PROJECT REPORT
ON
Network Traffic Analyzer for Detecting DoS attack using Machine Learning

Carried Out at

CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING
ELECTRONIC CITY, BANGALORE.

UNDER THE SUPERVISION OF

[Mr. Muraleedharan N]

C-DAC Bangalore

Submitted By

Anshul Agarwal (190851923002)
Chandan Singh (190851923007)
Suggala Bhargavi (190851923031)
Sumesh S Lobo (190851923033)
Madala Akhil (190851920039)

PG DAC & DITISS IN C-DAC , BANGALORE

Candidate's Declaration

We hereby certify that the work being presented in the report entitled Network Traffic Analyzer for Detecting DoS attack using Machine Learning, in the partial fulfillment of the requirements for the award of PG Diploma Certificate and submitted in the department of PG-DAC & PG-DITISS of the C-DAC Bangalore, is an authentic record of our work carried out during the period, 2nd December 2019 - 30th January 2020 under the supervision of **Mr. Muraleedharan N**, C-DAC Bangalore.

The matter presented in the report has not been submitted by us for the award of any degree of this or any other Institute/University.

(Name and Signature of Candidate)

Anshul Agarwal	**(190851923002)**
Chandan Singh	**(190851923007)**
Suggala Bhargavi	**(190851923031)**
Sumesh S Lobo	**(190851923033)**
Madala Akhil	**(190851920039)**

Counter Signed by

Mr. Muraleedharan N

ACKNOWLEDGMENT

We take this opportunity to express my gratitude to all those people who have been directly and indirectly with me during the completion of this project.

We sincerely thank **Mr. Muraleedharan N** who has given guidance and a light to me during this major project. His versatile knowledge about "title name" has eased me in the critical times during the span of this Final Project.

We acknowledge here out debt to those who contributed significantly to one or more steps. We take full responsibility for any remaining sins of omission and commission.

Student Name	
Anshul Agarwal	(190851923002)
Chandan Singh	(190851923007)
Suggala Bhargavi	(190851923031)
Sumesh S Lobo	(190851923033)
Madala Akhil	(190851920039)

ABSTRACT

At any given time the traffic entering the network may be genuine or can be some malicious traffic. There may be times when a definite pattern is established when the malicious traffic is highly active (based on timelines). It becomes necessary to analyze all the incoming traffic into a particular network/server to identify the types of traffic entering so that proactive measures can be taken to safeguard our systems against harmful traffic. NETWORK TRAFFIC ANALYSER analysis the incoming packets whether it is normal traffic or some malicious packets in particular denial of service attack. Using machine learning the system is trained to detect DOS attack. The tools used to achieve this are Wireshark, Pandas, Matplotlib and Scikit learn. Using Wireshark the incoming packets are captured and sent as a CSV file to Pandas which perform data analysis to differentiate normal packets and DOS packets and then we feed the system with DOS packets by using Scikit Learn. Later, we visualize DOS packets using Matplotlib.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

The project "Network Traffic Analyzer for Detecting DoS Attack Using Machine Learning" analyses the incoming packets and determines whether it is normal traffic or some malicious packets in particular denial of service attack. Using machine learning the system is trained to classify the incoming traffic as normal traffic or DoS attack. The tools used to achieve this are wireshark, anaconda navigator, jupyter notebook, Pandas, Matplotlib and Scikit learn. Using Tshark the incoming packets are captured and sent as a CSV file to Pandas which perform data analysis to differentiate normal packets and Dos packets and then the visualization (Bar graphs) whether normal and DoS packets by using Matplotlib later, we implement ML to train the System so that it will detect and classifies the Dos packets.

DoS Attack:

A Denial-of-Service (DoS) attack is an attack meant to shut down a machine or network, making it inaccessible to its intended users. DoS attacks accomplish this by flooding the target with traffic, or sending it information that triggers a crash. In both instances, the DoS attack deprives legitimate users (i.e. employees, members, or account holders) of the service or resource they expected.

Victims of DoS attacks often target web servers of high-profile organizations such as banking, commerce, and media companies, or government and trade organizations. Though DoS attacks do not typically result in the theft or loss of significant information or other assets, they can cost the victim a great deal of time and money to handle.

There are two general methods of DoS attacks: flooding services or crashing services. Flood attacks occur when the system receives too much traffic for the server to buffer, causing them to slow down and eventually stop. Popular flood attacks include:

- **Buffer overflow attacks** – the most common DoS attack. The concept is to send more traffic to a network address than the programmers have built the system to handle. It includes the attacks listed below, in addition to others that are designed to exploit bugs specific to certain applications or networks
- **ICMP flood** – leverages misconfigured network devices by sending spoofed packets that ping every computer on the targeted network, instead of just one specific machine. The network is then triggered to amplify the traffic. This attack is also known as the smurf attack or ping of death.
- **SYN flood** – sends a request to connect to a server, but never completes the handshake. Continues until all open ports are saturated with requests and none are available for legitimate users to connect to.

Other DoS attacks simply exploit vulnerabilities that cause the target system or service to crash. In these attacks, input is sent that takes advantage of bugs in the target that subsequently crash or severely destabilize the system, so that it can't be accessed or used.

An additional type of DoS attack is the Distributed Denial of Service (DDoS) attack. A DDoS attack occurs when multiple systems orchestrate a synchronized DoS

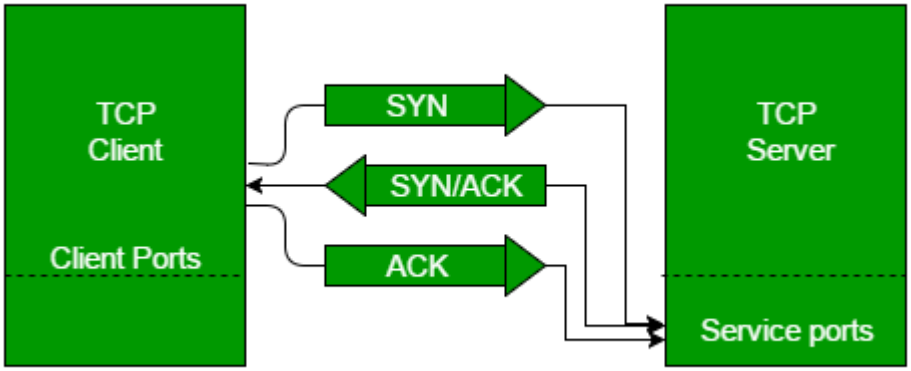
attack to a single target. The essential difference is that instead of being attacked from one location, the target is attacked from many locations at once. The distribution of hosts that defines a DDoS provide the attacker multiple advantages:

- Attacker can leverage the greater volume of machine to execute a seriously disruptive attack.
- The location of the attack is difficult to detect due to the random distribution of attacking systems (often worldwide)
- It is more difficult to shut down multiple machines than one.
- The true attacking party is very difficult to identify, as they are disguised behind many (mostly compromised) systems.
- Modern security technologies have developed mechanisms to defend against most forms of DoS attacks, but due to the unique characteristics of DDoS, it is still regarded as an elevated threat and is of higher concern to organizations that fear being targeted by such an attack.

TCP 3-Way Handshake Process

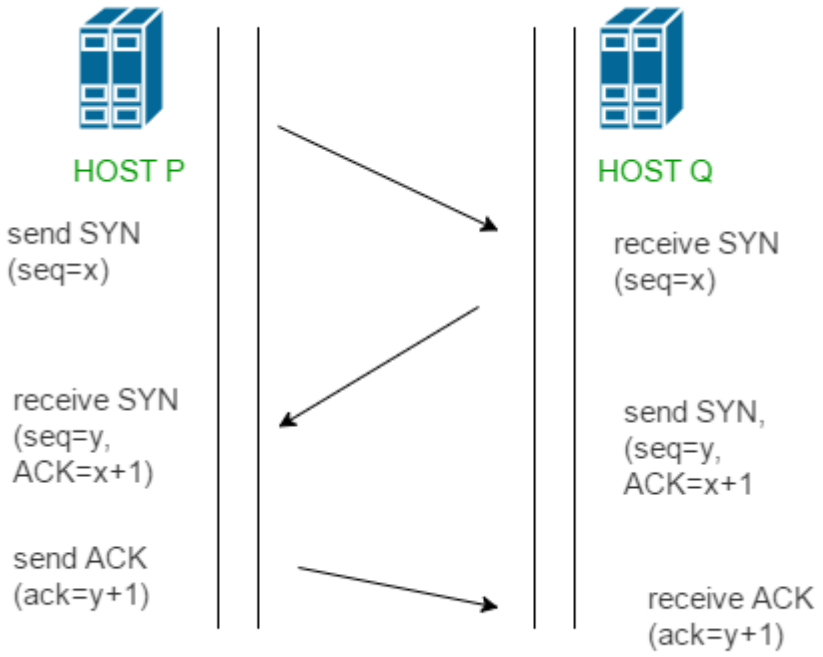
This could also be seen as a way of how TCP connection is established. TCP stands for Transmission Control Protocol which indicates that it does something to control the transmission of the data in a reliable way.

The process of communication between devices over the internet happens according to the current TCP/IP suite model (stripped out version of OSI reference model). The Application layer is a top pile of stack of TCP/IP model from where network referenced application like web browser on the client side establish connection with the server. From the application layer, the information is transferred to the transport layer. The two important protocols of this layer are – TCP, UDP (User Datagram Protocol) out of which TCP is prevalent (since it provides reliability for the connection established). However you can find application of UDP in querying the DNS server to get the binary equivalent of the Domain Name used for the website.



TCP provides reliable communication with something called Positive Acknowledgement with Re-transmission (PAR). The Protocol Data Unit (PDU) of the transport layer is called segment. Now a device using PAR resend the data unit until it receives an acknowledgement. If the data unit received at the receiver's end is damaged (It checks the data with checksum functionality of the transport layer that is used for Error Detection), then receiver discards the segment. So the sender has to resend the data unit for which positive acknowledgement is not received. You can realize from above mechanism that three

segments are exchanged between sender (client) and receiver (server) for a reliable TCP connection to get established. Let us delve how this mechanism works:



Step 1 (SYN) : In the first step, client wants to establish a connection with server, so it sends a segment with SYN(Synchronize Sequence Number) which informs server that client is likely to start communication and with what sequence number it starts segments with
Step 2 (SYN + ACK): Server responds to the client request with SYN-ACK signal bits set. Acknowledgement (ACK) signifies the response of segment it received and SYN signifies with what sequence number it is likely to start the segments with
Step 3 (ACK): In the final part client acknowledges the response of server and they both establish a reliable connection with which they will start the actual data transfer
The steps 1, 2 establish the connection parameter (sequence number) for one direction and it is acknowledged. The steps 2, 3 establish the connection parameter (sequence number) for the other direction and it is acknowledged. With these, a full-duplex communication is established.

Syn flooding:

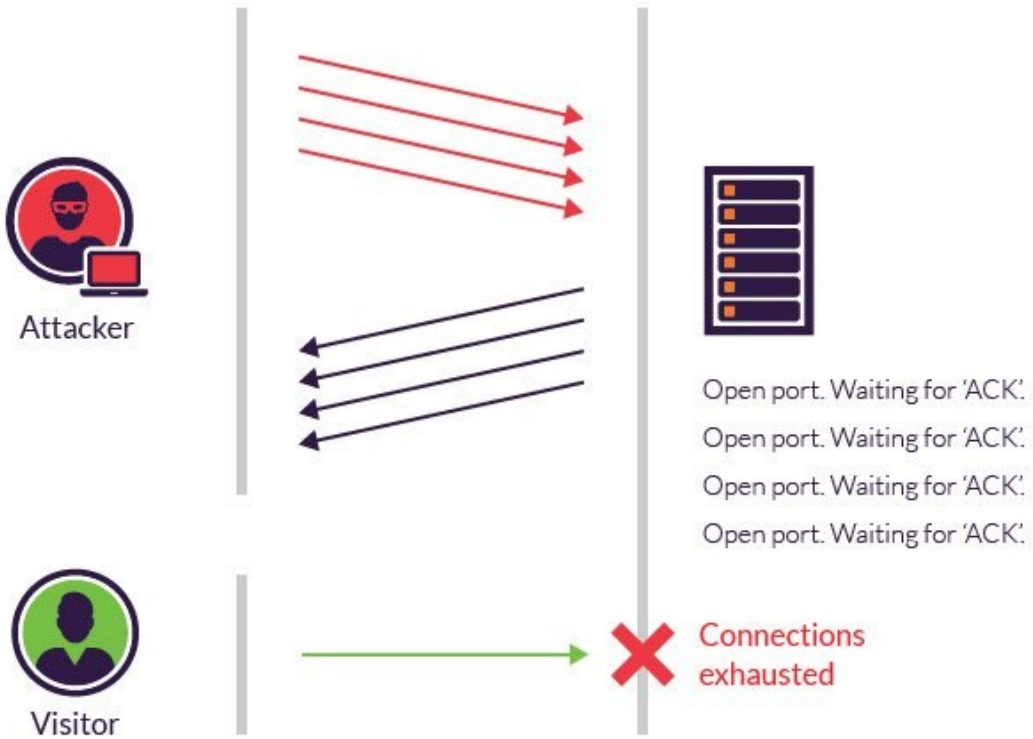
What is a SYN flood attack?
TCP SYN flood is a type of Distributed Denial of Service (DDoS) attack that exploits part of the normal TCP three-way handshake to consume resources on the targeted server and render it unresponsive.

Essentially, with SYN flood DDoS, the offender sends TCP connection requests faster than the targeted machine can process them, causing network saturation.

Attack description
When a client and server establish a normal TCP “three-way handshake,” the exchange looks like this:

Client requests connection by sending SYN (synchronize) message to the server.
Server acknowledges by sending SYN-ACK (synchronize-acknowledge) message back to the client.
Client responds with an ACK (acknowledge) message, and the connection is established.
In a SYN flood attack, the attacker sends repeated SYN packets to every port on the targeted server, often using a fake IP address. The server, unaware of the attack, receives multiple, apparently legitimate requests to establish communication. It responds to each attempt with a SYN-ACK packet from each open port.

The malicious client either does not send the expected ACK, or—if the IP address is spoofed—never receives the SYN-ACK in the first place. Either way, the server under attack will wait for acknowledgement of its SYN-ACK packet for some time.



During this time, the server cannot close down the connection by sending an RST packet, and the connection stays open. Before the connection can time out, another SYN packet will arrive. This leaves an increasingly large number of connections half-open – and indeed SYN flood attacks are also referred to as "half-open" attacks. Eventually, as the server's connection overflow tables fill, service to legitimate clients will be denied, and the server may even malfunction or crash.

While the "classic" SYN flood described above tries to exhaust network ports, SYN packets can also be used in DDoS attacks that try to clog your pipes with fake packets to achieve network saturation. The type of packet is not important. Still, SYN packets are often used because they are the least likely to be rejected by default.

SYN flooding is the process of sending half-open connections without completing the TCP handshake. These attacks are used to target individual access points, and most commonly firewalls. Firewalls do not treat these as actual connections as you are half-open connections, as a result, many half-open connections overwhelm the firewalls.

hping3:

Hping3 is a TCP/IP packet generator and analyzer. It is commonly used for generating packets. Because of its inherent functionality, many attackers utilize hping3 for denial of service attacks or for flooding.

Poc:

A simple DoS attack can be performed by using the following command:

hping3 -V -c 1000 -d 100 -S -p 21 –flood [IP ADDRESS]

What happens, is essentially a denial of service attack. The router will go down completely until you restart it!

- The -V is for verbose output.
- The -c command is essentially the number of packets you want to send to the particular target, In this case, 1000 packets.
- The -d command allows you to choose the size of a packet. For example, 100.
- To specify the type of packet, we need to add -S which is a SYN packet.
- After this, the -p command specifies the port, port 21 in this case, the FTP port.
- IP Address – Specify the IP address of the target.

Land Attacks:

LAND stands for, Local Area Network Denial Attack. It is essentially a denial of service attack where you send packets with the same source and destination IP to the same IP address. This is commonly referred to as IP spoofing.

Poc:

If my computer has an IP address, of 192.168.1.110, I would essentially send packets with the source and destination IP of 192.168.1.110 to my IP address.

hping3 -V -c 1000 -d 100 -S -p 21 -s 80 -k -a 192.168.1.110 192.168.1.110

- -v Is verbose output.
- -c Is to specify the number of packets.
- -d Is the size of the packets.
- -S is the SYN packets.
- -p Is the destination port.
- -s Is the source port. This only matters if you are doing it on an incognito mode. Set this to whatever you want.
- -k preserves the source port.
- -a Spoofs the source address.

ICMP Flooding:

In ICMP flooding the spoofed source address is used to send various or many ICMP packets to the entire network range, or to a specific network range and as a result, the devices on the network range will respond to these ICMP packets. The sheer amount of requests will

hping3 -1 –flood -a [IP OF TARGET] [NETWORK RANGE]

- The reason -1 is used, is because if you type in hping3 in terminal and press enter, you will see that we are trying to get away from the UDP/TCP, and go to the ICMP. So if we scroll up a bit, we can see that -1 corresponds with ICMP.

- We then add –flood.
- We want to spoof the source address, which is done using -a.
- You then add the IP address of the target (In my case, 192.168.1.103).

You then add the network range (In my case 192.186.1.1/24, a medium sized network

CHAPTER 2
LITERATURE SURVEY

A. Tan et. al. in (127) have studied the techniques for detecting DoS attacks to network services and have proposed an effective system for DoS attack detection occurring on Internet Cloud. Their methodology for detecting known as well as unknown DoS attacks was based on learning the patterns of legitimate network traffic and apply the idea of Multivariate Correlation Analysis (MCA) to network traffic characterization and attack recognition. A triangle area technique has been used to speed up the process of MCA. The influence of both non-normalized and normalized data on the performance of the detection system has also been examined. The algorithm has been validated using KDD'99 dataset. The authors claim to outperform the two state-of-the art approaches.

