, different evaluation metrics can be used. These metrics can portray many different aspects including accuracy, precision and error rates of the trained models. Possible evaluation metrics could include a confusion matrix, a PRC graph, and a ROC graph. Following is a description and example of each of the proposed evaluation metrics.

Confusion Matrix

A confusion matrix allows for visualization of how the model is classifying the inputted data. This is done by plotting the actual classification of the data on the top axis of the matrix with the predicted class down the side axis. This allows for the TP, TN, FP and FN values to be reported on the same graph. This can be done for both binary and multi-class classification algorithms. The example included in figure 3 below shows how a multi-classification confusion matrix would be implemented in a multi-class classification algorithm.

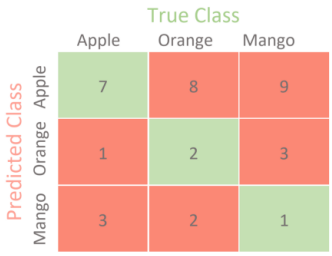


Figure 3: Example Multi-Class Confusion Matrix (Mohajon, 2020)

Taking the example of the Apple class from the above matrix, the values for TP, TN, FP and FN can be calculated as follows:

$$TP = 7$$
$$TN = 2 + 3 + 2 + 1 = 8$$
$$FP = 8 + 9 = 17 FN = 1 + 3 = 4$$

This form of evaluation can be scaled up and down to suit the number of classes that are required by the inputted or outputted data.

In relation to the data provided, the confusion matrix would be more complex and much larger in size due to the number of output classes that are available for the data. However, the main principles and calculations behind it would stay the same. An example of a confusion matrix for classifying network packet data can be seen in appendix E.

PRC graph

A PRC graph is a plot of precision vs recall, where the value for precision represent the positive predictive value and the value for recall correlates to the sensitivity value. To calculate the positive predictive value and the sensitivity, the equations as seen below can be used.

$$Positive\ Predictive\ Value = \frac{TP}{(TP+FP)}$$
$$Sensitivity = \frac{TP}{(TP+FN)}$$

As can be seen in the figure below, as the recall value approaches one, the precision drops.

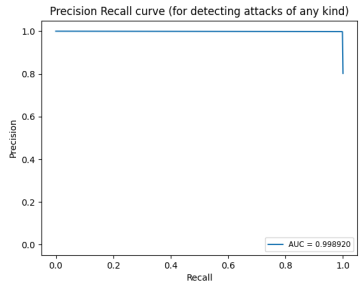


Figure 4: Example PRC graph (Self-Created)

Calculating the AUC represents how well the classifier performs the task given. Taking the AUC and multiplying the value by one hundred, provides a percentage for this value.

ROC graph

An ROC graph is plotted of True Positive Rate (TPR) vs False Positive Rate (FPR). The terms TPR and FPR can be seen defined as below.

$$\textbf{TPR /Recall / Sensitivity} = \frac{\textbf{TP}}{\textbf{TP + FN}}$$

$$\textbf{Specificity} = \frac{\textbf{TN}}{\textbf{TN + FP}}$$

$$\textbf{FPR = 1 - Specificity}$$

$$= \frac{\textbf{FP}}{\textbf{TN + FP}}$$

Figure 2: Defining Terms (Narkhede, 2021)

The AUC represents the ability the model has in distinguishing the different classifications of attack. Due to this, the accuracy value of the output of the model can be portrayed using this method. The value outputted is in the range of 0.00 to 1.00, multiplying this value by 100 gives you the accuracy percentage. This method can also be used to analyse the output of a multi-class model such as the ones recommended above by plotting a curve for each of the individual classifications. An example curve can be seen plotted below in figure 5.

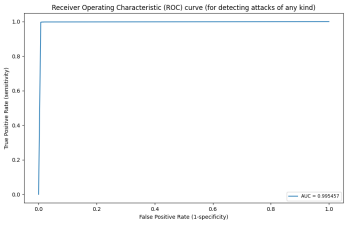


Figure 5: Example ROC graph (Self-Created)

4. Limitations & Challenges

Although the algorithms recommended above are able to carry out the multi-class classification that is required of this data, each of the algorithms have their own disadvantages that could impact the output. Some of the disadvantages and limitations can be seen described below.

Decision Trees:

- Can be prone to overfitting when applied to a full data set due to results possibly being more complex than is required. Overfitting is where the algorithm develops hypotheses in which reduce the error present in the training set but increase the error present in the testing set.
- If there is a small change in the data set, this can cause a need for a large change in structure which causes instability.
- A single decision tree may not be as accurate and would therefore require multiple, causing need for random forest trees.

Random Forest Tress:

- Can be prone to overfitting
- Slows in its progression due to each individual decision tree
- Can require more computational and power resource due to the number of trees being calculated.

Neural Networks:

- Can be quite hardware intensive and therefore could require more computational power.
- Can be considered complex to troubleshoot if something was to go wrong.
- Normally more complex than other machine learning algorithms.
- Complex issues can take a considerable amount of time to develop.

