Determining if cyberattacks can be detected and analysed by using network logs from devices

Myron Furtado
19703402

19703402@students.lincoln.ac.uk

I want to know if DDoS attacks be detected using network log data

School of Computer Science

College of Science

University of Lincoln

Submitted in partial fulfilment of the requirements for the
Degree of BSc(Hons) Computer Science

Supervisor: Dr Yvonne James

April 2022

Acknowledgements

Firstly, I would like to thank my supervisor Yvonne James that has been there to help me with ideas when looking for the topic and then answer questions. I would also like to thank my Tutor, Dr Heriberto Cuayahuiti for doing a Microsoft team meeting with me and being supportive when I was thinking about how to approach this project.

Abstract

This project tries to confirm if cyberattacks such as a DDoS attack can be detected using machine learning models on network log data collected from a network-connected computer. In today's world cyberattacks happen all too often and they cause financial and reputational damage to the target individuals/ organisations/ companies. This is a major motivator for cybersecurity personnel to work on methods to detect and mitigate the effects of Cyberattacks such as a DDoS attacks. This project explores coding a DDoS attack script in python to launch the attack, then utilises Wireshark to collect packet Network data. This data has features which can be classified by using a machine learning model. Then create a machine learning model to perform predictions on the dataset we collected from the DDoS attack. The results of this process can be used by network/system admins to help institute security measures and policies for more robust network security.

Keywords: Cyberattacks, network logs, Log file analysis, Network anomalies, NetFlow data, Machine learning Models, Classifying data, Cybercrime

Table of Contents

List of Figure

List of Tables

Chapter 1

Introduction

1.1 Introduction to the Cyberattacks

The internet is a vast space that has changed our world. It has reshaped technology, businesses, human interactions, society, and information exchange. Around 10 billion devices are connected to the internet as of recently. This however also increases the number of devices that can be at risk of being attacked and compromised. Cyberattacks can come in many forms such as Viruses, spyware, phishing attacks and so on. The attackers are after sensitive information so they can share it publicly or hold it at Ransome for financial or personal gain (Alghamdie, 2021). For companies, cyberattacks can add significantly to their operating cost by requiring additional security measures, litigation costs, fines and penalties, not to mention the loss of customers and reputation (Kamiya et al., 2021, 721). The figure below breaks from a 2012 study that breaks down the common motivation of attackers based on what they want. And the most prevalent reason, at 30% is to gain access to private/personal sensitive information.

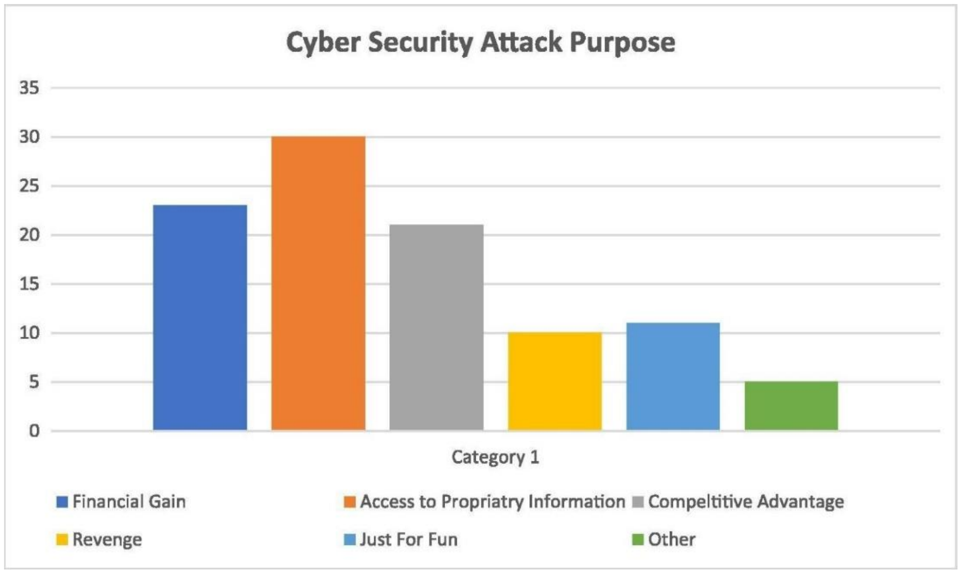


Figure 1- Common motivation behind Cyberattacks bar chart (Alghamdie, 2021, 4)

There are many types of attack vectors such as DDoS attacks, Viruses, spyware, SQL injections and so on. The attackers are after sensitive information that can either be made public or held at ransom till, they get paid to return the information (Alghamdie, 2021). For companies, cyberattacks can add significantly to their operating cost by requiring additional security measures, litigation costs, fines and penalties, not to mention the loss of customers and reputation (Kamiya et al., 2021, 721).

One method of detecting cyberattacks is to perform real-time analysis of network traffic and system files, but this is a very difficult process and cannot be relied upon a 100%. System Log analysis is another way of checking what has happened in the system as they provide concrete proof of the activities that happened in the machine. One of the newest methods of detection is the training of machine learning/deep learning models to learn and detect cyberattacks (Liu et al., 2021, 10985). To ensure that the models were making credible detections they are compared to the results of other methods of analysis (Liu et al., 2021, 10985). Deep learning and machine learning methods are far from perfect but are improving every day.

This project will try to study/analyse network log data using datasets collected by me from a Mac or Windows machine. It will contain network packet data received/sent from the machine, collected when a cyberattack was happening. This is supposed to identify abnormalities/irregularities in the network traffic data. I am confident this project will give me insights into some areas of the job role of a network engineer. As I am interested in applying for this role after completing university.

1.2 Main ideas to be examined, developed and discussed

The main question asked in this project is if cyberattacks can be detected by performing an analysis of network data that is recorded from the concerned computer. Cyber attacks come in many forms but this project will focus only on Denial of service (DoS) and Distributed Denial of Service (DDoS) attacks as going diving into other threats will greatly increase the size of the project. This is because not every threat can be detected using the same model, each threat needs a threat specific technology to be detected. DDoS attacks are very common on corporate entities and government agencies.

The DDoS script used for this project is not very powerful but still does send quite a lot of HTTP flood traffic over to the target. This script will perform the DDoS attack on the target device. Then Wireshark is used to collect the packet data using the npcap library and displays it on the screen. Wireshark is a packet sniffer tool available on Windows and macOS, both of which have been used in some stages of the project as the target devices. Using Wireshark network data was collected. Wireshark allows the user to record many components of network traffic data. For example, the project dataset needed the source and destination port numbers in the exported file, so these had to be selected during the export process. The next step will involve creating an ipynb file, which is a python file that can be used in the Jupiter notebook environment. Then import all the libraries that are going to be used for plotting and working with the data. The project dataset then gets imported into the python file and goes through phases that include data visualization, which is the stage where the data is explained using plots and tables and then the data is pre-processed to be used in the machine learning models, which is the next step.

The data is then run through a Machine learning model. In the project, the Machine learning model that was used is called k-nearest neighbor/kNeighbors Classifier (kNN) from the sklearn library, as it can predict the dataset used with really good accuracy. In addition to this, the author also implemented 2 additional ML models, a Linear Regression (LR) model, and a Random Forests Classifier (RFC) model both from the sklearn library as well.

1.2 Aims

This project aims to analyse and study Network packet log data by utilising Machine learning models to do the analysing and make the decision. This will help to determine if the machine models selected can be used to detect traces of DDoS attacks from the network packet logs. The data is collected on a macOS device as the target and the DDoS attack was performed on a Windows system using a python script.

1.3 Objectives and Milestones

1. Conduct a thorough Literature review
2. Research how to code a DDoS script, then code it in python
3. Test the DDoS script to make sure it works
4. Research what program to use to collect the Network log data
5. Launch DDoS attacks and record the resulting log data
6. Clean the Recorded data to remove Null, None and empty values

7. Select a Machine learning model from Scikit-learn to use for analysing data
8. Code the Machine learning part of the project, which includes importing the dataset to X and Y matrices, splitting the new dataset matrix into test and train, and training the model
9. The Model should output data that need to be displayed to the user. E.g., If DDoS has been detected or not, graphs breaking down and visualising the input dataset- IP address breakdown, port breakdown and so on
10. Write up the Final report explaining the process

Chapter 2

Literature Review

2.1 Background

Network logs can be analysed using machine learning algorithms to detect abnormalities in the log data. These models can then be used to predict cyberattacks. Bad actors can gain access to network systems through the network with the intent to cause disruption, breakdown, operational failure, and service interruptions. Studies such as (Ning and Jiang, 2021, 1154) have studied well-documented cyberattacks including Stuxnet on Siemens and Black energy that affected electrical grids in Ukraine. Both incidents caused physical and financial damage. One way to tackle this would be to use application network monitoring software but attacks on the hardware level will slip by undetected in this case, a better way to boost network security would be to study the network logs and analyse the data.

2.2 DDoS attacks

The internet has revolutionized the world since its inception. In their paper (Musumeci et al., 2021) say “it reshaped the technology, business communication, society, economic and many more.” They also note that it increases the level of risk for users connected to the internet. Distributed denial of service (DDoS) attacks is very dangerous for servers and systems that are connected to the internet. They explain that in a DDOS attack the attacker sends massive amounts of malicious traffic to cause the CPU or NIC (Network interface card). (Musumeci et al., 2021) notes that multiple vectors of attack can be deployed using dynamic/spoofed IP addresses to perform a combined attack and since there are massive numbers of IP addresses coming in making it is difficult to block them all by backlisting IP addresses consistently. “The most utilized DDoS attacks are typically grouped in the following categories: TCP SYN flood, UDP flood, ICMP flood and HTTP flood”. Another serious type of DDoS attack is the TCP SYN attack which is used a lot in today’s environment. This attack exploits the “TCP connections’ initiation packets to target the victim” (Musumeci et al., 2021). They discovered three ways to prevent and mitigate DDOS attacks. These are:

- Source based detection, where the incoming IP addresses are blacklisted
- Destination based detection, where the target system performs detection
- Network-based detection, where switches and routers are used to detect incoming attacks

In their paper (Musumeci et al., 2021) perform network-based detection by implementing defence mechanisms at the SDN (Software defined network) switches and block traffic at the data plane level while also saving the SDN controller from breakdown or going out of service.

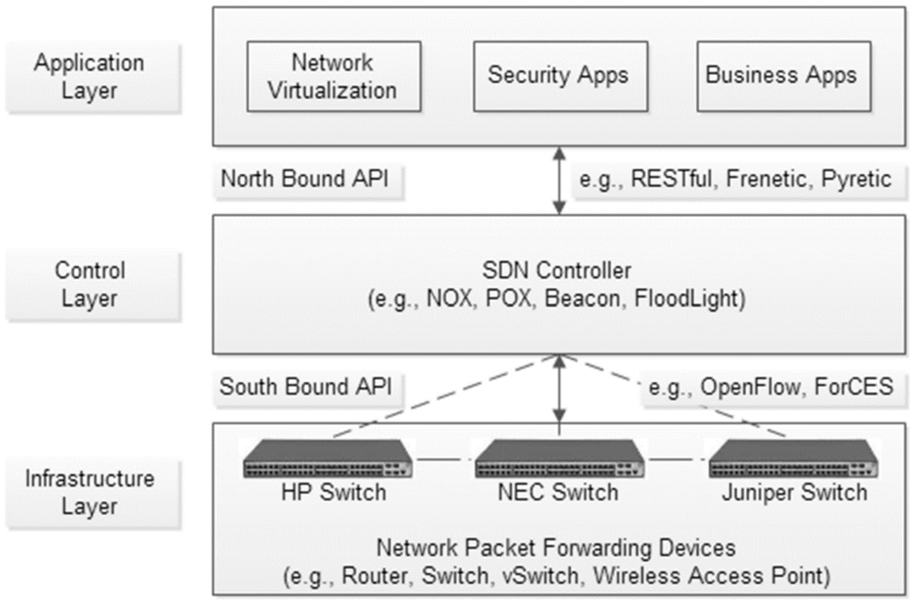


Figure 2- Basic SDN Architecture (Sahoo et al., 2019)

2.3 New emerging technologies to combat cyberthreats

Recent Developments: In their paper (Sahoo et al., 2019) they identified SDN (Software Defined Networks) as the new technology that many organisations need as a defence mechanism against cyber threats such as DDoS attacks. It helps solve network classification problems by using software to keep tabs on the traffic flow and root out DDoS or suspicious traffic. In my opinion, while this improves security it is not unbreakable as it is still possible to find new exploits in different layers of SDN. So, it is important to keep an eye on the logs in conjunction with SDN deployment.

The ELK stack is a package of open-source software (i.e., Elasticsearch, Logstash, and Kibana) that allows for the extraction of information for the log files and collects the required data metrics for processing (Liu et al., 2021, 10986). These provide a very concrete way of protecting potential high-risk targets like healthcare institutions. For example, healthcare devices can be vulnerable to Denial of Service (DoS) attacks due to the Software-defined Network (SDN) controller's limitation with flow tables (Huertas et al., 2021, 2719-2720). These can be mitigated by using policy-based architecture for Multi-Access edge computing (MEC) that allows it to detect and protect against cyberattacks that exploit SDN's weakness. This can help protect wearable health devices that are used by patients which will help save their lives and associated costs (Huertas et al., 2021, 2720). Research journals such as (Yang et al., 2019, 6344) have used log monitoring using the ELK stack to make networks safer. Collecting data for network logs and NetFlow logs using the Filebeats tool to convert the data to a visual format. Then combining that data with the NetFlow log data using the ceph file system (aka, CephFS) allowed them to compare the performance of the Reliable Autonomic Distributed Object Store (RADOS) gateway. This ensures that the people making the decisions on issues of network security have the most accurate, up-to-date, and safe information on the state of network security allowing them to make informed decisions.

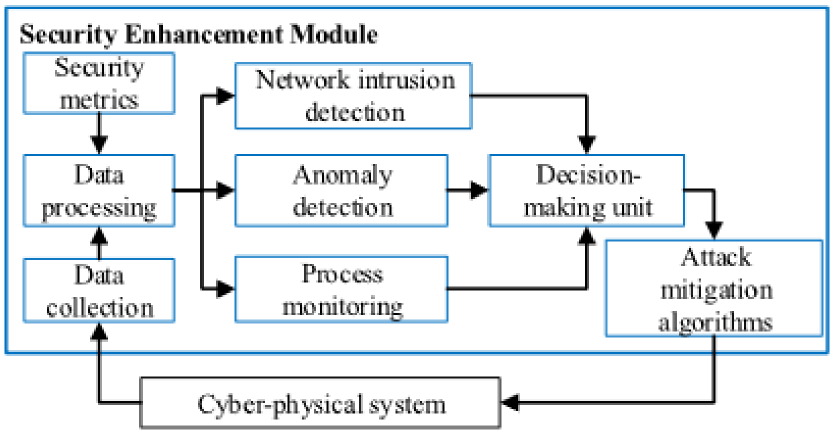


Figure 3 - Advanced ML models used for cyberattack detection (Ning and Jiang, 202, 1158)

Deep neural networks (DNNs) are Artificial intelligence models that are well known to be used for making predictions. First, DNNs are fed data (in our case network log data) and then the training data can be used on other test datasets to make predictions on the data we need to classify. DNN models do have a problem, when upper and lower neurons are fully connected the model can form a connection resulting in overfitting of the data (Vinayakumar et al., 2019).

The most cutting-edge technology being experimented on for providing a scalable cyberattack detector uses a deep neural network (DNN) as produced in the study by (R et al., 2019, 41525) the models used are flexible and effective at learning different types of unpredictable cyberattacks. The way the attacks are evolving requires a scalable fast-paced method of learning which facilitates the improvement of the algorithm which can accurately detect most of the attack types it has studied up to that point. The main advantage of using DNN is the vast number of datasets available out there that can be used to train the algorithm.

The project will need a plan to test if the analysis work can identify threats. This can be done by generating various attack scenarios to test if the systems fail to detect attempts of cyberattack. Second, it would be to test again with the highest intensity attack and lowest intensity attack to see what changes that could have (Ning and Jiang, 2021, 1156). The test results from here will be able to help give insight into what policies can be adopted to minimize and mitigate future attacks.

Another tool available for data visualization functionality that allows for tracking network traffic with more options for editing. This can then be aplite to machine learning models such as DNN. This advanced feature allows for real-time data management to interpret if a cyber-attack is taking place, if a cyberattack is happening then network administrators can be alerted to it. One downside of this is that the model can raise false positives.

Kozik developed a combo of NetFlow’s that contained a machine learning classifier, in which the Map reduce model is used. Before that Kiran and Chhabra investigated real-time classification of network data using a supervised model and achieved over 90% accuracy in correctly classifying their elephant and mice data. This model might be able to translate the accuracy of our dataset as well.

Chapter 3

Methodology

Project Methodology can be defined as a set of conventions that a team working on a project agrees to follow while working on a project. In their book, they explain that project management is meant to make better use of the existing resources by making the workflow horizontal as well as vertical (Kerzner, 2003). They explain Project management is the planning, organising, directing and controlling of resources to split up the objectives and complete them as set goals/objectives. This requires the project manager, in this case, just me since it is an individual project assigning a completion date for each objective. Since this is an individual project the project management does not allow for vertical scaling (e.g., assigning more people to a team). The project can only be completed faster by utilising horizontal scaling (e.g., working on more than one objective/task at a time). This can cause difficulties during project progression since there aren't other team members to check my progress and keep me in line making sure I am hitting my deadlines for the individual tasks. This could contribute to me losing sight of key goals and deviating from my original aims and going beyond the scope of my original project framework. This makes the Project management lifecycle (APM) necessary to increase the chance of successfully completing the project and set proper time management for it.

3.1 Project Management

This section will be used to discuss some of the tools and project management techniques that were available for use and discuss which ones were used or not used.

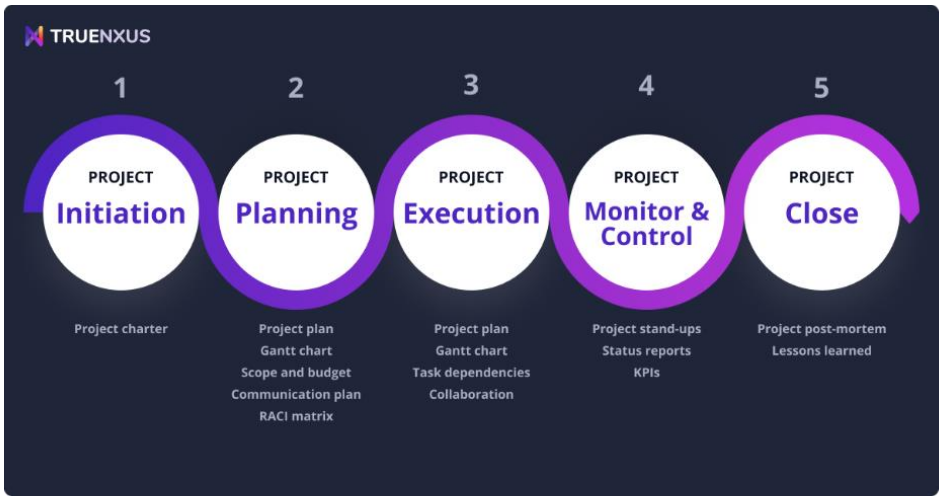


Figure 4 - Five phases of Project Management (Friedman, 2020)

Based on the 5 phases of the PM model the following aims were drawn up to start the project, they are listed below:

- Conduct a literature review to find learn about the subject and technologies involved
- Requirement gathering and making a list of things to set up before starting coding
- Research and code the DDoS script that will be required to collect the DDoS dataset needed for the next step
- Run the DDoS attack and collect the dataset
- Clean the dataset and export it to the required format

- Import libraries needed to perform the setup for the Machine learning environment
- Make a requirement list of what the Machine learning model should output
- Code the Machine learning model. Includes data pre-processing, training model, using the trained data on the test set, and output of the desired results on the screen
- Test the Machine learning model under different conditions
- Wire the final report and fulfil other requirements

3.1.1 Project management principles/Project Characteristics

This project is fundamentally not normal as normally it would be a team of people working on different tasks and then bringing it together to finish it, but here it is an individual project, it is only going to be me that manages the project all through the process.

As the sole author of the project, they also do not have much experience in both areas of the project, they will have to learn how to perform DDoS attacks, collect log data and Perform Machine learning operations. Sufficient time should be allocated to work on the Machine learning section of the artefact. Since these are relatively new topics for the user there will be a learning curve before they become proficient at programming DDoS scripts and machine learning programming.

The project also has a deadline which limits the amount of time that is available to complete the project. The author does not however have to worry about the budget of this project as most of the requirements and tools needed are either open source or available freely. The budget is therefore going to be very low or free of cost altogether.

The project overall had its major objectives all listed in the proposal but the smaller objectives were not decided on at the start. As the result of previous tasks would decide what the next task to do would be. An example of this is cleaning the DDoS dataset would require collecting the DDOS dataset first, that in turn would rely on having a DDoS script. This meant the projects had realistic and quantifiable goals but the smaller tasks to do were not clearly defined at the start.

Project scheduling was handled using a Gantt chart that had most of the main tasks that need to be carried out to complete this project and each task and section has due dates. Some of the tasks were planned and worked on in parallel to reduce the time spent on implementing the project artefact.

3.1.2 Gantt chart

A Gantt chart was created at the start of the project and submitted along with the Project Proposal, it contained most of the tasks that needed to be performed to complete the project and each task had a start and end date assigned to it. The chart was also separated into 3 sections based on the work that needed to be done for the 3 individual assignments that this project is comprised of and each section also had its due dates. These task entries were then updated as each task was progressed through and completed. The updated Gantt chart is shown in the figure below.

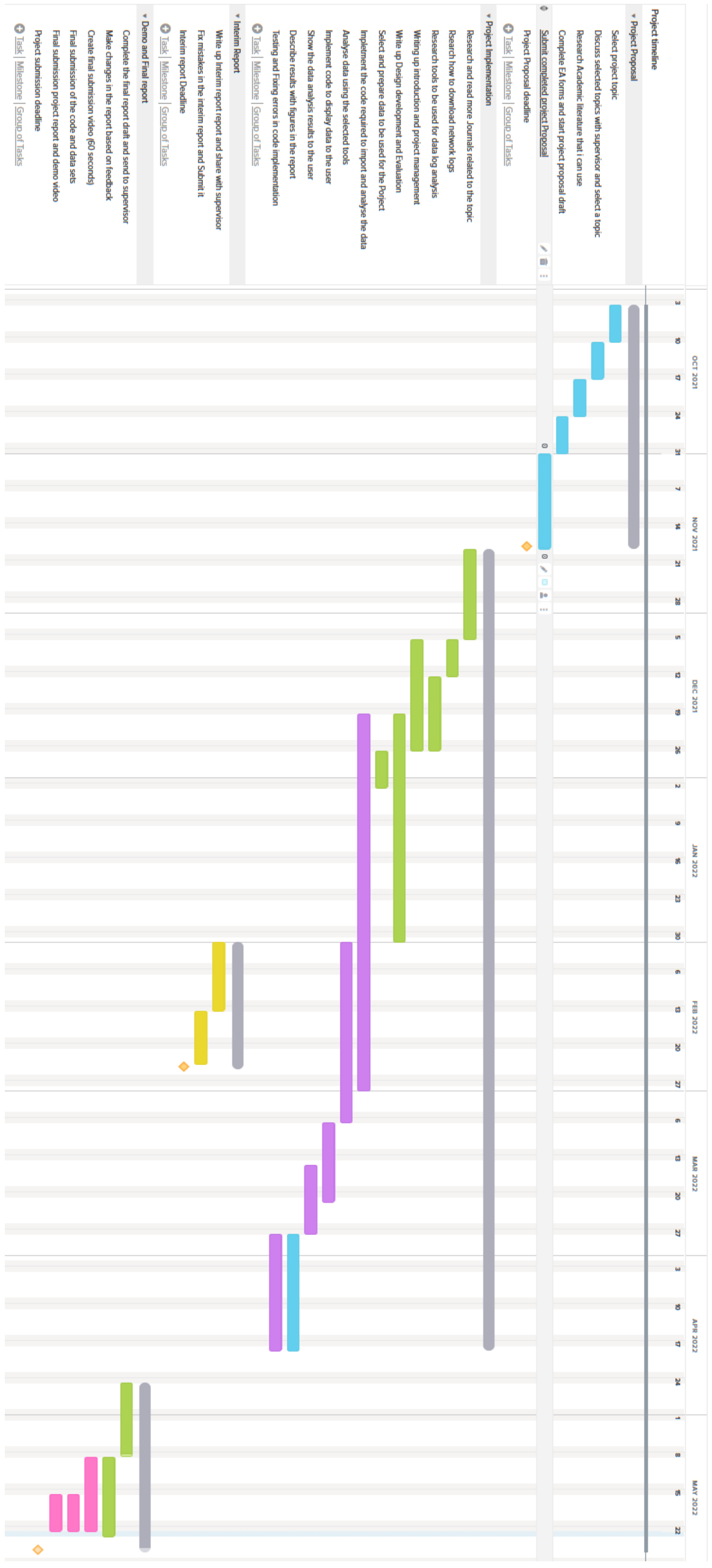


Figure 5 - Project timeline Gantt chart after completion

The Gantt chart was created to show the time scale that the tasks were to be completed but it is not completely accurate and might be missing some of the smaller tasks undertaken to add toward the completion of the project. Some of the tasks were completed earlier than expected. For example, researching academic literature was completed a week before its due date, also writing up the Interim report was completed weeks before the deadline date. At the same time not, all task progression was properly marked and documented. For example, writing up the Introduction, Project management was supposed to be finished by end of December but was not fully completed even by the 25th of May. It was the same case with writing up the section about Design development and evaluation that was supposed to be partially completed by the end of January but was not fully completed by the 25th of May. And lastly, the task where the machine learning model had to be implemented is called "Implement the code required to import and analyse the data" so the name here does not properly explain what the task accomplishes. Also, it was supposed to be completed by the end of February but, actually was not fully completed before the 15th of May.

The advantage of utilising a Gantt chart for this project was that it displayed a detailed view of all the big tasks and most small tasks that needed to be accomplished making it very difficult to miss tasks and milestones that needed to be worked on and what each task flow looks like. Conversely, the bigger tasks encompassed smaller tasks that were combined to complete the big task this means that the big task needed further planning to show what tasks needed to be done at the lower level. For example, the biggest task "Implement the code required to import and analyse the data" which is the Machine Learning implementation included lower-level tasks such as:

1) Import the .CSV format DDoS dataset file

2) Perform data pre-processing

3) Print dataset information and useful plots that helps the user understand the dataset

4) Split the input dataset into test and train datasets required for the machine learning model

5) Implement the machine learning model

6) Train the machine learning model on the training dataset

7) Test the trained model on the test DDoS dataset

8) Display the predictions made

The smaller steps above are the breakdown of the one step shown in the Gantt chart. It was not feasible to mention all these small steps in the Gantt chart so it tends to show generic overviews in some sections.

3.1.3 Risk Assessment

In their paper (Lavanya N. *et al.*, 2008) state that risk analysis is the process of identifying and analysing potential problems that could negatively impact the project, helping to avoid and mitigate the identified risks. It allows the team to be ready for any identified problems that might crop up and they can use the formulated strategy to deal with the problem and solve it. It is a very important part of Project management requirements

This section will look at some of the risks associated with the models that are going to be considered to be used for completing this project. As described earlier each model has its advantages and disadvantages, some of these disadvantages can give rise to risks and threats that will be explored below:

Table 1: The table below considers the risk involved in different Project management models Risk assessment

Risk	Chance of occurring	Impact on the Project	Mitigation steps
Some of the smaller tasks required are not completed	Low	High	Regularly check the Gantt chart to check which main task is in the process of being completed and make a requirement list for the smaller tasks present in it
Main DDoS datasets are too complex for the machine learning model	High	Low	Consider what columns of data are relevant to getting the results that are required. Clean the data before importing it into python. Also, perform data pre-processing

The model chosen does not produce the result wanted	Low	High	Make changes to the model like upgrades to the steps, layers, algorithms and so on. If the model cannot be adjusted then choose a new model if time allows
Hardware is not robust enough to perform the machine learning processing	Low	Medium	Test the machine learning model on the system being used at home, if it falls short then use the computers in the university labs. Also, consider dropping data columns that are not as important for the ML model
The dataset does not have enough features for training a machine	High	High	Research what features have been used to train ML models in previous studies. Use different

learning model			methods to collect data, e.g., Wireshark or tcpdump
The model fails to address the dates for tasks completion not being met	Low	Medium	Keep a month of time buffer from the due date of the project, if any problems arise then the model can be adjusted
Not enough time to finish the project before the due date	Low	High	Stick to the tasks due dates specified in the Gantt chart to complete the objectives on time
The model does not allow for testing and fixing bugs in the ML code	Low	Medium	Create a timeslot for testing and fixing bugs in the Gantt chart. Also, contact the supervisor if any help is needed with where to find resources to fix ML code

The risk analysis table above has brought to light some of the issues that can crop up during the duration of the process. This would allow for a predetermined plan to go into action to save time and the author would be ready if these problems came up.

