Detecting cyber threats from network edge using machine learning

Statement of Originality:

I declare herewith that this above-mentioned work project is my own original work. Any work not my own has been clearly referenced whether in the report or in the code.

Signed……… Sunny Mehta

Contents

Introduction

The goal of this report is to use the skills and knowledge learnt from the Data Science course and apply it to a project. This project specifically will be exploring and investing the UNSW-NB15 dataset from the University of South Wale Australia. It aims to apply Exploratory Data Analysis skills and Machine Learning Models in order to identify whether or not a datapoint can be classified into an attack packet or non-attack packet. It must be noted that while machine learning models are being used, the primary goal of the project is not to build robust models to detect and classify attack categories, but to use these to generate knowledge and insights to support future research in pinpointing the vulnerabilities and help develop solutions to the threat. In fact, while current research [1][2][3] has looked at developing broad systems to identify network intrusions and increase the network securities robustness, there has been little detailed examination of a particular attack category, and little examination of the patterns and trends of the specific attack category in the UNSW-NB15 dataset. This lack of exploration leads to missing insights that may only be gathered by focusing on specific attack categories and its relationship with the features in the dataset. The primary cyber-security threat this project will lock into are Denial-of-Service attacks (DoS). In exploring the dataset for insights and analysis into DoS attacks, the ability to identify potential insights into Distributed Denial-of-Service (DDoS) attacks will also be aided, as the features that are found to assist the report in identifying DoS attacks will also be the same features that will assist in battling DDoS attacks. Such insights will be extremely useful as cybersecurity continues to be a major threat in day-to-day activities.

The main objectives of the reports are the following: establish a correlation between the features in the dataset and the nature; establish which features have the strongest or closest correlation to the nature of the packet; establish which feature have little or no correlation in determining the nature of the packet as either normal or attack packet; determine which features are most important in deterring an attack, and finally, being able to pinpoint the smallest number of features that can be used in determining an attack packet. The final chapter, the conclusion, will assess to what extent these goals have been reached.

The structure of the report will be as followed: a brief introduction to the background of the dataset, and an explanation into how DoS and DDoS attacks occur, this is essential background knowledge to understand the rest of the data. Then, Chapter 1 will be a chapter focused on exploratory data analysis without using any machine learning models. Here patterns will be identified, categorical and numerical variables will be analysed through visual statistics, and a Pearson's Correlation index will be used to find features most correlated with the target variable label, and also find 'useless' features which may be removed in the machine learning model process. Chapter 2 will go ahead with the machine learning models, measuring how well these models do, and then the dataset will be split into half with the same tests run again. Chapter 2 will also give a step-by-step explanation for the choices made, as in accordance with the criteria of the Data Science report. This serves as the foundation chapter as no feature removal will be conducted here; this chapter will test the accuracy with all the features still intact and then compare the results to the future chapters. Chapter 3 will then run the same models, but with the ‘useless’ features found in Chapter 1 removed. The data will be split into half again, and the same tests will be run again to observe the differences. Chapter 4, finally, aims to select the fewest features by using feature selection on the best machine learning model from the previous chapters. Just as before, it will then split the data, and then discuss the findings. As mentioned before, the conclusion and reflection chapter will assess to what extent each chapter met the objectives of aimed in the introduction of the report, and also discuss what changes may be implemented in the future when conducting a thorough Data Analysis on the UNSW-NB15 dataset again. However, such reflections are also written in the individual chapters as well.

The report uses the Python programming language. A zip file will be attached alongside submission containing all the necessary notebooks and the datasets. The report acknowledges that understanding the function behind each feature is important in order to properly predict, and build models, in finding the ideal features in predicting DDoS attacks. However, such a written account would far exceed the word limit and thus the original UNSW-NB15 features description, which is short and brief, will be provided alongside the coding.

In regards to the ethical issues of this analysis. Loughborough University provides a checklist that must be filled out and approved. It is a questionnaire based checklist where if any of the questions are ticked yes then the research will not be carried forward. The report passed the checklist and thus has no ethical issues in conducting this research. Other ethical issues that do not arise in the questionnaire is the open-source nature of coding. As many solutions, coding, and models are shared online, the report makes it clear in the coding notebooks where a code has been copied from, if it has been. Finally, there is some risk management in running the notebook, which is that if Google Collaboratory is being used to read the notebook, the user must make sure that the datasets are completely uploaded before running any of the code. Running the code prior to complete upload will run errors in the machine learning models, if this occurs the solution is to delete the dataset files, re-upload, and then run again. This is not an issue in Jupyter Notebook.

By using both Data Analysis and Data Science skills accrued from the course this year, the report aims to provide useful insights into the nature of DoS attacks within the UNSW-NB15 dataset, and thusly, be valuable outside the dataset too.

Essential Background Information

The Dataset:

The dataset utilized for this project is the UNSW-NB15 dataset from the University of South Wale Australia. The network packets for the dataset were created using the IXIA PerfectStorm tool, combining 'real modern activities and synthetic contemporary attack behaviours."[4] The UNSW-NB15 dataset has nine types of attacks, Fuzzers, Analysis Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms. While the information regarding the other attack categories may be useful, and investigating the specific features which may be useful in detecting these other attacks may be even more useful, the goal of this report is to specifically look at DoS attacks. The fact that the dataset consists of not only natural attack behaviours but also synthetic one will affect the validity of the results as it may not be truly representative of real world behaviour. Nonetheless, it may provide accurate insights in which further research may be explored and investigated.

In total, there are 2,540,044 records stored into four CSV files: UNSW-NB15_1.csv, UNSW-NB15_2.csv, UNSW-NB15_3.csv and UNSW-NB15_4.csv. Due to the substantial amount of datapoints the report will focus on the partition of the dataset: the UNSW_NB15_training-set.csv and UNSW_NB15_testing-set.csv. This not only saves the huge amount of computing power required to analyse this data, but there also would not be enough time to conduct the analysis by due in date, thus the training and testing partition was used – which is still a substantive amount of data. The training set holds 175,341 datapoints; the testing set holds 82,332 datapoints.

The goal of this dataset was to provide a means to develop and improve current Network Intrusion Detection Systems (NIDS)[5]. This is due to the increase in technological dependency, as everyday life becomes more prone to cybersecurity attacks.[6] In this case, an intrusion is defined as actions that endanger the sanctity of a networks security.[7] An NID, therefore, mitigates this issue by monitoring the flow of traffic in order to capitalize on attacks. NID's perform this by using signatures and anomalies: signatures pair the existing knowledge of attacks to uncover intrusions, anomalies develop a profile based on the 'normal behaviour of the network' and, thus, any divergence from the normality is considered an attack[8]. However, previous evaluation metrics of NIDs had weaknesses, such as the amount of 'redundant records' which would affect the results of the detection; the missing records, and the failure to provide a modern representation of an attack environment.[9] The UNSW-NB15 dataset was created in response to this, providing an increasingly dynamic system that not only represents a more modern representation of attacks, but also a reliable dataset with minimum data missing.

Moustafa and Slay's statistical analysis show the effectiveness of the UNSW-NB15 dataset, demonstrating that the dataset is suitable for classification as both the training and the testing sets show mirror characteristics of 'non-linearity and non-normality', and that the skewness and kurtosis figures are in acceptable levels that when, for example, machine learning models are used, it is not widely inaccurate.[10]

Denial of Service Attacks (DoS):

Denial of Service attacks, also commonly referred to as DoS, is a common technique used to block web applications being used by genuine users. The attack aims to render a network catatonic by aiming at the networks connectivity. This is done by sending the victim network with a barrage of useless packets, the overload of which causes genuine users to be unable to access the desired client. The reason this is successful is because the series of needless packets have inauthentic return addresses, causing the server to waste its time in trying to authenticate the requested packet.[11] This problem is exacerbated when reminded of the multiple services that may be targeted, affecting day-to-day life. This goes from banking, emails, websites, or any service that are dependent upon a network. DoS attacks tend to take advantage of software and/or design flaws in order to harm the targeted network.[12]

There are five categories of DoS attacks: Network Device Level Attacks, OS Level Attacks, Application Level Attacks, Data Flood Attacks and Protocol Feature Level Attacks.[13] The Network Device level exploits bugs in the software or overwork the hardware; the OS level DoS uses the Ping of Death where the 'total data sizes greater than the maximum IP standard size', leading to a crash of the targeted machine; Application level attacks exploit novel bugs in the networks system, similar to OS level attacks, such that the resources are drained; Data Flood, as the name suggests, sends immense quantities of data causing the target to process more data than its bandwidth can handle, and finally, Protocol Feature level attacks exploits the protocol features themselves.[14]

Distributed Denial of Service (DDoS) is an upgraded form of attack compared to DoS. This is because traffic derives from multiple locations using multiple sources making it difficult to pinpoint the onslaught, and thus defend against it.[15]

One DDoS method is the User Datagram Protocol (UDP) flood attack. UDP is a protocol that allows data transfers and exchange of messages; these messages are partitioned into datagrams which get tunnelled into the network, finally reaching the 'destination/application server.'[16] A UDP flood attack takes advantage of this by sending huge numbers of UDP datagrams to their target serve. This slows down the network as bandwidth is unavailable for official requests to that system.[17]

Next, is the Internet Control Message Protocol Flood (ICMP). ICMP is used to diagnose error messages, checking whether the data is reaching its destination.[18] This is done through ping requests, thus a ICMP flood attack overwhelms the network by flooding it with request packets. The great number of requests lessens the bandwidth available on the target's network because the reply packets are not able to respond fast enough. This leads to similar consequences as the DDoS method.

Another common DDoS method is the Smurf Attack. It uses the same method as the ICMP flood attack, taking advantage of ping requests. However, a Smurf attack will attack all hosts on the network, the greater the amount of machines on the networks, the greater the Smurf attack.[19]

The final one that shall be looked at is the Ping of Death (POD). In this attack the culprit floods the targets network with 'malformed or malicious pings'.[20] This is done by taking advantage of maximum packet length, which is 65,535. The culprit divides the IP packets, manipulating them so when the IP packet is reassembled the target receives a packet larger than 65,535 bytes. The overload in memory causes the service to be unavailable for legitimate packets.[21]

While the report acknowledges that the dataset will be looking at DoS attacks rather than DDoS attacks, the features that are vulnerable to DoS attacks will be the same features that are vulnerable to DDoS attacks, thus an investigation into the attacks will help combat both types of attacks.