B. Noureldien A. Noureldien*, Izzedin M. Yousif Attaining high prediction accuracy in detecting anomalies in network traffic is a major goal in designing machine learning algorithms and in building Intrusion Detection Systems. One of the major network attack classes is Denial of Service (DoS) attack class that contains various types of attacks such as Smurf, Teardrop, Land, Back and Neptune. This paper examines the detection accuracy of a set of selected machine learning algorithms in detecting different DoS attack class types. The algorithms are belonging to different supervised techniques, namely, PART, Bayes Net, IBK, Logistic, J48, Random Committee and Input Mapped. The experimental work is carried out using NSL-KDD dataset and WEKA as a data mining tool. The results show that the best algorithm in detecting the Smurf attack is the Random Committee with an accuracy of 98.6161%, and the best algorithm in detecting the Neptune attack is the PART algorithm with an accuracy of 98.5539%, and on the average PART algorithm is the best algorithm in detecting DoS attacks while Input Mapped algorithm is the worst.

C. Project Neptune by daemon9 / route / infinity for Phrack Magazine TCP SYN flooding is a denial of service (DOS) attack.

Like most DOS attacks, it does not exploit a software bug, but rather a shortcoming in the implementation of a particular protocol. For example, mail bombing DOS attacks work because most SMTP agents are dumb and will accept whatever is sent their way. ICMP_ECHO floods exploit the fact that most kernels will simply reply to ICMP_ECHO request packets one after another, ad inifintum. We will see that TCP SYN flood DOS attacks work because of the current implementation of TCP's connection establishment protocol.

CHAPTER 3
Tools and libraries used in the project

Tools and libraries used in project:

- **WIRESHARK:**

- Wireshark is a free and open-source **packet analyzer**. It is used for network troubleshooting, analysis, software, and communications protocol development.
- It lets you see what's happening on your network at a microscopic level and is the de facto (and often de jure) standard across many commercial and non-profit enterprises, government agencies, and educational institutions.
- Wireshark is a cross-platform tool that runs on **Linux**, Microsoft Windows, macOS, BSD, Solaris, and other Unix-like operating systems.

Wireshark and covers the following areas:

- Web traffic and the default Wireshark column display
- Hiding columns
- Removing columns
- Adding columns
- Changing time to UTC
- Custom columns

Web Traffic and the Default Wireshark Column Display

Malware distribution frequently occurs through web traffic, and we also see this channel used for data exfiltration and command and control activity.

Wireshark's default column is not ideal when investigating such malware-based infection traffic. However, Wireshark can be customized to provide a better view of the activity.

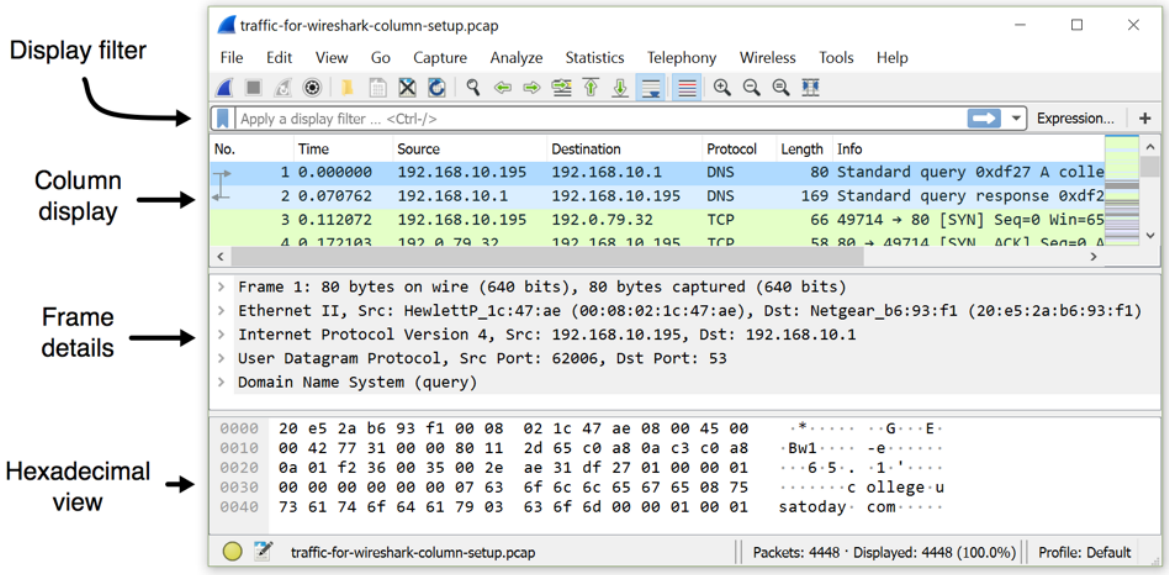


Fig 3.1: Viewing a pcap using Wireshark's default column display.

Wireshark's default columns are:

- **No**. -Frame number from the beginning of the pcap. The first frame is always 1.

- **Time** – Seconds broken down to the nanosecond from the first frame of the pcap. The first frame is always 0.000000.
- **Source** – Source address, commonly an IPv4, IPv6, or Ethernet address.
- **Destination** – Destination address, commonly an IPv4, IPv6, or Ethernet address.
- **Protocol** – Protocol used in the Ethernet frame, IP packet, or TCP segment (ARP, DNS, TCP, HTTP, etc.).
- **Length** – Length of the frame in bytes.

In our day-to-day work, we require the following columns in our Wireshark display:

- Date & time in UTC
- Source IP and source port
- Destination IP and destination port
- HTTP host
- HTTPS server
- Info

How can we reach this state?
First, we hide or remove the columns we do not want.

Hiding Columns We can easily hide columns in case we need them later. Right-click on any of the column headers to bring up the column header menu. Then left-click any of the listed columns to uncheck them. Figure 2 shows the **No.**, **Protocol**, and **Length** columns unchecked and hidden.

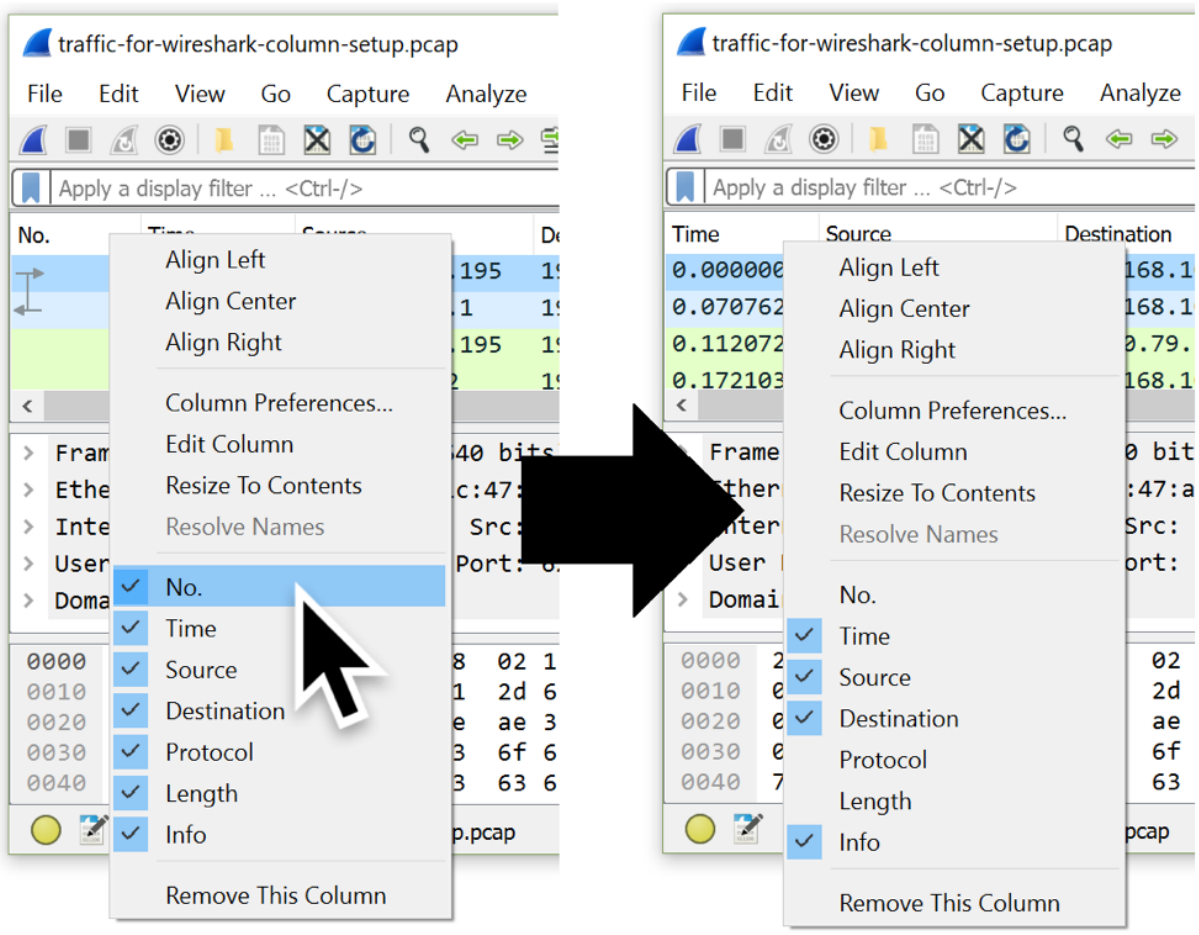


Figure 3.2: Before and after shots of the column header menu when hiding columns.

Removing columns, Because we never use the **No.**, **Protocol**, or **Length** columns, we completely remove them. To remove columns, right-click on the column headers you want to remove. Then select "Remove this Column…" from the column header menu.

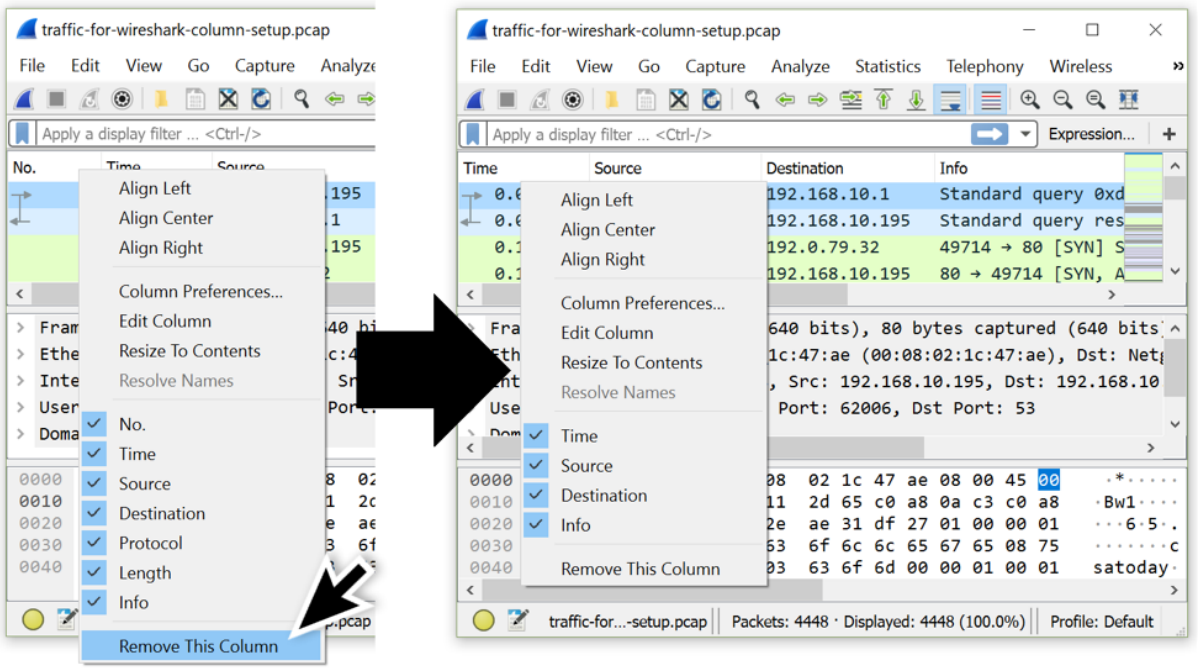


Fig 3.3: Before and after shots of the column header menu when removing columns.

At this point, whether hidden or removed, the only visible columns are Time, Source, Destination, Info. Adding Columns. To add columns in Wireshark, use the Column Preferences menu. Right-click on any of the column headers, then select "Column Preferences…"

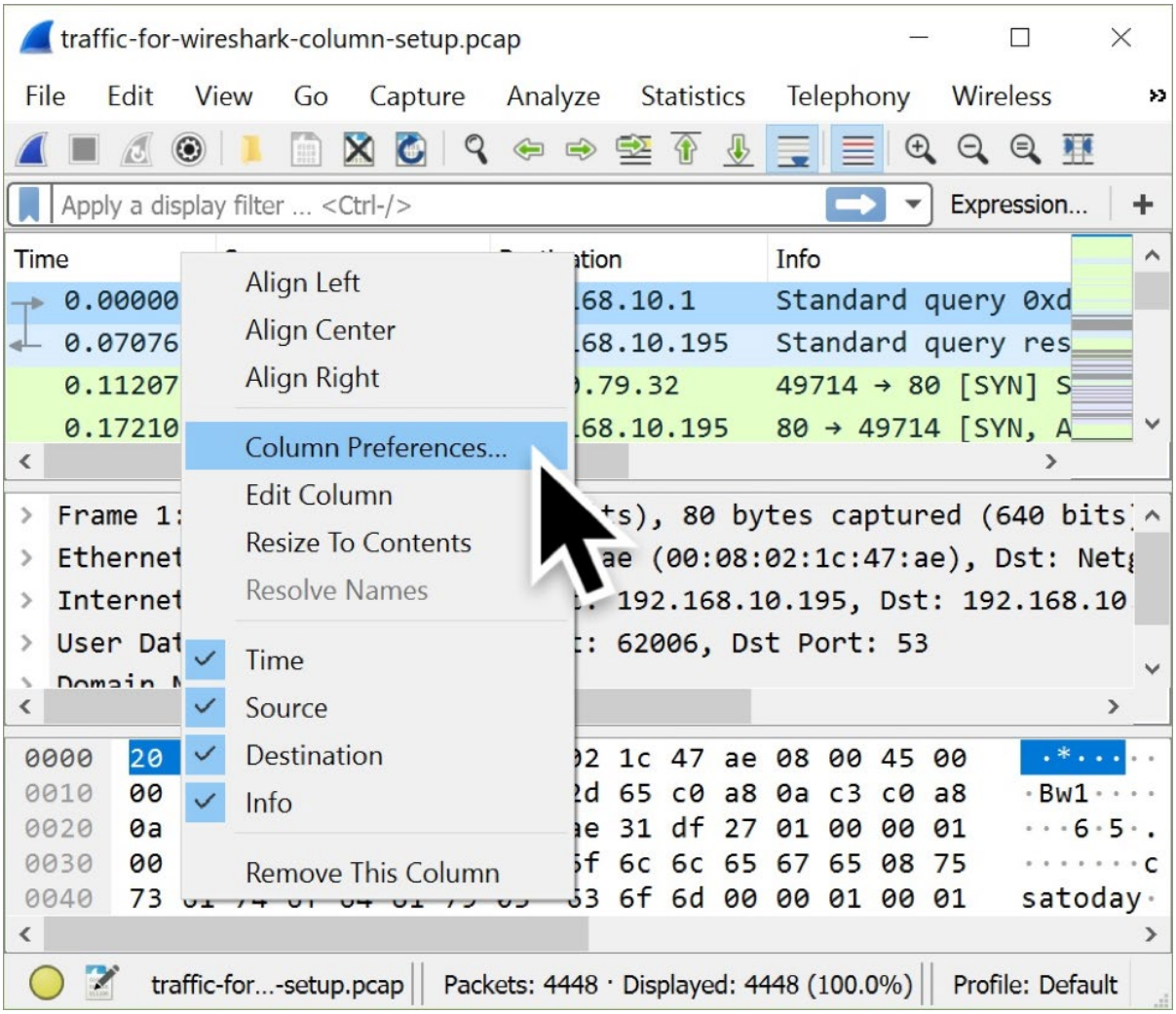


Fig 3.4: Getting to the Column Preferences menu by right-clicking on the column headers.

The Column Preferences menu lists all columns, viewed or hidden. Near the bottom left side of the Column Preferences menu are two buttons. One has a plus sign to add columns. The other has a minus sign to remove columns.

Left-click on the plus sign. An entry titled “New Column” should appear at the bottom of the column list.

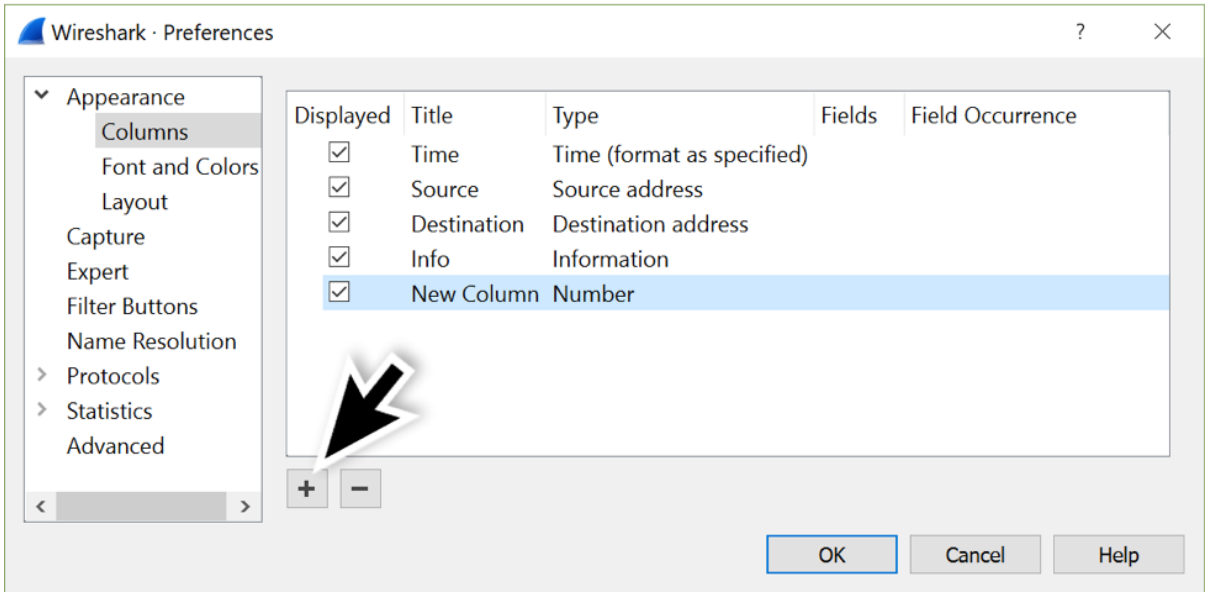


Figure 3.5: Adding a new column in the Column Preferences menu.