3.1.4 Supervisor Meetings

Another valuable resource throughout the project completion process was the ability to set up a meeting with the author's project supervisor. At the start of the project for about two months these meetings happened on a bi-weekly or sometimes weekly basis. The author in these meetings could bring up and discuss the problems in the project, the progress made since the last meeting, and also list what they plan on working on until the next meeting. This also allowed for a chance to ask for feedback on any work they wanted to show the supervisor and get feedback on it, then improve or make corrections to their work. Since these meetings were done throughout the project timescale it made progressing through the project one step at a time easier and make sure they were constantly making progress and not missing most of their objectives.

After January the meetings were not done on weekly basis and it was up to the author to set up the meeting by messaging the supervisor. This allowed the author to set up meetings when they felt like they had made enough progress to share, faced a problem or wanted to get feedback on their work. This put pressure on the author to work regularly and do tasks in small chunks.

Also, the author did not have any knowledge and experience in the field of writing DDoS scripts and data collection thereafter to make up the dataset. The tips from their supervisor motivated and pointed them in the right direction at the start of the project. Also, the help they were having issues getting started on the literature review as it is a very long and daunting process. The help they got for the supervisor during the initial weekly meeting helped them in completing the Literature review for all 3 reports.

3.1.5 Evaluation Methodologies

Software engineering frameworks can be considered as either Heavyweight or lightweight frameworks. Heavyweight frameworks are better suited for projects that have all their requirements and objectives already listed and require more documentation for the planning process, an example of this framework is the Waterfall model. Lightweight frameworks allow for more modular iterative planning and development, the agile model is an example of this framework.

A heavyweight framework makes it next to impossible to make changes to the project plan once it starts, to make changes current plan will need to be cancelled and a new plan created. Lightweight frameworks on the other hand make it easy to make making changes in requirements or objectives easier and we can accommodate them into the model.

Based on the characteristics of this project the Methodology framework would need the following requirements:

- Flexibility
- Adaptability
- Low budget
- Scale
- Small team/Individual work
- Timely delivery

Table 2 - Table to describe the relation between software engineering frameworks and project characteristics

Model Type		Heavyweight framework		Lightweight framework	
Project characteristics	Model name	Waterfall model	Spiral model	Agile model	SCRUM (Agile) model
Delivery time		On time	Early	Early	Early
Team size		Large	Small	Small	Small
The scale of the project		Large	Large	Large	Medium
Budget		Medium	Medium	Medium	Small
Risks		Low	Medium	Low	Low
Adaptability and Flexibility		Low	High	High	High
Project requirements planning		High	Low	Low	Low

After comparing the frameworks above and comparing different models that use them, the author of the project chose to go with the Lightweight framework and use the SCRUM model to plan and complete this project. As it allowed them to develop the project in small steps/iterations thus providing the flexibility to make changes in the project plan when needed rather than following the project structure to the T. This was important to the author as they did not have a detailed plan on every task they were going to undertake in the project, so it is important that they can slot in new tasks and objectives as the project progressed.

3.2 Software Development

This section will briefly explore the Software development techniques used for this project. This is important as it provides a structured plan for the author to follow so they don’t lose their focus from the objective.

KANBAN:

Kanban board was a project management framework created by Toyota. (Björkholm and Björkholm, 2015) They note that the Kanban model relies on the project manager to decide the roles of the team members, this would be for the author in this project as they are the only individual working on this project. This makes the team roles and responsibilities principles of this model useless.

Another principle of Kanban is its focus on working on the first card in order. This will be beneficial to the author as it will prevent the author from working on too many tasks/objectives at the same time and becoming overburdened with work and improve the effectiveness of the author leading to speedy delivery of the artefact. Kanban board also show the overall progress of the project in an easy-to-understand format making helping the user to keep track of how much work is left.

Kanban also has a feature called Work in progress Limits that tracks how much work the team is currently working on and prevents them from taking up additional work that might overburden them making them less effective (Björkholm and Björkholm, 2015). This would also be a good feature for the author as the sole person working on the project, they could have easily been overwhelmed by the taking on too much work but setting the WIP limit to 2 tasks at a time stopped them from being overburdened.

A downside of Kanban is that they don’t have timeframes for completing the tasks this could negatively affect the author when doing coding tasks where it is important to have due data for tasks/cards as the next task depends on it, so they need to be finished on time. This could make the author run out of time before finishing the machine learning artefact.

SCRUM

SCRUM is another variant of Agile methodology that suits the requirements of this project. It allows the user to organise and manage their work, this is don’t by doing tasks in iterative cycles that are done over equal periods of time (1 month or 1 week) called sprints (Rubin, 2013). Sprints are a very important concept to understand as they will make the process of planning the project simpler. At the end of each sprint, the team is supposed to hold a meeting called “Scrums” but these are not useful for our project as it is an individual assignment. This can have a negative effect as in a team project, team members can spot problems in the planning and can point them out. But for this project (Sommerville, 2016). The author also has other commitments that mean that they are not going to make the same amount of progress in every single sprint, this means each sprint might need to be slightly adjusted in terms of how much time each sprint gets. SCRUM also tracks the workflow using a Scrum board, the same as Kanban it can be digital or physical with cards that represent the tasks to be done. Scrum also does not require an end date to be decided this can lead to the author not being able to finish the project on time, leading to failure.

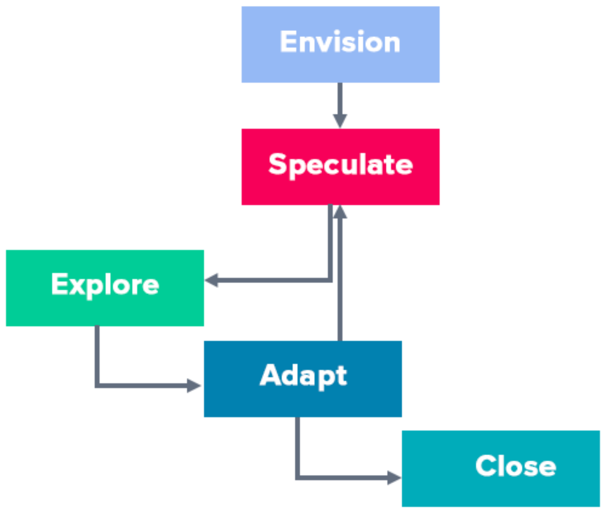


Figure 6 - SCRUM model stages (Cohen, 2022)

KANBAN VS SCRUM

Table 3 – This table compares the Kanban and SCRUM project management models

Scrub	**Kanban**
Does not rely on a project manager but on the team instead to manage tasks	Relies on the project manager to assign tasks
Has a smooth workflow that is very beneficial for software development work	Has a smooth workflow that is very beneficial for software development work

More flexible in the mid stages of the project as new/unforeseen requirements can be slotted in and worked on	Not as flexible as it does not allow to slot in new requirements in the middle of the project
Mandates need for team meetings at the end of each Sprint, but Scrum meetings are useless for this project	No requirement for team meetings but teams can choose to have them
Having a bigger team is beneficial, as they can also spot and point out problems in the project	More effective with a team of people with fixed roles/cards assigned
The project is completed in iterations of a fixed amount of time, which are called “Sprints”	The project is completed in iterations, where the cards are ordered in a list and the top cards are worked on first
Any of the tasks from the requirements can be worked on at any stage, with no top to bottom structure	Helps to focus by working on cards/tasks from the top card to the last card
No such system to prevent overburdening of team members	WIP limits prevent team members from being overburdened by working on too many cards at the same time
Not all sprints are going to have the same amount of work done due to other commitments the author has	Depending on the tasks being worked on, the cards decide how much need to be done
Has a time frame to complete each task called Sprint (e.g., 30 days, 7 days)	Does not have timeframes or due dates for each card can lead to the author running out of time

Very flexible and adaptable	Not as flexible and adaptable
No fixed end date can result in not finishing the project on time	No fixed end date can result in not finishing the project on time

For this project, the author decided to use the SCRUM model as it would fit their loose requirement list better and changes can be made to the model structure at any time as long as it fits the timeframe of the project. Scrum was chosen over Kanban because it has the Sprints and Kanban does not have a similar system that keeps the authors focus on the objective and get them to complete the given task before the next Sprint.

It also allowed for easier shifting of priorities and the tasks being worked on making it more flexible and adaptive compared to Kanban which has a more rigid structure

3.3 Toolsets and Machine Environments

This section is going to explore the tools and resources used to build and deploy the DDoS attack detection machine learning artefact and the software libraries used. Also, in addition to the version control system used to store and update the artefact code.

3.3.1 Programming Languages

Python

Python is a very powerful language that supports level two access to network services and Machine learning modules and libraries. Python is also an easier language to learn for new programmers. Python can access low-level sockets from the operating system, which allows to implement network communications. It also supports high-level protocols such as FTP, and HTTP (used in the project for DDoS script) (Python - Network Programming, 2022).

Python is one of the best for Machine learning and deep learning. A lot of programmers prefer using python because it has a big selection of libraries that can perform data manipulation, it also has libraries that contain numerous ML algorithms suck as Scikit-learn (used in this project), TensorFlow and so on (Lashchuck, 2022).

C++

C++ is a high-level programming language that can be used to perform networking and Machine learning. C++ does allow to connect two nodes on the network by using the Socket libraries that could be used to code a DDoS script with extensive research.

It can also be used in Machine learning but is comparatively harder to get into (learning and coding proficiency) and harder to code compared to python. It does execute faster than python.

Table 4 - Table comparing Programming languages that can be used for this project

	Python	C++
Socket support	Yes	Yes

Third-party library Support (including machine learning)	Yes	Partial support
GUI Coding Environment	Yes	Yes
Out plots for data visualisation	Yes, using external libraries	Yes, using external libraries
Language readability	Very Readable	Not very readable
Level	High level	High level
Open source	Yes	N/A
IDE support	Jupyter-notebook Visual Studio Code	Visual studio 2019

From the comparisons above the author decide to use Python for both phases (DDoS script and Machine learning) for the project as it is easier to learn for beginners. Also, the author has previous experience in using Machine learning models inside Jupyter- notebook IDE which will make working with it a bit easier and take less time to complete the project. IT also has excellent Machine library support (e.g., Scikit-learn/sklearn) that can be used for using pre-existing ML models

3.3.2 IDE

Integrated Development Environment (IDE) is the software that can consolidate basic tools required to code an application and execute it. IDEs contain also contain a code editor, compiler and interpreter and debugger that can be interacted with using a Graphical user interface (GUI). There were several IDE options available for both stages of the project including Jupyter-notebook, PyCharm, Visual Studio Code, and Spyder and the table below will compare some of these listed IDEs:

Table 5 - Compare some available IDEs to measure their effectiveness

	Jupyter Notebook	PyCharm	Visual Studio code	Google Collab
Cost	Free	Free version available Paid version available	Free version available	Free version available with limited capability Paid version available
Open Source	Yes	Yes	Yes	No
Licence	Modified BSD	Apache	MIT	Google

Python support	Yes	Yes	Yes	Yes
Line numbering	Partial	Yes	Yes	Yes
Built-in version control (e.g., Git)	No	Yes	Yes	Partial support
Code auto-complete	No	Yes	Yes	Yes
Debugger	Yes	Yes	Yes	Partial

Based on the comparisons above both Jupyter-notebook and Visual studio code were really good options for the author to code the artefact.

For the DDoS script, the Ide choice will be Visual studio code. The networking libraries like sockets can be imported directly. It also supports threading that allows the use of multiple threads for more requests per second when sending network traffic.

They decided Jupyter notebook would be a better option as they had previous experience working on Machine learning models using it. It also has many ML libraries which are a big part of the ML part of the artefact.

3.3.3 Project management tools

Gantt Chart

A Gantt chart was created to help with project planning to complete the project successfully. It helped with balancing a large number of tasks that otherwise would have been very overwhelming (Gantt Charts: Definitions, Features, & Uses | TeamGantt, 2022). For the project, the Gantt chart was created using an online service at www.teamgantt.com. This service was used as it was freely available and had lots of useful features such as tracking overall project progress and set start and end dates for each task.

Testing

The testing approach used for the SCRUM model used is described below:

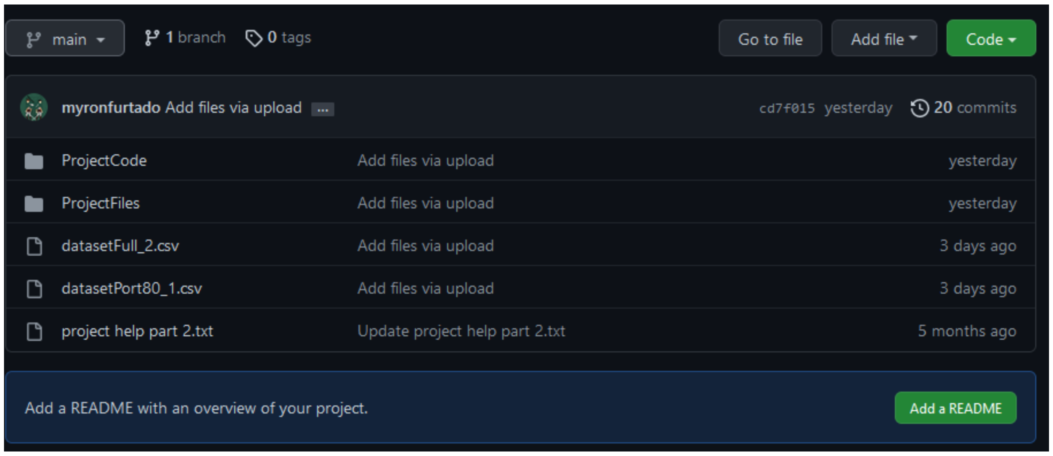
- A table of test cases will need to be prepared
- Test the artefact after it is deployed
- The results of the test will need to be displayed in a table

The benefit of Testing in a SCRUM model is that each deployment can be tested at deployment, alternatively, testing can also be performed at the end of each Sprint. This allows more flexibility to the author because they don't have to wait till the artefact is finished and can perform testing as each Sprint/ task is completed. This also allows for quick feedback from the project supervisor making it easy to fix bugs as early as possible, thus saving time.

Considering the advantages listed above a lightweight Agile strategy will be used for testing in this project artefact.

3.3.4 Github

Git/ Github is a version control system used to periodically update or sync code. Version control can be described as a system that records changes to a code file or any file over time so it can recover previous versions of the file can be restored if required (Cachon and Straub, 2014). Version control can also be accomplished by manually making copies of the file from time to time but this can be time-consuming, a better alternative for this is Github. It treats the file data as a series of snapshots of a miniature file system (Cachon and Straub, 2014).



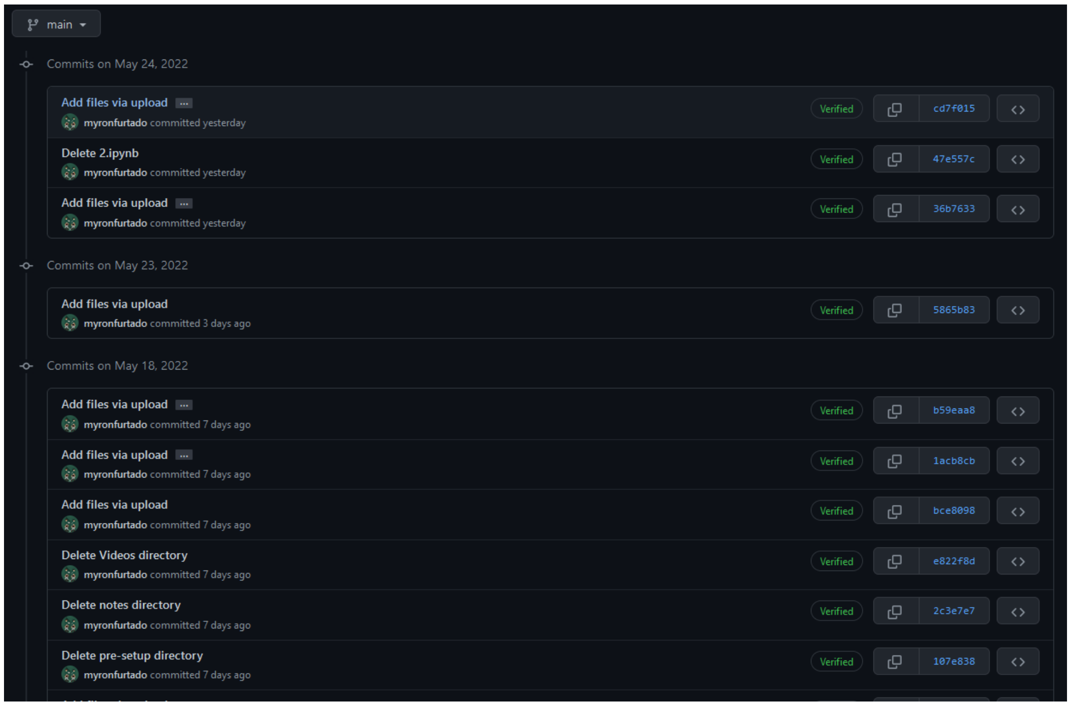
myronfurtado Add files via upload ... cd7f015 yesterday 20 commits

ProjectCode	Add files via upload	yesterday
ProjectFiles	Add files via upload	yesterday
datasetFull_2.csv	Add files via upload	3 days ago
datasetPort80_1.csv	Add files via upload	3 days ago
project help part 2.txt	Update project help part 2.txt	5 months ago

Add a README with an overview of your project.

Figure 7 - Screenshot of the main branch of the artefact files on Github

Git is the best too for use in this project because it allows the author to have different branches of code so they can compare and have previous versions of the files if they need to look at them. This allows the author to work on different lines of development without having to make changes to the main branch of the code (Cachon and Straub, 2014). IT also allows the user to fetch the code directly to the IDE if it is supported and also supports local download of the development code files.



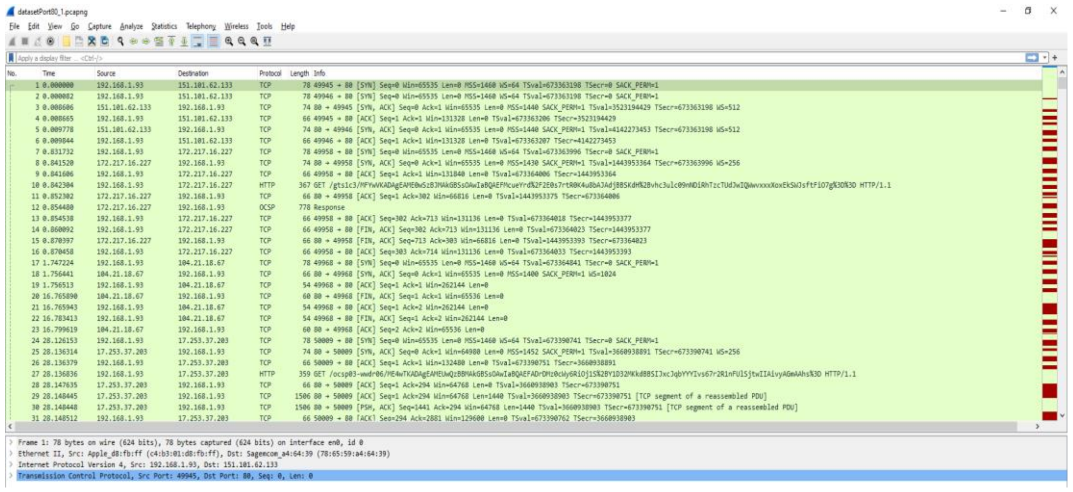
```
main

Commits on May 24, 2022

Add files via upload ...                                  Verified   cd7f015
myronfurtado committed yesterday
Delete 2.ipynb                                            Verified   47e557c
myronfurtado committed yesterday
Add files via upload ...                                  Verified   36b7633
myronfurtado committed yesterday

Commits on May 23, 2022

Add files via upload                                      Verified   5865b83
myronfurtado committed 3 days ago

Commits on May 18, 2022

Add files via upload ...                                  Verified   b59eaa8
myronfurtado committed 7 days ago
Add files via upload ...                                  Verified   1acb8cb
myronfurtado committed 7 days ago
Add files via upload                                      Verified   bce8098
myronfurtado committed 7 days ago
Delete Videos directory                                   Verified   e822f8d
myronfurtado committed 7 days ago
Delete notes directory                                    Verified   2c3e7e7
myronfurtado committed 7 days ago
Delete pre-setup directory                                Verified   107e838
myronfurtado committed 7 days ago
```

Figure 8 - List of all the commits to the Project Github main branch

3.3.5 Wireshark

Wireshark is a Program available for Many types of operating systems at are used for network monitoring. It captures packets sent over the network interface card by using the npcap library. It also provides filters that can be used to sort through captured data, capture filtered data, and look at the contents of captured packets. It also allows easy export of the captured data into the CSV format making it easy to work with the dataset.

It was preferred over built-in libraries like tcpdump, which are built into the Linux system because it provides a GUI and is easy to understanding layout. It is also easier to read the data because all data packets captured are colour coded making it easier to spot erroneous or odd connections



```
No. Time        Source          Destination     Protocol Length Info
  1 0.000000    192.168.1.93    151.101.62.133  TCP      78 49945 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=673363198 TSecr=0 SACK_PERM=1
  2 0.000082    192.168.1.93    151.101.62.133  TCP      78 49946 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=673363198 TSecr=0 SACK_PERM=1
  3 0.008606    151.101.62.133  192.168.1.93    TCP      74 80 → 49945 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1440 SACK_PERM=1 TSval=3523194429 TSecr=673363198 WS=512
  4 0.008665    192.168.1.93    151.101.62.133  TCP      66 49945 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0 TSval=673363206 TSecr=3523194429
  5 0.009778    151.101.62.133  192.168.1.93    TCP      74 80 → 49946 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1440 SACK_PERM=1 TSval=4142273453 TSecr=673363198 WS=512
  6 0.009844    192.168.1.93    151.101.62.133  TCP      66 49946 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0 TSval=673363207 TSecr=4142273453
  7 0.831732    192.168.1.93    172.217.16.227  TCP      78 49958 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=673363996 TSecr=0 SACK_PERM=1
  8 0.841520    172.217.16.227  192.168.1.93    TCP      74 80 → 49958 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 SACK_PERM=1 TSval=1443953364 TSecr=673363996 WS=256
  9 0.841606    192.168.1.93    172.217.16.227  TCP      66 49958 → 80 [ACK] Seq=1 Ack=1 Win=131840 Len=0 TSval=673364006 TSecr=1443953364
 10 0.842304    192.168.1.93    172.217.16.227  HTTP    367 GET /gts1c3/MFYwVKADAgEAME0wSzBJMAkGBSsOAwIaBQAEFMcueYrd%2F2E0s7rtR0K4uBbAJAdjBBSKdH%2Bvhc3ulc09nNDiRhTzcTUdJwIQWwvxxxXoxEkSWJsftFiO7g%3D%3D HTTP/1.1
 11 0.852302    172.217.16.227  192.168.1.93    TCP      66 80 → 49958 [ACK] Seq=1 Ack=302 Win=66816 Len=0 TSval=1443953375 TSecr=673364006
 12 0.854480    172.217.16.227  192.168.1.93    OCSP    778 Response
 13 0.854538    192.168.1.93    172.217.16.227  TCP      66 49958 → 80 [ACK] Seq=302 Ack=713 Win=131136 Len=0 TSval=673364018 TSecr=1443953377
 14 0.860092    192.168.1.93    172.217.16.227  TCP      66 49958 → 80 [FIN, ACK] Seq=302 Ack=713 Win=131136 Len=0 TSval=673364023 TSecr=1443953377
 15 0.870397    172.217.16.227  192.168.1.93    TCP      66 80 → 49958 [FIN, ACK] Seq=713 Ack=303 Win=66816 Len=0 TSval=1443953393 TSecr=673364023
 16 0.870458    192.168.1.93    172.217.16.227  TCP      66 49958 → 80 [ACK] Seq=303 Ack=714 Win=131136 Len=0 TSval=673364033 TSecr=1443953393
 17 1.747224    192.168.1.93    104.21.18.67    TCP      78 49968 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=673364841 TSecr=0 SACK_PERM=1
 18 1.756441    104.21.18.67    192.168.1.93    TCP      66 80 → 49968 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024
 19 1.756513    192.168.1.93    104.21.18.67    TCP      54 49968 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
 20 16.765890   104.21.18.67    192.168.1.93    TCP      60 80 → 49968 [FIN, ACK] Seq=1 Ack=1 Win=65536 Len=0
 21 16.765943   192.168.1.93    104.21.18.67    TCP      54 49968 → 80 [ACK] Seq=1 Ack=2 Win=262144 Len=0
 22 16.783413   192.168.1.93    104.21.18.67    TCP      54 49968 → 80 [FIN, ACK] Seq=1 Ack=2 Win=262144 Len=0
 23 16.799619   104.21.18.67    192.168.1.93    TCP      60 80 → 49968 [ACK] Seq=2 Ack=2 Win=65536 Len=0
 24 28.126153   192.168.1.93    17.253.37.203   TCP      78 50009 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=673390741 TSecr=0 SACK_PERM=1
 25 28.136314   17.253.37.203   192.168.1.93    TCP      74 80 → 50009 [SYN, ACK] Seq=0 Ack=1 Win=64980 Len=0 MSS=1452 SACK_PERM=1 TSval=3660938891 TSecr=673390741 WS=256
 26 28.136379   192.168.1.93    17.253.37.203   TCP      66 50009 → 80 [ACK] Seq=1 Ack=1 Win=132480 Len=0 TSval=673390751 TSecr=3660938891
 27 28.136836   192.168.1.93    17.253.37.203   HTTP    359 GET /ocsp03-wwdr06/ME4wTKADAgEAMEUwQzBBMAkGBSsOAwIaBQAEFADrDMz0cWy6RiOj1S%2BY1D32MKkdBBSIJxcJqbYYYIvs67r2R1nFUlSjtwIIAivyAGmAAhs%3D HTTP/1.1
 28 28.147635   17.253.37.203   192.168.1.93    TCP      66 80 → 50009 [ACK] Seq=1 Ack=294 Win=64768 Len=0 TSval=3660938903 TSecr=673390751
 29 28.148445   17.253.37.203   192.168.1.93    TCP    1506 80 → 50009 [ACK] Seq=1 Ack=294 Win=64768 Len=1440 TSval=3660938903 TSecr=673390751 [TCP segment of a reassembled PDU]
 30 28.148448   17.253.37.203   192.168.1.93    TCP    1506 80 → 50009 [PSH, ACK] Seq=1441 Ack=294 Win=64768 Len=1440 TSval=3660938903 TSecr=673390751 [TCP segment of a reassembled PDU]
 31 28.148512   192.168.1.93    17.253.37.203   TCP      66 50009 → 80 [ACK] Seq=294 Ack=2881 Win=129600 Len=0 TSval=673390762 TSecr=3660938903

> Frame 1: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface en0, id 0
> Ethernet II, Src: Apple_d8:fb:ff (c4:b3:01:d8:fb:ff), Dst: Sagemcom_a4:64:39 (78:65:59:a4:64:39)
> Internet Protocol Version 4, Src: 192.168.1.93, Dst: 151.101.62.133
> Transmission Control Protocol, Src Port: 49945, Dst Port: 80, Seq: 0, Len: 0
```

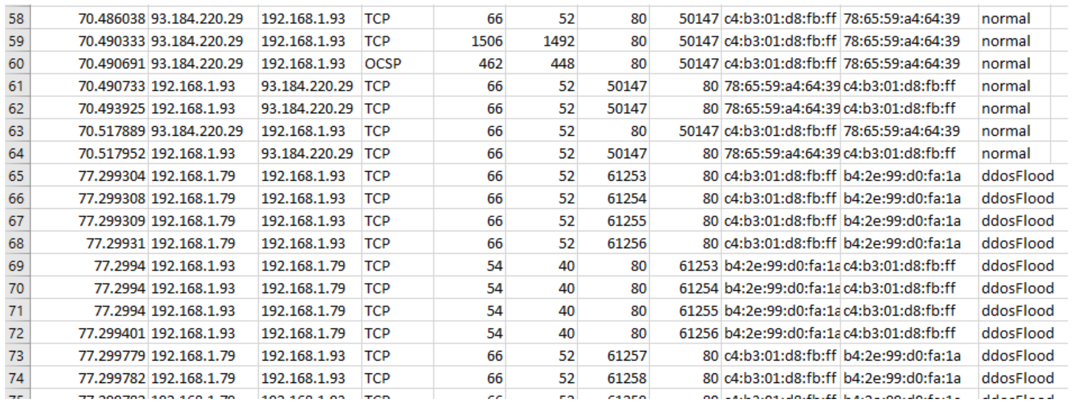
datasetPort80_1.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

Figure 9 - Original unfiltered data captured inside of Wireshark

3.3.6 Microsoft Excel

Microsoft Excel was used to clean and organise the data collected from Wireshark. This was done because excel allows the user of easy filters to remove unwanted columns and rows of data. Excel is also allowed for the dataset to be transferred out as CSV format after cleaning the dataset, this format is important as it makes importing data into pandas DataFrame a breeze.