Chapter 1: Exploratory Data Analysis

This chapter uses exploratory data analysis techniques (EDA) to examine the dataset and gather insights in order to achieve the overarching objectives laid out in the introduction. As originally defined by Tukey, data analysis uses “techniques for interpreting the results of such procedures, ways of planning the gathering of data to make its analysis easier, more precise and more accurate”.[23] The data will be described, and presented, through the use of descriptive statistics and plots; it will examine the relationship between the features and the target variable “label” (determining whether or not the nature of the packet is normal or not), and finally find features with the highest correlation with the target variable. Moreover, this chapter also hopes to present the skills and knowledge acquired from the conversion course by explaining the purpose behind most of the coding, for example why df.shape() was used and if it provided any information about the dataset A detailed written account of the EDA has advantages, compared to skipping to only explaining the descriptive statistics and the visuals, as it provides the literature with explanations to those unfamiliar with the field of coding. Furthermore, any inconsistencies with code or discrepancies with logic may further enhance the fields ability in detecting DoS and DDoS attacks as these would be improved upon, creating tighter and more astute models in selecting the optimum features for DDoS detection.

This chapter will be divided into two sections: firstly, an examination of the dataset before looking for connections with the target variable ‘label’, for example the use of df.info(), which helped provide direction on where to start the descriptive analytics; then, the second section observes and investigates the connections between the features and the target variable.

The insights and examination from this preliminary analysis will guide the machine learning models by providing a foundation in finding out what features are most likely to be important prior to using machine learning models for feature selection. The EDA will also investigate if the data needs to be cleaned in any shape or form, for example renaming certain features, as well as containing other pre-processing tasks which will be used throughout the chapters.

Dataset Examination

The following packages were used for this chapter:

- numpy
- pandas
- matplotlib
- seaborn
- missingno
- pickle
- warnings
- math, time, random, datetime

Firstly, both the UNSW-NB15 training dataset and the testing dataset were imported into pandas. Next, the datasets were concatenated. This is because combining both the files provides one with more data to examine, and this provides a more accurate outlook in investigating the patterns of the dataset. Also, any pre-processing demands needed in one will naturally need to be engineered on the other, thus saving time by cleaning it in one go.

The combined data frame (df) contains 257,673 rows and 45 features. Within the large data set, there were no missing values. Of the 45 features, only four were categorical variables – proto, service, state, and attack_cat - while the rest were a numeric mix between int64 and float64. No cleaning was necessary again as all the datatypes were apt to the features, however the object variables would be transformed into a numeric type for later feature selection with machine learning models, as well as for the Pearson's Correlation index. Also, as 'id' is not a useful variable, it will also be dropped. It is not useful as it des not provide the report with concrete information such as proto or service might, instead id is used to number its place in the dataset, which informs the report nothing in terms of identifying attack variables.

Df.describe showed interesting observations. Across the features, there seemed to be a great discrepancy between the mean and the maximum. For example, while the mean for dur (duration) was 1.246, and the median was even lower at 0.004285, the maximum was 59.999. This seemed to be a common occurrence across the features, where the maximum was disproportionately higher than the mean. This suggested that there may be a lot of outliers scattered in this dataset. Only 'label', the target variable, did not show this discrepancy. However this is also because 'label' may only be 0 , for no attack, and 1, for attack, thus this would be expected. The decision was also made to keep these outliers, this is because both orthodox and novel attacks have the possibility of being an outlier. If the outliers are removed, the machine learning models, in the future chapters, will decrease in efficacy due to its inability to classify an attack packet within the outliers, and thus decreasing in validity in the real-world.

Next, the categorical variables were examined.

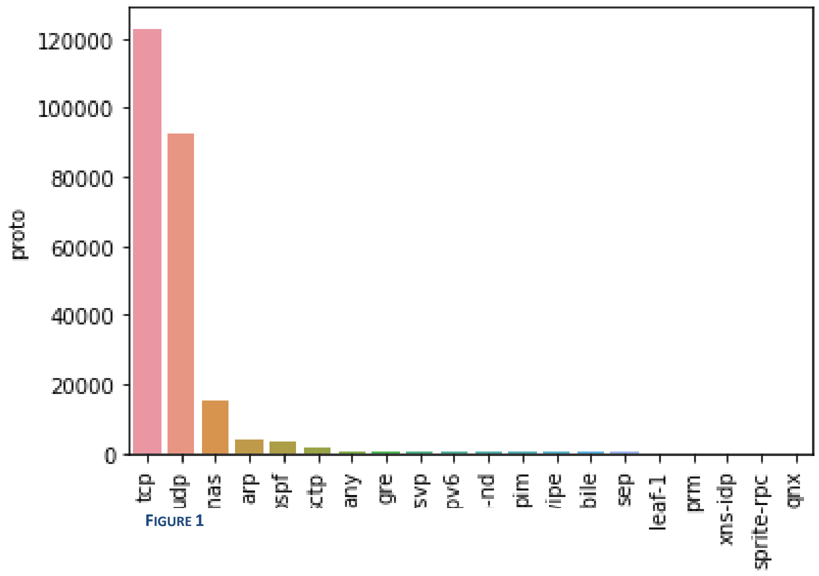

FIGURE 1

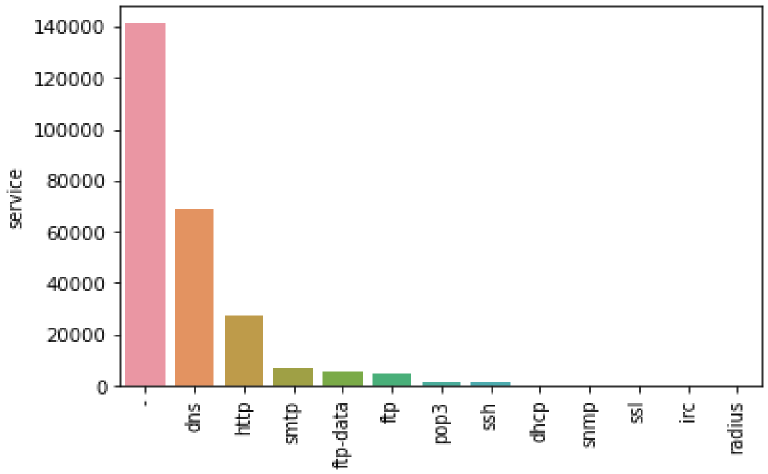

FIGURE 2

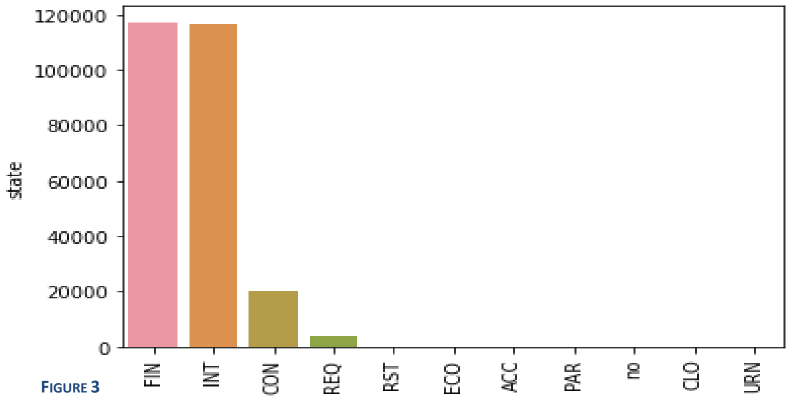

FIGURE 3

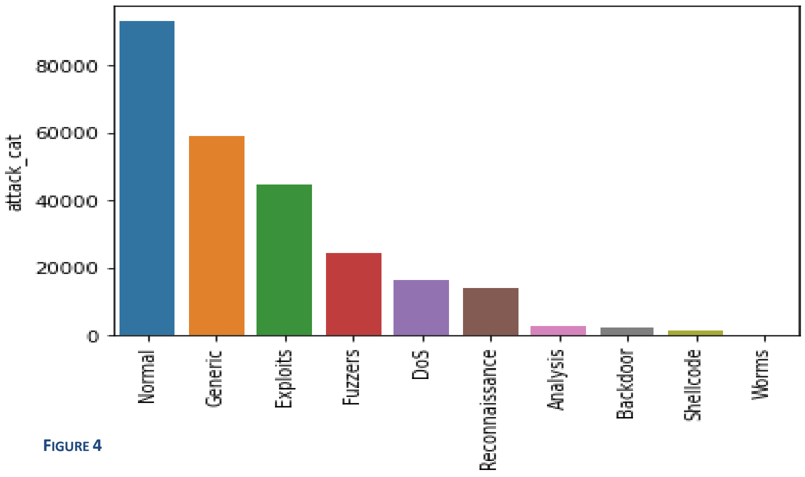

FIGURE 4

Figure 1 shows the top twenty protocol suites (proto) in the dataset. TCP and UDP comprised the majority of the protocol suites in this dataset, 123,041/257673 and 92,701/257,673 respectively. A much larger amount compared to the third highest, UNAS at 15,599, followed by the rest shared in small amounts. Background knowledge of the dataset would not find this surprising as TCP and UDP are the two most common protocol suites used in the real world, and the dataset is built-upon both natural and synthetic data. It may also suggest that these two, more likely than the others, may help with assessing whether a packet is a threat or not. However, this is only because it holds the majority, not because of an inexplicable link between the protocol and the chances of it being attacked. It may be argued that, from examining figure 1, only TCP, UDP and even UNAS should be kept for the feature selection, dropping the rest from the machine learning models. But this also means the ML model will not be able to examine threats beyond these protocol suites, reducing its ability in the real world. Moreover, features may interact differently with different protocol suites if the rest are removed attacks packets on those protocols will not be picked up. This, again, would reflect poorly in the real world which will not contain just 257,673 data points, but millions, and resultingly, there would be more data points for the remainder of the protocols in comparison to this dataset. Thus, this data alone does not justify the removal of these features.