Double-click on the "New Column" and rename it as "Source Port."

The column type for any new columns always shows "Number." Double-click on "Number" to bring up a menu, then scroll to "Src port (unresolved)" and select that for the column type.

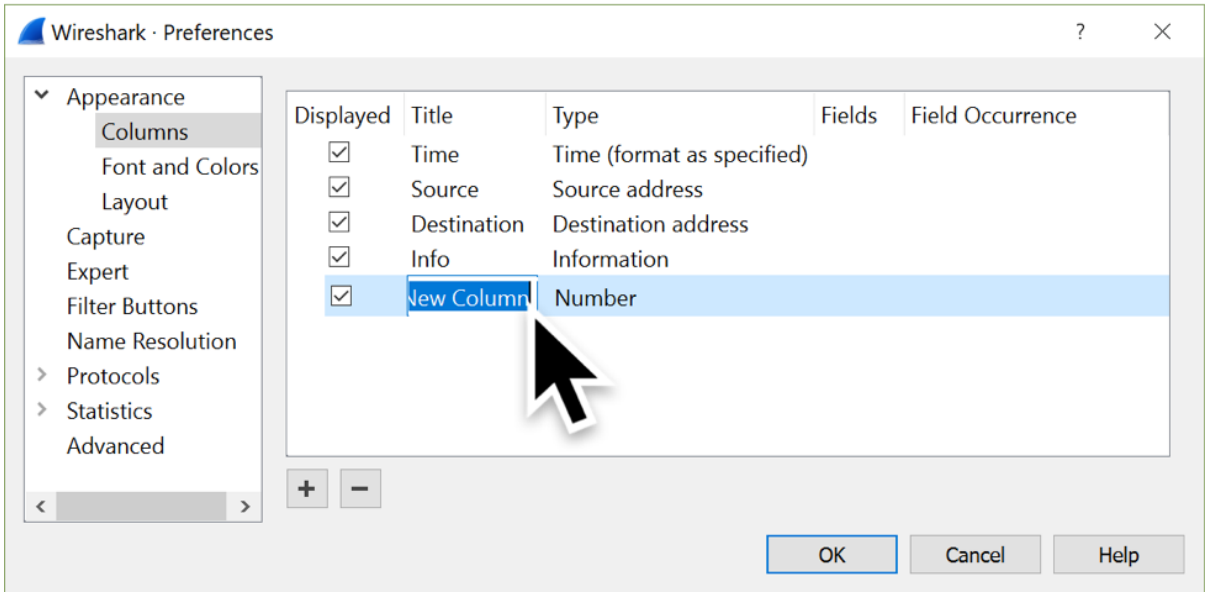


Figure 3.6: Changing the column title.

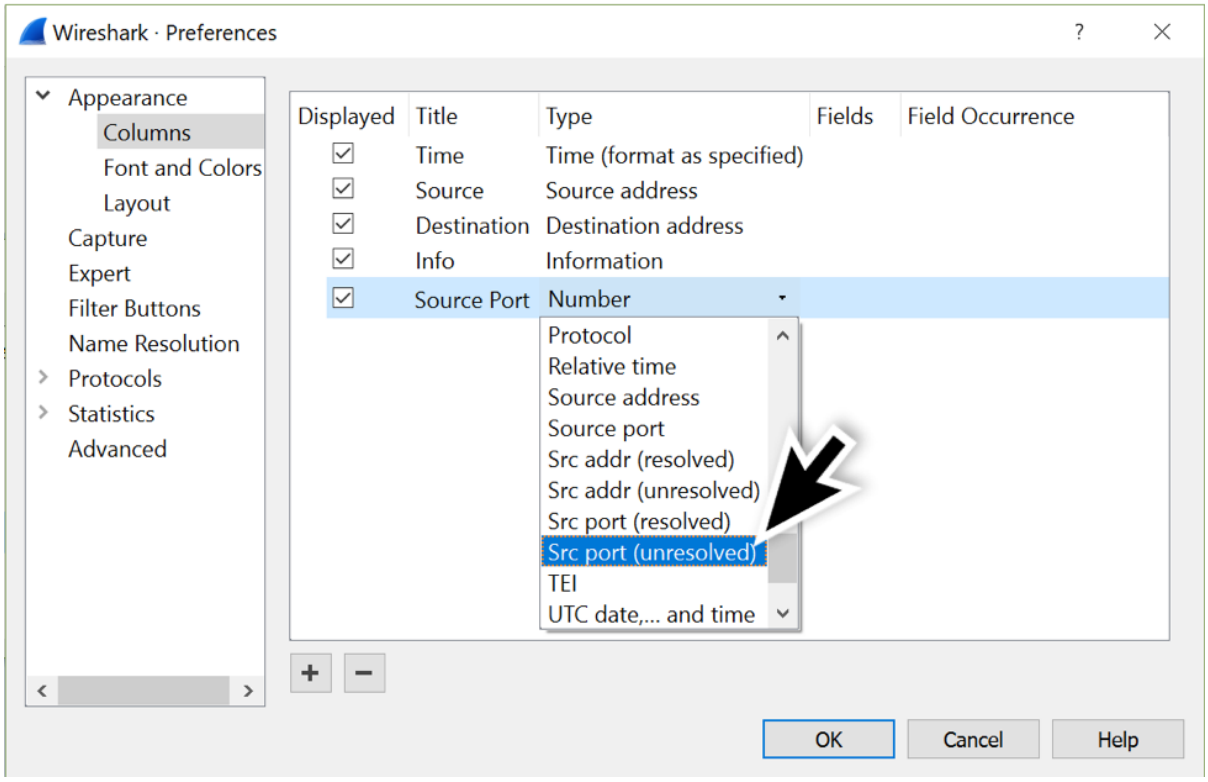


Figure 3.7: Changing the column type.

Our new column is now named “Source Port” with a column type of “Src port (unresolved).” Left-click on that entry and drag it to a position immediately after the source address.

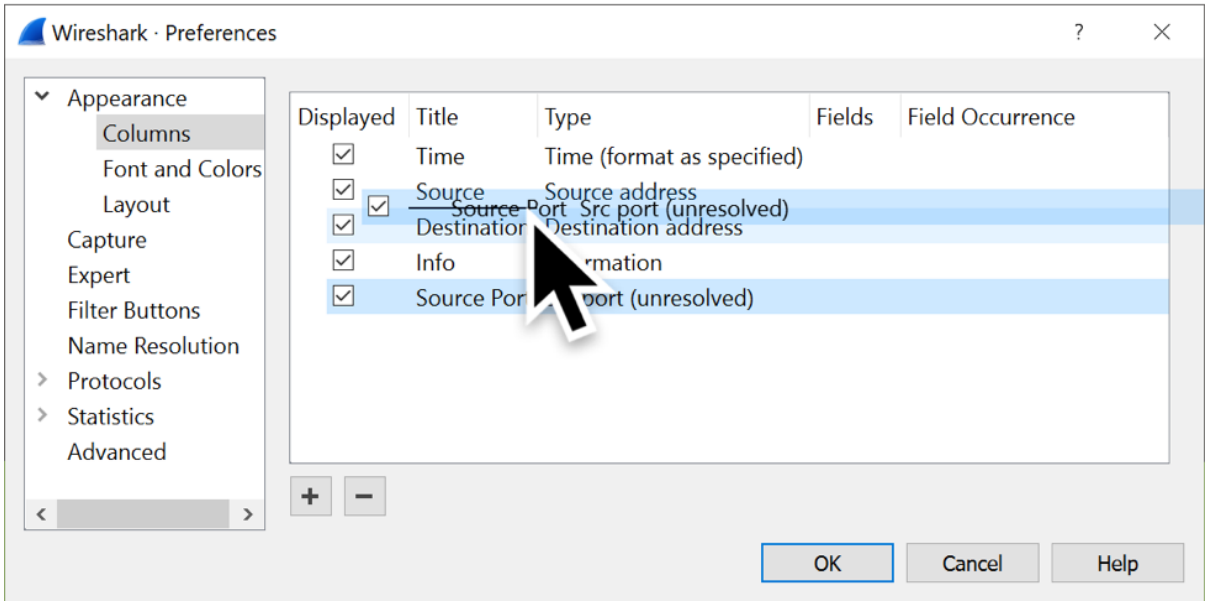


Figure 3.8: Changing the column position.

After the source port has been, add another column titled “Destination Port” with the column type “Dest

port (unresolved)."

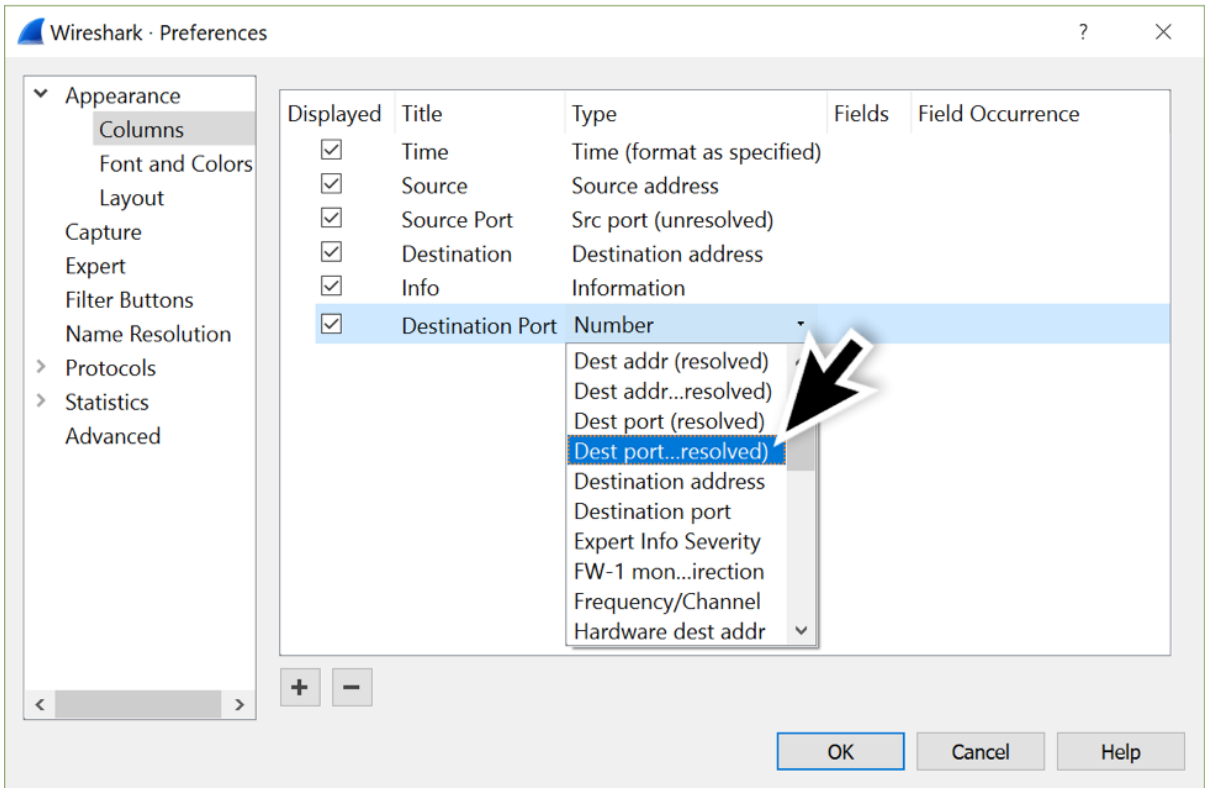


Figure 3.9: Adding another column for Destination Port.

Like we did with the source port column, drag the destination port to place it immediately after the Destination address. When you finish, your columns should appear as shown in Figure 10.

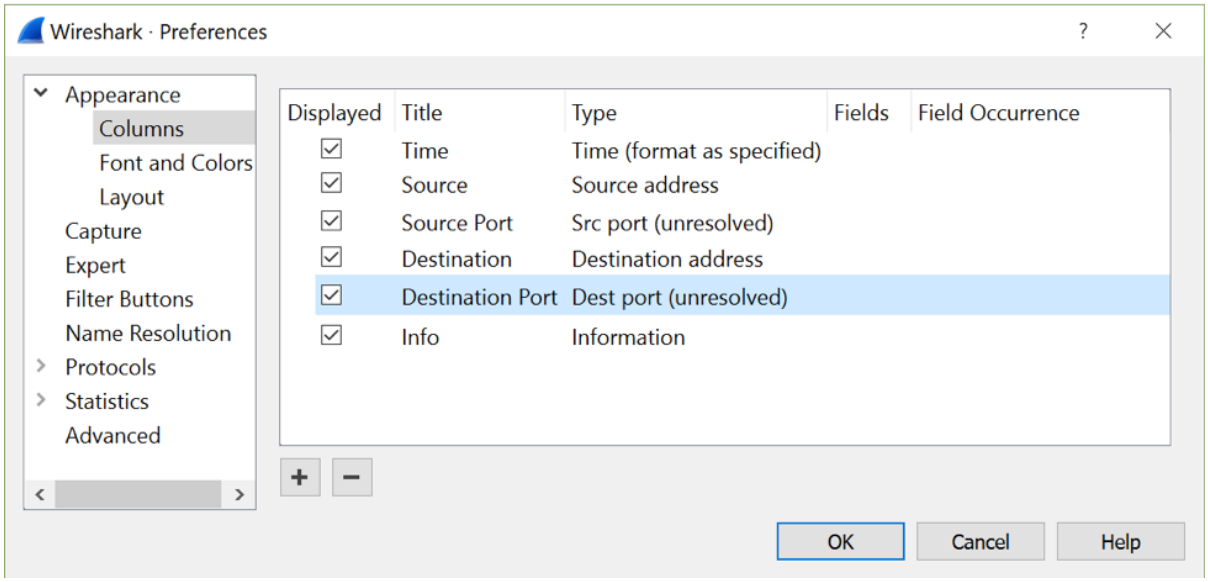


Figure 3.10: Final setup in the Column Preferences window.

After adding the source and destination port columns, click the “OK” button to apply the changes.

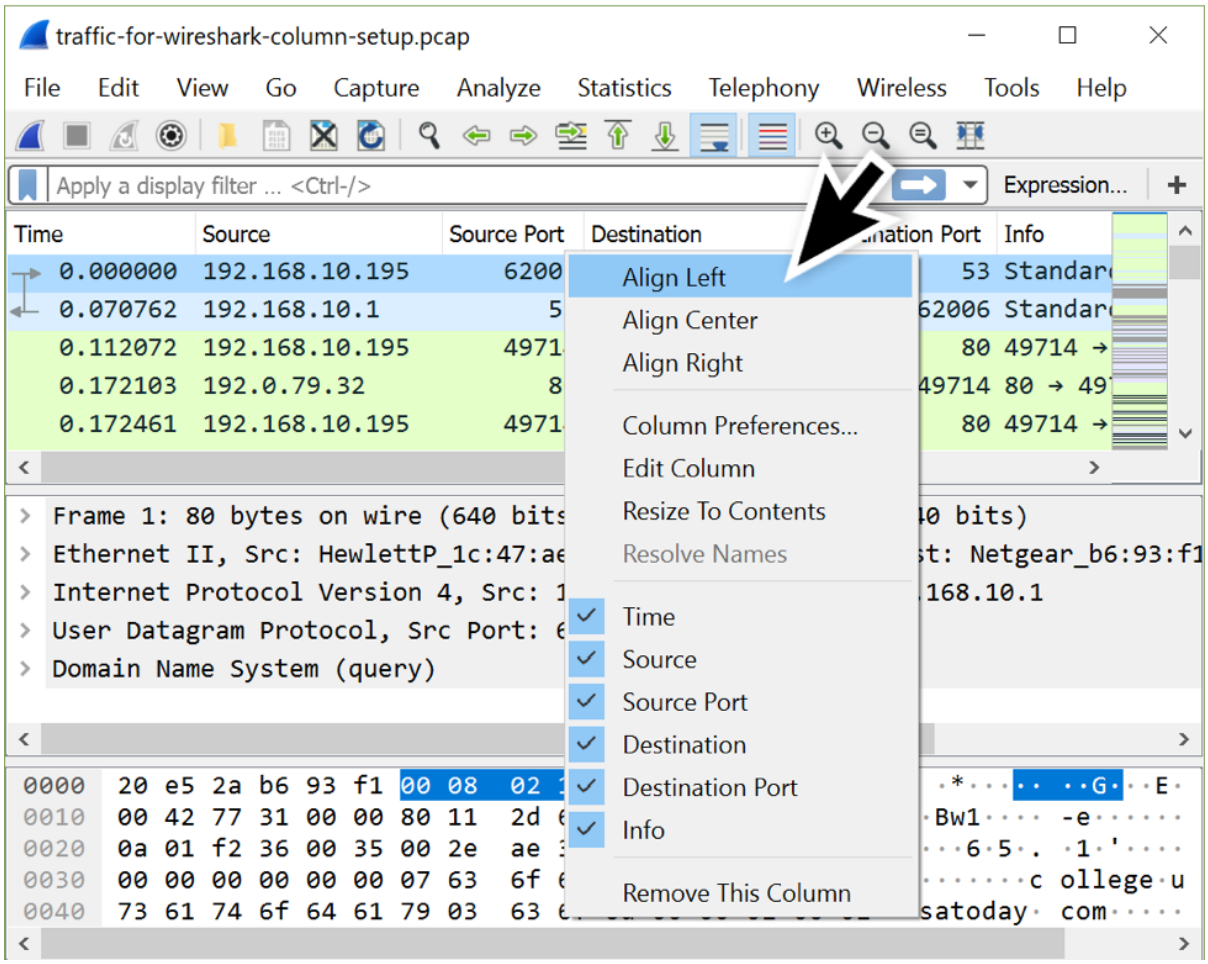


Figure 3.11: Aligning column displays in Wireshark.