Excel also allowed the author to enter the classifier/label column easily by entering the values into hundreds of columns at the same time.



58	70.486038	93.184.220.29	192.168.1.93	TCP	66	52	80	50147	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
59	70.490333	93.184.220.29	192.168.1.93	TCP	1506	1492	80	50147	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
60	70.490691	93.184.220.29	192.168.1.93	OCSP	462	448	80	50147	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
61	70.490733	192.168.1.93	93.184.220.29	TCP	66	52	50147	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
62	70.493925	192.168.1.93	93.184.220.29	TCP	66	52	50147	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
63	70.517889	93.184.220.29	192.168.1.93	TCP	66	52	80	50147	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
64	70.517952	192.168.1.93	93.184.220.29	TCP	66	52	50147	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
65	77.299304	192.168.1.79	192.168.1.93	TCP	66	52	61253	80	c4:b3:01:d8:fb:ff	b4:2e:99:d0:fa:1a	ddosFlood
66	77.299308	192.168.1.79	192.168.1.93	TCP	66	52	61254	80	c4:b3:01:d8:fb:ff	b4:2e:99:d0:fa:1a	ddosFlood
67	77.299309	192.168.1.79	192.168.1.93	TCP	66	52	61255	80	c4:b3:01:d8:fb:ff	b4:2e:99:d0:fa:1a	ddosFlood
68	77.29931	192.168.1.79	192.168.1.93	TCP	66	52	61256	80	c4:b3:01:d8:fb:ff	b4:2e:99:d0:fa:1a	ddosFlood
69	77.2994	192.168.1.93	192.168.1.79	TCP	54	40	80	61253	b4:2e:99:d0:fa:1a	c4:b3:01:d8:fb:ff	ddosFlood
70	77.2994	192.168.1.93	192.168.1.79	TCP	54	40	80	61254	b4:2e:99:d0:fa:1a	c4:b3:01:d8:fb:ff	ddosFlood
71	77.2994	192.168.1.93	192.168.1.79	TCP	54	40	80	61255	b4:2e:99:d0:fa:1a	c4:b3:01:d8:fb:ff	ddosFlood
72	77.299401	192.168.1.93	192.168.1.79	TCP	54	40	80	61256	b4:2e:99:d0:fa:1a	c4:b3:01:d8:fb:ff	ddosFlood
73	77.299779	192.168.1.79	192.168.1.93	TCP	66	52	61257	80	c4:b3:01:d8:fb:ff	b4:2e:99:d0:fa:1a	ddosFlood
74	77.299782	192.168.1.79	192.168.1.93	TCP	66	52	61258	80	c4:b3:01:d8:fb:ff	b4:2e:99:d0:fa:1a	ddosFlood

Figure 10 - Sample of the dataset showing both types of network data in excel (cleaned data)

3.3.7 Libraries Used

Sockets and Threading

The sockets module is a Python library that accesses a computer's socket interface from the operating system. It can open a socket instance at the target device port that is passed to it and send HTTP requests to other devices inside the network to communicate with it.

The threading library in python allows the code to start multiple threads/executions to execute at the same time, or in the case of python appears to run at the same time, but runs very fast in a sequence (Python, 2022). This will allow the DDOs Script to open multiple connections requests at the same time instead of sending them one request at a time making the attack more lethal

NumPy and Pandas

Both NumPy and pandas are open-source libraries that can be imported easily and used for high-performance data processing in machine learning. These are essential to perform data pre-processing on the dataset before the data can be fed into the ML model. They have really easy syntax and good performance allowing the calculations to be fast (Pandas vs NumPy - javatpoint, 2022).

Matplotlib and Seaborn

Matplotlib is an open-source Python 2D plotting library that allows the user to plot graphs and figures based on the data that is inputted into the system. Some examples of plots it can generate include Histograms, bar plots, line graphs and so on (Matplotlib, 2022).

Same as matplotlib, seaborn is also a plotting library with similar features. One difference is that seaborn can handle pandas DataFrame better than matplotlib.

Scikit-learn/ sklearn

Scikit-learn is another open-source machine learning library created for Python. It is similar to the very popular TensorFlow ML library, but a little less sophisticated and less advanced. It is more focused on performing predictive data analysis tasks (Scikit-learn, 2020). This library provides access to ML models such as Classifiers, regressors, and Clustering. This is very beneficial to the project as it means the author will not have to build models from scratch, a daunting task and would greatly increase the scale of the project.

3.4 Research Methods

Research methods are strategies or techniques that can be used in the collection of data and evidence for analysis, this allows the researcher to answer the question they proposed in their project (Booth D. *et al*., 2019). Research methods can be broken down into two categories: Qualitative and Quantitive research.

In this project the author uses Quantitative research data, this can be explained as numerical data that can be measured. In the project, Wireshark will be used to collect network traffic data that will contain two different types of data. The First will be normal traffic data and the second will be DDoS attack network traffic. This data will be used inside of machine learning models so that the model can predict and classify the data as either “normal” or “DDoS”. The results will allow us to determine if the machine learning models used can correctly classify the data or if they fail to do that, summarising the research question asked in this project.

Chapter 4

Design, Development and Evaluation

This project aims to collect a cyberattack (DDoS) dataset and determine if the data in it can be correctly predicted using Machine learning models. The analysis performed her is post analysis and not done in real time. The data is then visualised using Python libraries like matplotlib and seaborn and presented to the user. The last stage is the machine learning models are fed the pre-processed datasets and all 3 machine learning models are tested for their prediction accuracy.

4.1 Software Development Projects

I should be noted that since this is an individual project and the Scrum model is being used, that puts the author in charge of Project manager, developer and artefact owner and this make the Development process a little different from normal where a whole team would be working on it. It should also be noted that Agile models like SCRUM put more emphasis on the development of an artefact/product more than the need for comprehensive documentation. This can help to speed up the time taken for development.

1. **Requirement's elicitation, gathering, collection and analysis**

Requirement elicitation:

Software requirements elicitation is the act of studying and understanding the needs of the stakeholders and what they are going to need form the product developed (Sommerville, 2016). Since the stakeholder here is the author and has no one else to get ideals from, this was going to be a bit tricky as the author had to now decide what an actual stakeholder would want. To complete the requirements elicitation, process the author conducted a detailed literature review till the month of December to get a good ideal of the smaller tasks they would have to work on to get to the bigger milestones. This meant they had to go through a lot of documented studies and code snippets published online to research DDoS attacks and Machine learning models.

The project has 2 phases and both are equally important to be able to complete the project. One being the DDoS attack and data collection and second being the machine learning models.

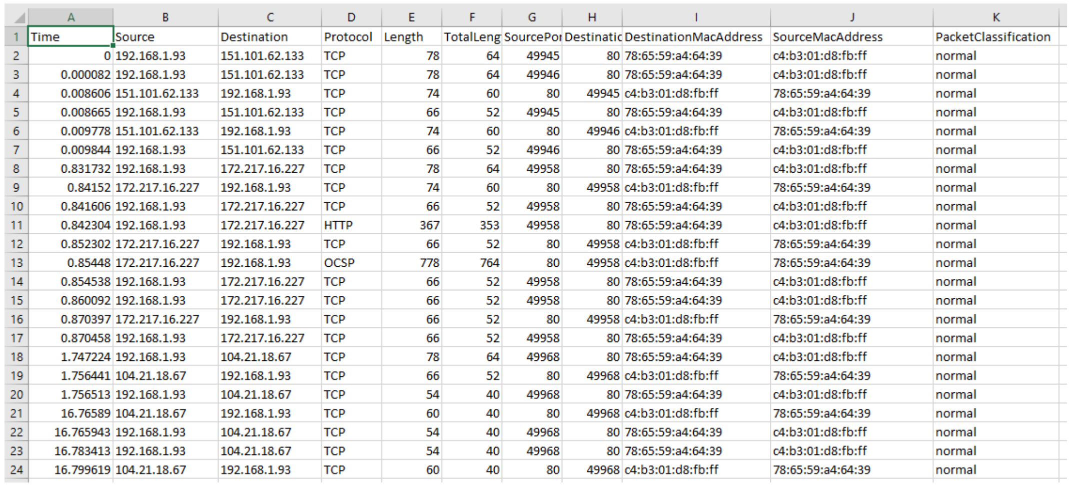
- Code a script in python that can perform a DDoS attack using an IP address
- Use Wireshark to capture the network data while a DDoS script is used to perform an attack on a selected device
- Depending on the type of dataset needed either filter the capture data and export or just export the whole capture into a CSV format file.
- Use Excel to open the datasets and clean the data. Drop columns that are not needed and add the label/ classification columns with values being either “normal” or “ddos”
- Set up the python environment in Jupyter notebook and import the necessary libraries and the datasets
- Perform data visualisation for the user. i.e., plots and graphs to describe the data
- Pre-process the data to be suitable for the ML models
- Split the dataset into test and train datasets
- Train the models using Train dataset

- Make predictions by predicting on the test dataset, and calculate the accuracy, confusion matrix for the model
- If DDoS attack is detected then print to the screen
- Show a graph of how each model performed

Upon defining the requirements, the author started with the preparation to start coding the DDoS script first so that the dataset could be collected.

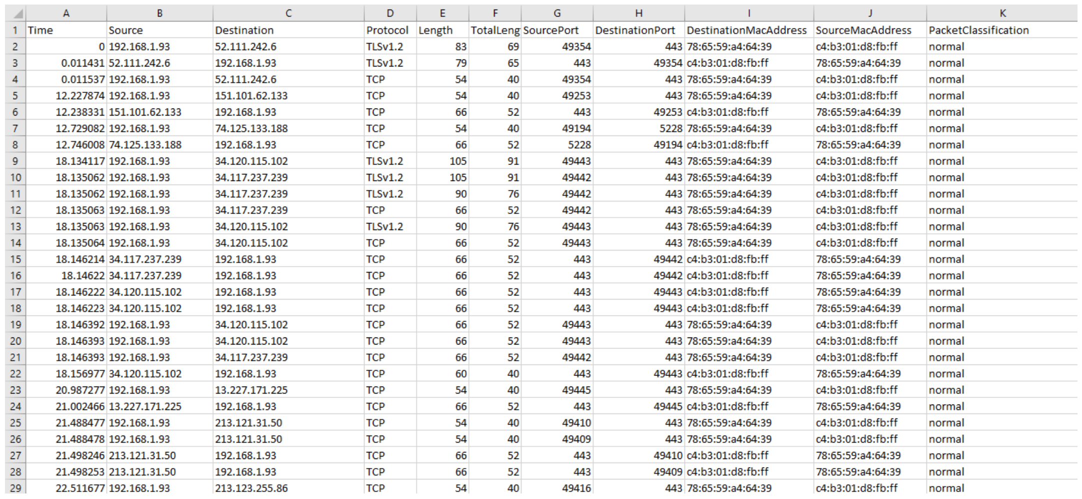
Requirement Dataset Collection and cleaning:

The dataset collected have columns like Time, source IP, destination IP, connection protocol used, and soon until the last column called Packet classification that set the packet as either normal or DDoS type. This column was added manually in Microsoft Excel after the dataset had been cleaned. Columns like info that contained useless information were dropped using the data cleaning processes, as well as records with other protocols that had “None” and “Null” or empty values.



	A	B	C	D	E	F	G	H	I	J	K
1	Time	Source	Destination	Protocol	Length	TotalLeng	SourcePor	Destinatic	DestinationMacAddress	SourceMacAddress	PacketClassification
2	0	192.168.1.93	151.101.62.133	TCP	78	64	49945	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
3	0.000082	192.168.1.93	151.101.62.133	TCP	78	64	49946	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
4	0.008606	151.101.62.133	192.168.1.93	TCP	74	60	80	49945	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
5	0.008665	192.168.1.93	151.101.62.133	TCP	66	52	49945	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
6	0.009778	151.101.62.133	192.168.1.93	TCP	74	60	80	49946	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
7	0.009844	192.168.1.93	151.101.62.133	TCP	66	52	49946	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
8	0.831732	192.168.1.93	172.217.16.227	TCP	78	64	49958	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
9	0.84152	172.217.16.227	192.168.1.93	TCP	74	60	80	49958	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
10	0.841606	192.168.1.93	172.217.16.227	TCP	66	52	49958	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
11	0.842304	192.168.1.93	172.217.16.227	HTTP	367	353	49958	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
12	0.852302	172.217.16.227	192.168.1.93	TCP	66	52	80	49958	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
13	0.85448	172.217.16.227	192.168.1.93	OCSP	778	764	80	49958	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
14	0.854538	192.168.1.93	172.217.16.227	TCP	66	52	49958	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
15	0.860092	192.168.1.93	172.217.16.227	TCP	66	52	49958	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
16	0.870397	172.217.16.227	192.168.1.93	TCP	66	52	80	49958	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
17	0.870458	192.168.1.93	172.217.16.227	TCP	66	52	49958	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
18	1.747224	192.168.1.93	104.21.18.67	TCP	78	64	49968	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
19	1.756441	104.21.18.67	192.168.1.93	TCP	66	52	80	49968	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
20	1.756513	192.168.1.93	104.21.18.67	TCP	54	40	49968	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
21	16.76589	104.21.18.67	192.168.1.93	TCP	60	40	80	49968	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
22	16.765943	192.168.1.93	104.21.18.67	TCP	54	40	49968	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
23	16.783413	192.168.1.93	104.21.18.67	TCP	54	40	49968	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
24	16.799619	104.21.18.67	192.168.1.93	TCP	60	40	80	49968	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal

Figure 11 - Dataset collected form the DDoS script on Port 80



	A	B	C	D	E	F	G	H	I	J	K
1	Time	Source	Destination	Protocol	Length	TotalLeng	SourcePort	DestinationPort	DestinationMacAddress	SourceMacAddress	PacketClassification
2	0	192.168.1.93	52.111.242.6	TLSv1.2	83	69	49354	443	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
3	0.011431	52.111.242.6	192.168.1.93	TLSv1.2	79	65	443	49354	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
4	0.011537	192.168.1.93	52.111.242.6	TCP	54	40	49354	443	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
5	12.227874	192.168.1.93	151.101.62.133	TCP	54	40	49253	443	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
6	12.238331	151.101.62.133	192.168.1.93	TCP	66	52	443	49253	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
7	12.729082	192.168.1.93	74.125.133.188	TCP	54	40	49194	5228	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
8	12.746008	74.125.133.188	192.168.1.93	TCP	66	52	5228	49194	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
9	18.134117	192.168.1.93	34.120.115.102	TLSv1.2	105	91	49443	443	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
10	18.135062	192.168.1.93	34.117.237.239	TLSv1.2	105	91	49442	443	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
11	18.135062	192.168.1.93	34.117.237.239	TLSv1.2	90	76	49442	443	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
12	18.135063	192.168.1.93	34.117.237.239	TCP	66	52	49442	443	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
13	18.135063	192.168.1.93	34.120.115.102	TLSv1.2	90	76	49443	443	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
14	18.135064	192.168.1.93	34.120.115.102	TCP	66	52	49443	443	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
15	18.146214	34.117.237.239	192.168.1.93	TCP	66	52	443	49442	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
16	18.14622	34.117.237.239	192.168.1.93	TCP	66	52	443	49442	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
17	18.146222	34.120.115.102	192.168.1.93	TCP	66	52	443	49443	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
18	18.146223	34.120.115.102	192.168.1.93	TCP	66	52	443	49443	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
19	18.146392	192.168.1.93	34.120.115.102	TCP	66	52	49443	443	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
20	18.146393	192.168.1.93	34.120.115.102	TCP	66	52	49443	443	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
21	18.146393	192.168.1.93	34.117.237.239	TCP	66	52	49442	443	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
22	18.156977	34.120.115.102	192.168.1.93	TCP	60	40	443	49443	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
23	20.987277	192.168.1.93	13.227.171.225	TCP	54	40	49445	443	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
24	21.002466	13.227.171.225	192.168.1.93	TCP	66	52	443	49445	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
25	21.488477	192.168.1.93	213.121.31.50	TCP	54	40	49410	443	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
26	21.488478	192.168.1.93	213.121.31.50	TCP	54	40	49409	443	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
27	21.498246	213.121.31.50	192.168.1.93	TCP	66	52	443	49410	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
28	21.498253	213.121.31.50	192.168.1.93	TCP	66	52	443	49409	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
29	22.511677	192.168.1.93	213.123.255.86	TCP	54	40	49416	443	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal

Figure 12 - Second dataset collected form the DDoS attack with all traffic

Requirement Analysis:

In this stage the first thing to work on is to get started on prototypes of the tasks listed in the requirements and makes sure they are attainable. This will be like a quick preliminary assessment of each task and its difficulty; it will allow for features that are too difficult or the ones that are not required to be either amended or removed from the project. Once the final requirements of the project have been decided on the author can start putting them in a backlog and start making increments to them as they follow the Gantt chart to decide order in which tasks are worked on. In addition to the Gantt chart the priority of a back log task is also decided other factors such as its risks, time needed, difficulty, its need to be complete before other tasks can start. This means the author might have to amend the dates and timeframes of certain tasks, this is where the SCRUM model really helps as it can accommodate changes in the plan such as changing current backlog items with other or picking up more backlog tasks to be worked on at the same time to save time and finish the project faster.

2. **Design**

The design for the code was designed at the start of the project by determining what type of application needed to be developed. This project only uses Python code to launch an DDoS attack and Machine learning models. Looking through the aim of the project it is specified that there are 4 major objectives in the project. From reading Academic journals and literature the author found out that it that they can use complete these 4 steps and bring them all together in the end to complete this project.

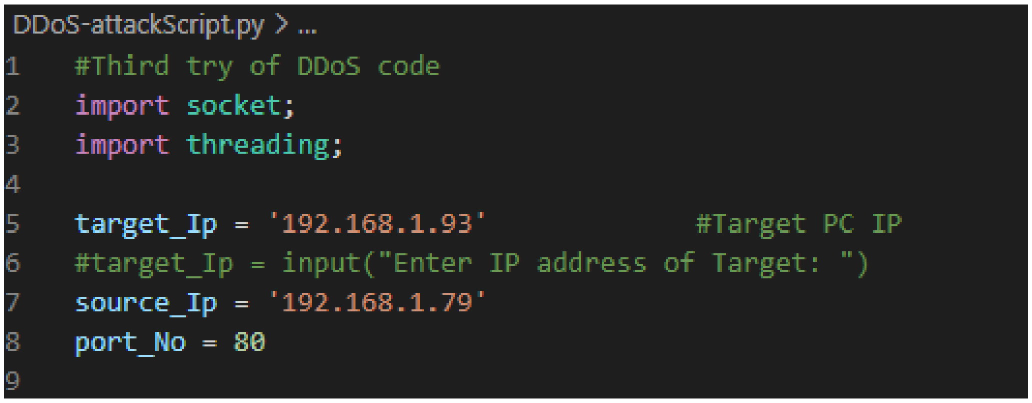
The overall design theme for the Software development life cycle (SDLC) is as follows:

1. Code a DDoS script and launch an attack on the target
2. User Wireshark to collect the data logs and export them to CSV file format
3. Clean the dataset and add the classifier/label column
4. Implement data visualisation and create 3 separate Machine learning classifier models

3. **Building and programming**

The artefact is designed and developed using the required specifications listed in this report. This stage will include screenshots and explanation of important code snippets from the artefact.

- **DDoS Script**

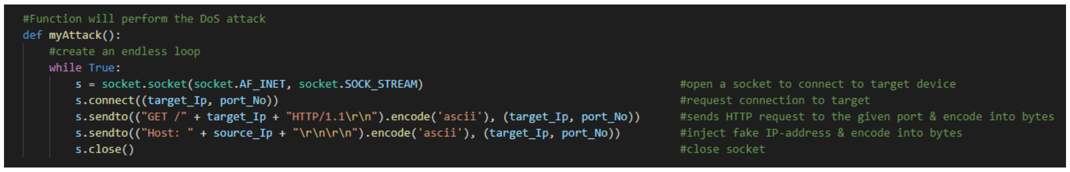


```python
DDoS-attackScript.py > ...
#Third try of DDoS code
import socket;
import threading;

target_Ip = '192.168.1.93'                #Target PC IP
#target_Ip = input("Enter IP address of Target: ")
source_Ip = '192.168.1.79'
port_No = 80

```

Figure 13 - DDoS script code 1

The code above is the start of the DDoS attack script created using Python. It starts by importing the socket and threading libraries. Socket is used to open a socket connection between two devices connected on a network. This allows the device to send and receive communication. Second, we import threading that allow multithread processing, used for sending HTTP flood requests using multiple threads instead of one core/tread.

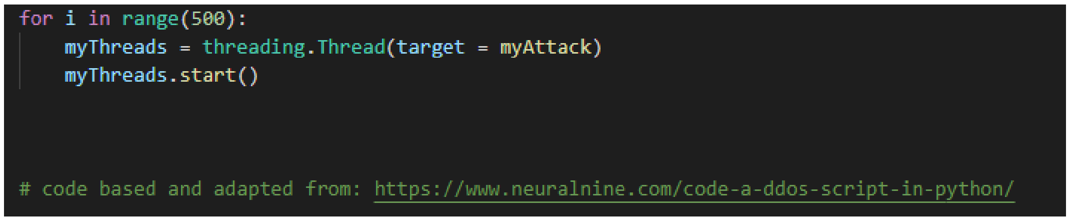
Next, we pass in the iPv4 address of the target device (target_Ip) and source device (source_Ip), this one can be a spoofed IP address and then we pass the port number that the number the target device where the source device will send the request.



```python
#Function will perform the DoS attack
def myAttack():
    #create an endless loop
    while True:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)                          #open a socket to connect to target device
        s.connect((target_Ip, port_No))                                                #request connection to target
        s.sendto(("GET /" + target_Ip + "HTTP/1.1\r\n").encode('ascii'), (target_Ip, port_No))      #sends HTTP request to the given port & encode into bytes
        s.sendto(("Host: " + source_Ip + "\r\n\r\n").encode('ascii'), (target_Ip, port_No))         #inject fake IP-address & encode into bytes
        s.close()                                                                      #close socket
```

Figure 14 - DDoS script code 2

The code above uses a function myAttack() that will run endlessly until the connection is blocked by the target device. On the first line the variable s opens a socket to connect to the target device. The code then keeps sending HTTP request in byte format (ascii) to the target deceive. Lastly, s.close() will close the connection when the code stops running when an error occurs (if target device blocks incoming requests)

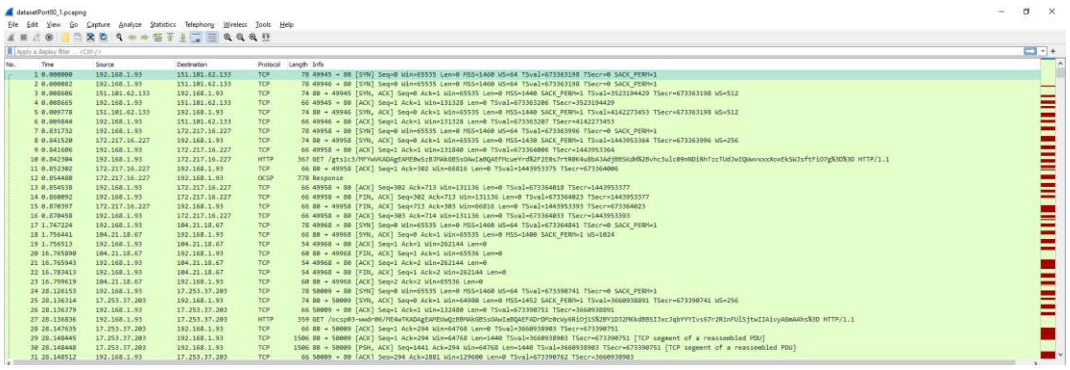


```python
for i in range(500):
    myThreads = threading.Thread(target = myAttack)
    myThreads.start()


# code based and adapted from: https://www.neuralnine.com/code-a-ddos-script-in-python/
```

Figure 15 - DDoS script code 3

The code above creates a loop so that the script can use threading to use multiple threads instead of a single thread, making the script more potent. It then calls the myAttack() function so the function can be executed on multiple threads at the same time.