Figure two shows the distribution of the services used. The first surprising point is the most used feature is not identified, named ‘ – ‘. It holds 141,321/257673. The lack of identification is a problem as it hampers the ability to pinpoint which services are most vulnerable to attacks. It cannot be removed either as removal will provide a biased feature selection, as it is possible some of the unidentified will contain attack packets. This will be renamed for the models. DNS and HTTP comprise the new two highest, 68,661 and 27,011 respectively. It follows a similar distribution as protocol suites as the more common services hold greater weight, barring the unidentified. For the same reasons, the rest will not be removed.

Figure three shows that the first two states, FIN and INT, are disproportionately represented in the dataset, even when compared to the previous figures. 117,164/257673 for FIN and 116,438 for INT. In this case, an argument may be made to remove the states with little weight in the dataset. For example, ACC and CLO only contain one datapoint in the dataset. On the other hand, it may be kept examining whether the ML model is able to successfully differentiate non-attack packets as well. Also, as part of the dataset is synthetic, it may be that the representations of these states are different in the real world. Nonetheless, these features are unlikely to make any impact in detecting an attack packet, but, again, this data alone does not suffice in the removal of these features.

Finally, the distribution of the attack categories is shown in figure 4. The majority of the attack category is normal, 93,000/257,673, which means no attacks. Unfortunately, DoS only comprises 16,353 out of the 257,673 datapoints. Firstly, as the report is focused with DoS attacks only, the rest will be dropped when examining the relationship between the attack packets and the target variable. The low number of DoS datapoints means the model will not be as accurate as it could be, as the greater amount of data the better the model. More data provides more information for the machine learning models to make its judgement. Furthermore, the low number of DoS datapoints also means that other features which may have an impact on judging the packet as attack or not will be missed as the size is low. This also means it will not be full representative of real-world data. This must be considered when concluding the results of the models in the upcoming chapters.

Relationship between the Features and ‘label’

In this section, features will be analysed with the target variable itself.

Firstly, a box plot between the attack categories and 'label' was coded. This was to make sure nothing as amiss, i.e. all attack categories should be on 1, and the normal category on 0, which was the case, thus the figure was not needed in the report.

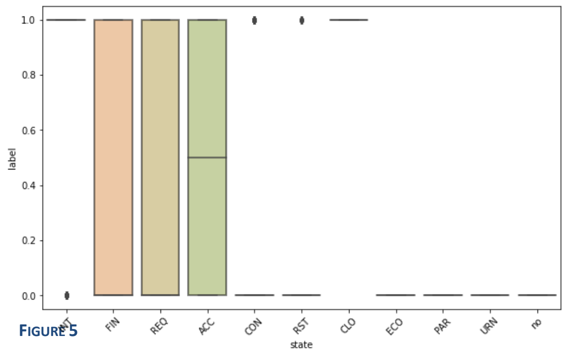

FIGURE 5

Figure 5 suggests that when the state used is INT, it is expected to also be an attack packet as the non-attack packet, 0, is shown to be an outlier. This is also true for CLO. The opposite is true for CON and RST. On the other hand, ECO PAR URN and no (none), show 0 correlation with the attack packet. This provides a reason to drop these features as it will not be helpful in identifying the attack packet, which is contrary to the stand-alone observation from figure 3, highlighting the importance of visualizing features with the target variable itself.

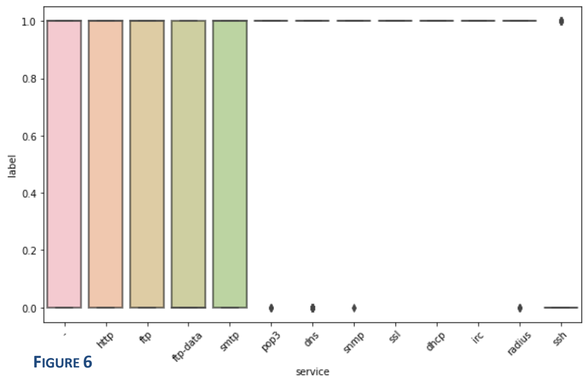

FIGURE 6

Figure 6 shows that POP3, DNS, SNMP and RADIUS, may be important features in identifying the attack packet, as its outliers are on 0, non-attack. The same is true for SSL, DHCP and IRC, except there are no outliers on the non-attack. SSH shows the opposite, suggesting little relationship with 'label', as the outliers are an attack variable.

The service SSH feature in figure 6 and the state CON and RST further strengthen the case case for not removing any outliers whatsoever. This is because the fact that outliers are present in the attack variable suggest that in the real-world, with more datapoints, the relationship between them and 'label' would be a lot stronger. Furthermore, it also presents the case that novel attacks may still occur, by removing outliers, it removes the ML's ability to identify novel attack packets. To prevent this, the report argues to not remove the outliers in the dataset. This case is strengthened when considering that not all of this is natural data, and it is not possible to distinguish between the natural and synthetic data. If, for example, SSH is a natural data point then it must remain so ML models may identify them, if it is synthetic, it may be removed as it is unrepresentative of real-world attacks. However, as there is no way of distinguishing the outliers, for this reason too, will remain.

After this, the relationship between the numeric features and the "label" category was examined. The object variables from before also turned into a numeric datatype using pd.get_dummies. Next, since DoS packets is the reports main concern, the other attack categories were removed. Leaving behind attack_cat_DoS and attack_cat_Normal. ID was also removed.

Finally, using pythons inbuilt corr() function, correlations between the numerical features and the target variable 'label' was found. Python's corr() function uses Pearson's correlation method, assigning the feature between -1 and 1. 1 would suggest a total positive correlation, -1 means a negative correlation, and 0 would mean no correlation at all.[24] Below were the results for features that had a Pearson's correlation value above at least 0.2:

TABLE 1

Feature	Pearson's correlation value.
sttl	0.624082
state_INT	0.516735
ct_state_ttl	0.476559
ct_dst_sport_ltm	0.371672
rate	0.335883
ct_src_dport_ltm	0.318518
ct_dst_src_ltm	0.299609
service_dns	0.259911
ct_src_ltm	0.252498
ct_srv_dst	0.247812
ct_srv_src	0.246596
ct_dst_ltm	0.240776

Table 1 shows that STTL is the strongest feature in identifying whether or not the datapoint is an attack packet, with a 0.62 correlation. State_INT is the second highest at 0.52, which is an interesting find. This is because figure 3 from earlier showed there were more State_FIN packets than State_INT. It would be expected for State_FIN to also show up on the list if State_INT has. This suggest there is an innate security vulnerability with State_INT, leading it to be more prone to DoS attacks. On the other hand, it may also be the distribution of synthetic data which has skewed the results. Regardless, this is an observation to pay attention to other cybersecurity datasets. Service_dns showing up and not the unidentified ' – ' is also surprising for the same reason, but it has a low Pearson index in the first place, consequently less impactful.

Following the Pearson Correlations, the features were then plotted with label, to present a numeric visual relationship. This was conducted near the end as the categorical variables were also turned into numeric at this point, for this analysis. The following features were found to have no correlation or relationship with the attack variable whatsoever: dur, **rate**, sloss, djit, tcprtt, trans_depth, ct_src_dport_ltm, ct_flw_http_mthd, proto_3pc, proto_aris, proto_br-sat-mon, proto_cphb, proto_ddp, proto_emcon, proto_ggp, proto_iatp, proto_idrp, proto_ip, proto_ipnip, proto_ipv6-opts, proto_iso-ip, proto_leaf-1, proto_micp, proto_netblt, proto_pim, proto_ptp, proto_rsvp, proto_sccopmce, proto_sep, proto_sprite-rpc, proto_sun-nd, proto_tp++, proto_unas, proto_vrrp, proto_xnx-idp, **service_dns**, service_pop3, service_ssl, **state_FIN,** state_URN. These will be referred to as the 'useless' features throughout the report.

From this, three features stood out as most unusual, and thus contemplation on whether or not to remove these features for the machine learning model as well.

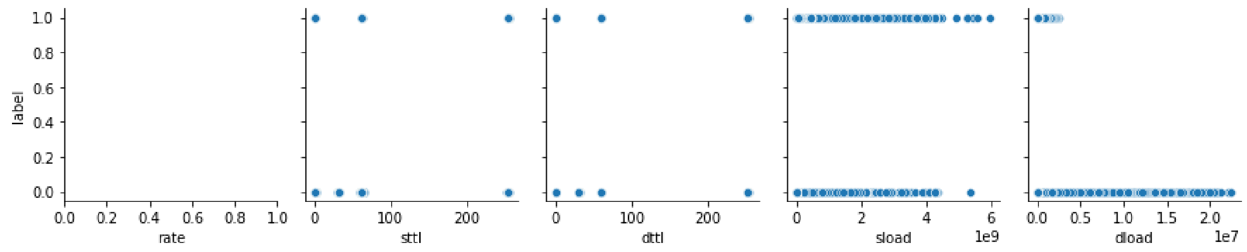

FIGURE 7

Rate was the first peculiar one. This is because it showed up with a 0.335883 Pearson Correlation index, which is normally low, but for this dataset that was quite significant. Next that stood out was service_dns:

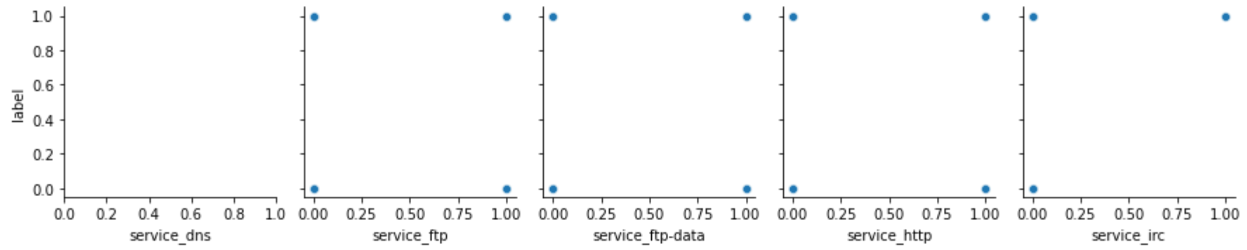

FIGURE 8

Figure 2 shows that service_dns had the second highest amount of datapoints out of the services, and the Pearson Correlation Index gave it a scoring higher than 0.2 as well. Thus far, it suggests the Pearson Correlation Index itself is not ideal in identifying attack packets, nor the relationship itself. Finally, state_FIN definitely provoked tentativeness and re-examination of the code, making sure there were no mistakes.