K-Nearest Neighbours:

- As the data set increases in size, the speed of the algorithm decreases drastically.
- Has no capability when it comes to dealing with missing values in the data set.
- Can require a large amount of computational power due to the need to store all of the training data.

Each of these disadvantages and limitations must be taken into consideration with the advantages each algorithm presents in order to find the best match that may suit the required scenario.

5. Conclusion

To conclude this paper, there are multiple options that are available when it comes to training a multi-class classification model. Provided in this paper was four algorithms that could be used, with their own advantages and disadvantages. The advantages and disadvantages of each algorithm must be considered before implementation to allow for the best result possible. With this in mind, following the procedure described above should allow for the implementation and evaluation of a multi-class classification model.

References

Azevedo, C. (2021) *On ROC and Precision-Recall curves*, *Medium*. Available at: https://towardsdatascience.com/on-roc-and-precision-recall-curves-c23e9b63820c (Accessed: 4 April 2021).

'Frameworks for Approaching the Machine Learning Process' (no date) *KDnuggets*. Available at: https://www.kdnuggets.com/a-general-approach-to-the-machine-learning-process.html/ (Accessed: 4 April 2021).

How Do Neural Network Systems Work? (2020) *CHM*. Available at: https://computerhistory.org/blog/how-do-neural-network-systems-work/ (Accessed: 9 April 2021).

Kreiger, J. R. (2020) *Evaluating a Random Forest model*, *Medium*. Available at: https://medium.com/analytics-vidhya/evaluating-a-random-forest-model-9d165595ad56 (Accessed: 4 April 2021).

Kumar, A., Glisson, W. and Cho, H. (2020) 'Network Attack Detection Using an Unsupervised Machine Learning Algorithm', in. doi: 10.24251/HICSS.2020.795.
Narkhede, S. (2021) *Understanding Confusion Matrix*, *Medium*. Available at: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62 (Accessed: 4 April 2021).

Mohajon, J. (2020) *Confusion Matrix for Your Multi-Class Machine Learning Model*, *Medium*. Available at: https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826 (Accessed: 1 April 2021).

Narkhede, S. (2021) *Understanding AUC - ROC Curve*, *Medium*. Available at: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5 (Accessed: 1 April 2021).

Saxena, R. (2017) 'How Decision Tree Algorithm works', *Dataaspirant*, 30 January. Available at: https://dataaspirant.com/how-decision-tree-algorithm-works/ (Accessed: 8 April 2021).

Simple guide to confusion matrix terminology (2014) *Data School*. Available at: https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/ (Accessed: 1 April 2021).

Thomas, M. (2019) 'Neural Networks: Advantages and Applications', *MarkTechPost*, 18 April. Available at: https://www.marktechpost.com/2019/04/18/introduction-to-neural-networks-advantages-and-applications/ (Accessed: 4 April 2021).

Wattanapongsakorn, N. *et al.* (2011) 'Classifying network attack types with machine learning approach', in *7th International Conference on Networked Computing*. *7th International Conference on Networked Computing*, pp. 98–102.
McCall, J. (2020) 'Web Application Firewalls Do More with Machine Learning', *XaaS Journal*, 18 August. Available at: https://www.xaasjournal.com/web-application-firewalls-do-more-with-machine-learning/ (Accessed: 18 April 2021).

What are Neural Networks? (2021). Available at: https://www.ibm.com/cloud/learn/neural-networks (Accessed: 9 April 2021).

Appendix A - Notation

Notation	Meaning
RFT	Random Forest Tree
NN	Neural Network
KNN	K-Nearest Neighbor
ANN	Artificial Neural Network
PRC	Precision Recall Curve
ROC	Receiver Operating Characteristic
TPR	True Positive Rate
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
FPR	False Positive Rate
AUC	Area Under Curve
IDS	Intrusion Detection Systems

Appendix B – Input Data Headings

id,dur,proto,service,state,spkts,dpkts,sbytes,dbytes,rate,sttl,dttl,sload,dload,sloss,dloss,sinpkt,dinpkt,sjit,djit,swin,stcpb,dtcpb,dwin,tcprtt,synack,ackdat,smean,dmean,trans_depth,response_body_len,ct_srv_src,ct_state_ttl,ct_dst_ltm,ct_src_dport_ltm,ct_dst_sport_ltm,ct_dst_src_ltm,is_ftp_login,ct_ftp_cmd,ct_flw_http_mthd,ct_src_ltm,ct_srv_dst,is_sm_ips_ports,attack_cat,label