- **Wireshark capture and Dataset cleaning**

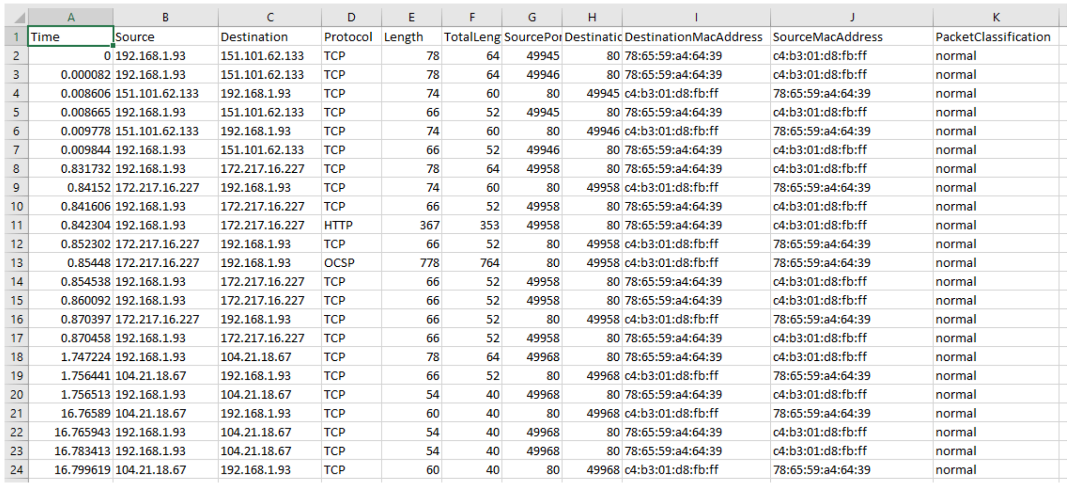


datasetPort80_1.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time       Source          Destination     Protocol Length Info
  1 0.000000   192.168.1.93    151.101.62.133  TCP      78 49945 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=673363198 TSecr=0 SACK_PERM=1
  2 0.000082   192.168.1.93    151.101.62.133  TCP      78 49946 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=673363198 TSecr=0 SACK_PERM=1
  3 0.008606   151.101.62.133  192.168.1.93    TCP      74 80 → 49945 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1440 SACK_PERM=1 TSval=3523194429 TSecr=673363198 WS=512
  4 0.008665   192.168.1.93    151.101.62.133  TCP      66 49945 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0 TSval=673363206 TSecr=3523194429
  5 0.009778   151.101.62.133  192.168.1.93    TCP      74 80 → 49946 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1440 SACK_PERM=1 TSval=4142273453 TSecr=673363198 WS=512
  6 0.009844   192.168.1.93    151.101.62.133  TCP      66 49946 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0 TSval=673363207 TSecr=4142273453
  7 0.831732   192.168.1.93    172.217.16.227  TCP      78 49958 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=673363996 TSecr=0 SACK_PERM=1
  8 0.841520   172.217.16.227  192.168.1.93    TCP      74 80 → 49958 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 SACK_PERM=1 TSval=1443953364 TSecr=673363996 WS=256
  9 0.841606   192.168.1.93    172.217.16.227  TCP      66 49958 → 80 [ACK] Seq=1 Ack=1 Win=131840 Len=0 TSval=673364006 TSecr=1443953364
 10 0.842304   192.168.1.93    172.217.16.227  HTTP    367 GET /gts1c3/MFYwVKADAgEAME0wSzBJMAkGBSsOAwIaBQAEFMcueYrd%2F2E0s7rtR0K4u8bAJAdjB8SKdM%2Bvhc3ulc09nNDiRhTzcTUdJwIQWwvxxxXoxEkSWJsftFiO7g%3D%3D HTTP/1.1
 11 0.852302   172.217.16.227  192.168.1.93    TCP      66 80 → 49958 [ACK] Seq=1 Ack=302 Win=66816 Len=0 TSval=1443953375 TSecr=673364006
 12 0.854480   172.217.16.227  192.168.1.93    OCSP    778 Response
 13 0.854538   192.168.1.93    172.217.16.227  TCP      66 49958 → 80 [ACK] Seq=302 Ack=713 Win=131136 Len=0 TSval=673364018 TSecr=1443953377
 14 0.860092   192.168.1.93    172.217.16.227  TCP      66 49958 → 80 [FIN, ACK] Seq=302 Ack=713 Win=131136 Len=0 TSval=673364023 TSecr=1443953377
 15 0.870397   172.217.16.227  192.168.1.93    TCP      66 80 → 49958 [FIN, ACK] Seq=713 Ack=303 Win=66816 Len=0 TSval=1443953393 TSecr=673364023
 16 0.870458   192.168.1.93    172.217.16.227  TCP      66 49958 → 80 [ACK] Seq=303 Ack=714 Win=131136 Len=0 TSval=673364033 TSecr=1443953393
 17 1.747224   192.168.1.93    104.21.18.67    TCP      78 49968 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=673364841 TSecr=0 SACK_PERM=1
 18 1.756441   104.21.18.67    192.168.1.93    TCP      66 80 → 49968 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024
 19 1.756513   192.168.1.93    104.21.18.67    TCP      54 49968 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
 20 16.765890  104.21.18.67    192.168.1.93    TCP      60 80 → 49968 [FIN, ACK] Seq=1 Ack=1 Win=65536 Len=0
 21 16.765943  192.168.1.93    104.21.18.67    TCP      54 49968 → 80 [ACK] Seq=1 Ack=2 Win=262144 Len=0
 22 16.783413  192.168.1.93    104.21.18.67    TCP      54 49968 → 80 [FIN, ACK] Seq=1 Ack=2 Win=262144 Len=0
 23 16.799619  104.21.18.67    192.168.1.93    TCP      60 80 → 49968 [ACK] Seq=2 Ack=2 Win=65536 Len=0
 24 28.126153  192.168.1.93    17.253.37.203   TCP      78 50009 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=673390741 TSecr=0 SACK_PERM=1
 25 28.136314  17.253.37.203   192.168.1.93    TCP      74 80 → 50009 [SYN, ACK] Seq=0 Ack=1 Win=64980 Len=0 MSS=1452 SACK_PERM=1 TSval=3660938891 TSecr=673390741 WS=256
 26 28.136379  192.168.1.93    17.253.37.203   TCP      66 50009 → 80 [ACK] Seq=1 Ack=1 Win=132480 Len=0 TSval=673390751 TSecr=3660938891
 27 28.136836  192.168.1.93    17.253.37.203   HTTP    359 GET /ocsp03-wwdr06/ME4wTKADAgEAMEUwQzBBMAkGBSsOAwIaBQAEFADrDMz0cWy6RiOj1S%2BY1D32MKkdBB5IJxcJqbYYYIvs67r2R1nFUl5jtwIIAivyAGmAAhs%3D HTTP/1.1
 28 28.147635  17.253.37.203   192.168.1.93    TCP      66 80 → 50009 [ACK] Seq=1 Ack=294 Win=64768 Len=0 TSval=3660938903 TSecr=673390751
 29 28.148445  17.253.37.203   192.168.1.93    TCP    1506 80 → 50009 [ACK] Seq=1 Ack=294 Win=64768 Len=1440 TSval=3660938903 TSecr=673390751 [TCP segment of a reassembled PDU]
 30 28.148448  17.253.37.203   192.168.1.93    TCP    1506 80 → 50009 [PSH, ACK] Seq=1441 Ack=294 Win=64768 Len=1440 TSval=3660938903 TSecr=673390751 [TCP segment of a reassembled PDU]
 31 28.148512  192.168.1.93    17.253.37.203   TCP      66 50009 → 80 [ACK] Seq=294 Ack=2881 Win=129600 Len=0 TSval=673390762 TSecr=3660938903
```

Figure 16 - Wireshark data capture of the Port_80 dataset

The image above shows the packet log data captured when a DDoS attack was launched and the traffic coming into port 80 was monitored for DDoS data. this data has many columns with a lot of data features that can be unique and could theoretically be considered for us in machine learning. But we cannot use this data as it is in the machine learning code, it needs to be cleaned first. What that means is dropping columns that don't have useful information. Dropping packet logs that have "Null" or empty value fields. This then needs to be exported int CSV file from Wireshark.



	A	B	C	D	E	F	G	H	I	J	K
1	Time	Source	Destination	Protocol	Length	TotalLeng	SourcePor	Destinatic	DestinationMacAddress	SourceMacAddress	PacketClassification
2	0	192.168.1.93	151.101.62.133	TCP	78	64	49945	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
3	0.000082	192.168.1.93	151.101.62.133	TCP	78	64	49946	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
4	0.008606	151.101.62.133	192.168.1.93	TCP	74	60	80	49945	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
5	0.008665	192.168.1.93	151.101.62.133	TCP	66	52	49945	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
6	0.009778	151.101.62.133	192.168.1.93	TCP	74	60	80	49946	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
7	0.009844	192.168.1.93	151.101.62.133	TCP	66	52	49946	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
8	0.831732	192.168.1.93	172.217.16.227	TCP	78	64	49958	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
9	0.84152	172.217.16.227	192.168.1.93	TCP	74	60	80	49958	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
10	0.841606	192.168.1.93	172.217.16.227	TCP	66	52	49958	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
11	0.842304	192.168.1.93	172.217.16.227	HTTP	367	353	49958	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
12	0.852302	172.217.16.227	192.168.1.93	TCP	66	52	80	49958	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
13	0.85448	172.217.16.227	192.168.1.93	OCSP	778	764	80	49958	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
14	0.854538	192.168.1.93	172.217.16.227	TCP	66	52	49958	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
15	0.860092	192.168.1.93	172.217.16.227	TCP	66	52	49958	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
16	0.870397	172.217.16.227	192.168.1.93	TCP	66	52	80	49958	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
17	0.870458	192.168.1.93	172.217.16.227	TCP	66	52	49958	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
18	1.747224	192.168.1.93	104.21.18.67	TCP	78	64	49968	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
19	1.756441	104.21.18.67	192.168.1.93	TCP	66	52	80	49968	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
20	1.756513	192.168.1.93	104.21.18.67	TCP	54	40	49968	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
21	16.76589	104.21.18.67	192.168.1.93	TCP	60	40	80	49968	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal
22	16.765943	192.168.1.93	104.21.18.67	TCP	54	40	49968	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
23	16.783413	192.168.1.93	104.21.18.67	TCP	54	40	49968	80	78:65:59:a4:64:39	c4:b3:01:d8:fb:ff	normal
24	16.799619	104.21.18.67	192.168.1.93	TCP	60	40	80	49968	c4:b3:01:d8:fb:ff	78:65:59:a4:64:39	normal

Figure 17 - CSV file for Port_80 dataset (after data cleaning)

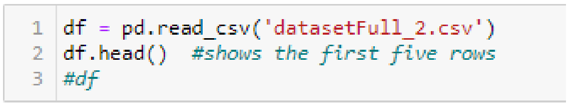
In the CSV file drop any leftover useless columns that might have slipped through from Wireshark. Second is a very important step where a new column needs to be added with the PacketClassification/label column name with values either being "normal" for normal traffic, and "ddos" for DDoS data.

- **Machine learning code**

There are 2 DDoS datasets one called "datasetFull_2.csv" which is the bigger dataset with traffic from all ports (any inbound and any outbound), and the second one is "datasetPort80_1.csv" which captured data only from port 80 (80 inbound and any outbound). Third, there is also a third dataset called "dataset_normalTrafficOnly.csv" which only has normal traffic data and therefore cannot be used with machine learning modes as that would require 2 or more type of unique data categories to exist in the dataset. So, the third dataset is useless in this situation

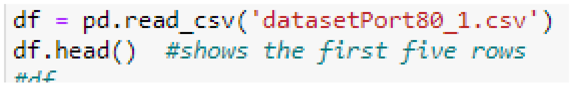
There are 2 separate machine learning, called "ML_DetectionModel_datasetPort_80" for the (datasetPort80_1.csv) dataset and the other is "ML_DetectionModel_datasetFull_2" for the (datasetFull_2.csv) dataset.

Both files are going to be the same except for I line of code that is where we are reading the dataset .csv file shown in the pictures below



```
df = pd.read_csv('datasetFull_2.csv')
df.head()  #shows the first five rows
#df
```

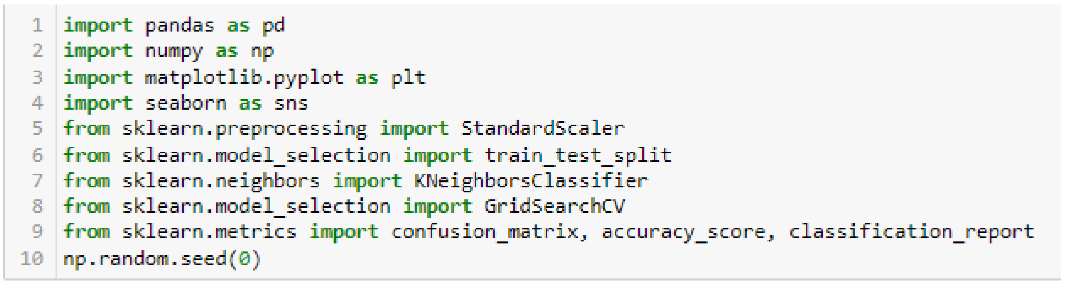
Figure 18 - ML code for datasetPort80_1



```
df = pd.read_csv('datasetPort80_1.csv')
df.head()  #shows the first five rows
```

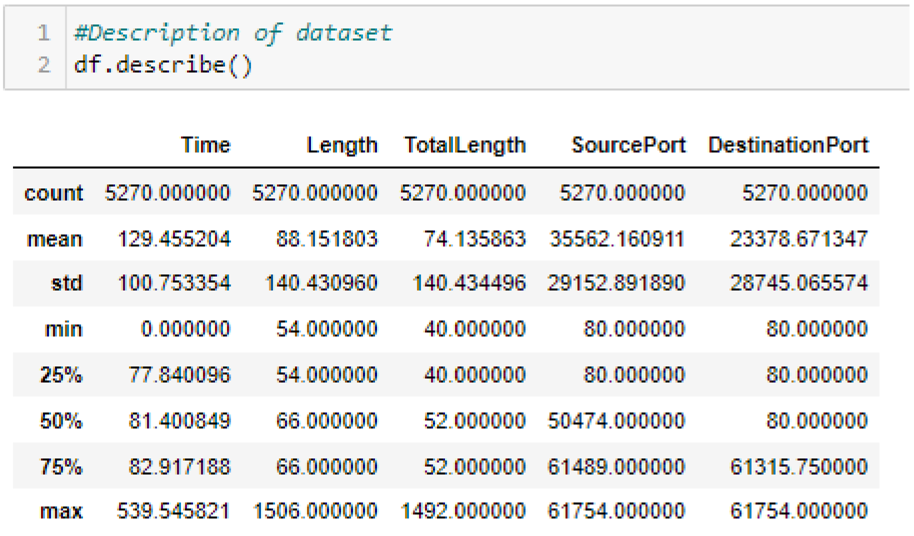
Figure 19 - ML code for datasetFull_2

Next, we import the required libraries shown below



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
np.random.seed(0)
```

Figure 20 – Python libraries



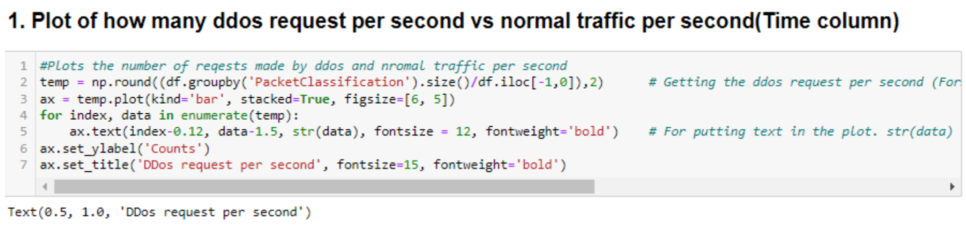
```python
#Description of dataset
df.describe()
```

	Time	Length	TotalLength	SourcePort	DestinationPort
count	5270.000000	5270.000000	5270.000000	5270.000000	5270.000000
mean	129.455204	88.151803	74.135863	35562.160911	23378.671347
std	100.753354	140.430960	140.434496	29152.891890	28745.065574
min	0.000000	54.000000	40.000000	80.000000	80.000000
25%	77.840096	54.000000	40.000000	80.000000	80.000000
50%	81.400849	66.000000	52.000000	50474.000000	80.000000
75%	82.917188	66.000000	52.000000	61489.000000	61315.750000
max	539.545821	1506.000000	1492.000000	61754.000000	61754.000000

21 - Summary of the dataset

From here on we are only going to look at the data from the Port 80 dataset but results for both datasets are similar to an extent. To check the results of the other dataset run the ipynb file inside Jupyter notebook environment.

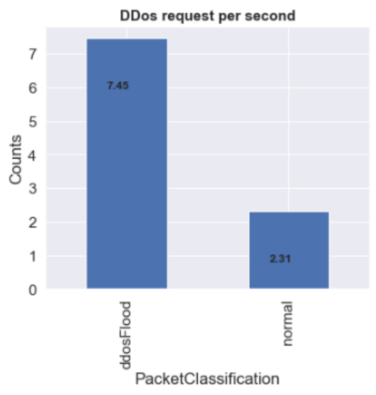
In the screenshot above it show the genereal description of the dataset.



1. Plot of how many ddos request per second vs normal traffic per second(Time column)

```python
#Plots the number of reqests made by ddos and nromal traffic per second
temp = np.round((df.groupby('PacketClassification').size()/df.iloc[-1,0]),2)      # Getting the ddos request per second (For
ax = temp.plot(kind='bar', stacked=True, figsize=[6, 5])
for index, data in enumerate(temp):
    ax.text(index-0.12, data-1.5, str(data), fontsize = 12, fontweight='bold')    # For putting text in the plot. str(data)
ax.set_ylabel('Counts')
ax.set_title('DDos request per second', fontsize=15, fontweight='bold')
```

```
Text(0.5, 1.0, 'DDos request per second')
```



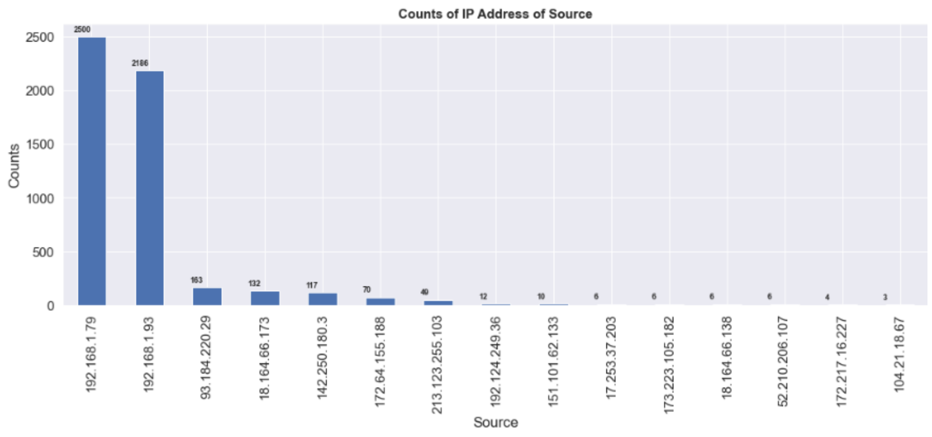
22 - Plot of DDoS requests vs normal requests per second

As can be seen in the plot during a ddos attack there are significantly more requests coming to the network device compared to when it is normal traffic. This is a sign of Traffic flood, a sign of an ongoing DDoS attack

2. Plot bar chart of ip address for source

```
#get the count of how many times an IP-address appears in the source column
temp = df.groupby('Source').size().sort_values(ascending=False)   # Getting the counts of Source IP and sorting into desce
ax = temp.plot(kind='bar', stacked=True, figsize=[18, 6])
for index, data in enumerate(temp):
    ax.text(index-0.30, data+50, str(data), fontsize = 9, fontweight='bold')    # For text in the plot
ax.set_ylabel('Counts')
ax.set_title('Counts of IP Address of Source', fontsize=15, fontweight='bold')
```

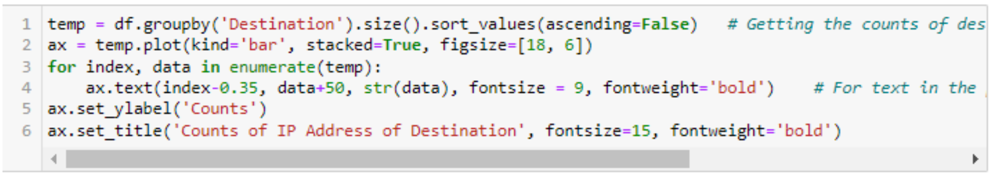
```
Text(0.5, 1.0, 'Counts of IP Address of Source')
```



23 - count of IP address in the source IP column

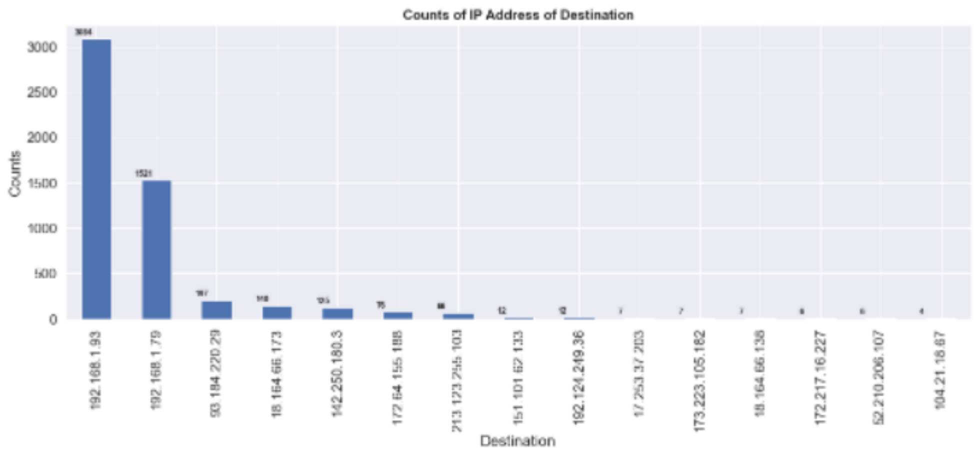
The plot above shows the count of how many times an IP address appears in the source IP column on the target device. This means that the IP 192.168.1.79 is the source of the ddos attack as can be seen by the massive amounts of traffic coming out from it.

3. Plot bar chart of ip address for destination



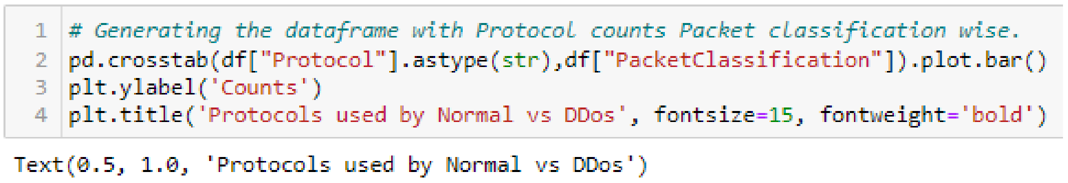
```
temp = df.groupby('Destination').size().sort_values(ascending=False)   # Getting the counts of des
ax = temp.plot(kind='bar', stacked=True, figsize=[18, 6])
for index, data in enumerate(temp):
    ax.text(index-0.35, data+50, str(data), fontsize = 9, fontweight='bold')     # For text in the
ax.set_ylabel('Counts')
ax.set_title('Counts of IP Address of Destination', fontsize=15, fontweight='bold')
```

```
Text(0.5, 1.0, 'Counts of IP Address of Destination')
```



24 - Plot of IP address in the destination IP columns

The plot above shows the count of how many times an IP address appears in the Destination IP column on the target device. This means that the IP 192,168.1.93 is the target of the ddos attack as can be seen by the massive amounts of traffic sent to it.



```
# Generating the dataframe with Protocol counts Packet classification wise.
pd.crosstab(df["Protocol"].astype(str),df["PacketClassification"]).plot.bar()
plt.ylabel('Counts')
plt.title('Protocols used by Normal vs DDos', fontsize=15, fontweight='bold')
```

```
Text(0.5, 1.0, 'Protocols used by Normal vs DDos')
```

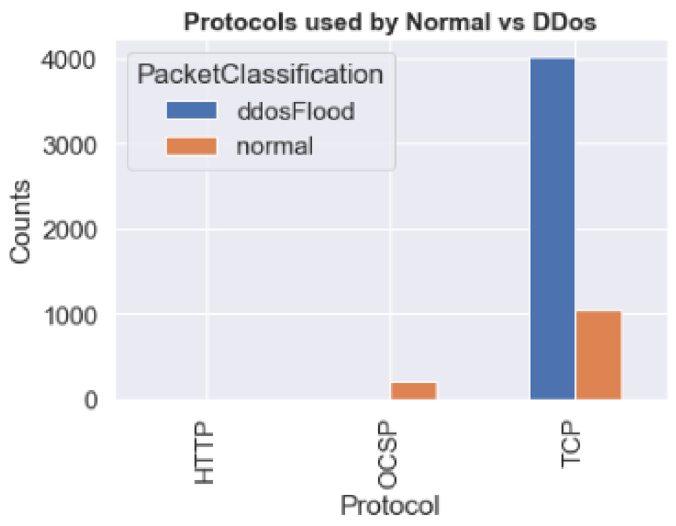
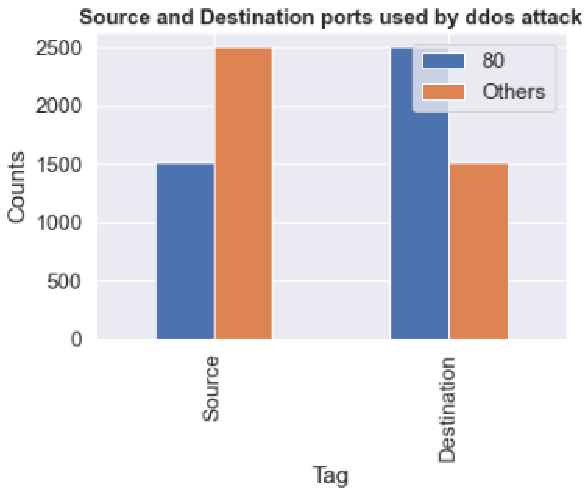


Figure 25 - Plot of Prtocol s sued by ddos and normal traffic

The plot above shows the distribution of the protocols used by DDoS traffic vs normal traffic. As can be seen most of the DDos traffic is TCP. That is because it use HTTP to send ddos request which is on the TCP-IP stack

```
#Plot
merged.plot.bar()
plt.ylabel('Counts')
plt.title('Source and Destination ports used by ddos attack', fontsize=15, fontweight='bold')
```

```
Text(0.5, 1.0, 'Source and Destination ports used by ddos attack')
```



26 - Plot of Source and destination ports used by DDoS attack traffic

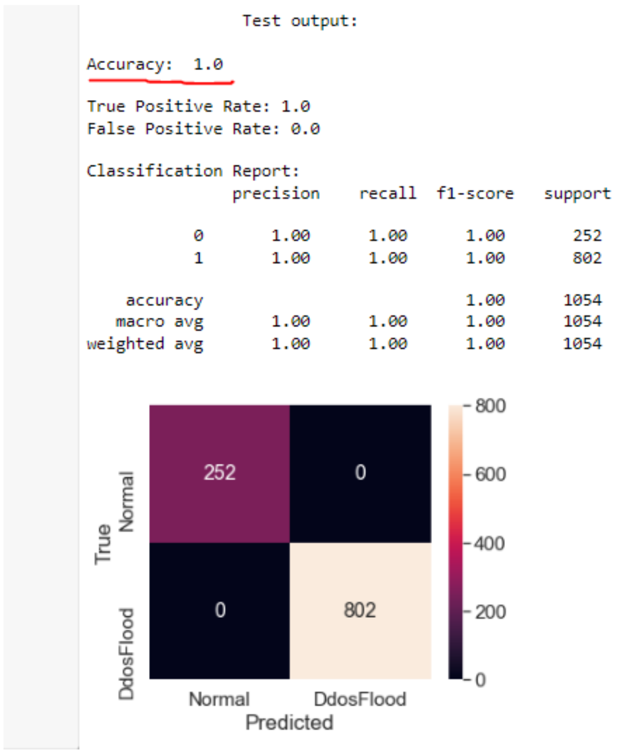
The plot above shows the distribution of port number that the source connection and destination connection used. As can be seen port 80is being over used on the incoming side by ddos traffic.