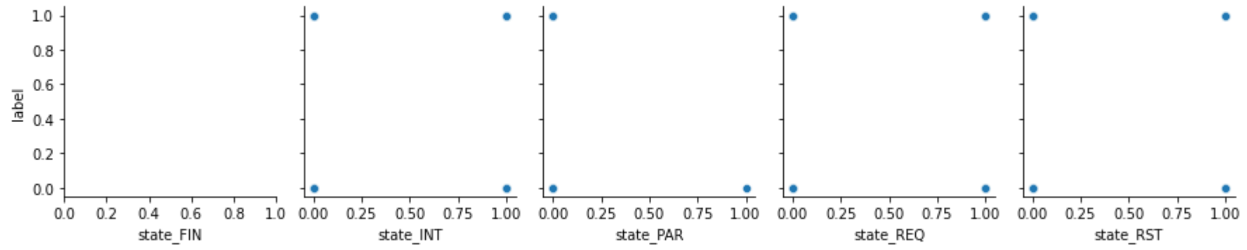

FIGURE 9

Figure 2 showed us that FIN was the most popular state in this dataset. This means it should be quite likely that there is an attack packet present in its datapoints. Furthermore, the boxplot in figure 5 also points towards this assumption. However, a closer examination between 'label' and the feature shows no relationship is present. Also, it did now show up in Pearson's Correlation index either, while state_INT did, further suggesting the removal of this feature.

The features which showed no relationship with label were removed, including state_INT, service_dns, and rate. Next, Pearson's correlation was used again, table 2 show the results:

TABLE 2

Feature	Pearson's correlation value.

sttl	0.624082
state_INT	0.516735
ct_state_ttl	0.476559
ct_dst_sport_ltm	0.371672
ct_src_dport_ltm	0.318518
ct_dst_src_ltm	0.299609
ct_src_ltm	0.252498
ct_srv_dst	0.247812
ct_srv_src	0.246596
ct_dst_ltm	0.240776

The results were the exact same. The only difference was the removal of rate and service_dns. The fact that it did not go lower shows that the removal of those features with no relationship between target variable was correct, because if those features had any impact, then some changes would be expected in the Pearson correlations after the removal.

This chapter, has thus, set the groundworks for the machine learning models that will be looked at in Chapter 2, 3 and 4. This is because it has established which features are definitely not needed, without needing to use feature selection methods from machine learning models. This saves CPU output when conducting these tests, and resultingly, it also saves time. The chapter also showcases the Exploratory Data Analysis skills acquired from the course thus far.

Chapter 2: Initial Machine Learning Models.

This chapter will examine the ability of machine learning models to accurately predict an attack packet. This chapter specifically will use all the features available, contrary to Chapter 1's finding. Chapter 3 will then implement the findings of Chapter One and compare the results, assessing if the removal of the 'useless' features was helpful. This will be evidenced by checking whether the machine learning model's accuracy has improved or not. Feature selection will then be used to extract the features that seem to be most useful in predicting an attack, and then these features will be compared to Chapter 1's Pearson Correlation Index. Afterwards, the dataset will be split in half, showcasing the results for 50% of the dataset. The split in the dataset will assist in assessing whether the number of datapoints impacts feature selection and prediction of an attack. For example, would 50% of the dataset lower in accuracy, and, would 50% of the dataset skip some features previously selected as important? Examining any such differences will be helpful in forming an accurate conclusion.

The chapter will be split into two sections: Methodology and Results & Discussion.

Methodology:

Packages:

The following packages were used for this chapter:

- numpy
- pandas
- matplotlib
- seaborn
- missingno
- pickle
- math, time, random, datetime

- sklearn

The following statistical models were used for the machine learning process:

- Logistic Regression
- K Neighbours Classifier
- Decision Tree Classifier
- Random Forest Classifier
- Gaussian NB
- Gradient Boosting Classifier

These are all supervised machine learning algorithms. The difference between a supervised and an unsupervised machine learning algorithm is that the former relies on labelled datasets, which means all the features are known. This is used to train the machine learning model to, in this case, classify the data accurately. Unsupervised, however, use unlabelled datasets, attempting to detect patterns without any prior information about the features. As mentioned prior in the introduction, the goals of this report means that supervised learning algorithms will be used.

Statistical Models:

Logistic Regression

Logistic Regression is the only regression analysis used in this report. Regression analysis is the method in which statistical procedures are used to estimate the relationship between the dependent variable and the independent variable.

Logistic Regression falls under this umbrella, however it differs from the more commonly used Linear Regression. This is because Logistic Regression deals with predicting the probability of a binary event occurring. Whereas Linear Regression deals with predictive modelling. For example, predicting a populations walking steps based on the weather outside. In this case, calculating the occurrence of an attack packet; label = 0 for no attack, label = 1 for DoS attacks. As label, the dependent variable fits into two categories, a machine learning model using logistic regression is apt.

K Neighbours

K-Neighbours, or also known as k-nearest neighbours (knn) classifies data by predicting the probability that a specific datapoint will belong to one group or another by examining the group closest to it. For example, imagine a random datapoint surrounded by two groups called X and Y. If the datapoint is closer to X it will assign it to the group X, and vice versa. In this case, knn will be examining how close a datapoint is to label = 0 versus label = 1, thus judging whether it is a non-attack packet or an attack packet. The calculation used to judge the distance between a datapoint and the groups will be the Euclidean distance.

Decision Tree

Decision Tree's essentially follows a flowchart structure, "with each internal node a test of an attribute, each branch is the outcome of that test, and each leaf contains a class label."[24] In this case, the internal nodes are the the independent variables – the features used to predict whether there is an attack packet or not – will be branched out until the datapoint is classified as either non-attack or an attack variable, which are the leaf nodes.. The internal nodes are the independent variables

Random Forest

Random forest is similar to the decision tree, the difference being that Random forest uses multiple decision trees to reach the end result. This is an advantages as the greater number of forests allows for a more astute prediction, leading to higher accuracy. As a result, it is expected that this algorithm should do better than the decision tree model. Furthermore, it also has the ability to handle greater amounts of data compared to decision tree's.

Gaussian NB

Gaussian Naïve Bayes is an extension of the Bayes theorem. The Bayes theorem, in this case, gathers all the non-attack packets and the attack packets, and then calculates the probability of each datapoint being non-attack or attack based on whether that datapoint was in the attack packet or not. Next a probability is given on whether that datapoint will be an attack packet or not.

Gradient Boosting

Gradient Boosting uses ensemble methods. This is a machine learning technique that blends multiple base models in order to generate a single optimum model. One such ensemble method is boosting. It combines many simple models, the more simple models there are the greater its prediction capabilities. Next, it uses gradient descent to minimize the 'loss function of the model'.[25]

All the models will be evaluated based on its accuracy and its cross-validation accuracy. Accuracy, essentially, denotes the fraction of predictions the model got right. It is the number of correct predictions divided by the total number of predictions. However, cross-validation is the more important metric in determining the efficacy of the model. Cross-validation accounts for overfitting or underfitting a model. An overfitted model fits too close, in some cases exactly, to the training data. Thus, when the model is provided with new data it performs poorly. Underfitting is the exact opposite problem, where the model does not fit the data well enough. Cross-validation helps prevent this. It separates the data into smaller subset, called folds. Then, the model iteratively trains on one of these folds while the remaining are used as a test set.[26]

Once the model has outputted the results, the cross-validated model with the highest accuracy will be chosen for the next Chapter. The model's best features will be checked and then those features will be used to rerun the model to see if the accuracy improves. Best features, in this report, simply means the top ten most significant features. This may seem like an arbitrary number, but it was found, as the results will show later on, the feature importance was unevenly distributed at the top features, thus ten is apt for the purposes of discussion. Normal accuracy will be used as a means of comparison, for example two models with similar accuracy but contrasting cross-validation accuracy will be observed, if such occurrences occur. This becomes more apparent as the data is split into 50%.

Coding:

The starting process is similar to the previous chapter. First the packages are imported, next both the training dataset and testing dataset are imported and then concatenated to create a larger dataset. The larger the dataset the more accurate the models will be, and, it will be more representative of the real world environment.

As found in the previous chapter, there was an unidentified feature under service labelled ' – '. This was renamed to 'None'. This was to avoid any bugging problems for the machine learning algorithms. The categorical variables were encoded into its numeric counterparts, and then the rest of the attack categories were removed apart from attack_cat_DoS and attack_cat_Normal. The data was then read again (using .head()) to double check that the only label categories are those that are attributed to attack category DoS. Once this was assured, attack category DoS and attack category Normal were removed from the dataset. There were two main reasons for this: firstly, target variable 'label' will only show 0 or 1 if the attack has been from DoS, from earlier processing, which means these two are no longer needed as they share the same information; secondly, if these were kept the model would have been biased, because of course attack category DoS will have an extremely high relationship with whether label = 1, which would skew the results for the rest of the features.

The data was then normalized. This is important because there are many features in this dataset, and these features do not all follow the same range. By normalizing the data, it all falls across the same scale. The benefits of this is that the models will be more accurate, and thus a better chance at predicting an attack packet.

Finally, the models were then implemented.

Results and Discussion

First, the results of the full dataset will be showcased. Next, the results of 50% of the dataset will be presented. Another aspect shown in the code but will not be discussed in the report is the execution time. This is because different users will have different times according to one's specific CPU power, therefore while it is a useful code to run, it does not merit discussion.