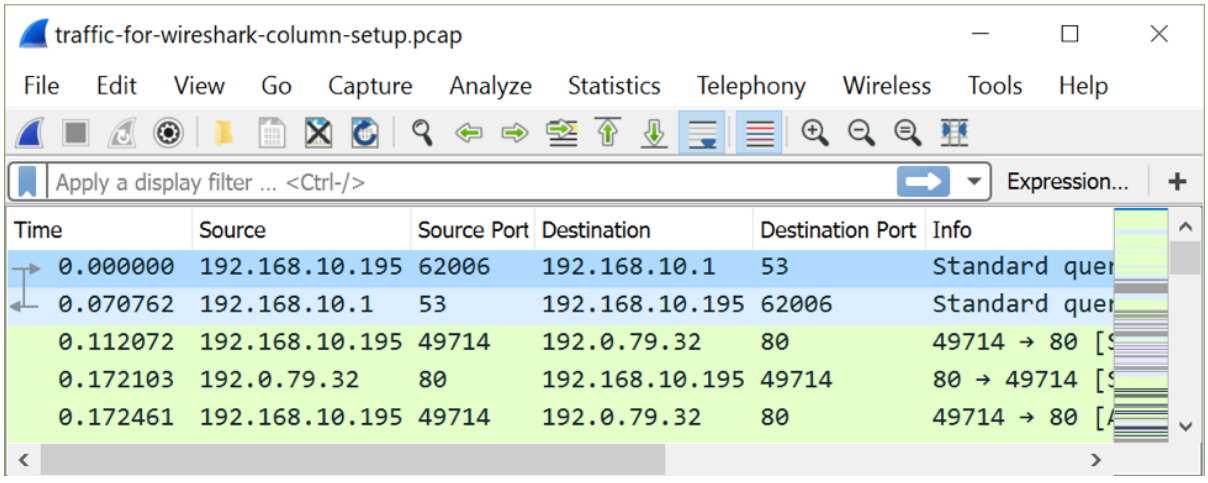


Figure 3.12: Column display after adding and aligning the source and destination ports.

In my day-to-day work, I often hide the source address and source port columns until I need them.

Changing Time to UTC.

To change the time display format, go the “View” menu, maneuver to “Time Display Format,” and change the value from “Seconds Since Beginning of Capture” to “UTC Date and Time of Day.” Use the same menu path to change the resolution from “Automatic” to “Seconds.” Figure 13 shows the menu paths for these options.

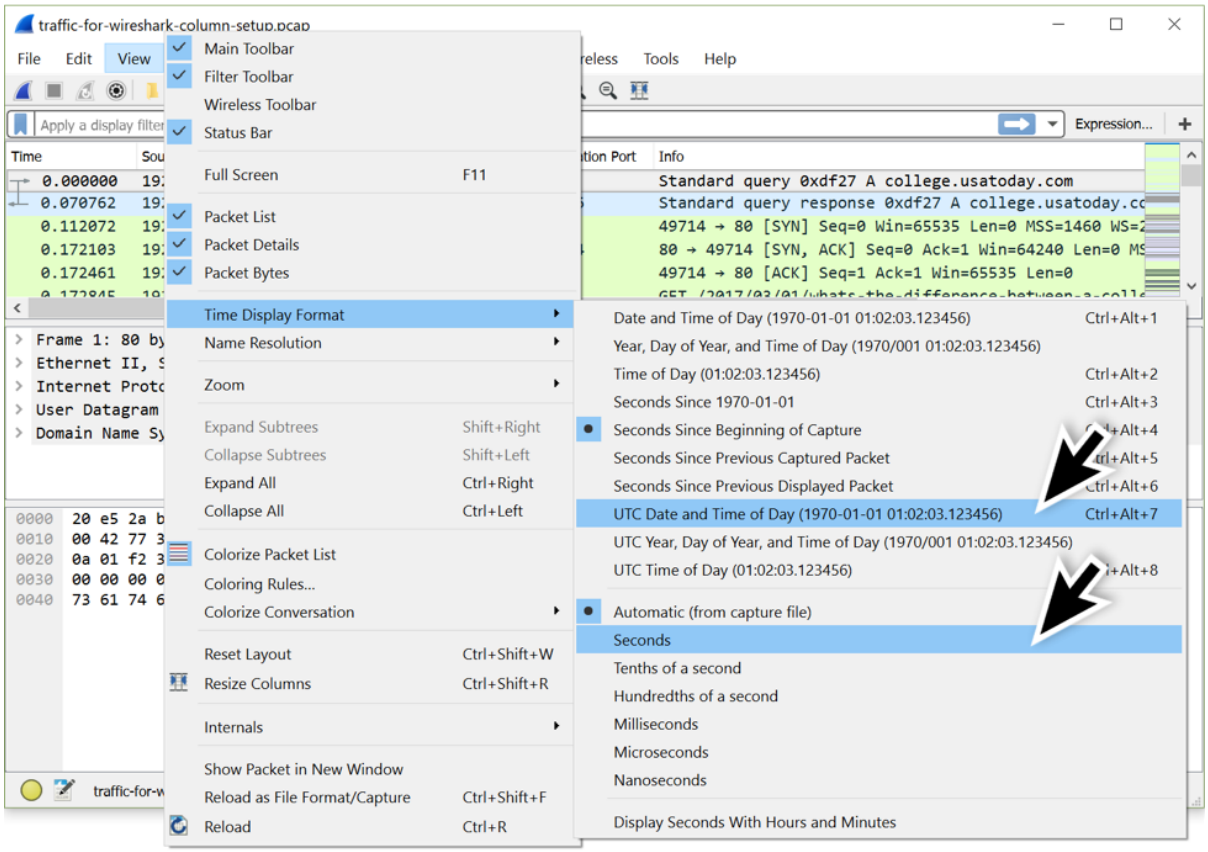


Figure 3.13: Changing the time display format to UTC date and time.

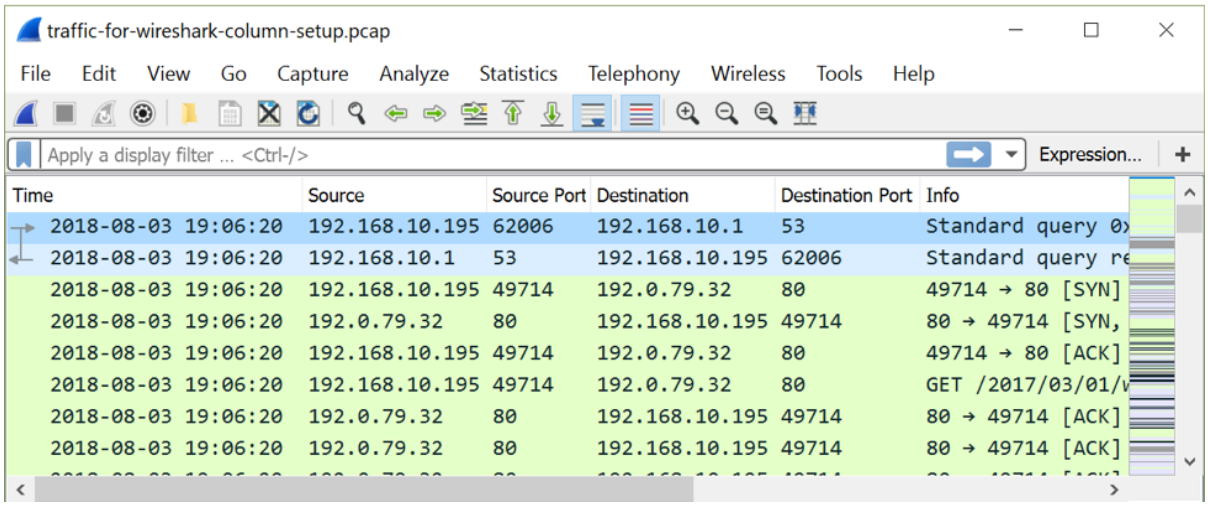


Figure 3.14: UTC date and time as seen in updated Wireshark column display.

Adding Custom Columns While we can add several different types of columns through the column preferences menu, we cannot add every conceivable value.

Fortunately, Wireshark allows us to add custom columns based on almost any value found in the frame details window.

This is how we add domain names used in HTTP and HTTPS traffic to our Wireshark column display.

To quickly find domains used in HTTP traffic, use the Wireshark filter **http.request** and examine the frame details window

.

In the frame details window, expand the line titled "Hypertext Transfer Protocol" by left clicking on the arrow that looks like a greater than sign to make it point down. This reveals several additional lines. Scroll down to the line starting with "Host:" to see the HTTP host name.

Left click on this line to select it. Right click on the line to bring up a menu. Near the top of this menu, select "Apply as Column." This should create a new column with the HTTP host name.

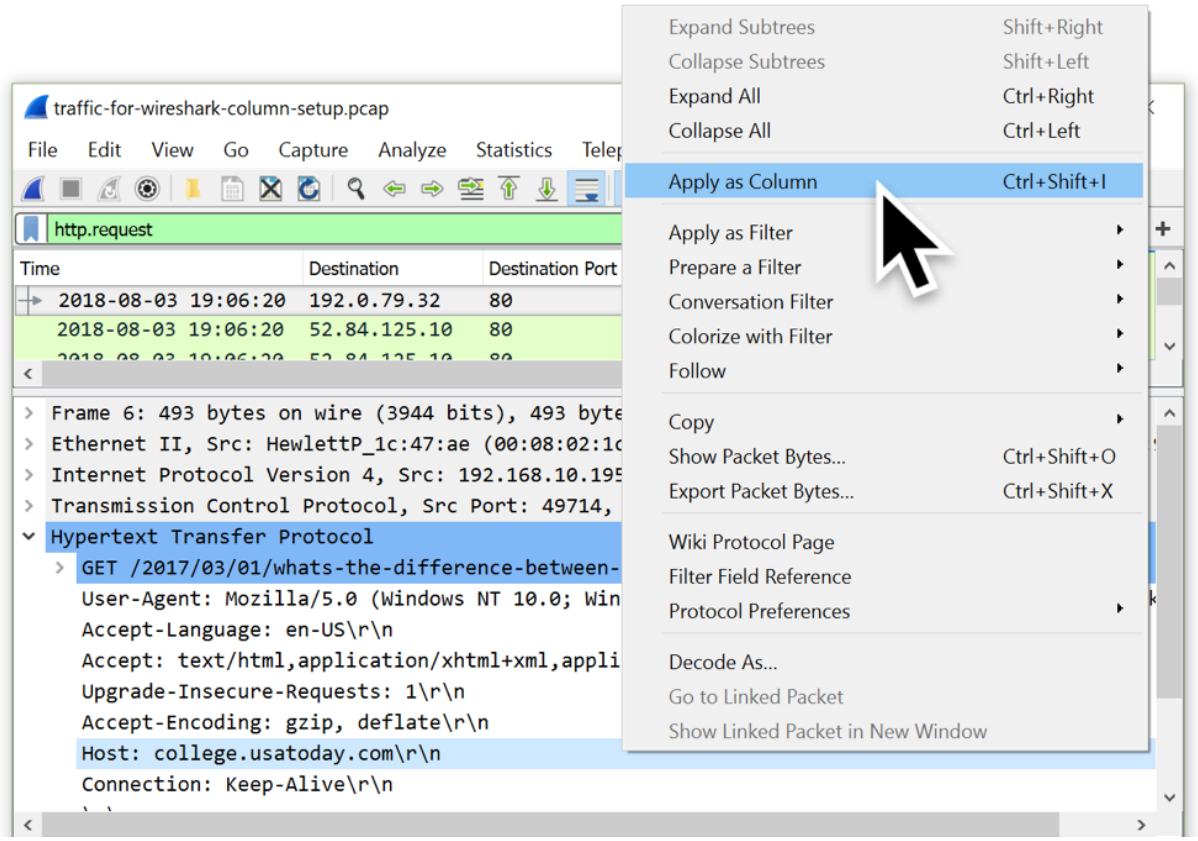


Figure 3.15: Applying the HTTP host name as a column.

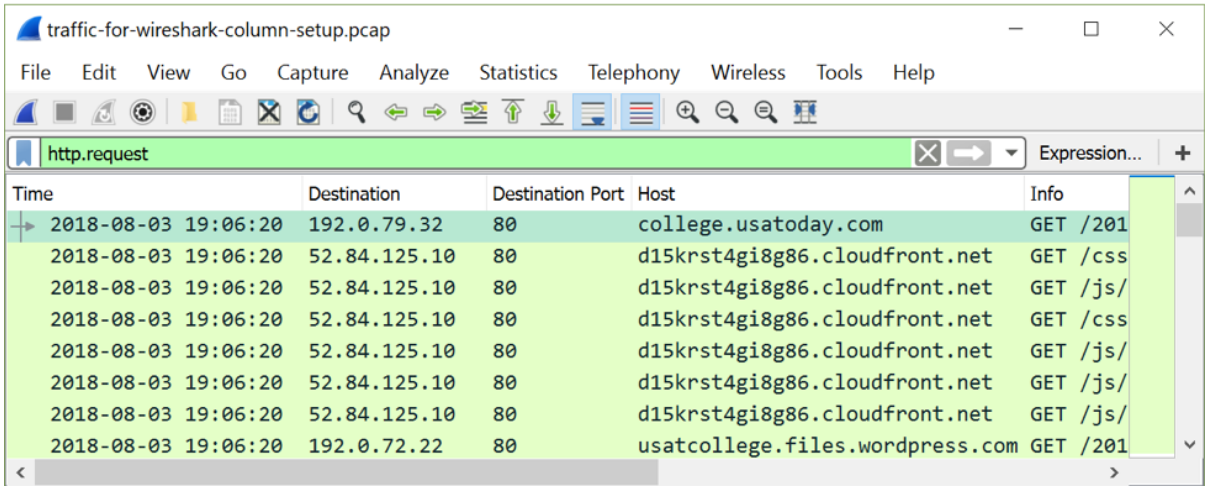


*Figure 3.16: HTTP host names in the column display when filtering on **http.request**.*

To find domains used in encrypted HTTPS traffic, use the Wireshark filter **ssl.handshake.type == 1** and examine the frame details window.

In the frame details window, expand the line titled "Secure Sockets Layer." Then expand the line for the TLS Record Layer. Below that expand another line titled "Handshake Protocol: Client Hello."

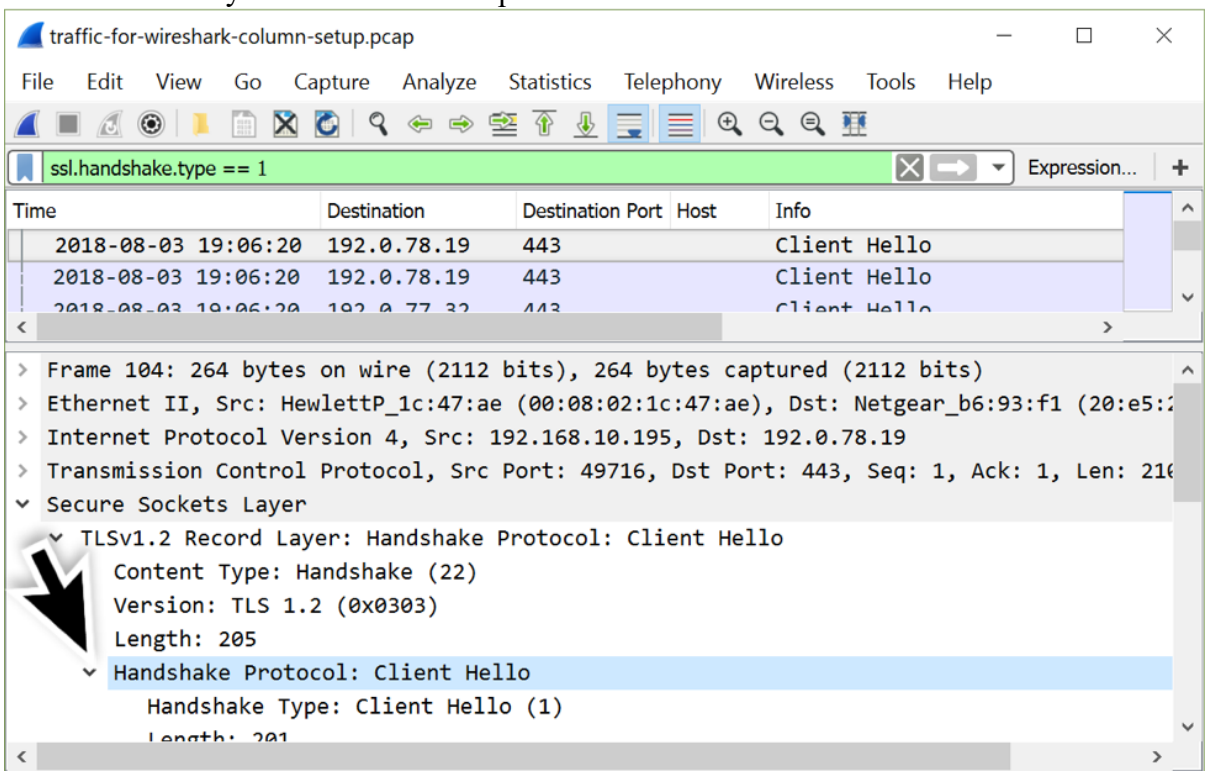


Figure 3.17: Filtering on SSL handshake type and working our way down.