Let’s move on to the Machine learning models now and start with the

KNeearestNeighbor

```
model = KNeighborsClassifier()    # Classifier

# Hyperparameters to Tune
parameter_space = {
    'n_neighbors': np.arange(1,21)
}

# Hyperparameter Tunning
clf = GridSearchCV(model, parameter_space, cv = 5, scoring = "accuracy", verbose = True) # model
clf.fit(X_train.values,y_train)
print(f'Best Parameters: {clf.best_params_}')
```

Figure 27 - Model code

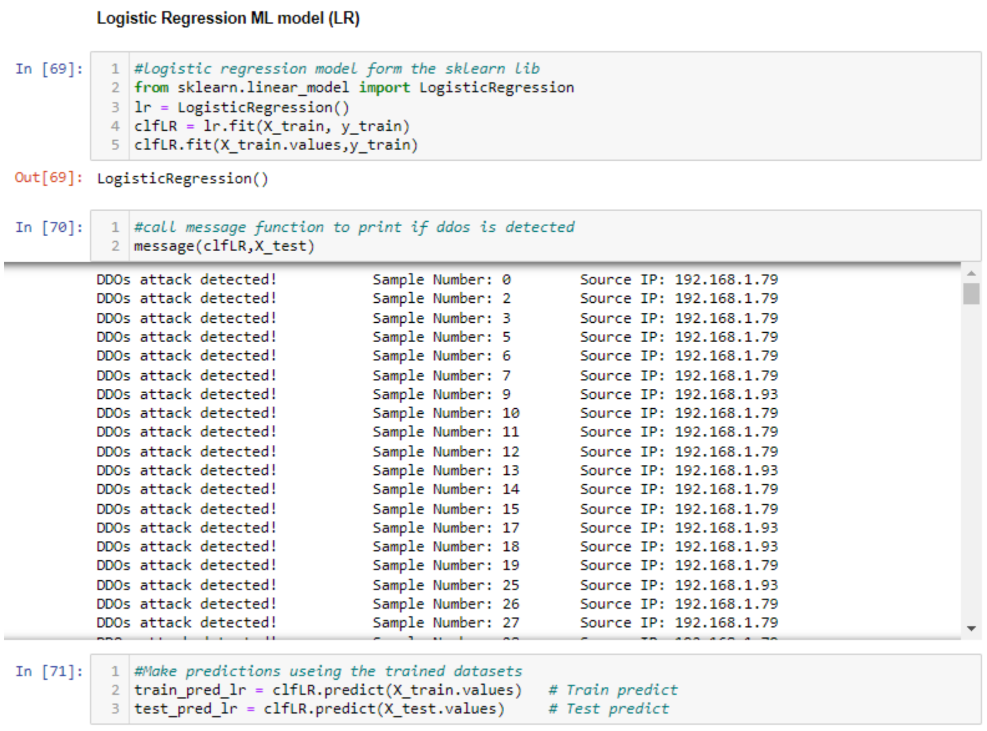


```
                Test output:

Accuracy:  1.0

True Positive Rate: 1.0
False Positive Rate: 0.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       252
           1       1.00      1.00      1.00       802

    accuracy                           1.00      1054
   macro avg       1.00      1.00      1.00      1054
weighted avg       1.00      1.00      1.00      1054
```

Figure 28 - Prediction Acuraccy for kNN model

As can be seen the KNearestNeighbour model has 100 percent accuracy on the prediction it made

Linear Regression model


Logistic Regression ML model (LR)

```
In [69]:
#logistic regression model form the sklearn lib
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
clfLR = lr.fit(X_train, y_train)
clfLR.fit(X_train.values,y_train)

Out[69]: LogisticRegression()

In [70]:
#call message function to print if ddos is detected
message(clfLR,X_test)

DDOs attack detected!          Sample Number: 0       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 2       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 3       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 5       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 6       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 7       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 9       Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 10      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 11      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 12      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 13      Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 14      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 15      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 17      Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 18      Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 19      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 25      Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 26      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 27      Source IP: 192.168.1.79

In [71]:
#Make predictions useing the trained datasets
train_pred_lr = clfLR.predict(X_train.values)   # Train predict
test_pred_lr = clfLR.predict(X_test.values)     # Test predict
```

Figure 29 - LR model code

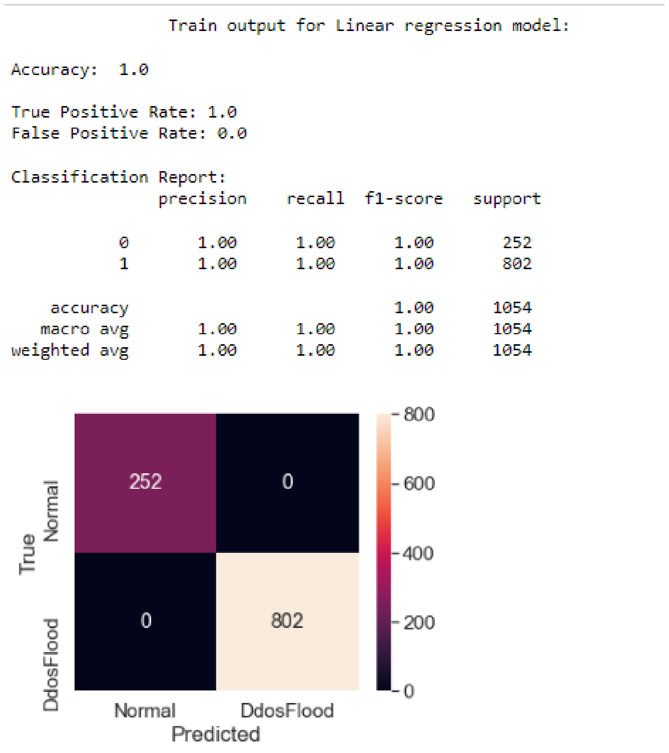
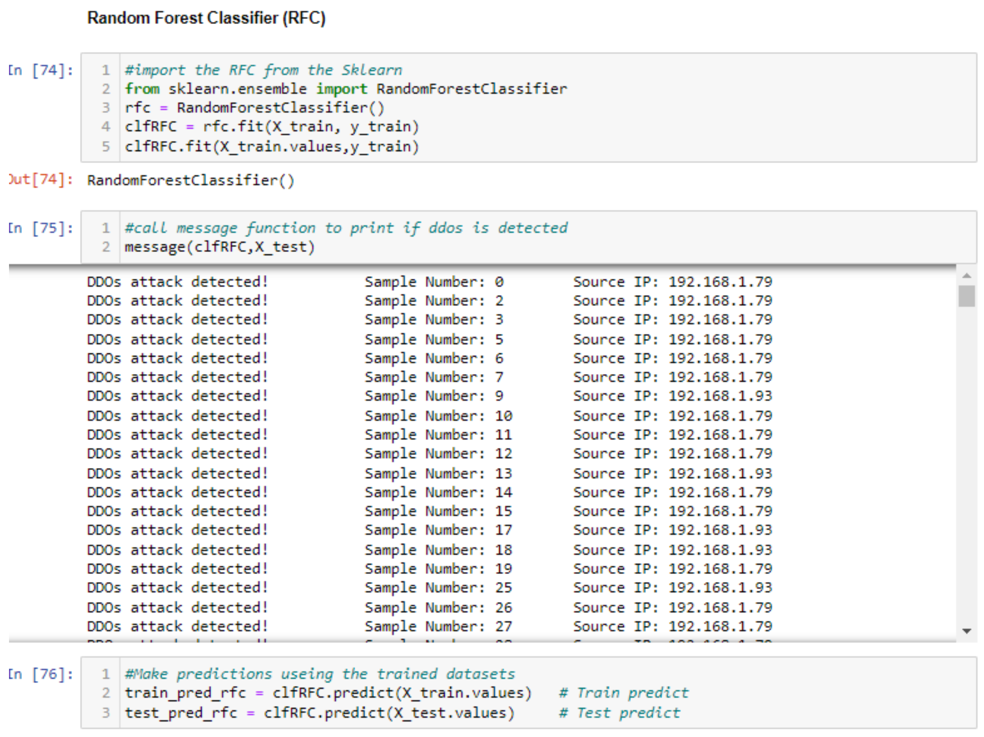

```
                  Train output for Linear regression model:

Accuracy:  1.0

True Positive Rate: 1.0
False Positive Rate: 0.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       252
           1       1.00      1.00      1.00       802

    accuracy                           1.00      1054
   macro avg       1.00      1.00      1.00      1054
weighted avg       1.00      1.00      1.00      1054
```

Figure 30 - Prediction Accuracy for LR model

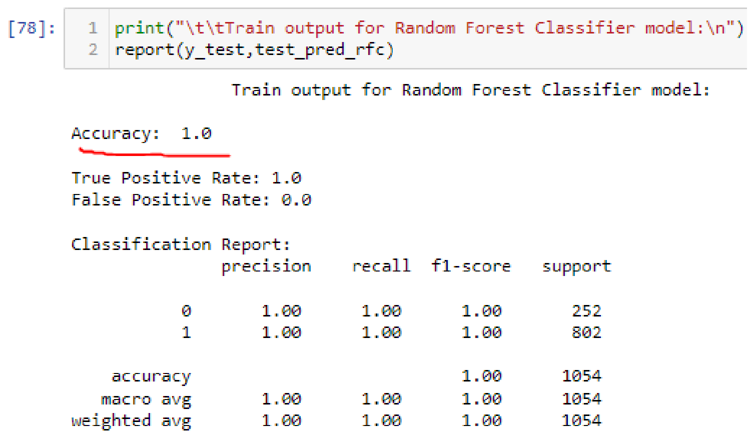
As can be seen the Accuracy of Linear regression model has 100 percent accuracy on the prediction it made.

Random Forest Classifier


Random Forest Classifier (RFC)

```
In [74]: #import the RFC from the Sklearn
         from sklearn.ensemble import RandomForestClassifier
         rfc = RandomForestClassifier()
         clfRFC = rfc.fit(X_train, y_train)
         clfRFC.fit(X_train.values,y_train)
Out[74]: RandomForestClassifier()

In [75]: #call message function to print if ddos is detected
         message(clfRFC,X_test)

DDOs attack detected!          Sample Number: 0        Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 2        Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 3        Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 5        Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 6        Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 7        Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 9        Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 10       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 11       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 12       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 13       Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 14       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 15       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 17       Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 18       Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 19       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 25       Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 26       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 27       Source IP: 192.168.1.79

In [76]: #Make predictions useing the trained datasets
         train_pred_rfc = clfRFC.predict(X_train.values)   # Train predict
         test_pred_rfc = clfRFC.predict(X_test.values)     # Test predict
```

Figure 31 - RFC model code



```
[78]:  print("\t\tTrain output for Random Forest Classifier model:\n")
       report(y_test,test_pred_rfc)

                Train output for Random Forest Classifier model:

Accuracy:  1.0

True Positive Rate: 1.0
False Positive Rate: 0.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       252
           1       1.00      1.00      1.00       802

    accuracy                           1.00      1054
   macro avg       1.00      1.00      1.00      1054
weighted avg       1.00      1.00      1.00      1054
```

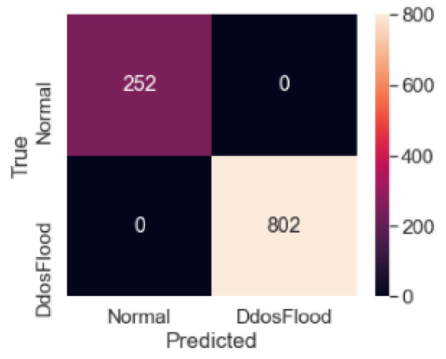
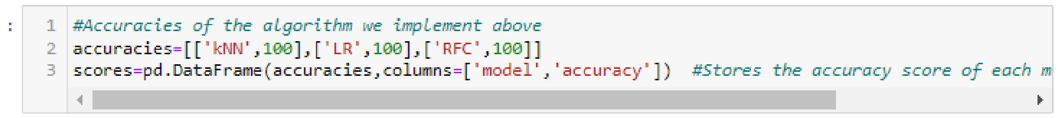
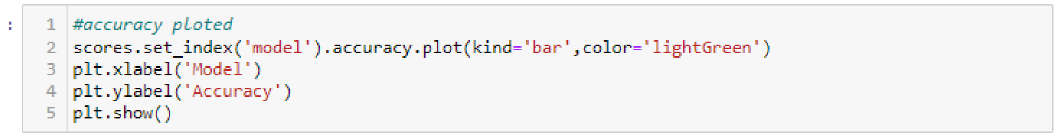


Figure 32 - Prediction Accuracy for RFC model

As can be seen the Accuracy of Random Forest classifier model has 100 percent accuracy on the prediction it made.



```
#Accuracies of the algorithm we implement above
accuracies=[['kNN',100],['LR',100],['RFC',100]]
scores=pd.DataFrame(accuracies,columns=['model','accuracy'])  #Stores the accuracy score of each m
```



```
#accuracy ploted
scores.set_index('model').accuracy.plot(kind='bar',color='lightGreen')
plt.xlabel('Model')
plt.ylabel('Accuracy')
plt.show()
```

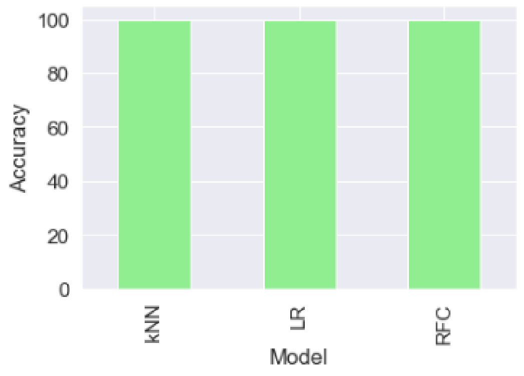


Figure 33 - Plot of ML model accuracy

This last plot shows the accuracies of all 3 models side by side and as can be seen all 3 of the models have full 100% accuracies rates for their prediction.

4. **Testing**

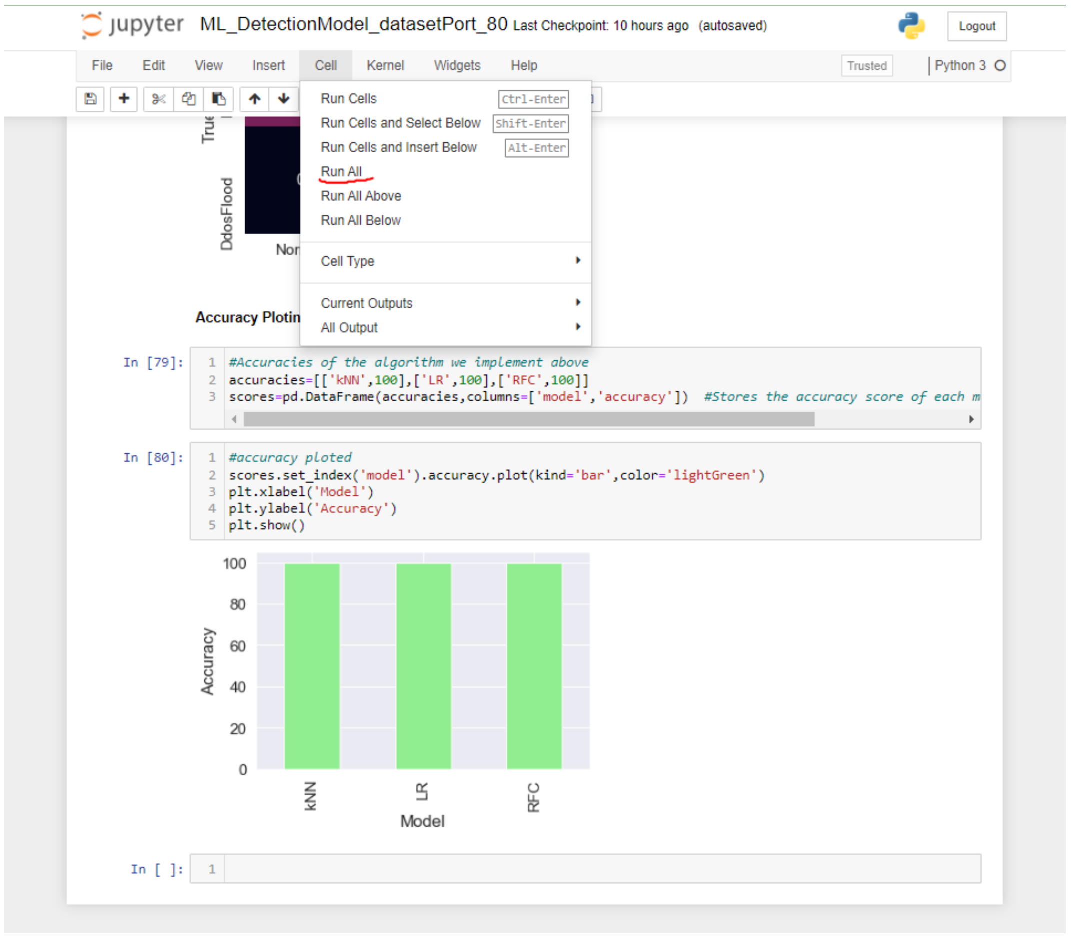
Table 6 - Risk analysis

Test	**Result**	**Fix**
Run the dataset “datasetnormalTraffic_Only”	Machine learning model runs into an error because the dataset only has one class	This dataset is incompatible with ML models does to its single class and it cannot be fixed
Launch a DDoS attack on a device without a firewall	The device was DDoSed	Activate the firewall to block incoming traffic floods

5. **Operation**

To run the application the user will need to download the “Project code” folder from the supporting documents and that can be downloaded and open it in Jupyter notebook. Install any packages required using the Anaconda CLI. A list of all the libraries needs and the command to install it in anaconda has been included in the supporting documents.

Now in Jupyter notebook click on “cell” and then click on “Run all” to run the ML module



```
In [79]:
#Accuracies of the algorithm we implement above
accuracies=[['kNN',100],['LR',100],['RFC',100]]
scores=pd.DataFrame(accuracies,columns=['model','accuracy'])  #Stores the accuracy score of each m
```

```
In [80]:
#accuracy ploted
scores.set_index('model').accuracy.plot(kind='bar',color='lightGreen')
plt.xlabel('Model')
plt.ylabel('Accuracy')
plt.show()
```

```
In [ ]:
```

Figure 34 - How to run ML model

Link to the demo video on YouTube: https://youtu.be/CTv38CVTYIU

Chapter 5

Conclusions

As detailed in the Abstract and introduction of this project it was designed to find out "if cyberattacks such as DDoS attacks could be detected using network log data in Machine learning models". At the end of the project, the Machine learning models were able to detect traces of cyber-attacks successfully on our datasets. In fact, all 3 models predicted the test dataset with 100% accuracy which meant that they were extremely efficient at making the predictions with 0 false positive and false negative predictions.

A form of lightweight Agile model called Scrum to structure and plan all stages of the software development life cycle (SDLC) for this project. This made it easy to manage this project as it was well suited to the loose requirements list that was made at the start of the project. Most key deadlines were met and most tasks were completed before the due date set for them. There were only 2 tasks that were not completed by their due date. The first one was the coding of the Machine learning model which was supposed to be completed by the end of March but was not completed till mid-May. This delay also had a landslide effect on the completion of the final report that was supposed to be completed a week before the due date of the project but the report was not completed until the 25th of May.

The list below are all the objectives identified in the aim, let us look at the progress through each of them below

1. Literature review:

The literature review was started at the start of the project and went on for 2 months till the end of December. This was a significant component of the project as before starting on the development requirements of the project Literature review was done to study the topic area. Initially, it was very difficult to find relevant literature as there is not much research done in the exact specific area of the project but eventually some sources were found that were closely related to the topic of the project. So, this objective was accomplished

2. Create a DDoS script:

This was the easiest part of the project. It was a day or two of research to find the libraries used for opening sockets that enable making network connections with other devices. This task was accomplished on time and to a very high standard. It even had multi-threading support.

3. Perform a DDoS attack and collect data:

This task was relatively short and was completed successfully within the time allotted. Using the DDoS script from the previous objective a DDoS attack was launched and on the target device, Wireshark was used to perform packet capture, a form of log data analysis. Wireshark is a very useful program that successfully captured the DDoS and normal packets and exported the captured database to the CSV format successfully.

4. Clean the data:

Cleaning the data was a lengthy process. this was the next step after the dataset was exported into CSV from Wireshark. Wireshark by default has some data columns that are not useful for Machine learning purposes. They contain additional information that is more human readable but not the data that is useful for ML. Some network packet entries also had missing and null values that needed to be filtered out and dropped. This task also had one of the most important steps in the project, adding the Packet Classifier/Label column that would specify if the packet record was a “ddos” or “normal” connection. This task was completed within the time limit.

5. Select 3 ML models to use with the dataset:

This step was easy to complete as at the start of the project the decision was made to use the scikit-learn machine library in python to complete this project. At this stage, the only thing that needed to be done is to select 3 models that would work with the 3 datasets we had unfortunately the normal dataset only had “Normal” traffic and ML models do not work with data that has only one classifier/Label so the only datasets that would were “dataset_port80” and the “dataset_full2”. So three models that were chosen are the following models from scikit-learn/sklearn

- KNearestNeighbor
- Linear regressor
- Random forests classifier

6. Import dataset and visualise the data:

This step involved a little bit of programming in python using python libraries such as Pandas, NumPy, matplotlib and seaborn. This contained plots shown in the previous section that helps the user visualise the data and find patterns in it. This task was completed within the time limit

7. Code ML models and get their accuracy:

All three ML models were successful coded and were able to make accurate predictions. This task was completed but unfortunately was delayed and was completed almost a month after its due date, also delaying the final report completion

8. Print to the user if DDoS is detected: this functionality was added to the code using a function that was created that checked if the packet data was predicted as a “ddos” attack packet and printed a warning message to the screen for the user to see

Does the artefact work well?

The main goal of the project was to detect cyber-attacks using log data in a machine learning model. The results and conclusion of this report are derived from the results that were obtained from the ML prediction models. Analysis of each of the model's results shows that all three models predicted the DDoS packets with full 100% accuracy and work very efficiently taking only about 10- 30 seconds to complete execution for both datasets. It was found out in the Literature review that other people had already successfully used machine learning models to detect cyberattacks and some even used DNNs for performing real-time attack detection at the bleeding edge of this field. Based on the results that were obtained from the artefact created for this project confirm the question asked in this project is possible does indeed work.

Chapter 6

Reflective Analysis

Throughout the timeline of this project, some things went smoothly, but there were also short tasks that were easy to complete and then there were larger objectives that were a lot of work to complete but were not difficult, then there were tasks small and big that was an absolute nightmare and very difficult to complete. This section is going to look at what went well and where things went wrong and what we could have done differently to avoid those problems.

What went well and the reasons why?

Many of the aspects of the project went well. In particular, the project planning phase as well as the plan drawn up and the model selected (SCRUM) were very flexible and suited to the project I was doing. At the start of the project, I was scared about choosing this topic as I felt I did not have enough knowledge about the topic the project was also made up of two phases (DDoS attack and Machine learning) but with the help of my supervisor and tutor, I was able to find resources that pointed me in the right direction and the initial tasks like Writing the DDoS scripts and collecting the datasets were easy to complete. It is worth noting though that the task of cleaning the dataset and getting ready for the machine learning was an easy task but it did take a very long time. And the Scrum model had to be used to adjust the initial due date by extending it by 3 weeks.

Another benefit of doing a project in networking is that it has given me a better understanding of how scripts can be coded to do different things with network components and I also learned about some of the networking libraries used in python. Writing the final report was also looking like a daunting task before I started the report after having written it I have learnt a lot about academic writing and the techniques used in it.

Another good aspect of the project was the tests I performed on codes of both phases after every Sprint helped me confirm that everything was working correctly

What could have gone better?

To start with I missed a couple of due dates for some of the tasks. Most important of which are: First, I was delayed by a lot in finishing the python Machine learning task. It was due at the end of March but I did not finish it until mid-May. The biggest problem was with implementing the functions for pre-processing, orienting the DDoS message and getting them to work properly. This also pushed back my plan to start the project report and ended up delaying the project report as well by a week and very close to the project submission deadline.

Another big problem I ran into was finding good literature at the start of the project. When I started to research journals, it was very hard to find matching sources that were closely related to my project topic, but eventually, I did end up finding a couple of journals that were based on related topics but they still were not extremely relevant.

Another issue is I wanted to have a dataset with just normal traffic as a classifier and pass that to the machine learning model as well but this is not possible because an ML model needs either 2 or more classifiers/labels to be able to classify results

What could have been better with more time?

Without the time barrier, I could have tried different DDoS scripts. The one used in this assignment only sends HTTP requests over the network, but there are many types of scripts that can be created that can attacks and exploit different layers of the OSI and TCP/IP model which would have provided much more depth to the project. And it certainly would have been very interesting to check the results from the data collected from these other attack types.

With more time I would also like to supply a script file with install commands for all the required python libraries and dependencies so that the user could run this file first and then they would not run into library not found errors if they don’t have them installed on their machines. Right now, it requires the user to find the python libraries that are used in the ML and Script code and install them by themselves before running the project code.

References

Alghamdie, M. (2021) Cyber security attack purpose. [image] Available from https://ars.els-cdn.com/content/image/1-s2.0-S2214785321029345-gr2.jpg [Accessed 16 November 2021]. (cit. on p. 2).

Vinayakumar, R., Alazab, M., Soman, K., Poornachandran, P., Al-Nemrat, A. and Venka-traman, S. (2019). Deep Learning Approach for Intelligent Intrusion Detection System. IEEE Access, [online] 7, pp.41525-41550. Available from https://ieeexplore-ieee-org.proxy.library.lincoln.ac.uk/stamp/stamp.jsp?tp=&arnumber=8681044&tag=1 [Accessed 23 February 2022]. (cit. on p. 6).

Musumeci, F., Fidanci, A., Paolucci, F., Cugini, F. and Tornatore, M., 2021. Machine-Learning-Enabled DDoS Attacks Detection in P4 Programmable Networks. Journal of Network and Systems Management, 30(1).

Sahoo, K., 2022. A Comprehensive Tutorial on Software Defined Network: The Driving Force for the Future Internet Technology. [image] Available from https://dl-acm-org.proxy.library.lincoln.ac.uk/doi/pdf/10.1145/2979779.2983928 [Accessed 22 May 2022]. (cit. on p. 7).

Sahoo, K., Panda, S., Sahoo, S., Sahoo, B. and Dash, R., 2019. Toward secure software-defined networks against distributed denial of service attack. The Journal of Supercomputing, 75(8), pp.4829-4874.

Alghamdie, M. (2021). A novel study of preventing the cyber security threats. Materials Today: Proceedings, [online] p.1-2. Available from https://www.sciencedirect.com/science/article/pii/S2214785321029345 [Accessed 16 No-vember 2021]. (cit. on p. 6).

Huertas Celdrán, A., Karmakar, K., Gómez Mármol, F. and Varadharajan, V. (2021). De-tecting and mitigating cyberattacks using software defined networks for integrated clinical environments. Peer-to-Peer Networking and

Applications, [online] 14(5), pp.2719-2734. Available from https://link-springer-com.proxy.library.lincoln.ac.uk/article/10.1007/s12083-021-01082-w [Accessed 18 November 2021].

Kamiya, S., Kang, J., Kim, J., Milidonis, A. and Stulz, R. (2021). Risk management, firm reputation, and the impact of successful cyberattacks on target firms. Journal of Financial Economics, [online] 139(3), pp.719-749. Available from https://www.sciencedirect.com/science/article/pii/S0304405X20300143 [Accessed 16 No-vember 2021].

Liu, J., Yang, C., Chan, Y., Kristiani, E. and Jiang, W. (2021). Cyberattack detection model using deep learning in a network log system with data visualization. The Journal of Super-computing, [online] 77(10), pp.10984-11003. Available from https://link-springer-com.proxy.library.lincoln.ac.uk/article/10.1007/s11227-021-03715-6 [Accessed 17 Novem-ber 2021].

Ning, X. and Jiang, J. (2021). Design, Analysis and Implementation of a Security Assess-ment/Enhancement Platform for Cyber-Physical Systems. IEEE Transactions on Industrial Informatics, [online] 18(2), pp.1154-1164. Available from https://ieeexplore-ieee-org.proxy.library.lincoln.ac.uk/document/9444805 [Accessed 18 November 2021].

Ning, X. and Jiang, J. (2021). Functional blocks in the security enhancement module.. [im-age] Available from https://ieeexplore-ieee-org.proxy.library.lincoln.ac.uk/mediastore_new/IEEE/content/media/9424/9594725/9444805/jiang4-3085543-small.gif [Accessed 18 November 2021].

R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat and S. Ven-katraman (2019). Deep Learning Approach for Intelligent Intrusion Detection System. in IEEE Access, [online] vol. 7, pp. 41525-41550, Available from https://ieeexplore-ieee-org.proxy.library.lincoln.ac.uk/document/8681044/authors#authors [Accessed 18 November 2021].

Vaishali, V. (2021). Types of Cyber-attacks. [image] Available from https://d1m75rqqgidzqn.cloudfront.net/2019/10/26-september-cyber-security-attack-infographic-1.jpg [Accessed 18 November 2021].

Yang, C., Kristiani, E., Wang, Y., Min, G., Lai, C. and Jiang, W. (2019). On construction of a network log management system using ELK Stack with Ceph. The Journal of Supercom-puting, [online] 76(8), pp.6344-6360. Available from https://link-springer-com.proxy.library.lincoln.ac.uk/article/10.1007%2Fs11227-019-02853-2 [Accessed 18 No-vember 2021].

NeuralNine. (2019). *Code A DDOS Script In Python - NeuralNine*. [online] Available from https://www.neuralnine.com/code-a-ddos-script-in-python/ [Accessed 21 May 2022].

CloudFlare. (2022). *What is a DDoS attack?*. [online] Available from https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/ [Accessed 21 May 2022].

Kerzner, H. (2003_. *Project management workbook*. Hoboken, N.J.: Wiley, p.4. (cit. on p. 10).

Friedman, J., 2020. *5 Phases of Project Management*. [image] Available from https://www.truenxus.com/blog/five-phases-of-project-management [Accessed 23 May 2022]. (cit. on p. 11).

Cohen, E. (2022). The Definitive Guide to Project Management Methodologies. [Blog] *workamajig*, Available from https://www.workamajig.com/blog/project-management-methodologies [Accessed 23 May 2022].

Cohen, E., (2022). *The Definitive Guide to Project Management Methodologies*. [image] Available from https://www.workamajig.com/blog/project-management-methodologies [Accessed 23 May 2022]

Lavanya, N. and Malarvizhi, T. (2008). *Risk Analysis And Management*. [online] Pmi.org. Available from https://www.pmi.org/learning/library/risk-analysis project management-7070 [Accessed 24 May 2022]

Björkholm, T. and Björkholm, J. (2015*) Kanban in 30 Days. Birmingham, UK: Impackt Publishing*. Available from: https://search.ebscohostcom.proxy.library.lincoln.ac.uk/login.aspx?direct=true&db=e020mww&AN=1021979&site=e ds-live&scope=site [accessed: 25 May 2022].