Algorithm	Accuracy
Decision Tree	99.74%
Random Forest	99.74%
K-Nearest Neighbours (KNN)	95.90%
Gradient Boosting Classifier (GBC)	93.38%
Logistic Regression (Log. Reg.)	90.10%
Gaussian Naïve Bayes (GNB)	50.50%

TABLE 1

The accuracy results show that both Decision Tree and Random Forest rank the highest in predicting whether or not a particular datapoint is an attack packet or not. This is to say that if a thousand attacks were to occur, the model would be able to accurately identify 997 (997.4 to be exact) of these attacks. One would initially expect Random Forest to do better than Decision Tree, even marginally, as it uses the same process but Random Forest go through multiple Decision Trees. It is also interesting to note the discrepancy in their prowess. All the models reached an accuracy rate of at least 90% or above, apart from Gaussian Naïve Bayes (GNB). To use the same example, out of the thousand DoS attacks, the model would only be able to identify 505 of the attacks. In a real world environment, this would not be good enough as even a fifty attacks have the potential to collapse an IoT (internet of things) or server. From this, it may be concluded that GNB is not a good model to use. However, as explained before, the cross-validation (CV) accuracy must also be noted, as good accuracy results may be due to overfitting rather than the evidence of a good model. Table 2 shows the CV accuracy.

Algorithm	CV Accuracy
Random Forest	92.6
GBC	92.09
Decision Tree	91.44
KNN	89.71
Logistic Regression	89.04
GNB	50.46

TABLE 2

After going through CV, it seems the 'true' accuracy of the models has been dropped. Random Forest's accuracy dropped by 7.14%; Decision Tree's was an even greater decrease by 8.3%, no longer tied in first with Random Forest. KNN had a 6.19% decrease; GBC only had a 1.29% decrease. Log Reg and GNB had similarly low decreases in value, 1.06% and 0.04% respectively. This shows us that going through cross-validation was a good counter-measure as all these models seem to have been overfitted to the training data. This also reaffirms the need for evaluating the model with different metrics to ensure efficacy; this is even more important when considering that even slight lapses may mean disastrous real-world results with DoS attacks.

The results also show that while Random Forest was considerably overfitted compared to the other models, even after cross-validation it still remained as one of the most accurate models in predicting an attack. It also affirms the hypothesis mentioned prior, that it is expected that Random Forest would do better than Decision Tree. GBC shows that it is less prone to overfitting, compared to these selection of models, and may still perform well in identifying attack packets. Future research may consider keeping GBC with more data, to assess whether having more datapoints will increase its ability to identify attacks. Decision Tree shows that it is prone to overfitting to the training data, making it less reliable to select the best features in identifying the attack packets. KNN followed the same pattern. Log Reg, while less prone to overfitting, still compares considerably worse compared to the top models, and the same is true for GNB.

After finding out that Random Forest was the best model, it was important to find out which features were the most significant in determining whether or not the datapoint was an attack packet. Table 3 shows the results of all the top ten significant features:

Feature	Importance
sttl	0.085595
ct_state_ttl	0.068380
sbytes	0.043514
smean	0.040248
ackdat	0.039816
ct_dst_src_ltm	0.039101
rate	0.038771
state_INT	0.038734
dttl	0.038217
sload	0.037963

TABLE 3

When compared to the Exploratory Data Analysis chapter, it seems STTL is still a good feature in predicting whether a datapoint is an attack packet or not. It is surprising to notice that DTTL, its counterpart (from source to destination) is considerably worse in identifying a DoS attack. The same discrepancy is noticed with sbytes and dbytes, where dbytes does not even make it to the top ten features. From this it may be inferred that a lot of the attacks are occurring in the source, rather than the destination. On the other hand, this is not a set pattern, as the remaining features (displayed in the code), shows some destination features being ranked with higher importance than source ones. Another discrepancy noticed, in comparison to the Pearson's Correlation index, is that Pearson disproportionately ranked ct features with higher correlation to the target variable, whereas only two ct variables appear with random forests feature selection. One possible explanation for this is that the ct variables have close correlations with each other, thus skewing the results in their favour. This underlies the importance of not relying on correlation in picking out the best features to identify an attack packet, because, while sttl and ct_state_ttl did show up in Pearson's correlation index, a lot of the other features had not. Furthermore, there was no evaluation of Pearson's correlation index's accuracy, thus lowering the efficacy of purely using EDA tools to capture important features. However, the next chapter will judge whether the EDA tools are useful in eliminating needless features. While the report does not aim for this, the next chapter may also shed some light on the extent to which EDA should be carried out for machine learning models, or if the machine learning model itself is enough to eliminate features. Nevertheless, it is expected that when the needless features found are removed, the cross-validation accuracy of the model will increase, granting greater confidence in the feature selection process.

50% of the Dataset

The dataset was split in half. The original size dataset contained 257,672 datapoints with 197 features. The 50% dataset contains 128,836 with 197 features. The models run here are expected to do worse in accuracy, as there is less data to work with. On the other hand, the top features selected for identifying the model is expected to be the same, only losing in slight prowess in identifying the attack packet. Table 4 shows the results of the machine learning models accuracy and Table 5 shows the cross-validation accuracy.

Algorithm	Accuracy
Decision Tree	99.81%
Random Forest	99.81%
K-Nearest Neighbours (KNN)	95.63%
Gradient Boosting Classifier (GBC)	93.44%
Logistic Regression (Log. Reg.)	90.03%
Gaussian Naïve Bayes (GNB)	50.48%

TABLE 4

Table 4 shows the same order as Table 1, with the complete dataset, showing that Decision Tree and Random Forest perform better. It must be noted that both their accuracies increased, compared to Table 1 by 0.7%. This suggests that when there is less data, Decision Tree and Random Forest perform worse. If this is the case, future research may find more datapoints helpful when running the two models because it may be hypothesised that the inverse is also true. As shown before, GNB is not a good model in predicting any attack packets compared to the other models.

Algorithm	CV Accuracy
Random Forest	94.63
GBC	93.38
Decision Tree	93.21
KNN	91.22
Logistic Regression	90.01
GNB	50.47

TABLE 5

After the cross-validation check, Random Forest has come up again. However, it must be noted that Random Forest, Decision Tree, Logistic Regression, and GNB all did worse in Table 2, suggesting that when there are more datapoints, its accuracy in predicting the attacks decreases. KNN, on the other hand, seemed to have done better with less datasets, even if still poor compared to its preceding models. This suggests that KNN may do better with less datapoints. It is not very practical as the real-world scenario are likely to have even greater amount of datapoints than the complete dataset.

Finally, as Random Forest did well again, the top ten significant features were checked again.

Feature	Importance
sttl	0.079113
ct_state_ttl	0.068052
sload	0.061455
dload	0.052190
synack	0.047375
dttl	0.045022
smean	0.044884
ct_srv_dst	0.043446

sbytes	0.042376
dinpkt	0.038391

TABLE 6

Table 6 shows the results. It seems even with 50% of the dataset, sttl and ct_state_ttl were both ranked as the two most important features in classifying an attack packet, with sttl still showing the greatest of importance by a larger margin. It is interesting to note that the importance of sttl decreased, while the importance of the other features have increased. This suggests that as datapoints increase, a greater number of datapoints that can be classified as an attack variable stem from sttl, thus, when the dataset is decreased, classifiers may misattribute the importance of the weaker features. It must also be noted that as the dataset decreased, new features showed up in the top ten importance list, while some have disappeared from the complete dataset. For example, the feature ackdat was previously ranked as the fifth most important feature in predicting attack packets, however this time it did not appear. Furthermore, features previously ranked lower with the full dataset have increased in importance with the dataset decreased. Sload and dttl are both examples of these, both gaining more importance as the dataset size decreases. A possible explanation for such an occurrence is that as the dataset increases, the linear seperability between the features may increase as well, which means the machine learning models find it easier to capture the best features in predicting an attack packet. Nonetheless, future research may find it helpful to increase their dataset size when classifying for an attack packet, as there may be relationships that show up that are unseen before. Having said that, it may be hypothesised, that it is still likely that sttl and ct_state_ttl will remain important in identifying the nature of an attack packet. Moreover, these models still contain the 'useless' features found in EDA, the next chapter will explore whether, firstly, the models improve with their removal, and if there is an improvement, whether the features selected as most important will remain the same.

Conclusion

Overall, this is a good start in finding the best features, and thus the most vulnerable features, in identifying attacks. These models will be compared to the results in Chapter 3, where the useless features in Chapter 1 were identified, and thusly removed. This chapter has provided a good control, which means that with subsequent improvements, the future Chapter models should theoretically be better than the results in this chapter.

Chapter 3: Machine Learning Models with Features Removed

This chapter will continue on using the machine learning models from Chapter Two in order to reach the objectives outlined in the introduction. However, this chapter will assess the results of removing the 'useless' features found in Chapter 1 and discuss the difference in results. It is expected that the model will perform better, which means the cross-validation accuracy should improve, as a lot of the needless features will be removed, optimising the model. It will use the same statistical models as before, it will also split the data in half to find out if it follows similar patterns observed before.

The chapter will be split into Methodology and Results & Discussion.

Methodology:

Packages:

There were no differences in the packages used:

- numpy
- pandas
- matplotlib
- seaborn
- missingno
- pickle
- math, time, random, datetime

The statistical models also remained the same:

- Logistic Regression
- K Neighbours Classifier
- Decision Tree Classifier
- Random Forest Classifier
- Gaussian Naïve Bayes
- Gradient Boosting Classifier

Once again, the best model will be chosen by its cross-validation accuracy rate.

Coding:

There was little difference in the coding process as the same steps were followed. The packages and data were imported, the datasets were concatenated. The unidentified ' – ' was renamed 'None' to avoid bugging problems when coding for the machine learning algorithms. The categorical variables service, state, proto, and attack categories were changed to numeric variables. Again, only attack category DOS and Normal were kept. The data was then normalized for increased accuracy.

After the normalization, the first change came into play. All the features that were found useless in Chapter One were swiftly removed. These features were: dur, rate, sloss, djit, tcprtt, trans_depth, ct_src_dport_ltm, ct_flw_http_mthd, proto_3pc, proto_aris, proto_br-sat-mon, proto_cphb, proto_ddp, proto_emcon, proto_ggp, proto_iatp, proto_idrp, proto_ip, proto_ipnip, proto_ipv6-opts, proto_iso-ip, proto_leaf-1, proto_micp, proto_netblt, proto_pim, proto_ptp, proto_rsvp, proto_sccopmce, proto_sep, proto_sprite-rpc, proto_sun-nd, proto_tp++, proto_unas, proto_vrrp, proto_xnx-idp, service_dns, service_pop3, service_ssl, state_FIN, state_URN. Once these results were removed, the models were run, following the exact same code and steps as the prior chapter.