Appendix C – Pre-processing of Input Data in R

```
colnames(data) =
c("id","dur","proto","service
","state","spkts","dpkts",
"sbytes", "dbytes", "rate",
"sttl", "dttl", "sload",
"dload", "sloss", "dloss",
                  "sinpkt",
"dinpkt", "sjit", "djit", "swin",
"stcpb", "dtcpb", "dwin",
"tcprtt", "synack",
                  "ackdat",
"smean", "dmean",
"trans_depth",
"response_body_len",
"ct_srv_src", "ct_state_ttl",
                  "ct_dst_ltm",
"ct_src_dport_ltm",
"ct_dst_sport_ltm",
"ct_dst_src_ltm",
"is_ftp_login",
                  "ct_ftp_cmd",
"ct_flw_http_mthd",
"ct_src_ltm", "ct_srv_dst",
"is_sm_ips_ports",
"attack_cat", "label")


# Preprocessing the data
data$id =
as.numeric(as.character(d
ata$id))
data$dur =
as.numeric(as.character(d
ata$dur))
vdata$proto =
factor(data$proto)
data$service =
factor(data$service)
data$state =
factor(data$state)
data$spkts =
as.numeric(as.character(d
ata$spkts))
data$dpkts =
as.numeric(as.character(d
ata$dpkts))
data$sbytes =
as.numeric(as.character(d
ata$sbytes))
data$dbytes =
as.numeric(as.character(d
ata$dbytes))
data$rate =
as.numeric(as.character(d
ata$rate))
data$sttl =
as.numeric(as.character(d
ata$sttl))
data$dttl =
as.numeric(as.character(d
ata$dttl))
data$sload =
as.numeric(as.character(d
ata$sload))
data$dload =
as.numeric(as.character(d
ata$dload))
data$sloss =
as.numeric(as.character(d
ata$sloss))
data$dloss =
as.numeric(as.character(d
ata$dloss))
data$sinpkt =
as.numeric(as.character(d
ata$sinpkt))
data$dinpkt =
as.numeric(as.character(d
ata$dinpkt))
data$sjit =
as.numeric(as.character(d
ata$sjit))
data$djit =
as.numeric(as.character(d
ata$djit))
data$swin =
as.numeric(as.character(d
ata$swin))
data$stcpb =
as.numeric(as.character(d
```

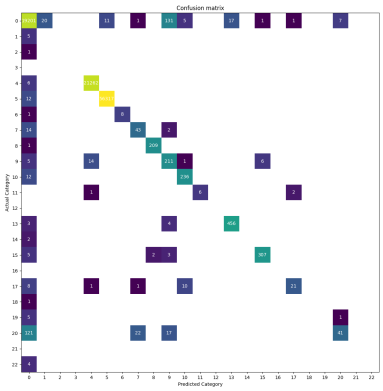
```
ata$stcpb))
data$dtcpb =
as.numeric(as.character(d
ata$dtcpb))
data$dwin =
as.numeric(as.character(d
ata$dwin))
data$tcprtt =
as.numeric(as.character(d
ata$tcprtt))
data$synack =
as.numeric(as.character(d
ata$synack))
data$ackdat =
as.numeric(as.character(d
ata$ackdat))
data$smean =
as.numeric(as.character(d
ata$smean))
data$dmean =
as.numeric(as.character(d
ata$dmean))
data$trans_depth =
as.numeric(as.character(d
ata$trans_depth))
data$response_body_len =
as.numeric(as.character(d
ata$response_body_len))
data$ct_srv_src =
as.numeric(as.character(d
ata$ct_srv_src))
data$ct_state_ttl =
as.numeric(as.character(d
ata$ct_state_ttl))
data$ct_dst_ltm =
as.numeric(as.character(d
ata$ct_dst_ltm))
data$ct_src_dport_ltm =
as.numeric(as.character(d
ata$ct_src_dport_ltm))
data$ct_dst_sport_ltm =
as.numeric(as.character(d
ata$ct_dst_sport_ltm))
data$ct_dst_src_ltm =
as.numeric(as.character(d
```

```
ata$ct_dst_src_ltm))
data$is_ftp_login =
as.numeric(as.character(d
ata$is_ftp_login))
data$ct_ftp_cmd =
as.numeric(as.character(d
ata$ct_ftp_cmd))
data$ct_flw_http_mthd =
as.numeric(as.character(d
ata$ct_flw_http_mthd))
data$ct_src_ltm =
as.numeric(as.character(d
ata$ct_src_ltm))
data$ct_srv_dst =
as.numeric(as.character(d
ata$ct_srv_dst))
data$is_sm_ips_ports =
as.numeric(as.character(d
ata$is_sm_ips_ports))
data$attack_cat =
factor(data$attack_cat)
```

Appendix D – Neural Network Hyperparameters

```
clf=
MLPClassifier(hidden_layer_sizes=(100,),
activation='relu', solver='adam',
alpha=0.0001, batch_size='auto',
learning_rate='constant',
learning_rate_init=0.001, power_t=0.5,
max_iter=200, shuffle=True,
random_state=None, tol=0.0001,
verbose=True, warm_start=False,
momentum=0.9,
nesterovs_momentum=True,
early_stopping=False,
validation_fraction=0.1, beta_1=0.9,
beta_2=0.999, epsilon=1e-08,
n_iter_no_change=10)
```

Appendix E – Example Confusion Matrix


Confusion matrix
Actual Category
Predicted Category
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
19201 20 11 1 131 5 17 1 1 7
5
1
6 21262
12 56317
1 8
14 43 2
1 209
5 14 211 1 6
12 236
1 6 2
3 4 456
2
5 2 3 307
8 1 1 10 21
1
5 1
121 22 17 41
4