Rubin, K. (2013) Essential Scrum. Upper Saddle River, NJ: Addison-Wesley.

Sommerville, I. (2016) Software engineering, 10th edition. Boston, USA: Pearson Education Limited

Tutorialspoint.com. (2022). *Python - Network Programming*. [online] Available from https://www.tutorialspoint.com/python/python_networking.htm [Accessed 25 May 2022]

Lashchuck, A. (2022). *8 Reasons Why Python is Good for Artificial Intelligence and Machine Learning*. [online] Medium. Available from https://towardsdatascience.com/8-reasons-why-python-is-good-for-artificial-intelligence-and-machine-learning-4a23f6bed2e6 [Accessed 25 May 2022]

Teamgantt.com. (2022). *Gantt Charts: Definitions, Features, & Uses | TeamGantt*. [online] Available from https://www.teamgantt.com/what-is-a-gantt-chart [Accessed 25 May 2022]

Chacon, S. and Straub, B. (2014) Pro git. Apress.

D. Booth. Libguides.newcastle.edu.au. (2019*). Libguides: Research Methods: What Are Research Methods?*. [online] Available from https://libguides.newcastle.edu.au/researchmethods [Accessed 25 May 2022].

Python, R. (2022). *An Intro to Threading in Python – Real Python*. [online] Realpython.com. Available from https://realpython.com/intro-to-python-threading/#what-is-a-thread [Accessed 25 May 2022]

www.javatpoint.com. (2022). *Pandas vs NumPy - javatpoint*. [online] Available from https://www.javatpoint.com/pandas-vs-numpy [Accessed 25 May 2022]

Matplotlib (2020) *Matplotlib N/A*: Matplotlib. Available from https://matplotlib.org/ [Accessed 25 May 2022].

Scikit-Learn.org (2020) *Scikit Learn*. Available from https://scikit-learn.org/stable/ [Accessed 25 May 2022]

University of Lincoln (2020) Referencing Handbook: Harvard. Lincoln: University of Lincoln.

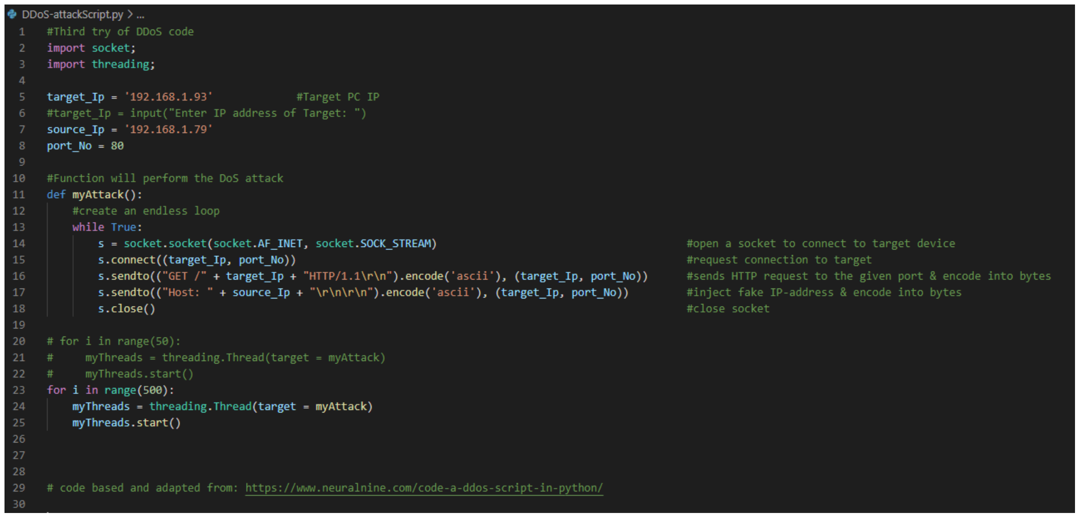
Word Count

11756

Appendix

Link to the demo video on YouTube: https://youtu.be/CTv38CVTYIU

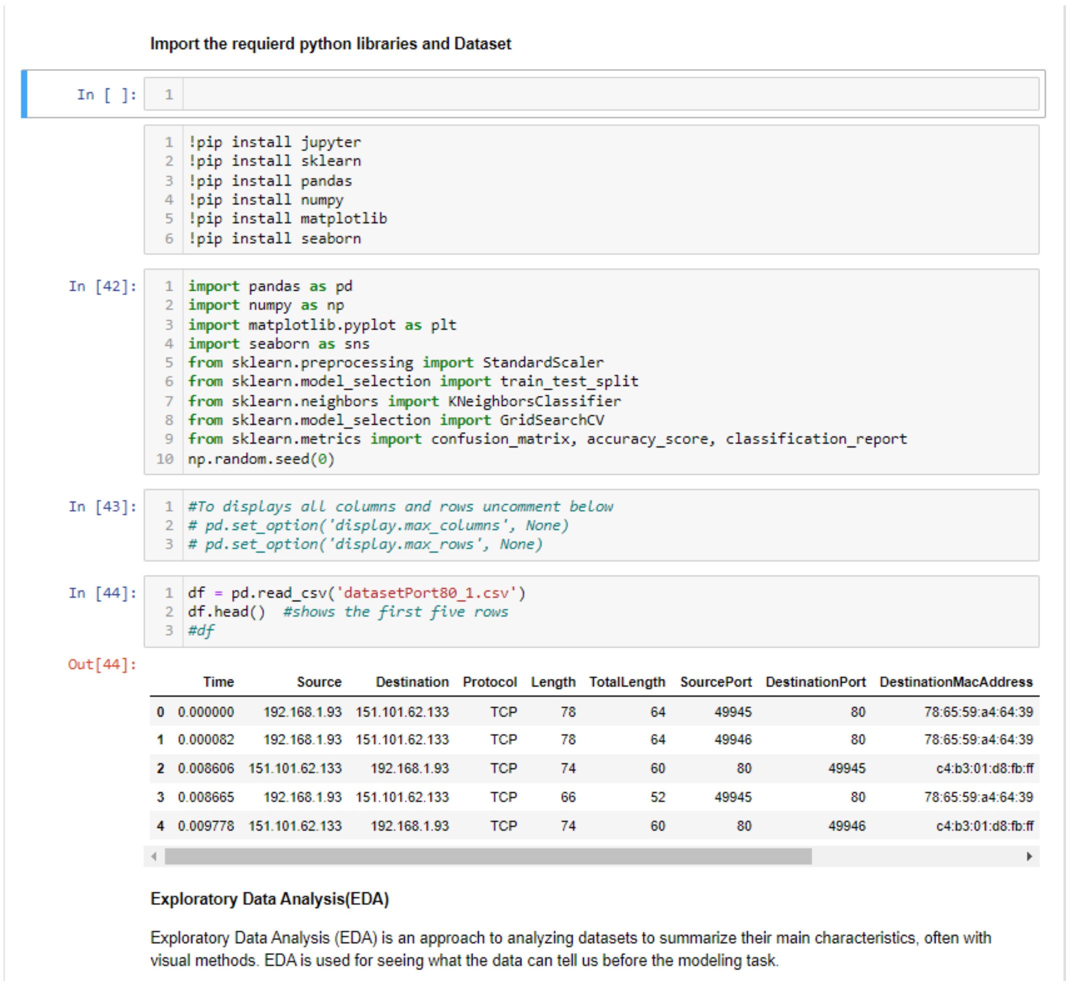
A.1 DDoS script code



DDoS-attackScript.py > ...

```python
#Third try of DDoS code
import socket;
import threading;

target_Ip = '192.168.1.93'              #Target PC IP
#target_Ip = input("Enter IP address of Target: ")
source_Ip = '192.168.1.79'
port_No = 80

#Function will perform the DoS attack
def myAttack():
    #create an endless loop
    while True:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)                                   #open a socket to connect to target device
        s.connect((target_Ip, port_No))                                                        #request connection to target
        s.sendto(("GET /" + target_Ip + "HTTP/1.1\r\n").encode('ascii'), (target_Ip, port_No))      #sends HTTP request to the given port & encode into bytes
        s.sendto(("Host: " + source_Ip + "\r\n\r\n").encode('ascii'), (target_Ip, port_No))        #inject fake IP-address & encode into bytes
        s.close()                                                                               #close socket

# for i in range(50):
#     myThreads = threading.Thread(target = myAttack)
#     myThreads.start()
for i in range(500):
    myThreads = threading.Thread(target = myAttack)
    myThreads.start()



# code based and adapted from: https://www.neuralnine.com/code-a-ddos-script-in-python/

```

A.2 ML_DetectionModel_datasetPort_80 Code



Import the requierd python libraries and Dataset

In [ ]:

```

```

```
!pip install jupyter
!pip install sklearn
!pip install pandas
!pip install numpy
!pip install matplotlib
!pip install seaborn
```

In [42]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
np.random.seed(0)
```

In [43]:

```
#To displays all columns and rows uncomment below
# pd.set_option('display.max_columns', None)
# pd.set_option('display.max_rows', None)
```

In [44]:

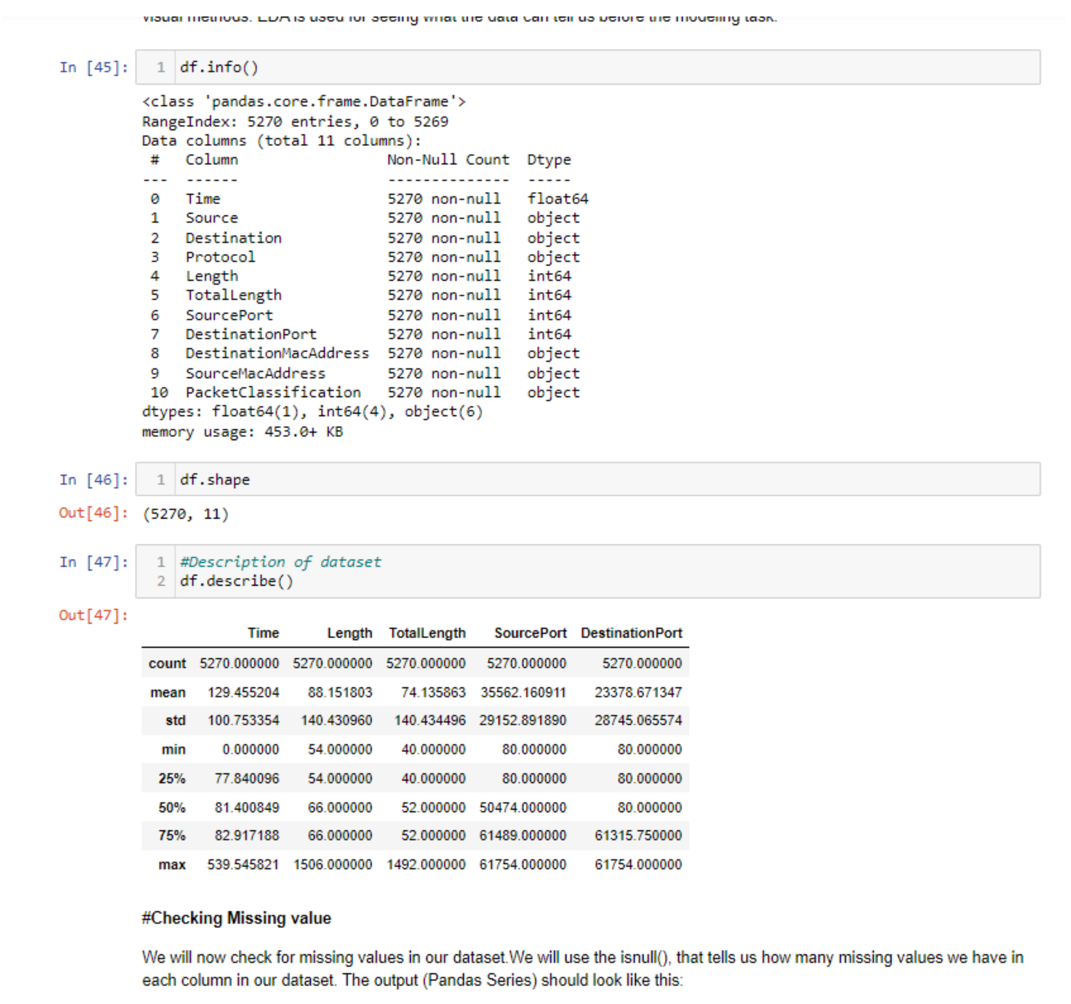
```
df = pd.read_csv('datasetPort80_1.csv')
df.head()  #shows the first five rows
#df
```

Out[44]:

	Time	Source	Destination	Protocol	Length	TotalLength	SourcePort	DestinationPort	DestinationMacAddress
0	0.000000	192.168.1.93	151.101.62.133	TCP	78	64	49945	80	78:65:59:a4:64:39
1	0.000082	192.168.1.93	151.101.62.133	TCP	78	64	49946	80	78:65:59:a4:64:39
2	0.008606	151.101.62.133	192.168.1.93	TCP	74	60	80	49945	c4:b3:01:d8:fb:ff
3	0.008665	192.168.1.93	151.101.62.133	TCP	66	52	49945	80	78:65:59:a4:64:39
4	0.009778	151.101.62.133	192.168.1.93	TCP	74	60	80	49946	c4:b3:01:d8:fb:ff

Exploratory Data Analysis(EDA)

Exploratory Data Analysis (EDA) is an approach to analyzing datasets to summarize their main characteristics, often with visual methods. EDA is used for seeing what the data can tell us before the modeling task.



visual methods. EDA is used for seeing what the data can tell us before the modeling task.

In [45]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5270 entries, 0 to 5269
Data columns (total 11 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Time                   5270 non-null   float64
 1   Source                 5270 non-null   object 
 2   Destination            5270 non-null   object 
 3   Protocol               5270 non-null   object 
 4   Length                 5270 non-null   int64  
 5   TotalLength            5270 non-null   int64  
 6   SourcePort             5270 non-null   int64  
 7   DestinationPort        5270 non-null   int64  
 8   DestinationMacAddress  5270 non-null   object 
 9   SourceMacAddress       5270 non-null   object 
 10  PacketClassification   5270 non-null   object 
dtypes: float64(1), int64(4), object(6)
memory usage: 453.0+ KB
```

In [46]:
```
df.shape
```

Out[46]:
```
(5270, 11)
```

In [47]:
```
#Description of dataset
df.describe()
```

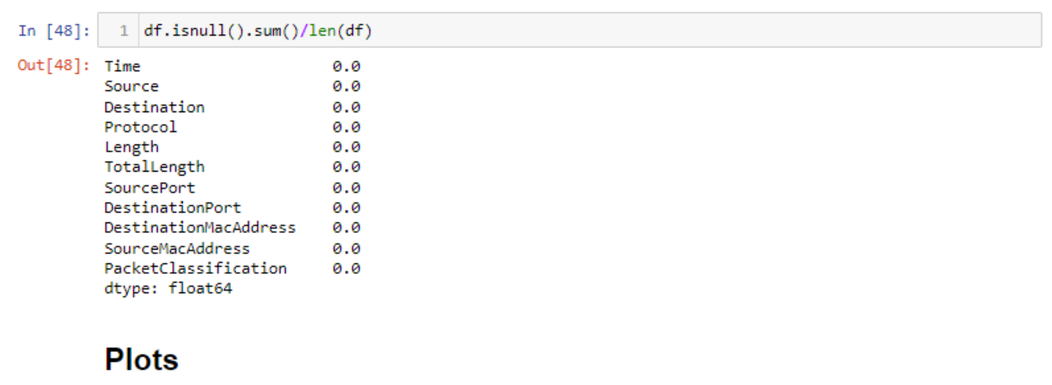
Out[47]:

	Time	Length	TotalLength	SourcePort	DestinationPort
count	5270.000000	5270.000000	5270.000000	5270.000000	5270.000000
mean	129.455204	88.151803	74.135863	35562.160911	23378.671347
std	100.753354	140.430960	140.434496	29152.891890	28745.065574
min	0.000000	54.000000	40.000000	80.000000	80.000000
25%	77.840096	54.000000	40.000000	80.000000	80.000000
50%	81.400849	66.000000	52.000000	50474.000000	80.000000
75%	82.917188	66.000000	52.000000	61489.000000	61315.750000
max	539.545821	1506.000000	1492.000000	61754.000000	61754.000000

#Checking Missing value

We will now check for missing values in our dataset.We will use the isnull(), that tells us how many missing values we have in each column in our dataset. The output (Pandas Series) should look like this:

each column in our dataset. The output (Pandas Series) should look like this:



In [48]:
```
df.isnull().sum()/len(df)
```

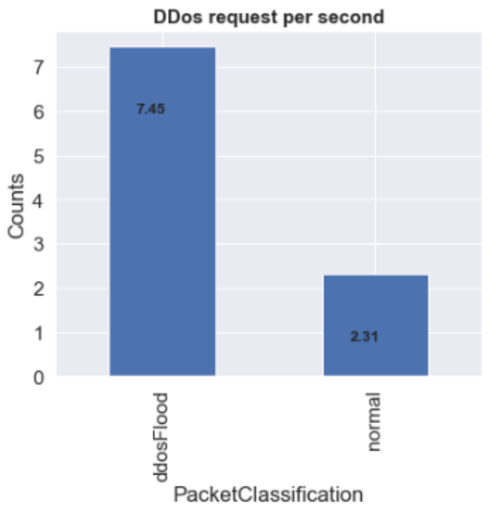
Out[48]:
```
Time                     0.0
Source                   0.0
Destination              0.0
Protocol                 0.0
Length                   0.0
TotalLength              0.0
SourcePort               0.0
DestinationPort          0.0
DestinationMacAddress    0.0
SourceMacAddress         0.0
PacketClassification     0.0
dtype: float64
```

Plots

1. Plot of how many ddos request per second vs normal traffic per second(Time column) ¶

```
9]:
#Plots the number of reqests made by ddos and nromal traffic per second
temp = np.round((df.groupby('PacketClassification').size()/df.iloc[-1,0]),2)      # Getting the dd
ax = temp.plot(kind='bar', stacked=True, figsize=[6, 5])
for index, data in enumerate(temp):
    ax.text(index-0.12, data-1.5, str(data), fontsize = 12, fontweight='bold')    # For putting te
ax.set_ylabel('Counts')
ax.set_title('DDos request per second', fontsize=15, fontweight='bold')
```

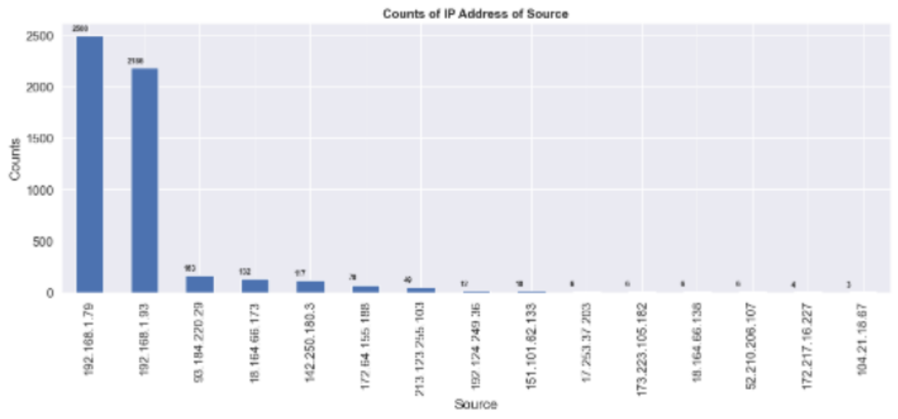
```
9]: Text(0.5, 1.0, 'DDos request per second')
```



2. Plot bar chart of ip address for source

```
0]:
#get the count of how many times an IP-address appears in the source column
temp = df.groupby('Source').size().sort_values(ascending=False)    # Getting the counts of Source I
ax = temp.plot(kind='bar', stacked=True, figsize=[18, 6])
for index, data in enumerate(temp):
    ax.text(index-0.30, data+50, str(data), fontsize = 9, fontweight='bold')     # For text in the
ax.set_ylabel('Counts')
ax.set_title('Counts of IP Address of Source', fontsize=15, fontweight='bold')
```

Out[50]:
```
Text(0.5, 1.0, 'Counts of IP Address of Source')
```

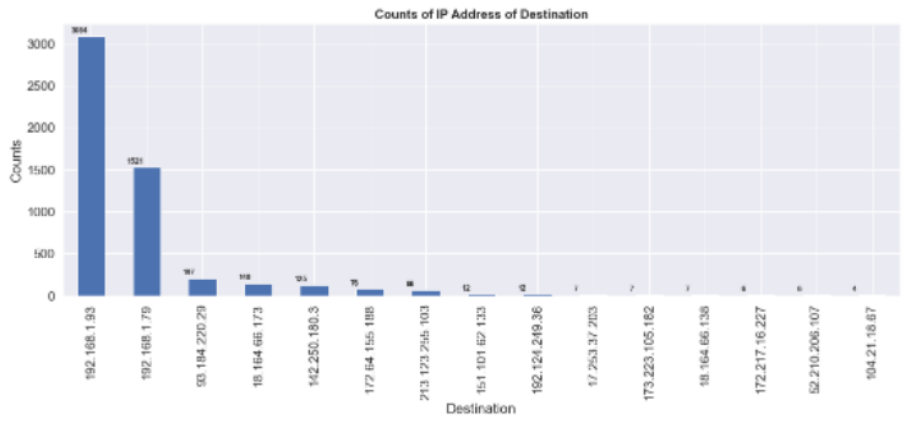


3. Plot bar chart of ip address for destination

In [51]:
```python
temp = df.groupby('Destination').size().sort_values(ascending=False)   # Getting the counts of des
ax = temp.plot(kind='bar', stacked=True, figsize=[18, 6])
for index, data in enumerate(temp):
    ax.text(index-0.35, data+50, str(data), fontsize = 9, fontweight='bold')    # For text in the
ax.set_ylabel('Counts')
ax.set_title('Counts of IP Address of Destination', fontsize=15, fontweight='bold')
```

Out[51]:
```
Text(0.5, 1.0, 'Counts of IP Address of Destination')
```

Out[51]:
```
Text(0.5, 1.0, 'Counts of IP Address of Destination')
```



4. Plot of protocol used by ddos vs normal

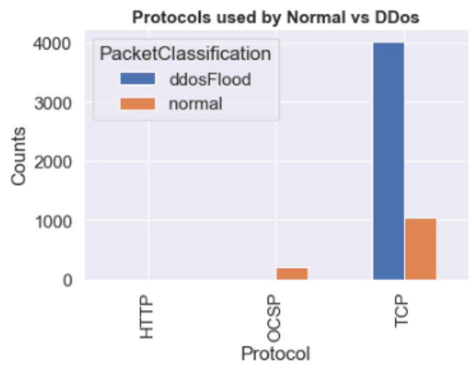
In [52]:
```python
#Protocol breakdown based on AttackTyoe/PacketClassification
pd.crosstab(df["Protocol"],df["PacketClassification"])
```

Out[52]:

PacketClassification	ddosFlood	normal
Protocol		
HTTP	0	4
OCSP	0	201
TCP	4021	1044

In [53]:
```
# Generating the dataframe with Protocol counts Packet classification wise.
pd.crosstab(df["Protocol"].astype(str),df["PacketClassification"]).plot.bar()
plt.ylabel('Counts')
plt.title('Protocols used by Normal vs DDos', fontsize=15, fontweight='bold')
```

Out[53]:
```
Text(0.5, 1.0, 'Protocols used by Normal vs DDos')
```



5. Plot of source and destination ports used by ddos attack

In [54]:
```
#create new df Source which counts unique Source port numbers used by ddos request
Source = df[df['PacketClassification'] == 'ddosFlood'].groupby('SourcePort').size().sort_values(as
#print(Source)
Source['Others'] = Source.iloc[:, 1:].sum(axis=1)  #sums the value in all columns after 1st column
Source['Others']
s = Source[[80,'Others']]
print(s)
```

```
SourcePort     80  Others
0            1521    2500
```

In [55]:
```
#create new df Dest which counts unique destination port numbers used by ddos request
Dest = df[df['PacketClassification'] == 'ddosFlood'].groupby('DestinationPort').size().sort_values
Dest['Others'] = Dest.iloc[:, 1:].sum(axis=1)   #sums the value in all columns after 1st column(i.
d  = Dest[[80,'Others']]
print(d)
```

```
DestinationPort     80  Others
0                 2500    1521
```

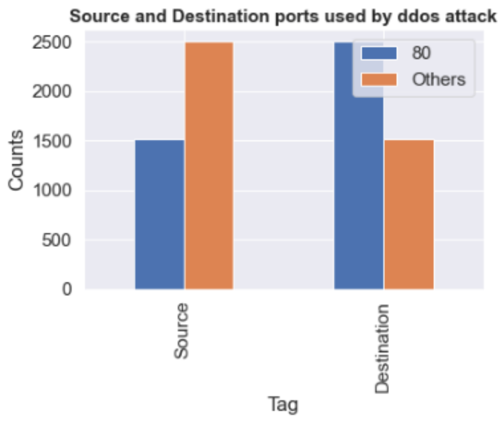
In [56]:
```python
#merge the results from s and d into one table/dataframe
merged = s.append(d)
merged['Tag'] = ['Source','Destination']    #create row headers/titles
merged = merged.set_index('Tag')
merged
```

Out[56]:

Tag	80	Others
Source	1521	2500
Destination	2500	1521

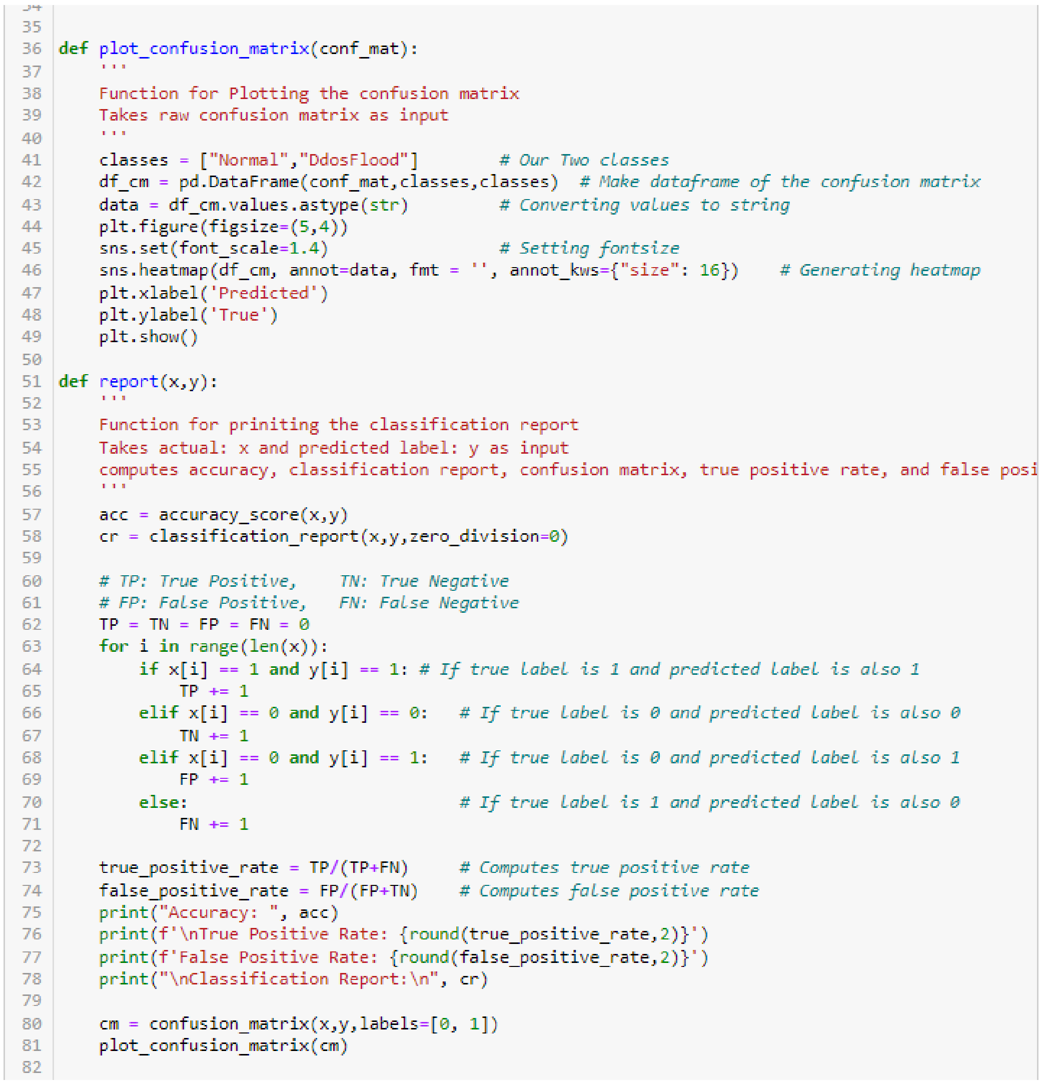
In [57]:
```python
#Plot
merged.plot.bar()
plt.ylabel('Counts')
plt.title('Source and Destination ports used by ddos attack', fontsize=15, fontweight='bold')
```