Results and Discussion

Table 1 displays the results for Accuracy; Table 2 displays the results for cross-validation accuracy.

Algorithm	Accuracy %
Decision Tree	99.73
Random Forest	99.73
K-Nearest Neighbours (KNN)	95.87
Gradient Boosting Classifier (GBC)	93.32
Logistic Regression (Log. Reg.)	89.90
Gaussian Naïve Bayes (GNB)	42.28

TABLE 1

Table 1 shows that, yet again, Decision Tree and Random Forest rank the highest in predicting whether or not a particular datapoint is an attack packet. To use the analogy from the prior chapter, if one thousand attacks were to occur the model would be accurately identified 997 (997.3) as DoS attacks, for both Decision Tree and Random Forest. It must also be noted that the rest of the top – those ranked at least 90% and above - models did not display any large decrease after the removal of the 'useless' features':

- Decision Tree and Random Forests accuracy decreased by 0.01%
- KNN decreased by 0.03%
- GBC decreased by 0.06%
- Logistic Regression showed a significant decrease, 0.2%
- GNB showed the most significant decrease, 8.22%

This shows, however, that there is no evidence for a significant decrease in the efficacy of the model, after the 'useless' features were removed. It also suggests that Logistic Regression and GNB were relying on features that were removed in order to identify the attacks, whereas, for example, Decision Tree and Random Forest clearly viewed those features as less important and thus the lack of decrease. It is interesting to note that while many features were removed, Random Forest still held tied for first with Decision Tree. This further affirms the need for evaluation of the accuracy results. The models were also ranked in the same exact order. Table 2 displays the cross-validation accuracies:

Algorithm	CV Accuracy %
Random Forest	92.65
GBC	92.13
Decision Tree	91.47
KNN	89.77
Logistic Regression	88.97
GNB	42.24

TABLE 2

The cross-validation accuracy, again, displays that the models had overfitted to the dataset as the accuracies decreased. It still follows the same pattern as before; the rankings are the same and Decision Tree did worsen after cross-validation. Random Forest's accuracy decreased by 8%; GBC's decreased by 1.19%. Decision Tree's accuracy decreased by 8.26%; KNN decreased by 6.1%. Logistic Regression only decreased by 0.93%; and GNB decreased by 0.04%. Just as before, while Random Forest saw a significant decrease in efficacy, it still remains to be one of the better models in classifying an attack packet. GBC, again, showed a pattern of being less prone to overfitting, and still does quite well, compared to the selection presented here, in classifying DoS attacks. Therefore, the previous comment that future research may consider keeping GBC with more data remains unchanged.

Next, it must be noted that the models performed better with the useless features removed, which is interesting to note because its normal Accuracy score decreased, but the cross-validated accuracy score has increased:

- Random Forest's accuracy increased by 0.05%
- GBC's increased by 0.04%
- Decision Tree increased by 0.03%
- KNN increased by 0.06%
- Logistic Regression and GNB, however, saw a decrease in efficacy: 0.07% and 8.22% respectively.

While these are not significant increases in accuracy by any means, the fact that the model's classification ability has improved suggests that the features that were removed do not show any significance in assisting with the identification of attack packets. In fact, this reaches one of the reports goal of identifying features that are not useful in predicting attacks whatsoever. However, as these are not large increases it also suggests that there are many features still within the data that may need to be removed in order to optimise the ability to identify attack packets. The cross-validation results also further reaffirm the inadequacy of using KNN and Logistic Regression. The fact that these decreased in accuracy, while the top models increased in accuracy, suggests these models were using features irrelevant to the cause of finding DoS packets. Future research may ignore these models completely.

Again, as Random Forest was the best model, feature selection was used to find out which features contributed the most in identifying the attack packets. Table 3 shows the results of the top ten significant features.

Feature	Importance
ct_state_ttl	0.083699
sload	0.063967
sttl	0.059481
dload	0.059384
sbytes	0.054462
dttl	0.051652
dinpkt	0.048836
smean	0.044197
synack	0.041624
ct_srv_dst	0.039468

TABLE 3

Table 3 shows the results. It seems that small improvements in the accuracy of the model has produced major changes in feature importance; the individual features strength in identifying an attack packet. Previously, both in Pearson's Correlation index and the random forest features selection in the previous chapter both found that sttl was the most significant feature in identifying an attack packet. This was true even when the dataset was cut at 50%. However, with the 'useless' features removed, it ranks significantly lower than before. Going from 0.085585 to 0.059481. Ct_state_ttl, on the other hand, has increased in importance in identifying attack packets. The same is true for s load, which was ranked ten in the previous chapter, and is now the second best feature for classification purposes. A possible explanation for this is the removed variables interacted with sttl which would increase its importance, the removal of those features meant that sttl by itself is not a good predictor of an attack packet. Future research may follow this line of enquiry because this opens up the possibility of grouping features together to help identify classification. Nevertheless, while it is no longer the most important feature, it is still significant enough for classification. Another interesting pattern that has emerges is that, previously, there were no features with an importance between 0.05 – 0.06; sttl had an importance of 0.085, and ct_state_ttl had an importance of 0.068, the rest were lower than 0.044. However, with the 'useless' features removed, the significance of the remaining features have increased. This may be explained by the fact that there are less features now, and thus naturally there is more weight to share around. Similar patterns still exist, however, compared to the previous chapter. For example, it seems source features have a greater weighting in deciding whether a datapoint is an attack packet compared to destination packets, but, again, this is not true at all times. This does raise questions when deciding which model to use for feature selection, this is because it seems odd that while the model increased marginally, the feature importance changed so drastically. On the other hand, it does affirm the fact that those features were not needed, as evidenced by the improvement in the model.

50% of the Dataset

After splitting it in half the dataset had 128,836 datapoints, however, this time there were only 157 features due to the prior removal. Prior predictions remain the same, where the model is expected to have lower cross-validation accuracy but the top ten features stay somewhat similar. Table 4 shows the normal accuracy of the machine learning models and Table 5 shows the cross-validation accuracy.

Algorithm	Accuracy
Decision Tree	99.80%
Random Forest	99.80%
K-Nearest Neighbours (KNN)	95.62%
Gradient Boosting Classifier (GBC)	93.43%
Logistic Regression (Log. Reg.)	89.89%
Gaussian Naïve Bayes (GNB)	42.17%

TABLE 4

In a similar vein as the previous chapter, Table 4 shows the same order ranking as Table 1 showing, again, that Decision Tree and Random Forest perform the best. The same pattern emerges as the previous chapter, that as the dataset size decreases, Decision Tree and Random Forest perform with higher accuracy. Not only that, just like the previous chapter, the increase in their accuracies, from Table 1, is 0.7%. It may be argued that this is peculiar, suggesting some pre-processing bugs, as the number of features had reduced, thus the extent to which it increases in accuracy should not be the exact same even if there is a general trend of increasing accuracy with lower datasets. It may also be the result of synthetic and natural data being mixed together, skewing the results, or even a coincidence. This may be more likely as exact decreases in the other models are not visible, if it was a coding inadequacy then the pattern would emerge for all the models as a function was created to run all these models in the first place. Future research, however, may elaborate on this pattern. GNB is still bad.

Algorithm	CV Accuracy %
Random Forest	94.71
GBC	93.33
Decision Tree	93.25
KNN	91.17
Logistic Regression	89.84
GNB	42.17

TABLE 5

The cross-validation yields similar results to the previous chapter. Random Forest was the best model in classifying attack packets at 50% of the dataset, again reiterating its ability in classifying DoS attacks. The same pattern is seen again where Random Forest, GBC, Decision Tree, Logistic Regression and GNB have higher cross-validation accuracy when the dataset size is decreased; KNN, however, increased again. For the models that have increased, future research may test the extent to which these models get worse as more data is introduced, or if there is a plateau at some point.

Random Forest's feature importance was examined again. Table 6 shows the results for the top ten features:

Feature	Importance
sttl	0.102895
ct_state_ttl	0.084435

sbytes	0.059444
sload	0.049951
dttl	0.048940
dload	0.048068
smean	0.042490
dbytes	0.038187
ct_srv_dst	0.038122
sjit	0.035720

TABLE 6

Once the dataset was cut in half, contrary to Table 3, sttl goes back in top as the most important feature in predicting an attack packet, and by a more significant margin as well compared to the previous chapter. Future research must include a greater amount of datapoints because it seems the datasets size makes a considerably difference in the models use of features and predicting DoS attacks. This is only evidenced by sttl becoming a feature with more importance, but also because it has followed the same pattern as the previous chapter wherein features not deemed important with the complete dataset are suddenly significant when the data is cut in half, and also, the ranking order also changes in relation to the dataset size. Dttl is another example of this. While it does have higher importance compared to Table 3, it is still ranked as the sixth most important feature. Furthermore, synack, which had an importance rate of 0.041 did not show up in this dataset cut in half. Thus, a consistent pattern of feature importance is observed across the chapters, even with the elimination of many features in this chapter. Therefore, a conclusion for the best features to use for detecting DoS attacks can only be made for this sized dataset; larger datasets may follow different results. Nevertheless, it is clear that sttl and ct_state_ttl seem to show consistent importance in identifying a DoS packet, suggesting some vulnerabilities within these features, the rest of the features are less clear cut.

Conclusion

In conclusion, removal of the 'useless' features has improved the models accuracy in identifying DoS packets, even if it was by a small margin. Chained with this observation, however, is that the most important features seem to have completely changed with the elimination of some features, suggesting that some of the eliminated features had some dependency with the features remaining, thus removal of those features may have provided a cleaner insight into the more important features. This line of enquiry may be further pursued in future research of this dataset.

Chapter 4: Top Features for Classifying DoS attacks

After going through some machine learning experiments, this will be the final one in aiming to reach the goals laid out in the introduction. The machine learning models in this chapter will be run by, firstly, eliminating the 'useless' variables which was found to have increased the accuracy of the classification process, then it will select the for the best features found in the previous Chapter. At the end the dataset will be split into half again, and the results will be discussed in a similar manner to the previous chapters.