Below the "Handshake Protocol: Client Hello" line, expand the line that starts with "Extension: server_name." Under that is "Server Name Indication extension" which contains several Server Name value types when expanded. Select the line that starts with "Server Name:" and apply it as a column.

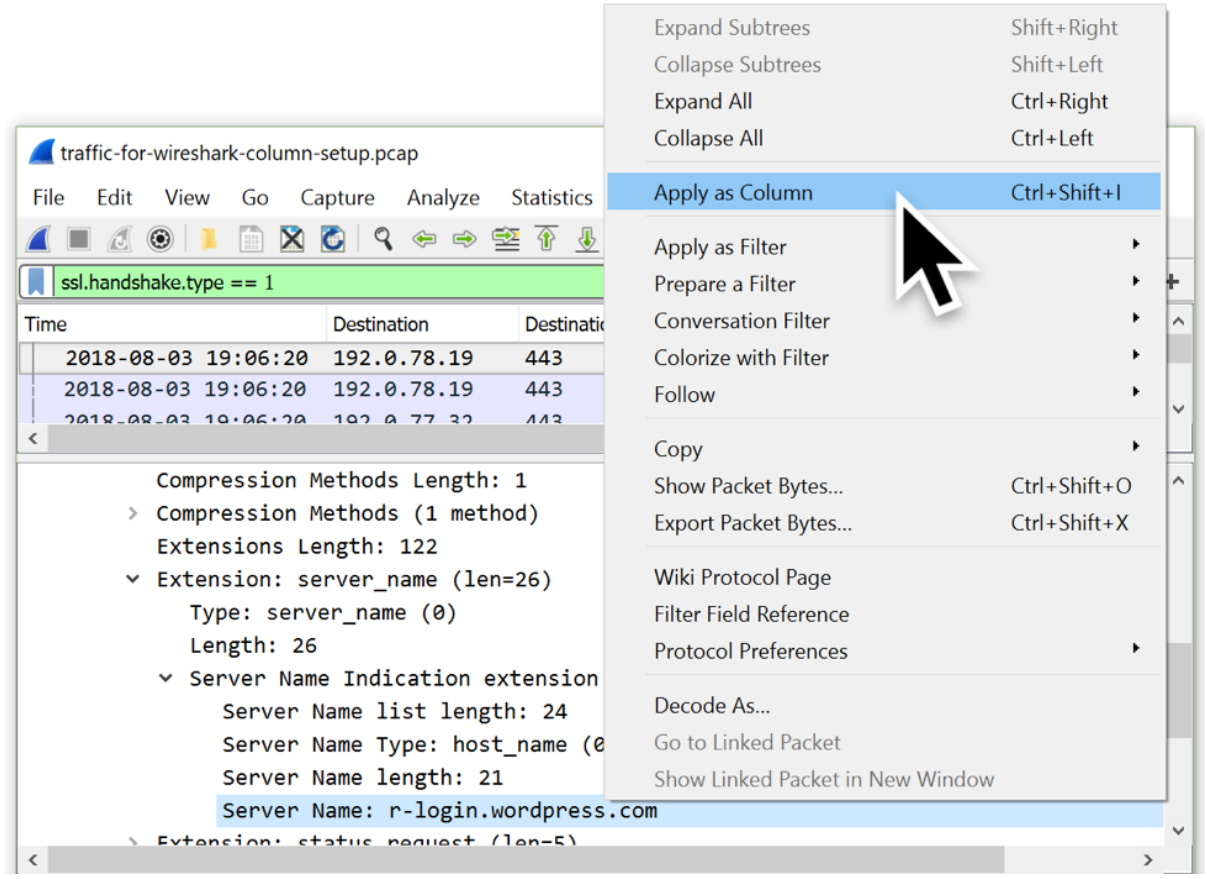


Figure 3.18: Applying the HTTPS server name as a column.

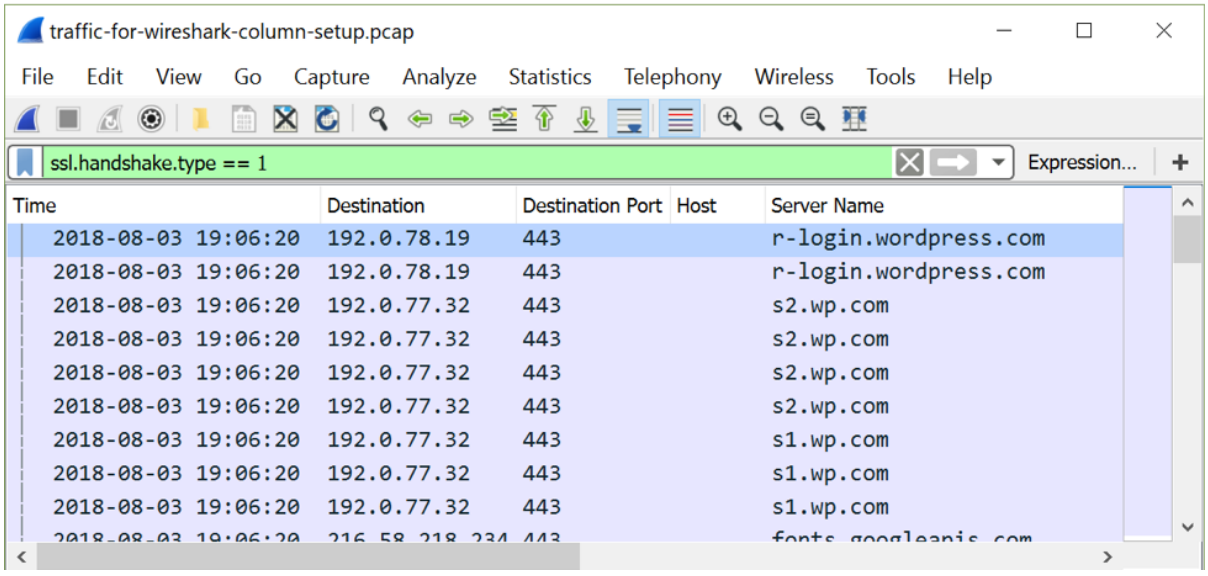


*Figure 3.19: HTTP server names in the column display when filtering on **ssl.handshake.type == 1**.*

With this customization, we can filter on **http.request or ssl.handshake.type == 1** as shown in Figure 20. This gives us a much better idea of web traffic in a pcap than using the default column display in Wireshark.

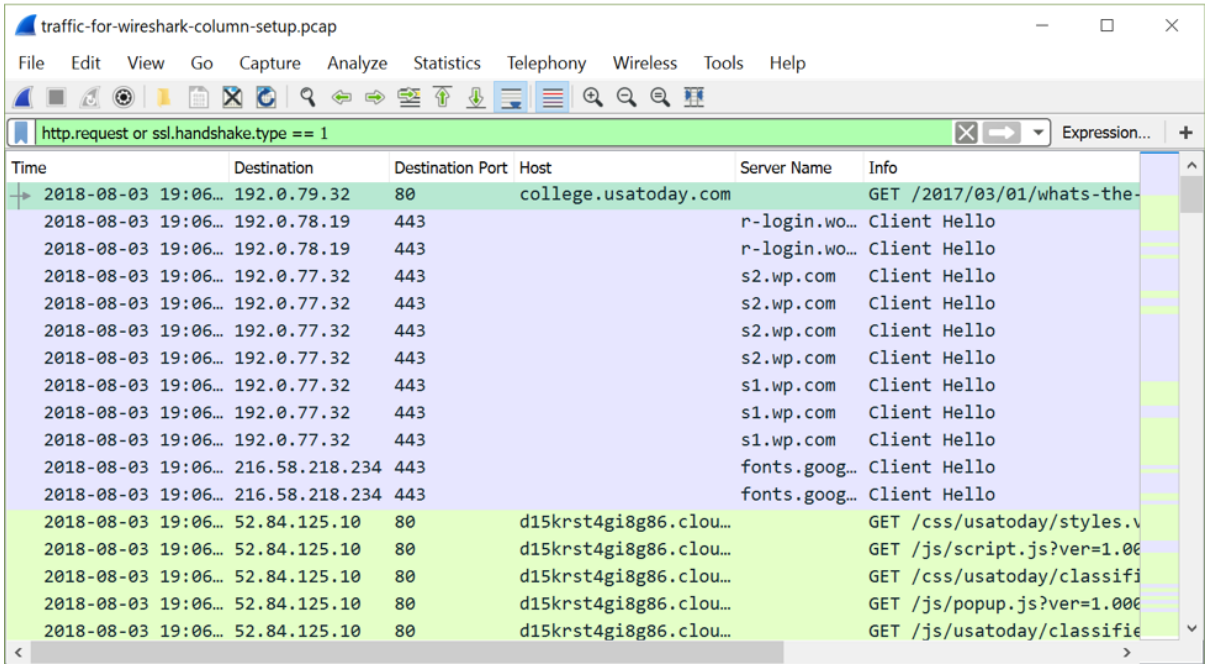


*Figure 3.20: Filtering on **http.request or ssl.handshake.type == 1** in the pcap for this tutorial.*

- **ANACONDA NAVIGATOR:**

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

The applications we can access using Navigator are as follows,

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- Spyder
- VSCode
- Glueviz
- Orange 3 App
- RStudio

- **JUPYTER NOTEBOOK:**

Introduction

The Jupyter Notebook is an **interactive computing environment** that enables users to author notebook documents that include:

- Live code
- Interactive widgets –

- Plots
- Narrative text
- Equations
- Images
- Video

Components

The Jupyter Notebook combines three components:

- **The notebook web application**:

  An interactive web application for writing and running code interactively and authoring notebook documents.

- **Kernels**:

  Separate processes started by the notebook web application that runs users' code in a given language and returns output back to the notebook web application. The kernel also handles things like computations for interactive widgets, tab completion and introspection.

- **Notebook documents**:

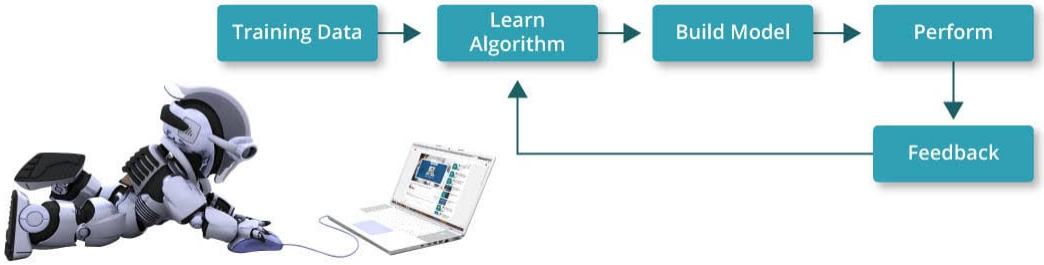
  Self-contained documents that contain a representation of all content visible in the notebook web application, including inputs and outputs of the computations, narrative text, equations, images, and rich media representations of objects. Each notebook document has its own kernel.

➢ **PANDAS:**

- Pandas Stands for "Python Data Analysis Library.
- Pandas takes data (like a CSV or TSV file, or a SQL database) and creates a Python object with rows and columns called data frame that looks very similar to table in a statistical software.
- NumPy stands for Numerical phython.
- It is a library consisting of multi-dimensional array objectsand a collection of routines for processing thise Array.
- Using NumPy mathematical and logical operation on arrays can be performed.
- Matplotlib.pyplot is a plotting library used for 2D graphics in python programming language. It can be used in python scripts, shell, web application servers and other graphical user interface toolkits.

➢ **SCIKIT-LEARN:**

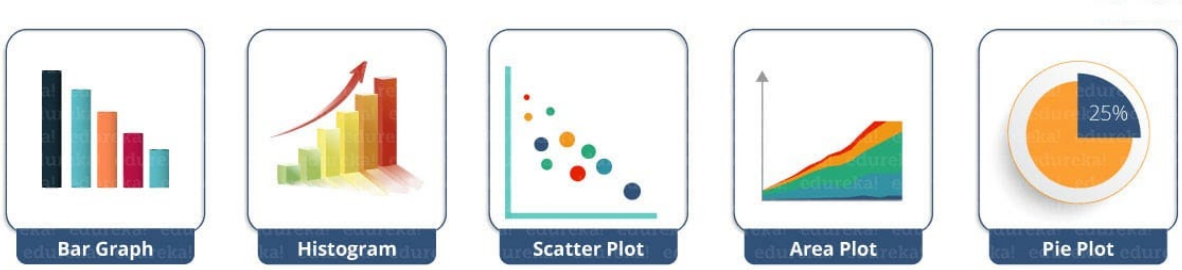
- Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language.
- It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.



➢ **MATPLOTLIB:**

Matplotlib.pyplot is a plotting library used for 2D graphics in python programming language. It can be used in python scripts, shell, web application servers and other graphical user interface toolkits.

- Python Matplotlib : Types of Plots
  - There are various plots which can be created using python matplotlib. Some of them are listed below:



CHAPTER 4
SOFTWARE AND HARDWARE INTERFACES

Software Interfaces:

Software configuration for back-end Services

- Windows 10
- Linux
- Wireshark
- Panadas /Numpy
- Scikit-learn

Software configuration for front-end Services

- Matplotlib

Hardware and Network Interfaces:

Back-end Server Configuration

- Intel Pentium-IV
- 128 MB RAM
- 1 Raid Controller Card
- 32 bit Ethernet Controller (100 Base T)
- 8 x 2.0 GB Fast SCSI/2 with Raid Support
- 2.88 MB FDD
- 48 x CD ROM Drive

Front-end Client Configuration

- Intel Pentium-III @ 650 MHz
- 128 MB SDRAM
- 10 GB Hard Disk Drive
- 1.44 MB Floppy Disk Drive
- 15” SVGA Digital Color Monitor
- One Serial, One Parallel pond One USB port.
- 104 Keys Keyboard
- PS2 Mouse with pad •
- 32 bit PCI Ethernet Card
- 48X CD Drive

CHAPTER 5

SYSTEM DESIGN

Flow chart for the seven machine learning steps:

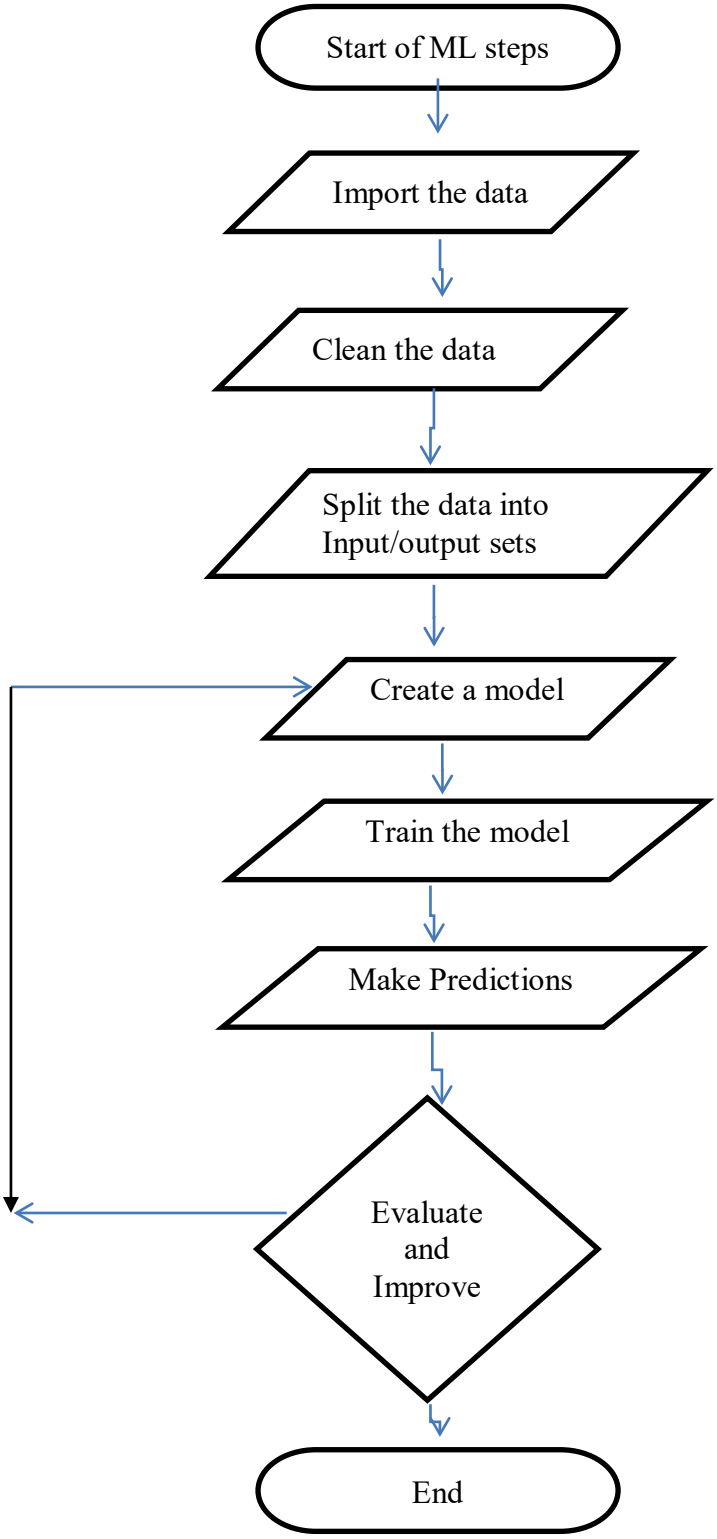


Fig 5.1: system design flow chart representation

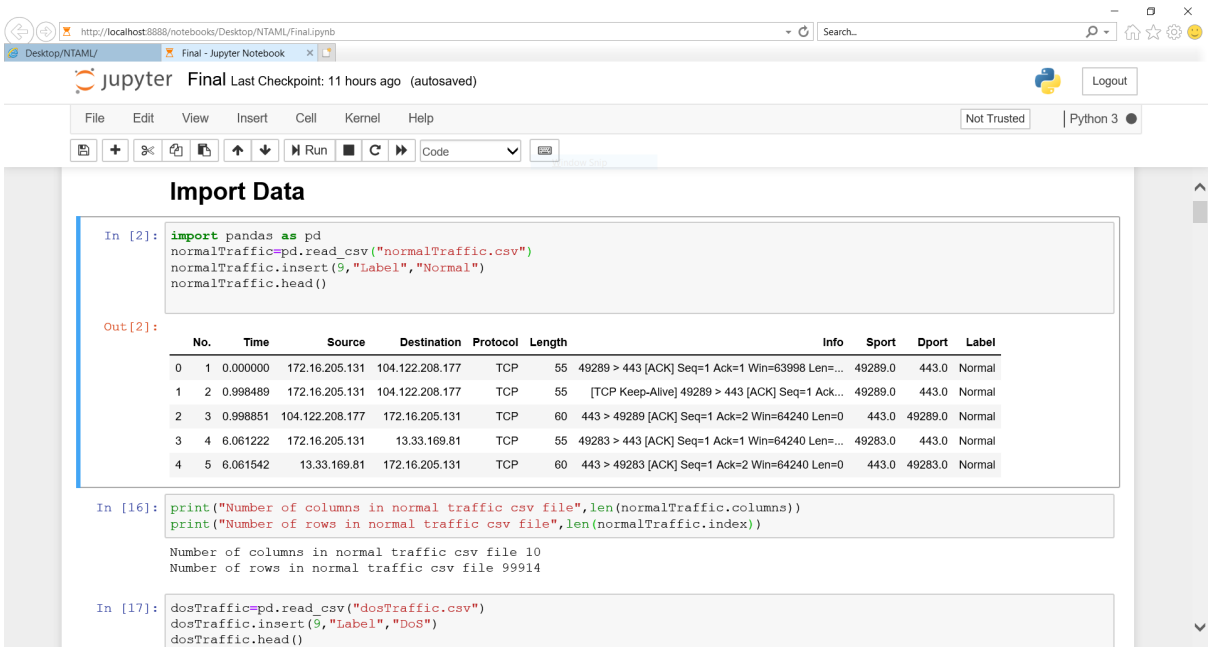
Performing Machine Learning 7 steps:

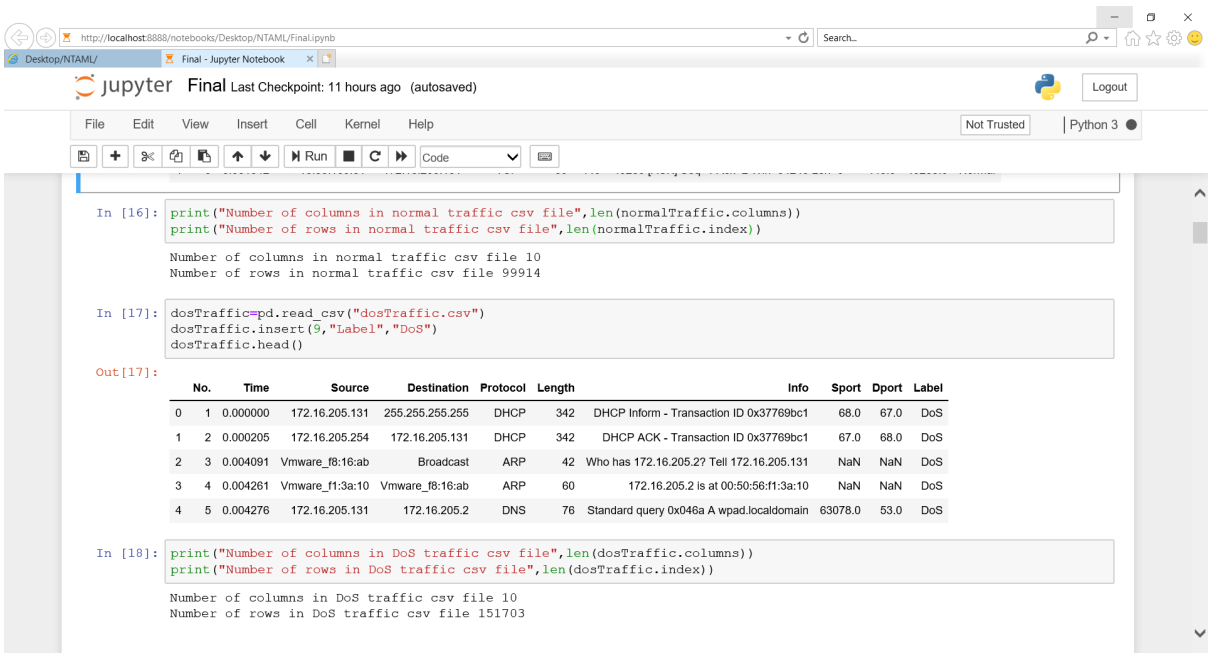
- Import the data.
- Clean the data
- Split the data into Input/output sets.
- Create a model
- Train the model
- Make Predictions
- Evaluate and Improve

STEP 1: **Importing the data as csv file**

By using the following command we have importee CSV file into Jupyter notebook using pandas library.

Command : import pandas as pd
From sklearn.tree import DecisionTreeClassifier
Countriesdata=pd.read_csv('countries.csv')





```
In [16]: print("Number of columns in normal traffic csv file",len(normalTraffic.columns))
         print("Number of rows in normal traffic csv file",len(normalTraffic.index))

         Number of columns in normal traffic csv file 10
         Number of rows in normal traffic csv file 99914

In [17]: dosTraffic=pd.read_csv("dosTraffic.csv")
         dosTraffic.insert(9,"Label","DoS")
         dosTraffic.head()
```

Out[17]:

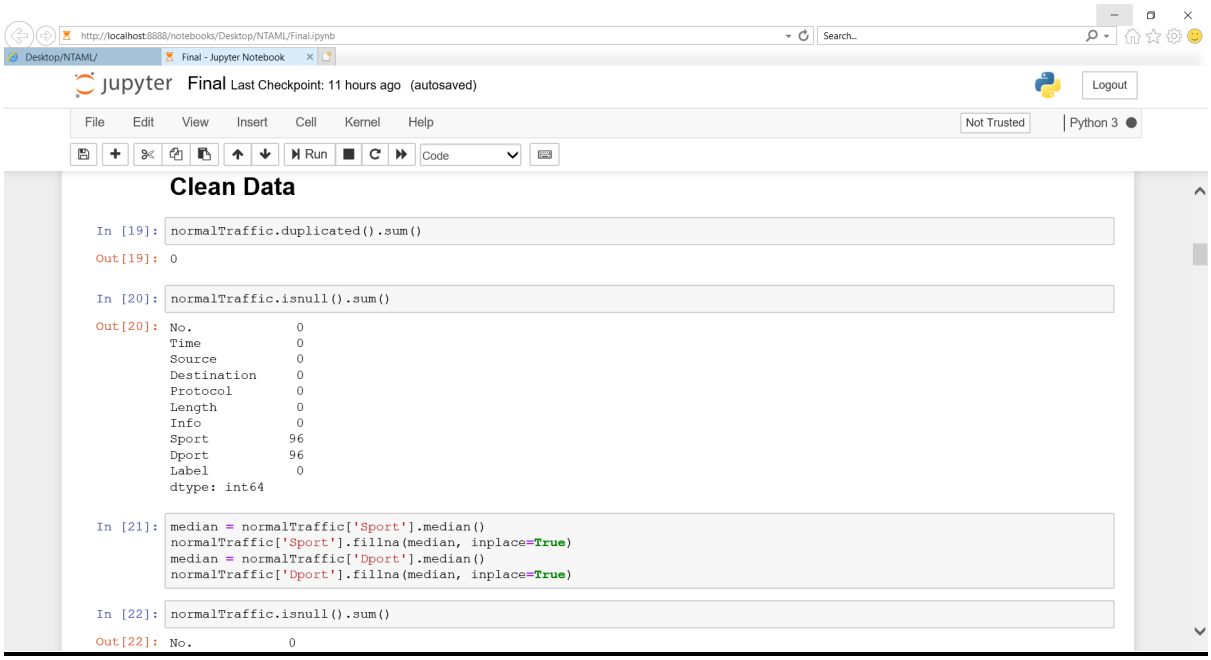
	No.	Time	Source	Destination	Protocol	Length	Info	Sport	Dport	Label
0	1	0.000000	172.16.205.131	255.255.255.255	DHCP	342	DHCP Inform - Transaction ID 0x37769bc1	68.0	67.0	DoS
1	2	0.000205	172.16.205.254	172.16.205.131	DHCP	342	DHCP ACK - Transaction ID 0x37769bc1	67.0	68.0	DoS
2	3	0.004091	Vmware_f8:16:ab	Broadcast	ARP	42	Who has 172.16.205.2? Tell 172.16.205.131	NaN	NaN	DoS
3	4	0.004261	Vmware_f1:3a:10	Vmware_f8:16:ab	ARP	60	172.16.205.2 is at 00:50:56:f1:3a:10	NaN	NaN	DoS
4	5	0.004276	172.16.205.131	172.16.205.2	DNS	76	Standard query 0x046a A wpad.localdomain	63078.0	53.0	DoS

```
In [18]: print("Number of columns in DoS traffic csv file",len(dosTraffic.columns))
         print("Number of rows in DoS traffic csv file",len(dosTraffic.index))

         Number of columns in DoS traffic csv file 10
         Number of rows in DoS traffic csv file 151703
```

Fig: 5.2 import the data

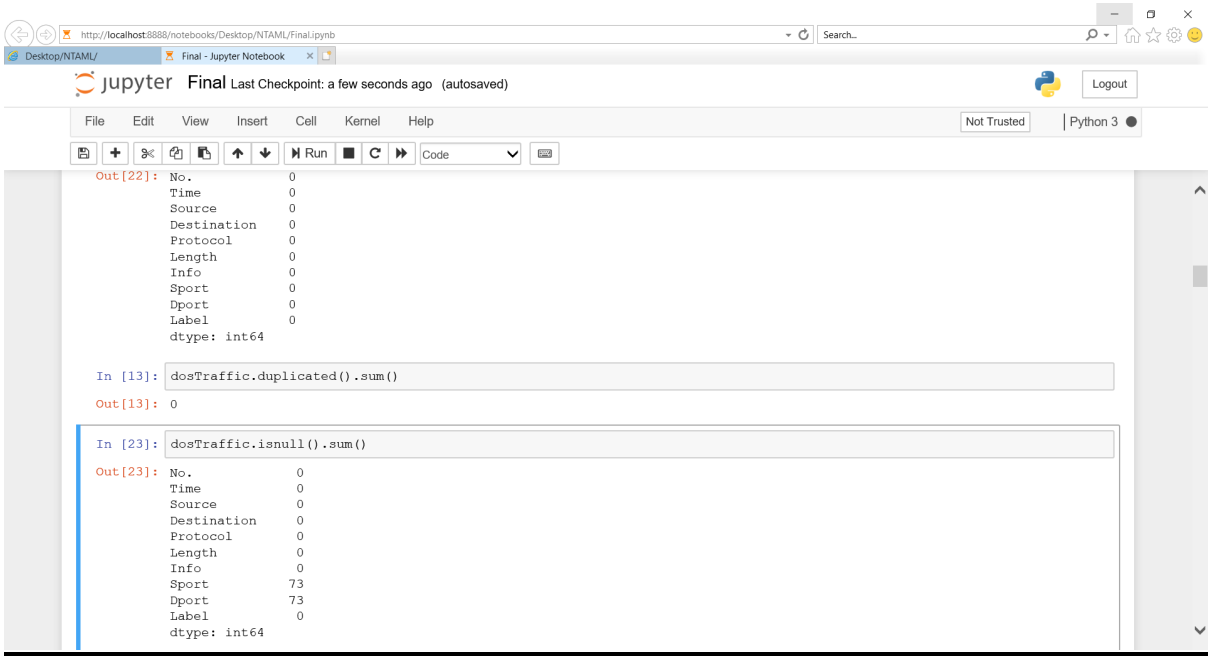
STEP 2: Cleaning the data

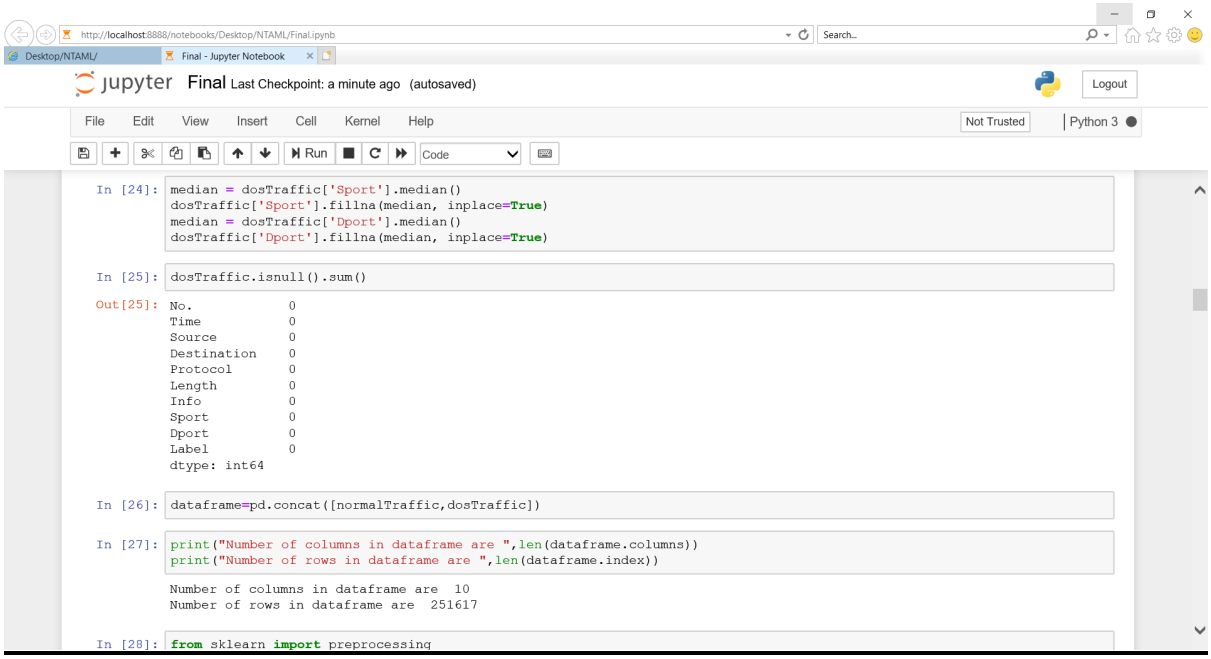
In this step, we'll clean the data i.e., removing the duplicate and null data (to ensure high accuracy).