Out[57]:
```
Text(0.5, 1.0, 'Source and Destination ports used by ddos attack')
```



Some Utility Functions

In [58]:
```python
def preprocessing(df):
    '''
    Function for preprocessing of the data
    Takes dataframe as input
    Does following preprocessing,
    1) drops time column
    2) Does One hot encoding of Categorical columns
    3) Does Label encoding of Target Variable
    4) Standardize the data

    Return final dataframe
    '''
    df = df.drop(columns=['Time'])

    for col in ['Source', 'Destination', 'Protocol', 'DestinationMacAddress', 'SourceMacAddress']:
        # Get dummies of the column and right merge into the original dataframe
        df = pd.merge(left=df,
                      right=pd.get_dummies(df[col], prefix=col, prefix_sep='_'),
                      left_index=True,
                      right_index=True
        )
        df = df.drop(columns = [col])

    # Replacing Normal with 0 and ddosFlood with 1
    df = df.replace({'PacketClassification':{'normal':0,'ddosFlood':1}})

    # Scaling only specific columns
    col = ['Length', 'TotalLength', 'SourcePort', 'DestinationPort']
    scaler = StandardScaler()
    X = scaler.fit_transform(df[col])
    df[col] = X

    return df
```



```python

def plot_confusion_matrix(conf_mat):
    '''
    Function for Plotting the confusion matrix
    Takes raw confusion matrix as input
    '''
    classes = ["Normal","DdosFlood"]        # Our Two classes
    df_cm = pd.DataFrame(conf_mat,classes,classes)  # Make dataframe of the confusion matrix
    data = df_cm.values.astype(str)         # Converting values to string
    plt.figure(figsize=(5,4))
    sns.set(font_scale=1.4)                 # Setting fontsize
    sns.heatmap(df_cm, annot=data, fmt = '', annot_kws={"size": 16})    # Generating heatmap
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.show()

def report(x,y):
    '''
    Function for priniting the classification report
    Takes actual: x and predicted label: y as input
    computes accuracy, classification report, confusion matrix, true positive rate, and false posi
    '''
    acc = accuracy_score(x,y)
    cr = classification_report(x,y,zero_division=0)

    # TP: True Positive,    TN: True Negative
    # FP: False Positive,   FN: False Negative
    TP = TN = FP = FN = 0
    for i in range(len(x)):
        if x[i] == 1 and y[i] == 1: # If true label is 1 and predicted label is also 1
            TP += 1
        elif x[i] == 0 and y[i] == 0:   # If true label is 0 and predicted label is also 0
            TN += 1
        elif x[i] == 0 and y[i] == 1:   # If true label is 0 and predicted label is also 1
            FP += 1
        else:                           # If true label is 1 and predicted label is also 0
            FN += 1

    true_positive_rate = TP/(TP+FN)     # Computes true positive rate
    false_positive_rate = FP/(FP+TN)    # Computes false positive rate
    print("Accuracy: ", acc)
    print(f'\nTrue Positive Rate: {round(true_positive_rate,2)}')
    print(f'False Positive Rate: {round(false_positive_rate,2)}')
    print("\nClassification Report:\n", cr)

    cm = confusion_matrix(x,y,labels=[0, 1])
    plot_confusion_matrix(cm)

```

```python

def message(clf,data):
    '''
    Function for printing the message when DDos attack is detected
    '''
    # Loop through all data
    for i in range(len(data)):
        #   Predict data
        pred = clf.predict(data.iloc[i].values.reshape(1,-1))
        # If DDos detected
        if pred == 1:
            # Get all columns whose names start with Source_
            filter_col = [col for col in data if col.startswith('Source_')]

            # Get column name which has 1 as data
            h = data[filter_col].iloc[i].isin([1])

            # Find the ip address and remove Source_ from that
            ip = str(h[h].index.values[0]).replace('Source_','')

            # Display message
            print(f'DDOs attack detected!\t\tSample Number: {i}\tSource IP: {ip}')
```

```python
df = preprocessing(df)       # Preprocessing data
```

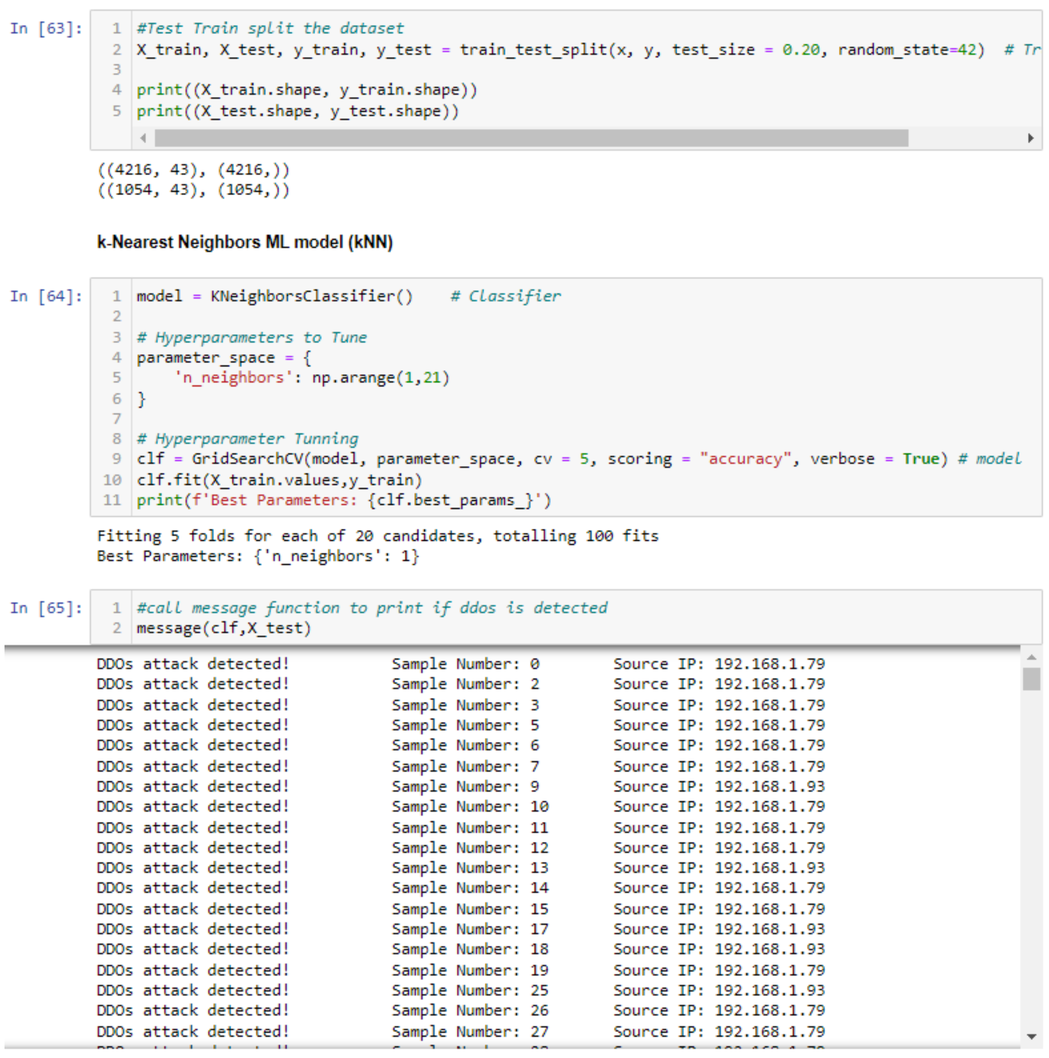
```python
pd.set_option('display.max_columns', None)  #shows all 44 columns
df.head()
```

```
     Length  TotalLength  SourcePort  DestinationPort  PacketClassification  Source_104.21.18.67  Source_142.250.180.3  Source_
0 -0.072297    -0.072182    0.493406        -0.810605                     0                    0                     0
1 -0.072297    -0.072182    0.493440        -0.810605                     0                    0                     0
2 -0.100784    -0.100668   -1.217222         0.924293                     0                    0                     0
3 -0.157757    -0.157639    0.493406        -0.810605                     0                    0                     0
4 -0.100784    -0.100668   -1.217222         0.924327                     0                    0                     0
```

```python
df.shape          # Shape after preprocessing
```

```
(5270, 44)
```

```python
x = df.drop(columns=['PacketClassification'])        # Features
y = df['PacketClassification'].values                # Labels
```



In [63]:
```python
#Test Train split the dataset
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, random_state=42)  # Tr

print((X_train.shape, y_train.shape))
print((X_test.shape, y_test.shape))
```

```
((4216, 43), (4216,))
((1054, 43), (1054,))
```

k-Nearest Neighbors ML model (kNN)

In [64]:
```python
model = KNeighborsClassifier()     # Classifier

# Hyperparameters to Tune
parameter_space = {
    'n_neighbors': np.arange(1,21)
}

# Hyperparameter Tunning
clf = GridSearchCV(model, parameter_space, cv = 5, scoring = "accuracy", verbose = True) # model
clf.fit(X_train.values,y_train)
print(f'Best Parameters: {clf.best_params_}')
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
Best Parameters: {'n_neighbors': 1}
```

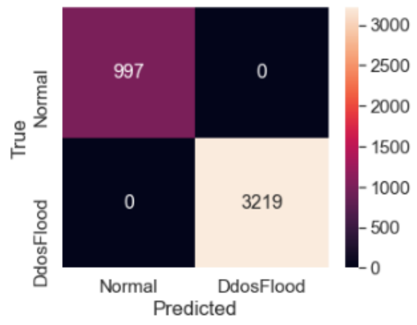
In [65]:
```python
#call message function to print if ddos is detected
message(clf,X_test)
```

```
DDOs attack detected!          Sample Number: 0       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 2       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 3       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 5       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 6       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 7       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 9       Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 10      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 11      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 12      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 13      Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 14      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 15      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 17      Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 18      Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 19      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 25      Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 26      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 27      Source IP: 192.168.1.79
```

```
In [66]:  #Make predictions useing the trained datasets
          train_pred = clf.predict(X_train.values)   # Train predict
          test_pred = clf.predict(X_test.values)     # Test predict
```

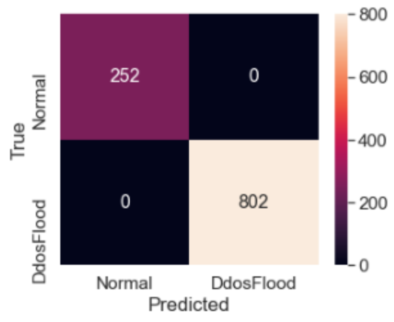
```
In [67]:  print("\t\tTrain output:\n")
          report(y_train,train_pred)
```

```
                Train output:

Accuracy:  1.0

True Positive Rate: 1.0
False Positive Rate: 0.0

Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00       997
           1       1.00      1.00      1.00      3219

    accuracy                           1.00      4216
   macro avg       1.00      1.00      1.00      4216
weighted avg       1.00      1.00      1.00      4216
```



```
In [68]:  print("\t\tTest output:\n")
          report(y_test,test_pred)
```

```
                Test output:

Accuracy:  1.0

True Positive Rate: 1.0
False Positive Rate: 0.0

Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00       252
           1       1.00      1.00      1.00       802

    accuracy                           1.00      1054
   macro avg       1.00      1.00      1.00      1054
weighted avg       1.00      1.00      1.00      1054
```



Logistic Regression ML model (LR)

```
In [69]:
#logistic regression model form the sklearn lib
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
clfLR = lr.fit(X_train, y_train)
clfLR.fit(X_train.values,y_train)
```

```
Out[69]: LogisticRegression()
```

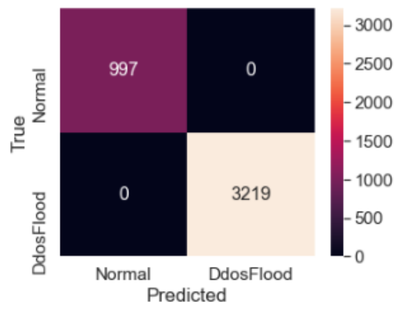
```
In [70]:
#call message function to print if ddos is detected
message(clfLR,X_test)
```

```
DDOs attack detected!          Sample Number: 0       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 2       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 3       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 5       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 6       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 7       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 9       Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 10      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 11      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 12      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 13      Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 14      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 15      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 17      Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 18      Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 19      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 25      Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 26      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 27      Source IP: 192.168.1.79
```

```
In [71]:
#Make predictions useing the trained datasets
train_pred_lr = clfLR.predict(X_train.values)   # Train predict
test_pred_lr = clfLR.predict(X_test.values)     # Test predict
```

```
In [72]:
print("\t\tTrain output for Linear regression model:\n")
report(y_train,train_pred_lr)
```

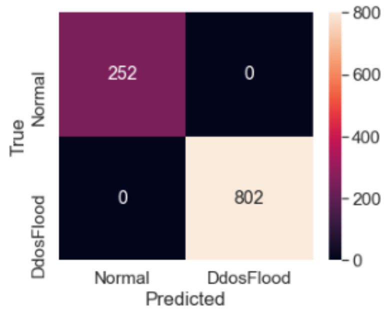
```
                Train output for Linear regression model:

Accuracy:  1.0

True Positive Rate: 1.0
False Positive Rate: 0.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       997
           1       1.00      1.00      1.00      3219

    accuracy                           1.00      4216
   macro avg       1.00      1.00      1.00      4216
weighted avg       1.00      1.00      1.00      4216
```



Predicted

In [73]:
```
print("\t\tTrain output for Linear regression model:\n")
report(y_test,test_pred_lr)
```

```
                Train output for Linear regression model:

Accuracy:  1.0

True Positive Rate: 1.0
False Positive Rate: 0.0

Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00       252
           1       1.00      1.00      1.00       802

    accuracy                           1.00      1054
   macro avg       1.00      1.00      1.00      1054
weighted avg       1.00      1.00      1.00      1054
```



Random Forest Classifier (RFC)

In [74]:
```
#import the RFC from the Sklearn
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
clfRFC = rfc.fit(X_train, y_train)
clfRFC.fit(X_train.values,y_train)
```

Out[74]:
```
RandomForestClassifier()
```

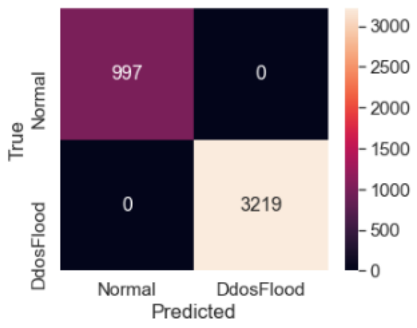
In [75]:
```
#call message function to print if ddos is detected
message(clfRFC,X_test)
```

```
DDOs attack detected!           Sample Number: 76       Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 78       Source IP: 192.168.1.93
DDOs attack detected!           Sample Number: 79       Source IP: 192.168.1.93
DDOs attack detected!           Sample Number: 83       Source IP: 192.168.1.93
DDOs attack detected!           Sample Number: 84       Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 85       Source IP: 192.168.1.93
DDOs attack detected!           Sample Number: 87       Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 88       Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 89       Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 90       Source IP: 192.168.1.93
DDOs attack detected!           Sample Number: 91       Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 92       Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 93       Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 94       Source IP: 192.168.1.93
DDOs attack detected!           Sample Number: 95       Source IP: 192.168.1.93
DDOs attack detected!           Sample Number: 96       Source IP: 192.168.1.93
DDOs attack detected!           Sample Number: 97       Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 98       Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 99       Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 100      Source IP: 192.168.1.79
```

In [76]:
```
#Make predictions useing the trained datasets
train_pred_rfc = clfRFC.predict(X_train.values)   # Train predict
test_pred_rfc = clfRFC.predict(X_test.values)     # Test predict
```

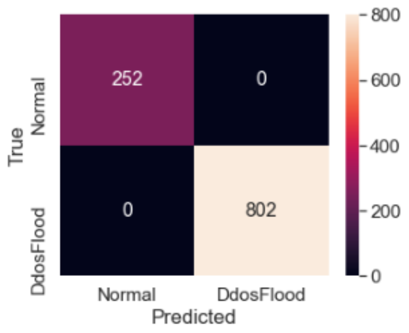
```
In [77]: print("\t\tTrain output for Random Forest Classifier model:\n")
         report(y_train,train_pred_rfc)
```

```
                Train output for Random Forest Classifier model:

Accuracy:  1.0

True Positive Rate: 1.0
False Positive Rate: 0.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       997
           1       1.00      1.00      1.00      3219

    accuracy                           1.00      4216
   macro avg       1.00      1.00      1.00      4216
weighted avg       1.00      1.00      1.00      4216
```



```
In [78]: print("\t\tTrain output for Random Forest Classifier model:\n")
         report(y_test,test_pred_rfc)
```

```
                Train output for Random Forest Classifier model:

Accuracy:  1.0

True Positive Rate: 1.0
False Positive Rate: 0.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       252
           1       1.00      1.00      1.00       802

    accuracy                           1.00      1054
   macro avg       1.00      1.00      1.00      1054
weighted avg       1.00      1.00      1.00      1054
```

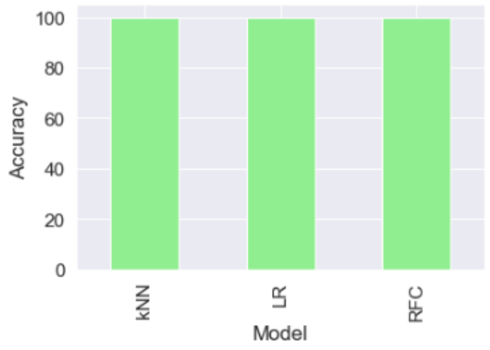


Accuracy Ploting

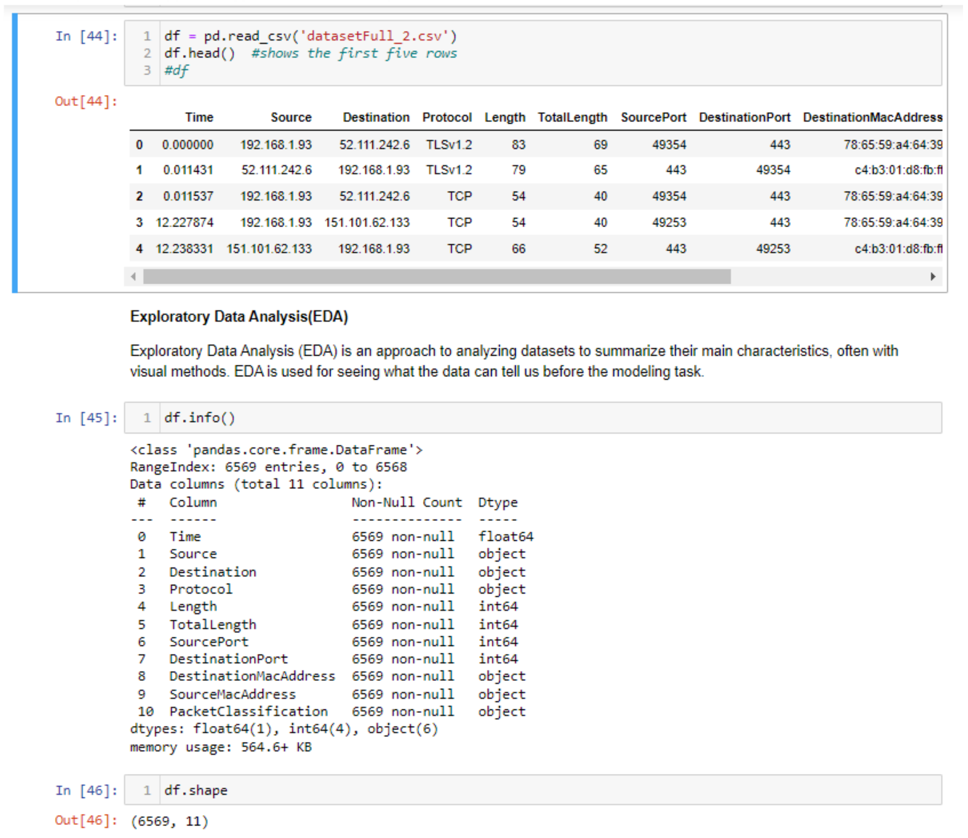
Accuracy Ploting

```
In [79]:
#Accuracies of the algorithm we implement above
accuracies=[['kNN',100],['LR',100],['RFC',100]]
scores=pd.DataFrame(accuracies,columns=['model','accuracy'])  #Stores the accuracy score of each m
```

```
In [80]:
#accuracy ploted
scores.set_index('model').accuracy.plot(kind='bar',color='lightGreen')
plt.xlabel('Model')
plt.ylabel('Accuracy')
plt.show()
```



A.2 ML_DetectionModel_datasetFull_2 Code



```
In [44]: df = pd.read_csv('datasetFull_2.csv')
         df.head()  #shows the first five rows
         #df
```

Out[44]:

	Time	Source	Destination	Protocol	Length	TotalLength	SourcePort	DestinationPort	DestinationMacAddress
0	0.000000	192.168.1.93	52.111.242.6	TLSv1.2	83	69	49354	443	78:65:59:a4:64:39
1	0.011431	52.111.242.6	192.168.1.93	TLSv1.2	79	65	443	49354	c4:b3:01:d8:fb:fl
2	0.011537	192.168.1.93	52.111.242.6	TCP	54	40	49354	443	78:65:59:a4:64:39
3	12.227874	192.168.1.93	151.101.62.133	TCP	54	40	49253	443	78:65:59:a4:64:39
4	12.238331	151.101.62.133	192.168.1.93	TCP	66	52	443	49253	c4:b3:01:d8:fb:fl

Exploratory Data Analysis(EDA)

Exploratory Data Analysis (EDA) is an approach to analyzing datasets to summarize their main characteristics, often with visual methods. EDA is used for seeing what the data can tell us before the modeling task.

```
In [45]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6569 entries, 0 to 6568
Data columns (total 11 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Time                   6569 non-null   float64
 1   Source                 6569 non-null   object
 2   Destination            6569 non-null   object
 3   Protocol               6569 non-null   object
 4   Length                 6569 non-null   int64
 5   TotalLength            6569 non-null   int64
 6   SourcePort             6569 non-null   int64
 7   DestinationPort        6569 non-null   int64
 8   DestinationMacAddress  6569 non-null   object
 9   SourceMacAddress       6569 non-null   object
 10  PacketClassification   6569 non-null   object
dtypes: float64(1), int64(4), object(6)
memory usage: 564.6+ KB
```

```
In [46]: df.shape
Out[46]: (6569, 11)
```

In [47]:
```
#Description of dataset
df.describe()
```

Out[47]:

	Time	Length	TotalLength	SourcePort	DestinationPort
count	6569.000000	6569.000000	6569.000000	6569.000000	6569.000000
mean	119.062792	237.026336	222.998021	33518.690364	24608.078094
std	57.358208	445.186000	445.197451	29362.736769	28390.503285
min	0.000000	54.000000	40.000000	80.000000	80.000000
25%	75.281392	54.000000	40.000000	443.000000	80.000000
50%	80.810659	66.000000	52.000000	49457.000000	443.000000
75%	192.577095	66.000000	52.000000	63221.000000	49472.000000
max	206.740153	1506.000000	1492.000000	63549.000000	63549.000000

Checking Missing value

We will now check for missing values in our dataset.We will use the isnull(), that tells us how many missing values we have in each column in our dataset. The output (Pandas Series) should look like this:

In [48]:
```
#checking for the null values
df.isnull().sum()/len(df)
```

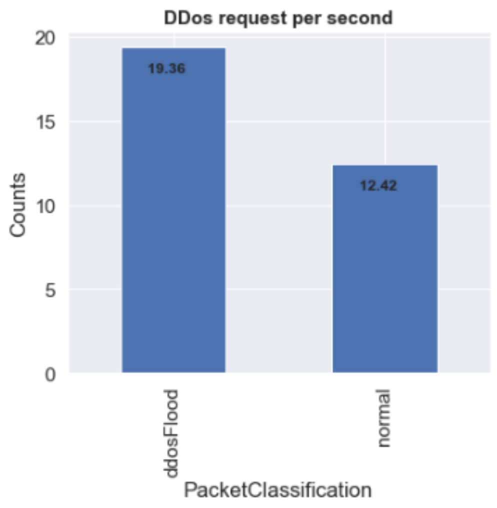
Out[48]:
```
Time                     0.0
Source                   0.0
Destination              0.0
Protocol                 0.0
Length                   0.0
TotalLength              0.0
SourcePort               0.0
DestinationPort          0.0
DestinationMacAddress    0.0
SourceMacAddress         0.0
PacketClassification     0.0
dtype: float64
```

Plots

1. Plot of how many ddos request per second vs normal traffic per second(Time column)

[49]:
```
#Plots the number of reqests made by ddos and nromal traffic per second
temp = np.round((df.groupby('PacketClassification').size()/df.iloc[-1,0]),2)      # Getting the dd
ax = temp.plot(kind='bar', stacked=True, figsize=[6, 5])
for index, data in enumerate(temp):
    ax.text(index-0.12, data-1.5, str(data), fontsize = 12, fontweight='bold')    # For putting te
ax.set_ylabel('Counts')
ax.set_title('DDos request per second', fontsize=15, fontweight='bold')
```

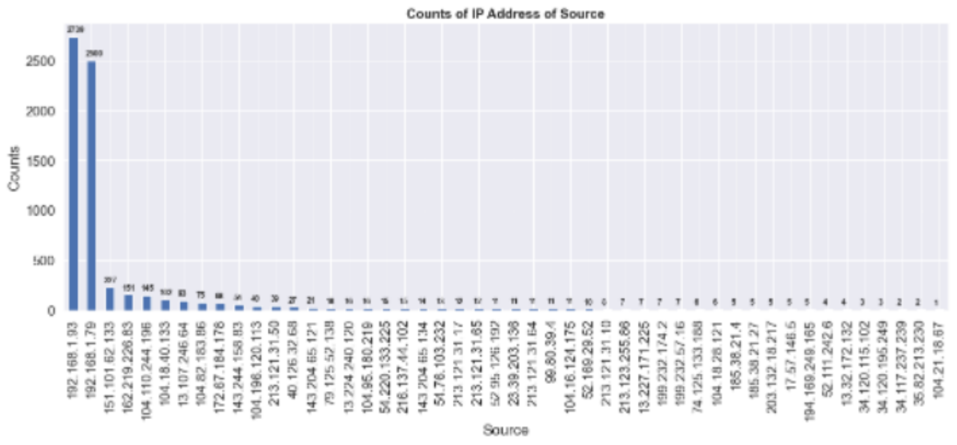
[49]:
```
Text(0.5, 1.0, 'DDos request per second')
```



2. Plot bar chart of ip address for source

In [50]:
```python
#get the count of how many times an IP-address appears in the source column
temp = df.groupby('Source').size().sort_values(ascending=False)    # Getting the counts of Source I
ax = temp.plot(kind='bar', stacked=True, figsize=[18, 6])
for index, data in enumerate(temp):
    ax.text(index-0.30, data+50, str(data), fontsize = 9, fontweight='bold')     # For text in the
ax.set_ylabel('Counts')
ax.set_title('Counts of IP Address of Source', fontsize=15, fontweight='bold')
```