The chapter will also be split into two sections: Methodology and Results & Discussion.

Methodology:

Packages:

The same packages were used for this chapter:

- numpy
- pandas
- matplotlib
- seaborn
- missingno
- pickle
- math, time, random, datetime

The statistical models:

- Logistic Regression
- K Neighbours Classifier
- Decision Tree Classifier
- Random Forest Classifier
- Gaussian Naïve Bayes
- Gradient Boosting Classifier

The best model will be chosen by its cross-validation accuracy rate. It is hypothesised that the models run in this Chapter should be the best model so far. This is because not only will it remove the 'useless' variables found in the EDA, but it will also be refined by only containing the features that ranked at least 0.001 or higher in the previous chapter. The rest will not be used in assessing whether or not the model is able to accurately identify an attack packet. It will be compared to the models in Chapter 3 as well, but not the models to Chapter 2. This is because Chapter 3's model performed the best, and thus it makes sense to compare it to the most reliable one so far.

Coding:

As the coding process has not significantly changed in this chapter, it will move onto the section where these features were selected.

After the normalization process and the dropping of the 'useless' variables, a new data frame called rf_features was created to extract only those features that ranked 0.001 or higher. The following features were selected: ct_state_ttl, sload, sttl, dload, sbytes, dttl, dinpkt, smean, synack, ct_serv_dst, dbytes, sinpkt, ct_dst_ltm, ackdat, dmean, dpkts, ct_srv_src, sjit, state_INT, ct_src_ltm, ct_dst_ltm, spkts, ct_dst_sport_ltm, stcpb, dloss, and finally label as this is the target variable. Altogether, 26 features, including label, were chosen. This is a significant decrease from Chapter 2, containing 197 features.

Results and Discussion

Below are the tables displaying the results for accuracy and cross-validation accuracy.

Algorithm	Accuracy
Decision Tree	99.68%
Random Forest	99.68%
K-Nearest Neighbours (KNN)	95.62%
Gradient Boosting Classifier (GBC)	92.68%
Logistic Regression (Log. Reg.)	88.22%

Gaussian Naïve Bayes (GNB)	75.88%

TABLE 1

Decision Tree and Random Forest rank the highest in identifying attack packets, showing a consistency across all chapters. This trend did not change either with feature removal, nor feature selection. However, compared to the previous chapter, the accuracy did diminish:

- Decision Tree and Random Forest's accuracy decreased by 0.05%
- KNN's decreased by 0.25%
- GBC decreased by 0.64%
- GNB increased by 33.6%

GNB had an extremely significant increased compared to the other models. This suggests that GNB performs better with less amount of features, or that these features were optimum for GNB to accurately classify an attack packet. It is unlikely to be the first assumption, as Chapter 3 had removed features as well but its accuracy had decreased instead. Unfortunately, in comparison to the other models, it still performs quite poorly, and thus not suited for the task of predicting DoS attacks.

Algorithm	CV Accuracy %
Random Forest	92.02
GBC	91.32
Decision Tree	90.97
KNN	89.22
Logistic Regression	87.54
GNB	75.95

TABLE 2

The cross-validation accuracy has also conformed to the trend from the previous chapters. All models performed worse after this evaluation, and the ranking of the models did not change either with Random Forest still performing the best. The accuracy change was the following, compared to Table 1:

- Random Forest decreased by 7.66%
- GBC decreased by 1.36%
- Decision Tree decreased by 8.71%
- KNN decreased by 6.4%
- Logistic Regression decreased by 0.68%
- GNB increased by 0.7%

This has also broken the trend seen in previous pattern. Whereas before all of the models would show signs of overfitting to the data, thus the decrease in accuracy after cross-validation, GNB was found to be underfitting the data, hence its increase after cross-validation. As hinted in previous chapters, evaluation metrics are crucial in optimising the machine learning model.

Table 2 also shows us that when the features were reduced to only the 26 selected, the accuracy of the model suffered compared to the previous chapter in which there was only the elimination of the 'useless' features.

- Random Forests accuracy decreased by 0.63%
- GBC decreased by 0.81%
- Decision Tree decreased by 0.5%
- KNN decreased by 0.55%
- Logistic Regression decreased by 1.43%
- GNB's accuracy increased by 33.71%

It may be inferred that the decrease in accuracy suggests that some of the features removed, those ranked 0.001 or below may have in fact been useful in predicting whether or not a datapoint was classified as an attack packet or not. This may be because of the nature of DoS attacks itself, in that novel attacks will always occur and thus those features not ordinarily vulnerable may still be the target of an attack. However, this has the impractical implication that no features should be removed so the models may be prepared for all forms of attacks, both the orthodox and the novel.

The results from Table 2 also show us that, while there has been a decrease in accuracy, it was not a large decrease at all. Out of a thousand DoS attacks, the Random Forest model from the previous chapter would detect 926 of these attacks, whereas the model in this chapter would detect 920. The report does acknowledge that even some DoS attacks left undetected my cause a mountain of problems, but the small decrease in accuracy means that the features selected in this chapter are definitely useful in predicting DoS attacks, only not by themselves.

Next, the top ten significant features are displayed below:

Feature	Importance
sttl	0.184078
ct_state_ttl	0.132480
dload	0.070340
sload	0.067643
sbytes	0.059520
smean	0.051875
ct_srv_dst	0.050054
dmean	0.040127
sinpkt	0.038400
ct_srv_src	0.036386

TABLE 2

The significant increase in importance is unsurprising as there are only 26 features left in the dataset, which means the feature importance will only need to be distributed among these features. However, what is slightly surprising is the extent to which sttl is given importance compared to ct_state_ttl. This is because the previous chapter found that with the 'useless' features removed, ct_state_ttl was more significant in determining an attack packet. Thus, with a greater number of features removed it is expected that ct_state_ttl to continue being the most significant, however that is not the case. A possible explanation for this is that the features that remained in the previous chapter had some linear dependency with ct_state_ttl, and those features remained. Whereas the features that were removed also happened to be the ones that had more linear dependency with sttl, and thus ranked lower. Nevertheless, a consistent finding across all the chapters is that sttl and ct_state_ttl are the most important features in this dataset in detecting DoS attacks. One trend has continued where it may be confirmed that sttl and ct_state_ttl will remain the top two significant features, the order of the rest however differ from that of the previous chapter. This inconsistency in order rankings suggest that a greater number of features should remain in the dataset until it this change is more thoroughly explored, this may also explain why this Chapters model did slightly worse.

50% of the Dataset

Algorithm	Accuracy
Decision Tree	99.80%
Random Forest	99.80%
K-Nearest Neighbours (KNN)	95.62%
Gradient Boosting Classifier (GBC)	93.43%
Logistic Regression (Log. Reg.)	89.89%
Gaussian Naïve Bayes (GNB)	42.17%

TABLE 4

It is interesting to note that the accuracy results here were the exact same as the previous chapters, after it was cut in half. From this it may be inferred that the 26 features were a lot more concise in identifying attack packets than when the complete dataset was provided. Apart from this, the same trends are followed. It must be noted, even if the model has performed poorly, that once the dataset was cut in half GNB went back to 40% accuracy rates. This may suggest that as the dataset size increase, and the features are accurately optimised, GNB may a useful model in classifying attack packets.

Algorithm	CV Accuracy %
Random Forest	94.68
GBC	93.33
Decision Tree	93.25
KNN	91.17
Logistic Regression	89.84
GNB	42.17

TABLE 5

The exact same pattern emerges after the cross-validation evaluation of the accuracy. After splitting the dataset in half, the results were the exact same as the previous chapters. The only difference was Random Forest which was only decreased by 0.03%. This provides greater weight to the fact that the 26 features selected this chapter ay be the most important features in classifying an attack packet. This is on top of the fact that the full sized model only had a small decrease in accuracy.

Feature	Importance
sttl	0.121206
ct_state_ttl	0.082166
sbytes	0.057532
sload	0.052262
dttl	0.049676
dload	0.047616
smean	0.042376
dbytes	0.041695

ct_srv_dst	0.040628
sjit	0.037318

TABLE 6

Sttl and ct_state_ttl were both ranked as most important even after the dataset was cut in half. Destination features comprise three of the top ten important features (dttl, dload, dbytes), compared to the full dataset where it only comprised of two (dmean, dload). The juggling of feature importance, apart from the top two sttl and ct_state_ttl, still remain a constant pattern when cutting the dataset in half. It may be safely assumed that the more data there is available, the greater the accuracy in assessing the strengths of the features, and thus while the 50% dataset show us different features throughout, the features selected from the complete dataset is where the focus should be. Nevertheless, cutting the dataset by 50% has provided with interesting findings, such as finding the similarity in Table 4 and 5 in this chapter and the previous chapter.

Conclusion: Have the goals been reached?

These were the goals set out in the beginning of the report: establish a correlation between the features in the dataset and the nature; establish which features have the strongest or closest correlation to the nature of the packet; establish which feature have little or no correlation in determining the nature of the packet as either normal or attack packet; determine which features are most important in deterring an attack, and finally, being able to pinpoint the smallest number of features that can be used in determining an attack packet. The conclusion will go through each chapter and assess to what extent were these questions answered. Reflection for what may be done in future research will also be embedded.