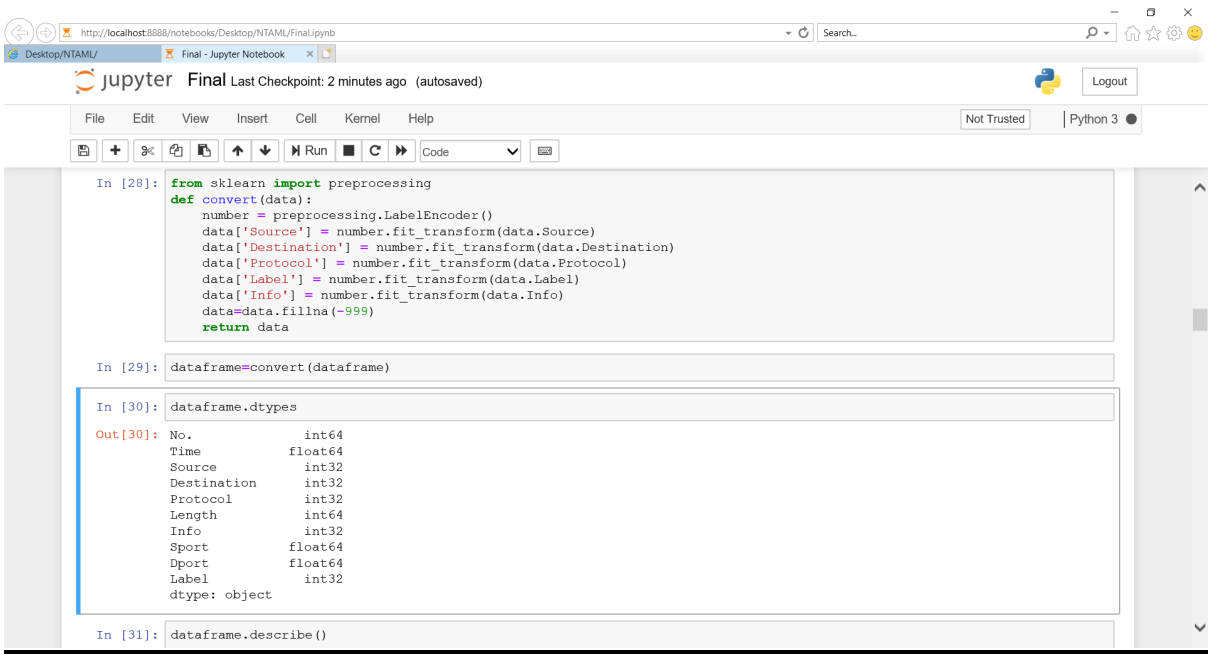


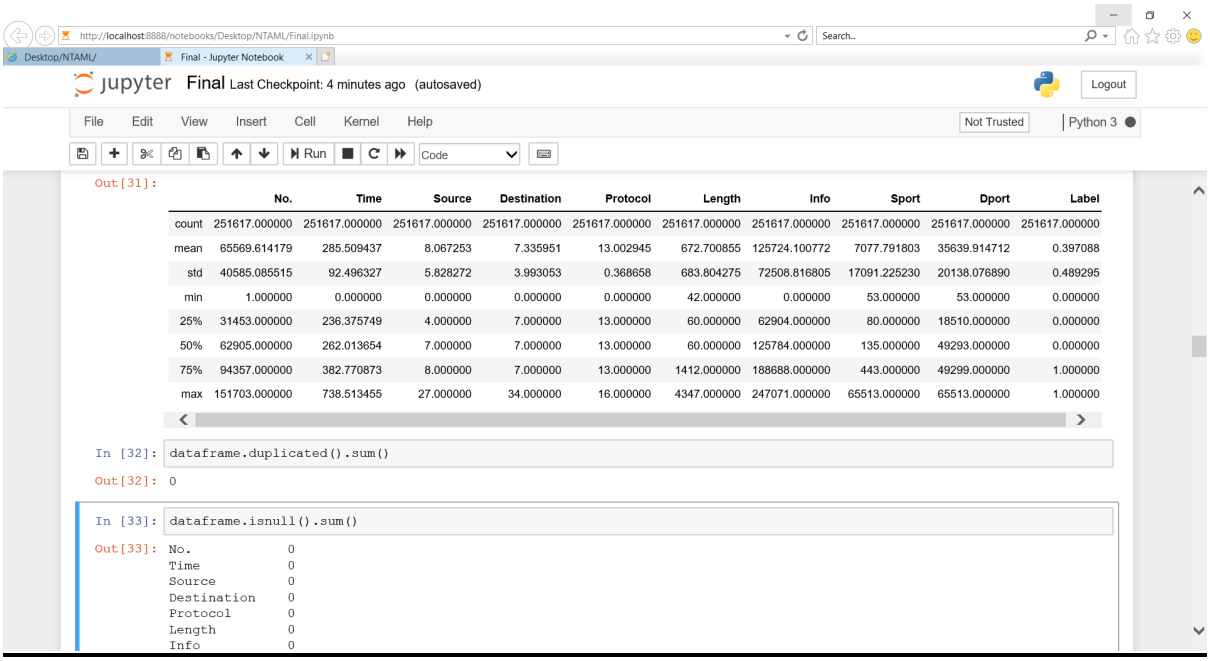
Clean Data

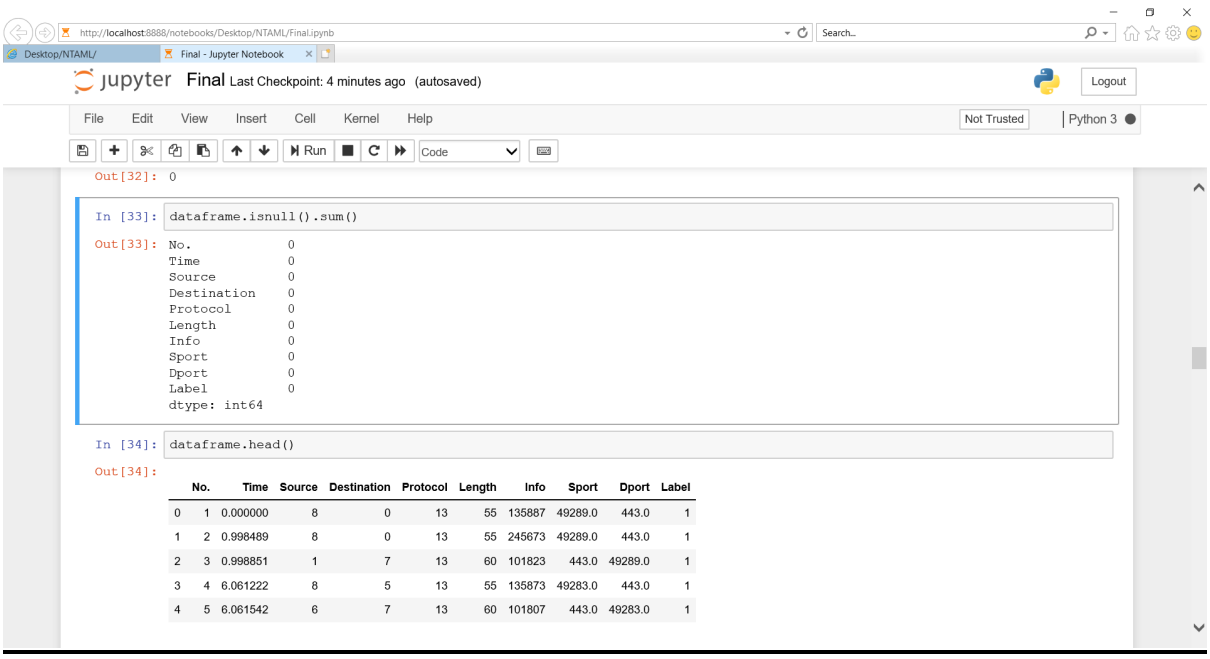
```
In [19]: normalTraffic.duplicated().sum()

Out[19]: 0

In [20]: normalTraffic.isnull().sum()

Out[20]: No.            0
         Time           0
         Source         0
         Destination    0
         Protocol       0
         Length         0
         Info           0
         Sport         96
         Dport         96
         Label          0
         dtype: int64

In [21]: median = normalTraffic['Sport'].median()
         normalTraffic['Sport'].fillna(median, inplace=True)
         median = normalTraffic['Dport'].median()
         normalTraffic['Dport'].fillna(median, inplace=True)

In [22]: normalTraffic.isnull().sum()

Out[22]: No.            0
```


http://localhost:8888/notebooks/Desktop/NTAML/Final.ipynb
Desktop/NTAML/
Final - Jupyter Notebook
jupyter Final Last Checkpoint: a few seconds ago (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Trusted
Python 3
Run
Code
Out[22]:
No. 0
Time 0
Source 0
Destination 0
Protocol 0
Length 0
Info 0
Sport 0
Dport 0
Label 0
dtype: int64
In [13]:
dosTraffic.duplicated().sum()
Out[13]:
0
In [23]:
dosTraffic.isnull().sum()
Out[23]:
No. 0
Time 0
Source 0
Destination 0
Protocol 0
Length 0
Info 0
Sport 73
Dport 73
Label 0
dtype: int64


http://localhost:8888/notebooks/Desktop/NTAML/Final.ipynb
Desktop/NTAML/
Final - Jupyter Notebook
jupyter Final Last Checkpoint: a minute ago (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Trusted
Python 3
Run
Code
In [24]:
median = dosTraffic['Sport'].median()
dosTraffic['Sport'].fillna(median, inplace=True)
median = dosTraffic['Dport'].median()
dosTraffic['Dport'].fillna(median, inplace=True)
In [25]:
dosTraffic.isnull().sum()
Out[25]:
No. 0
Time 0
Source 0
Destination 0
Protocol 0
Length 0
Info 0
Sport 0
Dport 0
Label 0
dtype: int64
In [26]:
dataframe=pd.concat([normalTraffic,dosTraffic])
In [27]:
print("Number of columns in dataframe are ",len(dataframe.columns))
print("Number of rows in dataframe are ",len(dataframe.index))
Number of columns in dataframe are 10
Number of rows in dataframe are 251617
In [28]:
from sklearn import preprocessing


jupyter Final Last Checkpoint: 2 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Trusted
Python 3
Run
Code
In [28]:
from sklearn import preprocessing
def convert(data):
number = preprocessing.LabelEncoder()
data['Source'] = number.fit_transform(data.Source)
data['Destination'] = number.fit_transform(data.Destination)
data['Protocol'] = number.fit_transform(data.Protocol)
data['Label'] = number.fit_transform(data.Label)
data['Info'] = number.fit_transform(data.Info)
data=data.fillna(-999)
return data
In [29]:
dataframe=convert(dataframe)
In [30]:
dataframe.dtypes
Out[30]:
No. int64
Time float64
Source int32
Destination int32
Protocol int32
Length int64
Info int32
Sport float64
Dport float64
Label int32
dtype: object
In [31]:
dataframe.describe()


jupyter Final Last Checkpoint: 4 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Trusted
Python 3
Run
Code
Out[31]:
No. Time Source Destination Protocol Length Info Sport Dport Label
count 251617.000000 251617.000000 251617.000000 251617.000000 251617.000000 251617.000000 251617.000000 251617.000000 251617.000000 251617.000000
mean 65569.614179 285.509437 8.067253 7.335951 13.002945 672.700855 125724.100772 7077.791803 35639.914712 0.397088
std 40585.085515 92.496327 5.828272 3.993053 0.368658 683.804275 72508.816805 17091.225230 20138.076890 0.489295
min 1.000000 0.000000 0.000000 0.000000 0.000000 42.000000 0.000000 53.000000 53.000000 0.000000
25% 31453.000000 236.375749 4.000000 7.000000 13.000000 60.000000 62904.000000 80.000000 18510.000000 0.000000
50% 62905.000000 262.013654 7.000000 7.000000 13.000000 60.000000 125784.000000 135.000000 49293.000000 0.000000
75% 94357.000000 382.770873 8.000000 7.000000 13.000000 1412.000000 188688.000000 443.000000 49299.000000 1.000000
max 151703.000000 738.513455 27.000000 34.000000 16.000000 4347.000000 247071.000000 65513.000000 65513.000000 1.000000
In [32]:
dataframe.duplicated().sum()
Out[32]:
0
In [33]:
dataframe.isnull().sum()
Out[33]:
No. 0
Time 0
Source 0
Destination 0
Protocol 0
Length 0
Info 0



	No.	Time	Source	Destination	Protocol	Length	Info	Sport	Dport	Label
0	1	0.000000	8	0	13	55	135887	49289.0	443.0	1
1	2	0.998489	8	0	13	55	245673	49289.0	443.0	1
2	3	0.998851	1	7	13	60	101823	443.0	49289.0	1
3	4	6.061222	8	5	13	55	135873	49283.0	443.0	1
4	5	6.061542	6	7	13	60	101807	443.0	49283.0	1

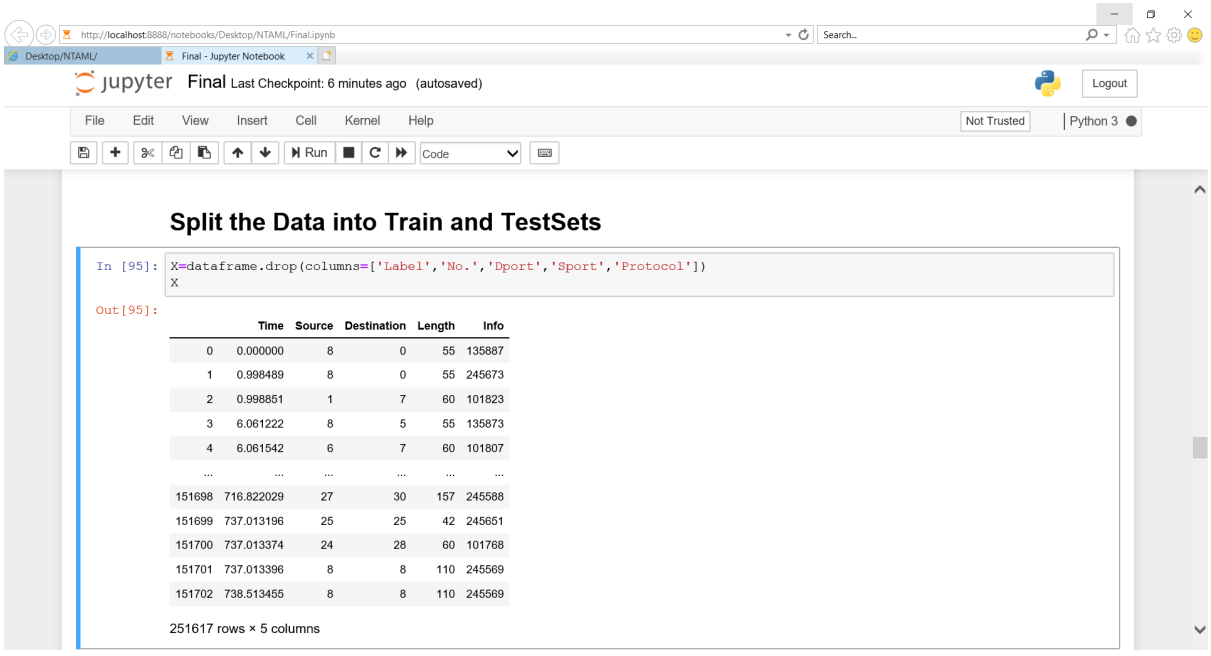
Fig: 5.3 cleaning data

STEP 3: **Split the data into Input/output sets**

Here, we use the commands like,

INPUTSET=countriesdata.drop(columns=[‘population’])

outputset=countriesdata[‘population’])



	Time	Source	Destination	Length	Info
0	0.000000	8	0	55	135887
1	0.998489	8	0	55	245673
2	0.998851	1	7	60	101823
3	6.061222	8	5	55	135873
4	6.061542	6	7	60	101807
...	...	...	...	...	...
151698	716.822029	27	30	157	245588
151699	737.013196	25	25	42	245651
151700	737.013374	24	28	60	101768
151701	737.013396	8	8	110	245569
151702	738.513455	8	8	110	245569

251617 rows × 5 columns

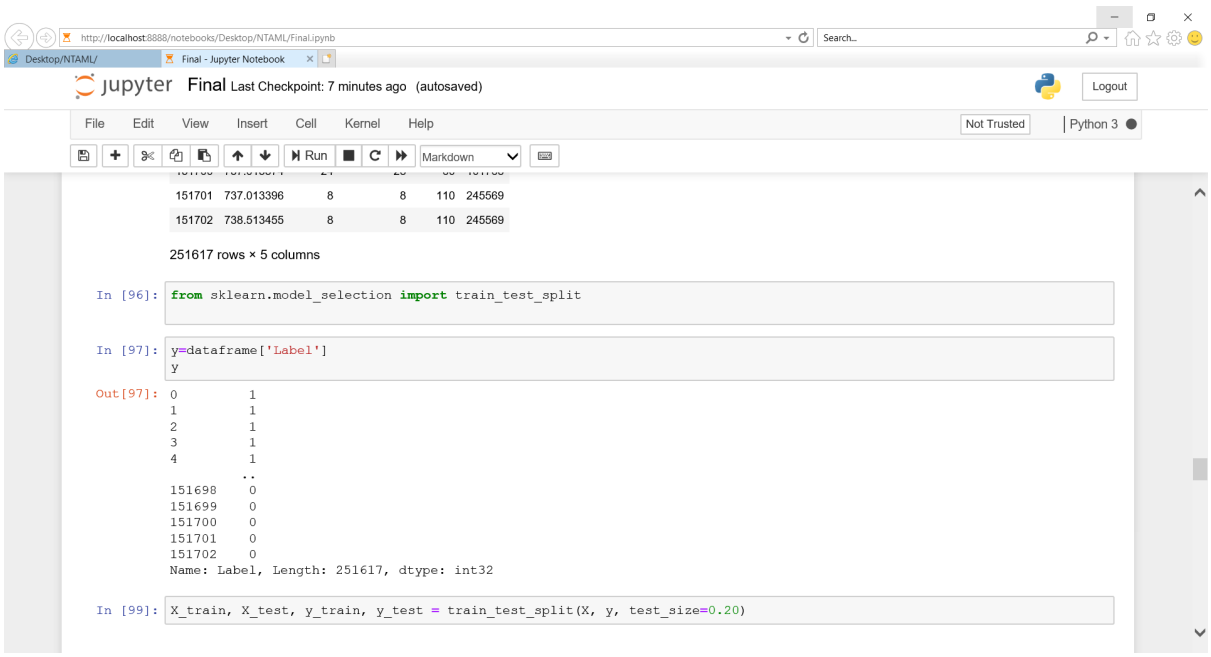


Fig: 5.4 split data

STEP 4: Create a model

This step involves, selecting an Machine Learning algorithm for analysing the data (Algorithm to be used is yet to be finalised).

Here, the command is as follows

Model=DecisionTreeClassifier()

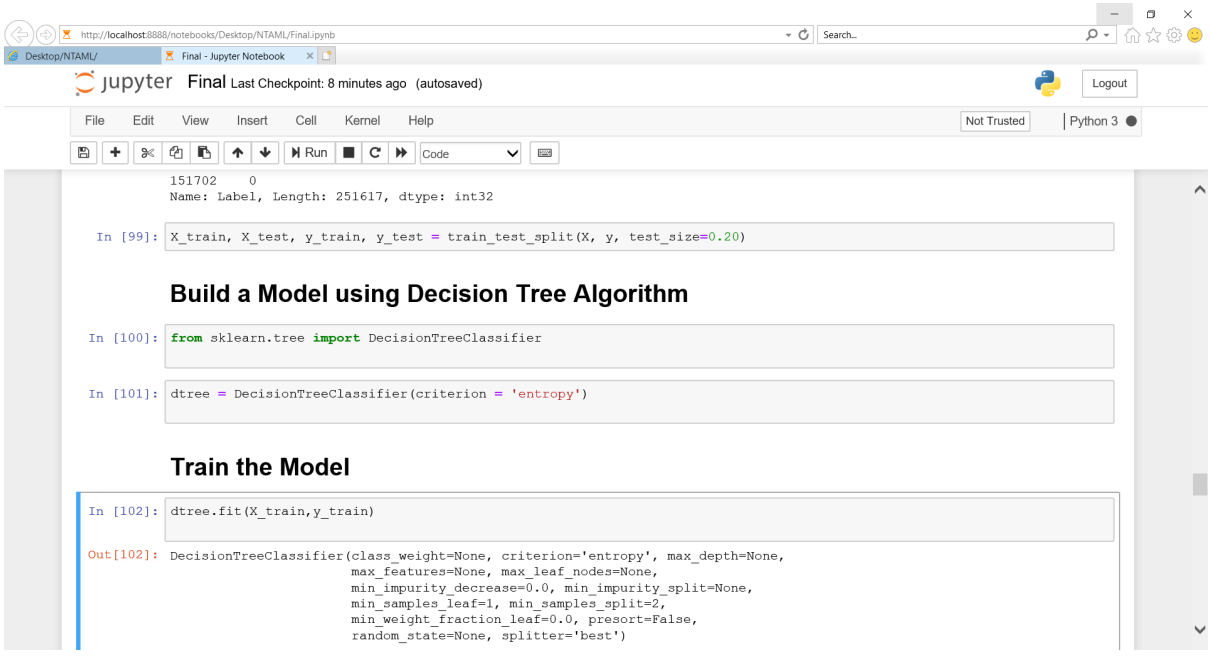


Fig:5.5 create model

CHAPTER 6

IMPLEMENTATION

Data science provides a plethora of classification algorithms such as logistic regression, support vector machine, naive Bayes classifier, and decision trees. But near the top of the classifier hierarchy is the random forest classifier (there is also the random forest regressor but that is a topic for another day).