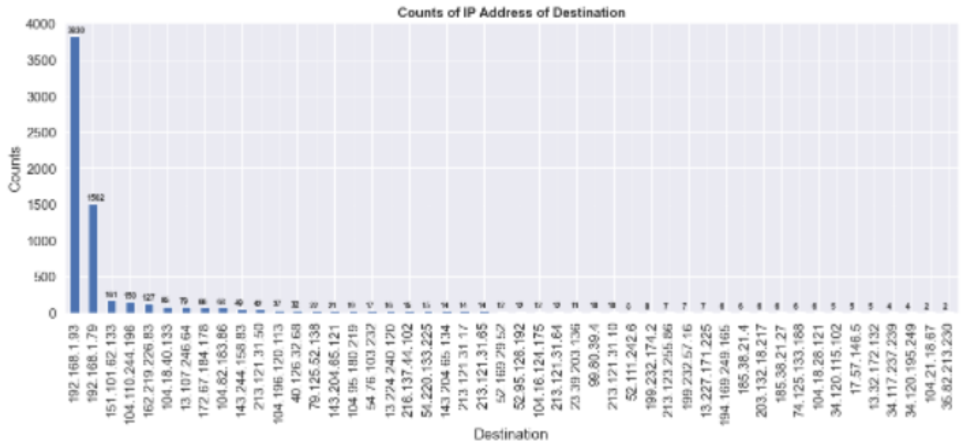
Out[50]:
```
Text(0.5, 1.0, 'Counts of IP Address of Source')
```



3. Plot bar chart of ip address for destination

[51]:
```python
temp = df.groupby('Destination').size().sort_values(ascending=False)    # Getting the counts of des
ax = temp.plot(kind='bar', stacked=True, figsize=[18, 6])
for index, data in enumerate(temp):
    ax.text(index-0.35, data+50, str(data), fontsize = 9, fontweight='bold')     # For text in the
ax.set_ylabel('Counts')
ax.set_title('Counts of IP Address of Destination', fontsize=15, fontweight='bold')
```

t[51]:
```
Text(0.5, 1.0, 'Counts of IP Address of Destination')
```



4. Plot of protocol used by ddos vs normal

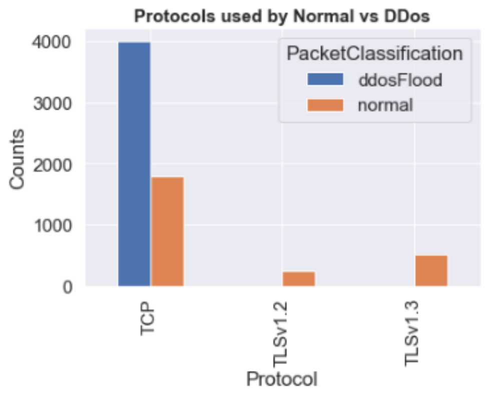
[52]:
```python
#Protocol breakdown based on AttackTyoe/PacketClassification
pd.crosstab(df["Protocol"],df["PacketClassification"])
```

t[52]:

PacketClassification	ddosFlood	normal
Protocol		
TCP	4002	1800
TLSv1.2	0	250
TLSv1.3	0	517

In [53]:
```python
# Generating the dataframe with Protocol counts Packet classification wise.
pd.crosstab(df["Protocol"].astype(str),df["PacketClassification"]).plot.bar()
plt.ylabel('Counts')
plt.title('Protocols used by Normal vs DDos', fontsize=15, fontweight='bold')
```

Out[53]:
```
Text(0.5, 1.0, 'Protocols used by Normal vs DDos')
```



5. Plot of source and destination ports used by ddos attack

In [54]:
```
#create new df Source which counts unique Source port numbers used by ddos request
Source = df[df['PacketClassification'] == 'ddosFlood'].groupby('SourcePort').size().sort_values(as
#print(Source)
Source['Others'] = Source.iloc[:, 1:].sum(axis=1)  #sums the value in all columns after 1st column
Source['Others']
s = Source[[80,'Others']]
print(s)
```

```
SourcePort     80  Others
0            1502    2500
```

In [55]:
```
#create new df Dest which counts unique destination port numbers used by ddos request
Dest = df[df['PacketClassification'] == 'ddosFlood'].groupby('DestinationPort').size().sort_values
Dest['Others'] = Dest.iloc[:, 1:].sum(axis=1)   #sums the value in all columns after 1st column(i.
d  = Dest[[80,'Others']]
print(d)
```

```
DestinationPort     80  Others
0                 2500    1502
```

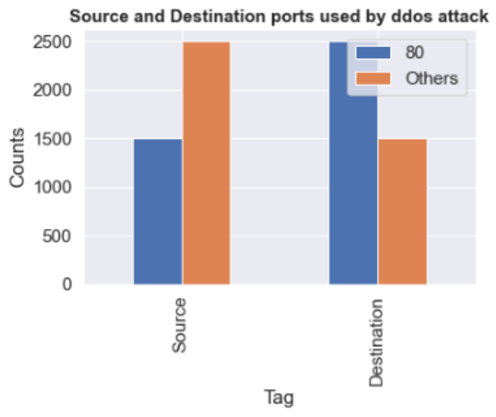
In [56]:
```
#merge the results from s and d into one table/dataframe
merged = s.append(d)
merged['Tag'] = ['Source','Destination']   #create row headers/titles
merged = merged.set_index('Tag')
merged
```

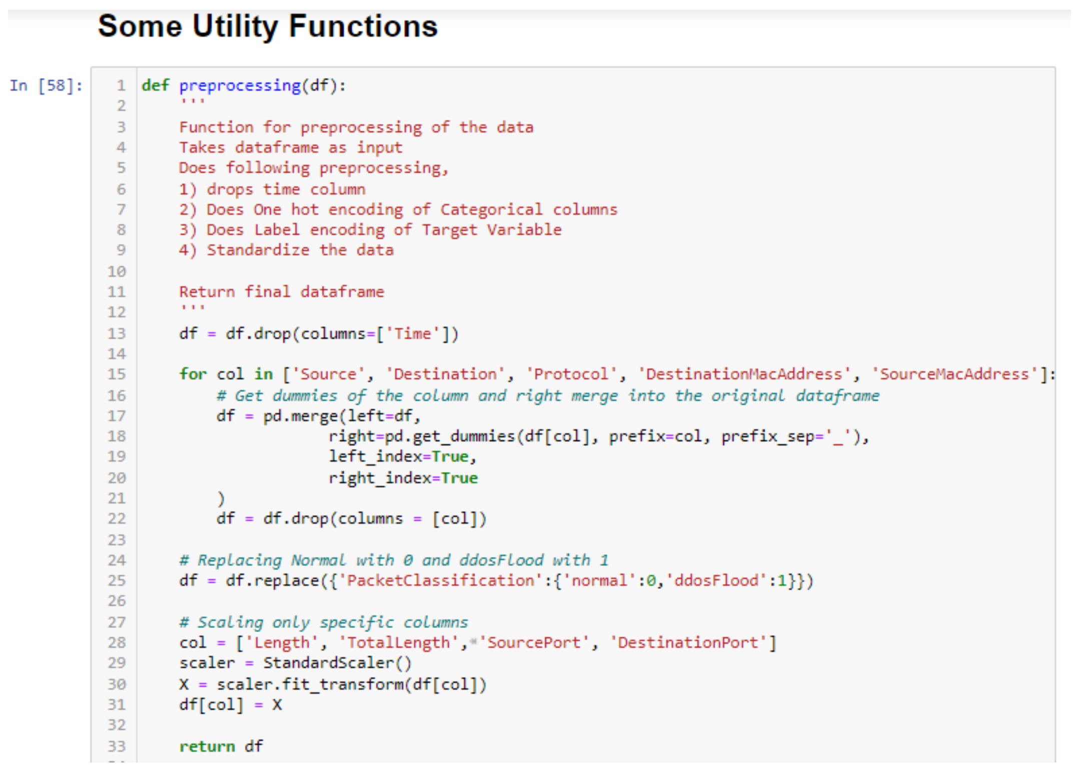
Out[56]:

	80	Others
Tag		
Source	1502	2500
Destination	2500	1502

In [57]:
```
#Plot
merged.plot.bar()
plt.ylabel('Counts')
plt.title('Source and Destination ports used by ddos attack', fontsize=15, fontweight='bold')
```

Out[57]: `Text(0.5, 1.0, 'Source and Destination ports used by ddos attack')`

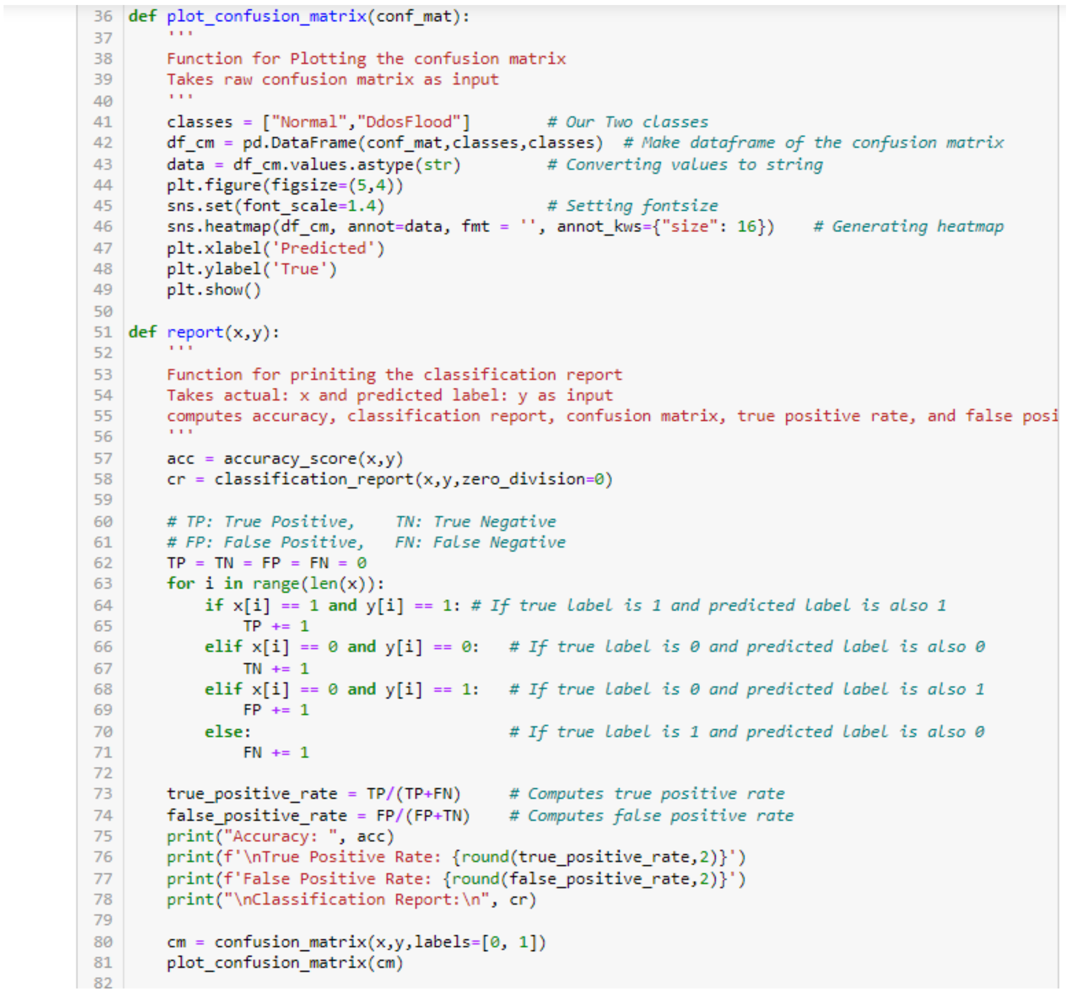




Some Utility Functions

In [58]:

```python
def preprocessing(df):
    '''
    Function for preprocessing of the data
    Takes dataframe as input
    Does following preprocessing,
    1) drops time column
    2) Does One hot encoding of Categorical columns
    3) Does Label encoding of Target Variable
    4) Standardize the data

    Return final dataframe
    '''
    df = df.drop(columns=['Time'])

    for col in ['Source', 'Destination', 'Protocol', 'DestinationMacAddress', 'SourceMacAddress']:
        # Get dummies of the column and right merge into the original dataframe
        df = pd.merge(left=df,
                      right=pd.get_dummies(df[col], prefix=col, prefix_sep='_'),
                      left_index=True,
                      right_index=True
        )
        df = df.drop(columns = [col])

    # Replacing Normal with 0 and ddosFlood with 1
    df = df.replace({'PacketClassification':{'normal':0,'ddosFlood':1}})

    # Scaling only specific columns
    col = ['Length', 'TotalLength','SourcePort', 'DestinationPort']
    scaler = StandardScaler()
    X = scaler.fit_transform(df[col])
    df[col] = X

    return df
```



```python
def plot_confusion_matrix(conf_mat):
    '''
    Function for Plotting the confusion matrix
    Takes raw confusion matrix as input
    '''
    classes = ["Normal","DdosFlood"]        # Our Two classes
    df_cm = pd.DataFrame(conf_mat,classes,classes)  # Make dataframe of the confusion matrix
    data = df_cm.values.astype(str)         # Converting values to string
    plt.figure(figsize=(5,4))
    sns.set(font_scale=1.4)                 # Setting fontsize
    sns.heatmap(df_cm, annot=data, fmt = '', annot_kws={"size": 16})    # Generating heatmap
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.show()

def report(x,y):
    '''
    Function for priniting the classification report
    Takes actual: x and predicted label: y as input
    computes accuracy, classification report, confusion matrix, true positive rate, and false posi
    '''
    acc = accuracy_score(x,y)
    cr = classification_report(x,y,zero_division=0)

    # TP: True Positive,    TN: True Negative
    # FP: False Positive,   FN: False Negative
    TP = TN = FP = FN = 0
    for i in range(len(x)):
        if x[i] == 1 and y[i] == 1: # If true label is 1 and predicted label is also 1
            TP += 1
        elif x[i] == 0 and y[i] == 0:   # If true label is 0 and predicted label is also 0
            TN += 1
        elif x[i] == 0 and y[i] == 1:   # If true label is 0 and predicted label is also 1
            FP += 1
        else:                           # If true label is 1 and predicted label is also 0
            FN += 1

    true_positive_rate = TP/(TP+FN)     # Computes true positive rate
    false_positive_rate = FP/(FP+TN)    # Computes false positive rate
    print("Accuracy: ", acc)
    print(f'\nTrue Positive Rate: {round(true_positive_rate,2)}')
    print(f'False Positive Rate: {round(false_positive_rate,2)}')
    print("\nClassification Report:\n", cr)

    cm = confusion_matrix(x,y,labels=[0, 1])
    plot_confusion_matrix(cm)
```

```python

def message(clf,data):
    '''
    Function for printing the message when DDos attack is detected
    '''
    # Loop through all data
    for i in range(len(data)):
        #   Predict data
        pred = clf.predict(data.iloc[i].values.reshape(1,-1))
        # If DDos detected
        if pred == 1:
            # Get all columns whose names start with Source_
            filter_col = [col for col in data if col.startswith('Source_')]

            # Get column name which has 1 as data
            h = data[filter_col].iloc[i].isin([1])

            # Find the ip address and remove Source_ from that
            ip = str(h[h].index.values[0]).replace('Source_','')

            # Display message
            print(f'DDOs attack detected!\t\tSample Number: {i}\tSource IP: {ip}')
```

]:
```python
df = preprocessing(df)       # Preprocessing data
```

]:
```python
pd.set_option('display.max_columns', None)  #shows all 110 columns
df.head()
```

]:

	Length	TotalLength	SourcePort	DestinationPort	PacketClassification	Source_104.110.244.196	Source_104.16.124.175	Sou
0	-0.346008	-0.345936	0.539341	-0.851232	0	0	0	
1	-0.354994	-0.354921	-1.126537	0.871693	0	0	0	
2	-0.411155	-0.411080	0.539341	-0.851232	0	0	0	
3	-0.411155	-0.411080	0.535901	-0.851232	0	0	0	
4	-0.384198	-0.384124	-1.126537	0.868135	0	0	0	

]:
```python
df.shape          # Shape after preprocessing
```

]:
```
(6569, 110)
```

]:
```python
x = df.drop(columns=['PacketClassification'])         # Features
y = df['PacketClassification'].values                 # Labels
```

In [63]:
```python
#Test Train split the dataset
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, random_state=42)  # Tr

print((X_train.shape, y_train.shape))
print((X_test.shape, y_test.shape))
```

```
((5255, 109), (5255,))
((1314, 109), (1314,))
```

k-Nearest Neighbors ML model (kNN)

In [64]:
```python
model = KNeighborsClassifier()    # Classifier

# Hyperparameters to Tune
parameter_space = {
    'n_neighbors': np.arange(1,21)
}

# Hyperparameter Tunning
clf = GridSearchCV(model, parameter_space, cv = 5, scoring = "accuracy", verbose = True) # model
clf.fit(X_train.values,y_train)
print(f'Best Parameters: {clf.best_params_}')
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
Best Parameters: {'n_neighbors': 1}
```

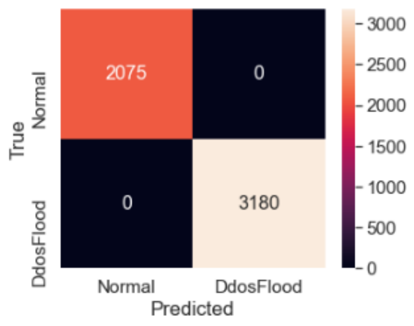
In [65]:
```python
#call message function to print if ddos is detected
message(clf,X_test)
```

```
DDOs attack detected!           Sample Number: 0        Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 1        Source IP: 192.168.1.93
DDOs attack detected!           Sample Number: 2        Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 3        Source IP: 192.168.1.93
DDOs attack detected!           Sample Number: 4        Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 5        Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 7        Source IP: 192.168.1.93
DDOs attack detected!           Sample Number: 8        Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 10       Source IP: 192.168.1.93
DDOs attack detected!           Sample Number: 11       Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 12       Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 13       Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 14       Source IP: 192.168.1.93
DDOs attack detected!           Sample Number: 15       Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 16       Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 19       Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 22       Source IP: 192.168.1.93
DDOs attack detected!           Sample Number: 23       Source IP: 192.168.1.93
DDOs attack detected!           Sample Number: 25       Source IP: 192.168.1.79
DDOs attack detected!           Sample Number: 26       Source IP: 192.168.1.93
```

```
In [66]:  #Make predictions useing the trained datasets
          train_pred = clf.predict(X_train.values)    # Train predict
          test_pred = clf.predict(X_test.values)      # Test predict
```

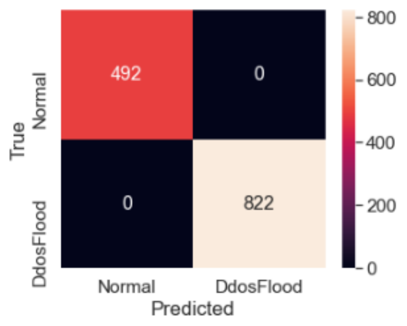
```
In [67]:  print("\t\tTrain output:\n")
          report(y_train,train_pred)
```

```
                Train output:

Accuracy:  1.0

True Positive Rate: 1.0
False Positive Rate: 0.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2075
           1       1.00      1.00      1.00      3180

    accuracy                           1.00      5255
   macro avg       1.00      1.00      1.00      5255
weighted avg       1.00      1.00      1.00      5255
```



```
In [68]:  print("\t\tTest output:\n")
          report(y_test,test_pred)
```

```
                Test output:

Accuracy:  1.0

True Positive Rate: 1.0
False Positive Rate: 0.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       492
           1       1.00      1.00      1.00       822

    accuracy                           1.00      1314
   macro avg       1.00      1.00      1.00      1314
weighted avg       1.00      1.00      1.00      1314
```



Logistic Regression ML model (LR)

```
In [69]:
#logistic regression model form the sklearn lib
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
clfLR = lr.fit(X_train, y_train)
clfLR.fit(X_train.values,y_train)
```

```
Out[69]: LogisticRegression()
```

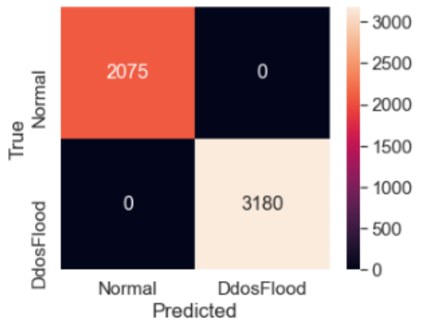
```
In [70]:
#call message function to print if ddos is detected
message(clfLR,X_test)
```

```
DDOs attack detected!          Sample Number: 0       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 1       Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 2       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 3       Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 4       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 5       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 7       Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 8       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 10      Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 11      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 12      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 13      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 14      Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 15      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 16      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 19      Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 22      Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 23      Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 25      Source IP: 192.168.1.79
```

```
In [71]:
#Make predictions useing the trained datasets
train_pred_lr = clfLR.predict(X_train.values)   # Train predict
test_pred_lr = clfLR.predict(X_test.values)     # Test predict
```

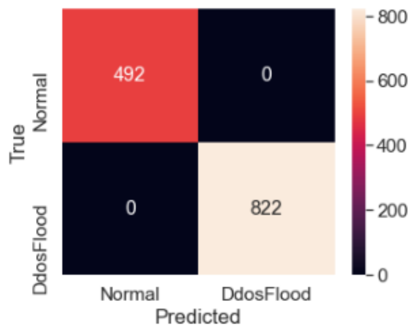
```
In [72]:
print("\t\tTrain output for Linear regression model:\n")
report(y_train,train_pred_lr)
```

```
                Train output for Linear regression model:

Accuracy:  1.0

True Positive Rate: 1.0
False Positive Rate: 0.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2075
           1       1.00      1.00      1.00      3180

    accuracy                           1.00      5255
   macro avg       1.00      1.00      1.00      5255
weighted avg       1.00      1.00      1.00      5255
```



```
In [73]:  print("\t\tTrain output for Linear regression model:\n")
          report(y_test,test_pred_lr)
```

```
                Train output for Linear regression model:

Accuracy:  1.0

True Positive Rate: 1.0
False Positive Rate: 0.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       492
           1       1.00      1.00      1.00       822

    accuracy                           1.00      1314
   macro avg       1.00      1.00      1.00      1314
weighted avg       1.00      1.00      1.00      1314
```



Random Forest Classifier (RFC)

```
In [74]:  #import the RFC from the Sklearn
          from sklearn.ensemble import RandomForestClassifier
          rfc = RandomForestClassifier()
          clfRFC = rfc.fit(X_train, y_train)
          clfRFC.fit(X_train.values,y_train)

Out[74]:  RandomForestClassifier()
```

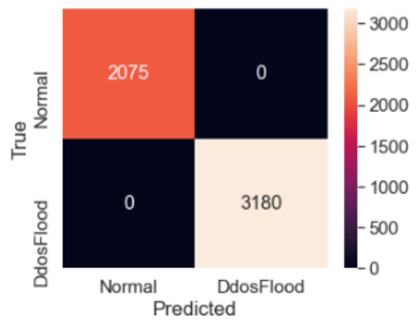
```
In [75]:  #call message function to print if ddos is detected
          message(clfRFC,X_test)
```

```
DDOs attack detected!          Sample Number: 0        Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 1        Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 2        Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 3        Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 4        Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 5        Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 7        Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 8        Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 10       Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 11       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 12       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 13       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 14       Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 15       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 16       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 19       Source IP: 192.168.1.79
DDOs attack detected!          Sample Number: 22       Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 23       Source IP: 192.168.1.93
DDOs attack detected!          Sample Number: 25       Source IP: 192.168.1.79
```

```
In [76]:  #Make predictions useing the trained datasets
          train_pred_rfc = clfRFC.predict(X_train.values)    # Train predict
          test_pred_rfc = clfRFC.predict(X_test.values)      # Test predict
```

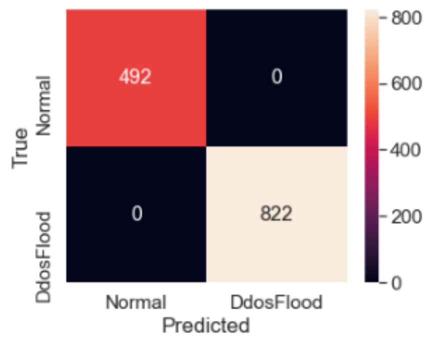
```
In [77]:
print("\t\tTrain output for Random Forest Classifier model:\n")
report(y_train,train_pred_rfc)
```

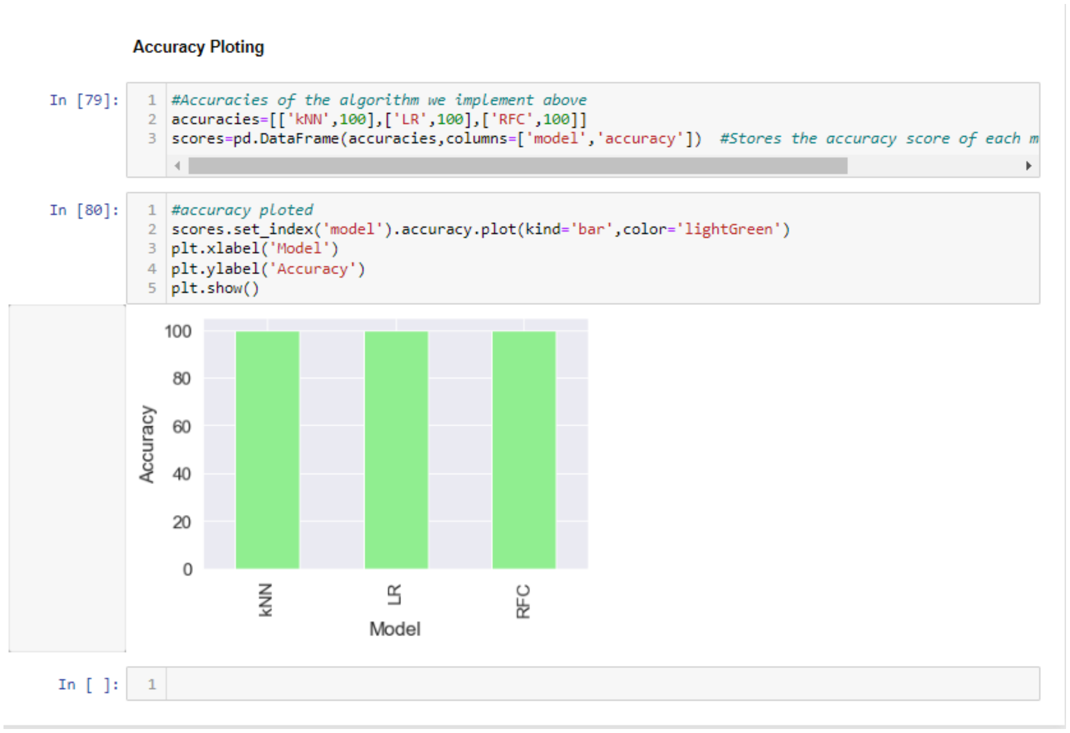
```
                Train output for Random Forest Classifier model:

Accuracy:  1.0

True Positive Rate: 1.0
False Positive Rate: 0.0

Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00      2075
           1       1.00      1.00      1.00      3180

    accuracy                           1.00      5255
   macro avg       1.00      1.00      1.00      5255
weighted avg       1.00      1.00      1.00      5255
```



```
[78]:
print("\t\tTrain output for Random Forest Classifier model:\n")
report(y_test,test_pred_rfc)
```

```
                Train output for Random Forest Classifier model:

Accuracy:  1.0

True Positive Rate: 1.0
False Positive Rate: 0.0

Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00       492
           1       1.00      1.00      1.00       822

    accuracy                           1.00      1314
   macro avg       1.00      1.00      1.00      1314
weighted avg       1.00      1.00      1.00      1314
```

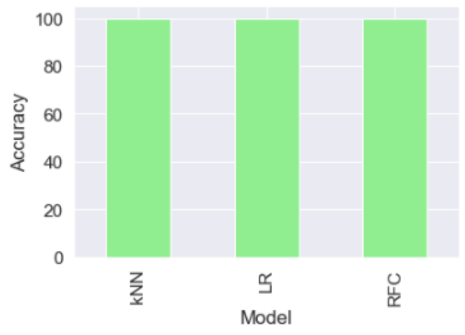




Accuracy Ploting

In [79]:
```
#Accuracies of the algorithm we implement above
accuracies=[['kNN',100],['LR',100],['RFC',100]]
scores=pd.DataFrame(accuracies,columns=['model','accuracy'])  #Stores the accuracy score of each m
```

In [80]:
```
#accuracy ploted
scores.set_index('model').accuracy.plot(kind='bar',color='lightGreen')
plt.xlabel('Model')
plt.ylabel('Accuracy')
plt.show()
```



In [ ]:
```
```