Chapter One, the exploratory data analysis chapter attempted to establish a correlation between the features in the dataset and the nature of the packet, however, Pearson's Correlation index was not reliable enough to warrant the adherence of these results. It did establish features with close correlation to the nature of the packet, however, its accuracy was not measured and it was found in subsequent chapters that machine learning models fared better. This chapter did, however, affirmatively establish which features were not important at all in determining the nature of the packet. These features were:

dur, **rate**, sloss, djit, tcprtt, trans_depth, ct_src_dport_ltm, ct_flw_http_mthd, proto_3pc, proto_aris, proto_br-sat-mon, proto_cphb, proto_ddp, proto_emcon, proto_ggp, proto_iatp, proto_idrp, proto_ip, proto_ipnip, proto_ipv6-opts, proto_iso-ip, proto_leaf-1, proto_micp, proto_netblt, proto_pim, proto_ptp, proto_rsvp, proto_sccopmce, proto_sep, proto_sprite-rpc, proto_sun-nd, proto_tp++, proto_unas, proto_vrrp, proto_xnx-idp, **service_dns**, service_pop3, service_ssl, **state_FIN,** state_URN, categorized as the 'useless' features throughout the report. This was a success as it showed that EDA was able to eliminate features without need for machine learning algorithms. It did not determine which features were most important in deterring an attack as it only determined the correlation levels, neither did it identify what the smallest number of features could be. Nevertheless, it did overall contribute to this report, and overall research, as it was able to determine features that were not important to the goal of classifying a DoS attack packet. In reflection, a future EDA may spend more time preprocessing in order to find methods to combine source and destination features, those marked with s and d such as sttl or dttl. This will be elaborated on down below in concluding the subsequent chapters.

Chapter Two, the initial machine learning models where there was no manipulation of the features did help establish features that were strong in determining an attack packet. Sttl and ct_state_ttl showed up in the complete dataset and the 50% dataset, both these features had appeared near the top in Chapter Ones correlation check as well. As the goal was to create a control model for future chapters to test against, it did not establish features with low importance. The fact it repeatedly showed sttl and ct_state_ttl as significant in classifying an attack packet shows these two are definitely the more important features in deterring an attack. It did not find the smallest number of features that may be used in determining an attack packet. For future research, more machine learning algorithms may be implemented as different results may show up. This chapter also affirmed the strength of Random Forest in classifying objects. However, out of the goals laid out, it only helped in establishing the importance of sttl and ct_state_ttl. It was also an important chapter for comparisons in subsequent chapters.

Chapter 3 saw a huge improvement in reaching closer to the projects goals. Firstly, some features were established as being extremely strong in classifying and deterring an attack. These features were sttl and ct_state_ttl which showed up consistently across the chapters. However, it was not able to establish an ordered ranking apart from these two, because as this chapter removed the 'useless' features, the rankings for the remaining features also changed. Secondly, the fact that the model had improved after the 'useless' features were eliminated shows that the goal of determining features with little significance was reached. It did not, however, pinpoint the smallest number of features that could be used. Future research may decide to examine a dataset with more datapoints. This is because Chapter 3 showed, as well as Chapter 2 and 4, that the size of the dataset changes the importance of the features strength in determining an attack packet, which is to say, that as the dataset increases, it increases in accurately identifying which features are doing the best. Having said this, it is still unlikely that sttl and ct_state_ttl will be absconded from the best features in classifying attack packets.

Chapter 4 continued to reaffirm the findings of the previous chapters, which is that sttl and ct_state_ttl were the strongest in classifying an attack packet, and thus, the best at deterring an attack as well. It may be argued that it did in fact establish features that had little to no importance in determining the nature of the packet. This is because even though only 26 features remained, the extent to which the accuracy of the model decreased was quite small. Furthermore, when the dataset was cut in half, the results were nearly the exact same. The only difference was Random Forest's cross-validation accuracy which decreased by 0.03%, an insignificant decrease. This also means that this Chapter reached the goal the previous chapters did not which is to find the smallest number of features that can be used in determining an attack. The lack of significant decrease in accuracy suggests that the 26 features chosen in the end - ct_state_ttl, sload, sttl, dload, sbytes, dttl, dinpkt, smean, synack, ct_serv_dst, dbytes, sinpkt, ct_dst_ltm, ackdat, dmean, dpkts, ct_srv_src, sjit, state_INT, ct_src_ltm, ct_dst_ltm, spkts, ct_dst_sport_ltm, stcpb, dloss and label – is the smallest amount of features needed in order to predict a DoS attack packet. This has important real-world implications as it saves a lot more computing output and resources, thus saving time as well, allowing for faster results.

Overall, correlations between the features of the dataset and the nature of the packet was established in Chapter 1 through Pearson's Correlation Index; the report found that sttl and ct_state_ttl are the two strongest features in classifying the nature of the packet, and thus also important in future research in deterring an attack; dur, **rate**, sloss, djit, tcprtt, trans_depth, ct_src_dport_ltm, ct_flw_http_mthd, proto_3pc, proto_aris, proto_br-sat-mon, proto_cphb, proto_ddp, proto_emcon, proto_ggp, proto_iatp, proto_idrp, proto_ip, proto_ipnip, proto_ipv6-opts, proto_iso-ip, proto_leaf-1, proto_micp, proto_netblt, proto_pim, proto_ptp, proto_rsvp, proto_sccopmce, proto_sep, proto_sprite-rpc, proto_sun-nd, proto_tp++, proto_unas, proto_vrrp, proto_xnx-idp, **service_dns**, service_pop3, service_ssl, **state_FIN,** and state_URN were found to definitely have no impact or significance in determining the nature of a packet, which was later affirmed by the improvement in the models performance, and finally, the smallest number of features that can be used in determining an attack packet is 26: ct_state_ttl, sload, sttl, dload, sbytes, dttl, dinpkt, smean, synack, ct_serv_dst, dbytes, sinpkt, ct_dst_ltm, ackdat, dmean, dpkts, ct_srv_src, sjit, state_INT, ct_src_ltm, ct_dst_ltm, spkts, ct_dst_sport_ltm, stcpb, dloss and label.

It must be noted that Random Forest did the best throughout in providing these results. Future research should attempt to introduce more models and compare the results in order to optimize for the best features. Furthermore, a future re-examination of the dataset would focus more on the pre-processing and EDA stage rather than using the machine learning models. This is to find ways to combine the source and destination features, because the results throughout the report seemed to show that if, for example, sttl or sload showed importance in determining an attack packet, dttl or dload would be close behind. At every turn there was a connection between the source and destination packets. Combining these may help strengthen models in defeating DoS and DDoS attacks. This also suggests an innate vulnerability with features with sttl and ct_state_ttl that future research can focus on.

References:

[1] Moustafa, Nour, et al. "Anomaly Detection System Using Beta Mixture Models and Outlier Detection." *Progress in Computing, Analytics and Networking. Springer, Singapore, 2018.* 125-135.

[2] Moustafa, Nour, et al. "Flow Aggregator Module for Analysing Network Traffic." *Progress in Computing, Analytics and Networking. Springer, Singapore, 2018.* 19-29.

[3] Moustafa, Nour, et al. "A Network Forensic Scheme Using Correntropy-Variation for Attack Detection." *IFIP International Conference on Digital Forensics. Springer, Cham, 2018.*

[4] UNSW_NB15 | Kaggle. 2021. UNSW_NB15 | Kaggle. [ONLINE] Available at: https://www.kaggle.com/mrwellsdavid/unsw-nb15.

[5] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1-6, doi: 10.1109/MilCIS.2015.7348942.

[6] Gurley, R., 2000. *Intrusion Detection*. Sams Publishing.

[7] W. Hu, W. Hu and S. Maybank, "AdaBoost-Based Algorithm for Network Intrusion Detection," in *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 2, pp. 577-583, April 2008, doi: 10.1109/TSMCB.2007.914695.

[8] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems

[9] ibid

[10] Moustafa, Nour, and Jill Slay. "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset." *Information Security Journal: A Global Perspective* (2016): 1-14.

[11] Understanding Denial-of-Service Attacks | CISA. 2021. Understanding Denial-of-Service Attacks | CISA. [ONLINE] Available at: https://us-cert.cisa.gov/ncas/tips/ST04-015

[12] Sílvia Farraposo, Laurent Gallon, Philippe Owezarski, "Network Security and DoS Attacks",April 2005.

[13] ijirset.com. 2021. No page title. [ONLINE] Available at: http://ijirset.com/upload/2016/june/247_A%20Review.pdf.

[14] ijirset.com. 2021. No page title. [ONLINE] Available at: http://ijirset.com/upload/2016/june/247_A%20Review.pdf.

[15] D. Moore, G.M Voelker and S. Savage, "Inferring Internet Denial of Service activity," *ACM Trans. Comput Syst.,* vol. 24, no. 2, pp. 115-139, 2006.

[16] ijirset.com. 2021. No page title. [ONLINE] Available at: http://ijirset.com/upload/2016/june/247_A%20Review.pdf.

[17] F. Lau, S. H. Rubin, M. H. Smith and L. Trajkovic, "Distributed denial of service attacks," *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics. 'cybernetics evolving to systems, humans, organizations, and their complex interactions' (cat. no.0*, 2000, pp. 2275-2280 vol.3, doi: 10.1109/ICSMC.2000.886455.

[18] Forouzan, Behrouz A. (2007). Data Communications And Networking (Fourth ed.). Boston: McGraw-Hill. pp. 621–630.

[19] M.Aijaz, S. Parveen," Analysis of Dos and DDoS Attacks", International Journal of Emerging Research in Management &Technology, Volume-5, Issue-5.2012

[20] Sonar, K. and Upadhyay, H., 2014. A survey: DDOS attack on Internet of Things. *International Journal of Engineering Research and Development*, *10*(11), pp.58-63.

[21] Sonar, K. and Upadhyay, H., 2014. A survey: DDOS attack on Internet of Things

[22] Tukey, John W. "The Future of Data Analysis." *The Annals of Mathematical Statistics*, vol. 33, no. 1, 1962, pp. 1–67.

[23] ijirset.com. 2021. No page title. [ONLINE] Available at: http://ijirset.com/upload/2016/june/247_A%20Review.pdf.

[24] DeepAI. 2021. Decision Tree Definition | DeepAI. [ONLINE] Available at: https://deepai.org/machine-learning-glossary-and-terms/decision-tree. [Accessed 27 August 2021].

[25] GreatLearning Blog: Free Resources what Matters to shape your Career!. 2021. What is Gradient Boosting and How is it different from AdaBoost?. [ONLINE] Available at: https://www.mygreatlearning.com/blog/gradient-boosting/

[26] GreatLearning Blog: Free Resources what Matters to shape your Career!. 2021. Valentine's Day: Invest in Education or Roses? - Great Learning. [ONLINE] Available at: https://www.mygreatlearning.com/blog/cross-validation/v