In this post, we will examine how basic decision trees work, how individual decisions trees are combined to make a random forest, and ultimately discover why random forests are so good at what they do.

STEP 5: **Train a model**

In this step, we are going to feed the training data and Testing Data preferably 80% for training and 20% for Testing for high accuracy.

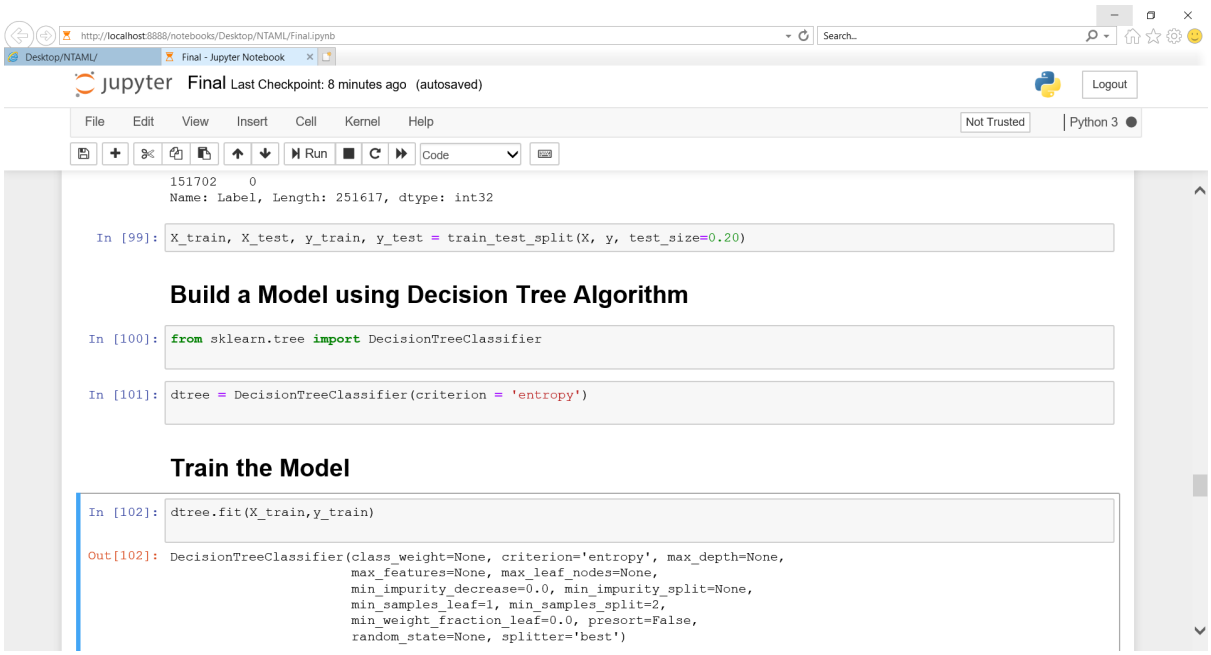


Fig:6.1 training the model

Decision Tree Algorithm:

Decision trees are the building blocks of the random forest model. Fortunately, they intuitively arrive at a decision.

Simple Decision Tree Example:

It's probably much easier to understand how a decision tree works through an example.

Imagine that our dataset consists of the numbers at the top of the figure to the left. We have two 1s and five 0s (1s and 0s are our classes) and desire to separate the classes using their features. The features are color (red vs. blue) and whether the observation is underlined or not. So how can we do this?

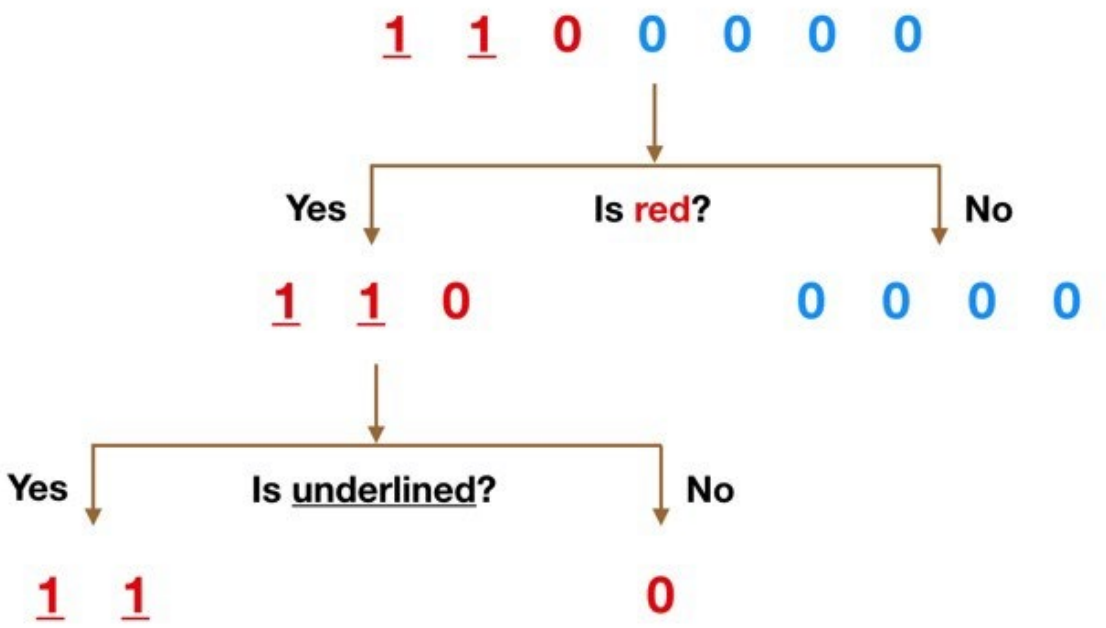


Fig 6.2 Decision tree

Color seems like a pretty obvious feature to split by as all but one of the 0s are blue. So we can use the question, "Is it red?" to split our first node. You can think of a node in a tree as the point where the path splits into two — observations that meet the criteria go down the Yes branch and ones that don't go down the No branch.

The No branch (the blues) is all 0s now so we are done there, but our Yes branch can still be split further. Now we can use the second feature and ask, "Is it underlined?" to make a second split.

The two 1s that are underlined go down the Yes sub branch and the 0 that is not underlined goes down the right sub branch and we are all done. Our decision tree was able to use the two features to split up the data perfectly.

Obviously in real life our data will not be this clean but the logic that a decision tree employs remains the same. At each node, it will ask

What feature will allow me to split the observations at hand in a way that the resulting groups are as different from each other as possible (and the members of each resulting subgroup are as similar to each other as possible)?

STEP 6: **Make predictions**

It will predict whether it is normal packet or DoS (Denial of Service) attack based on our model.

STEP 7: **Evaluate and improve**

If the predictions go wrong then we have to make some improvements or else we have to change the model.

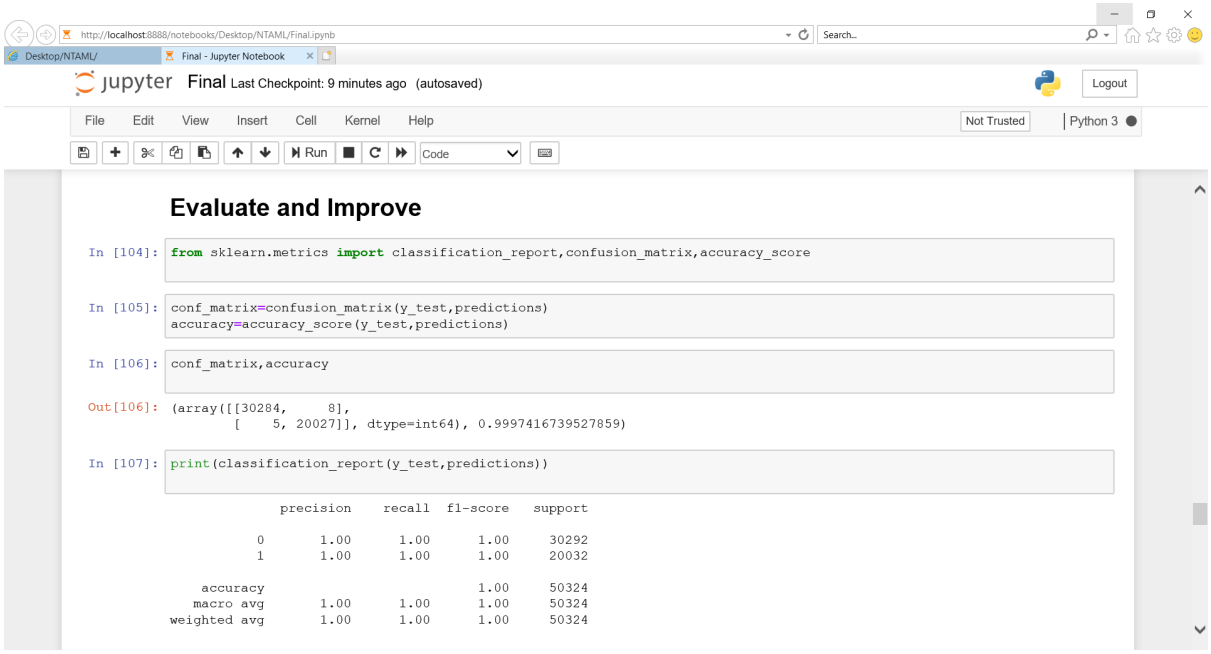


Fig: 6.3 Evaluate and improve

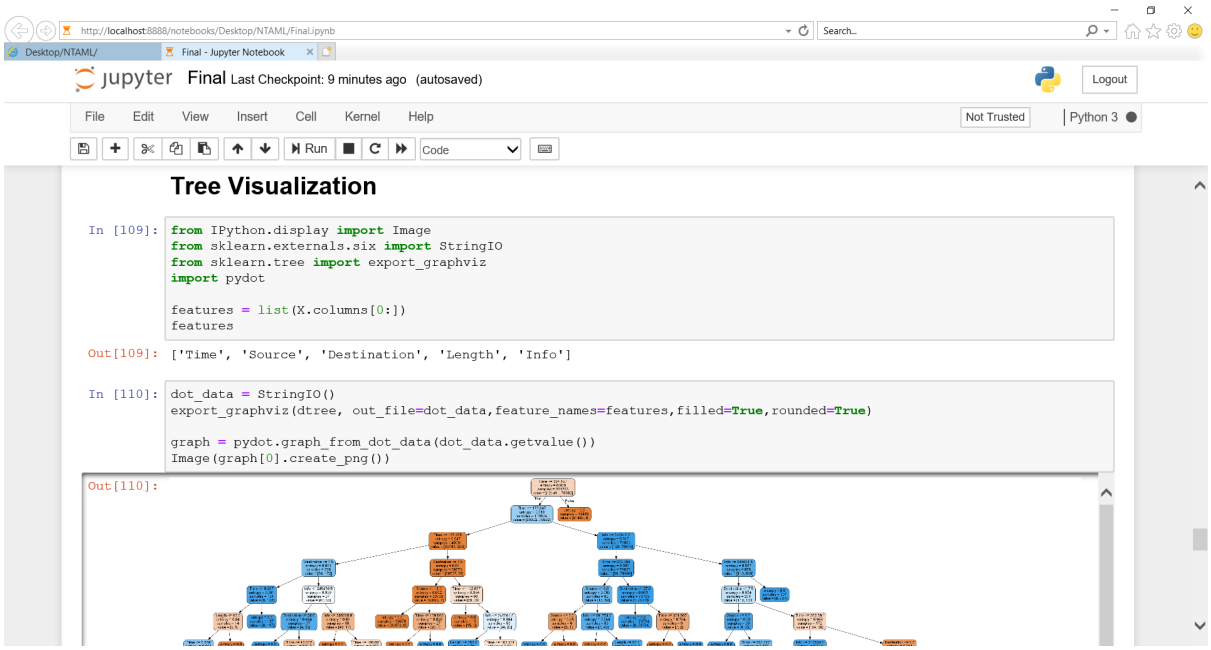
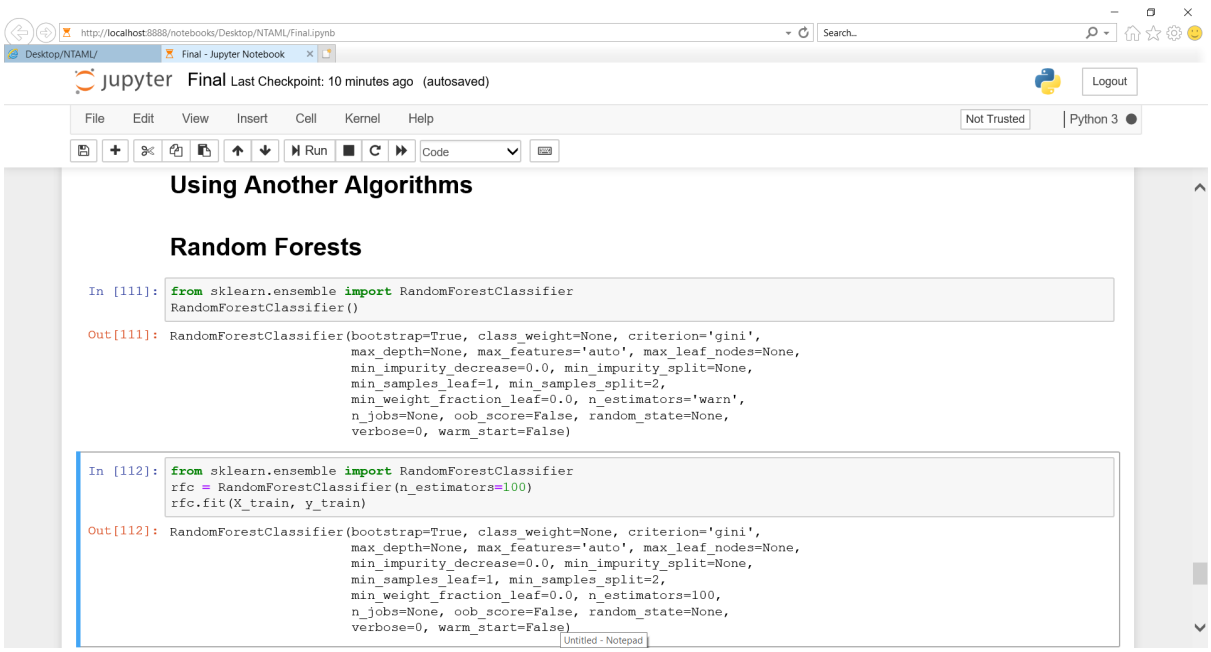


Fig: 6.4 Tree visualization



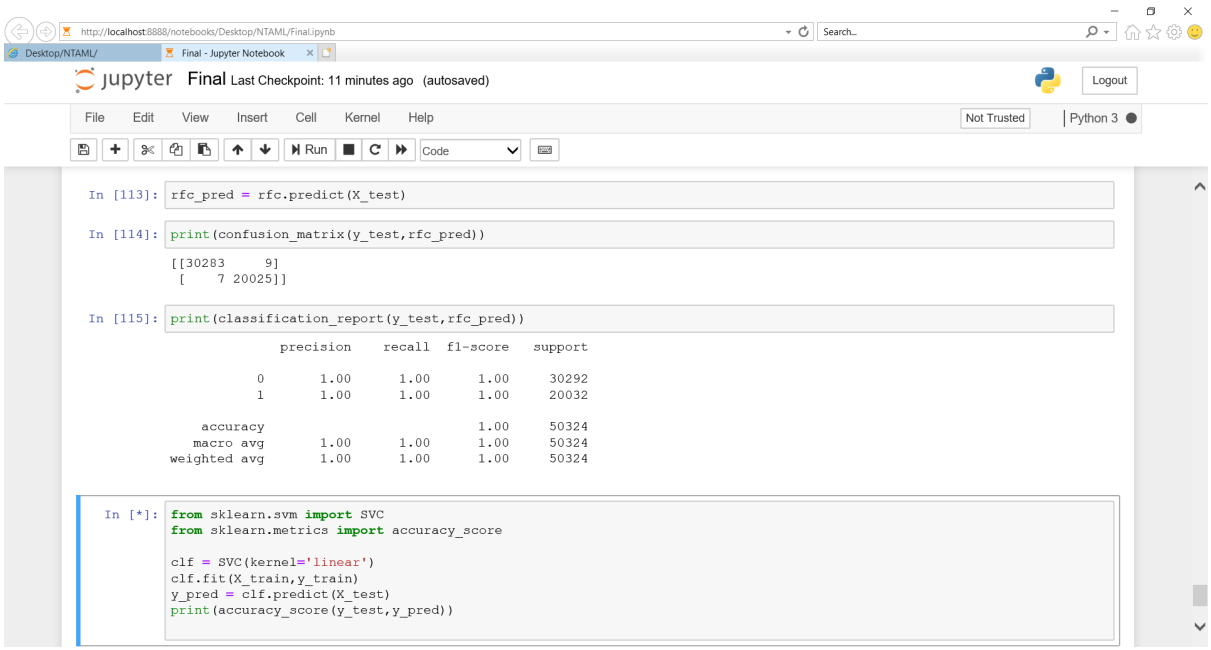


Fig:6.5 Random forest algorithm

Random Forest Decision Tree Algorithm

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction (see figure below).

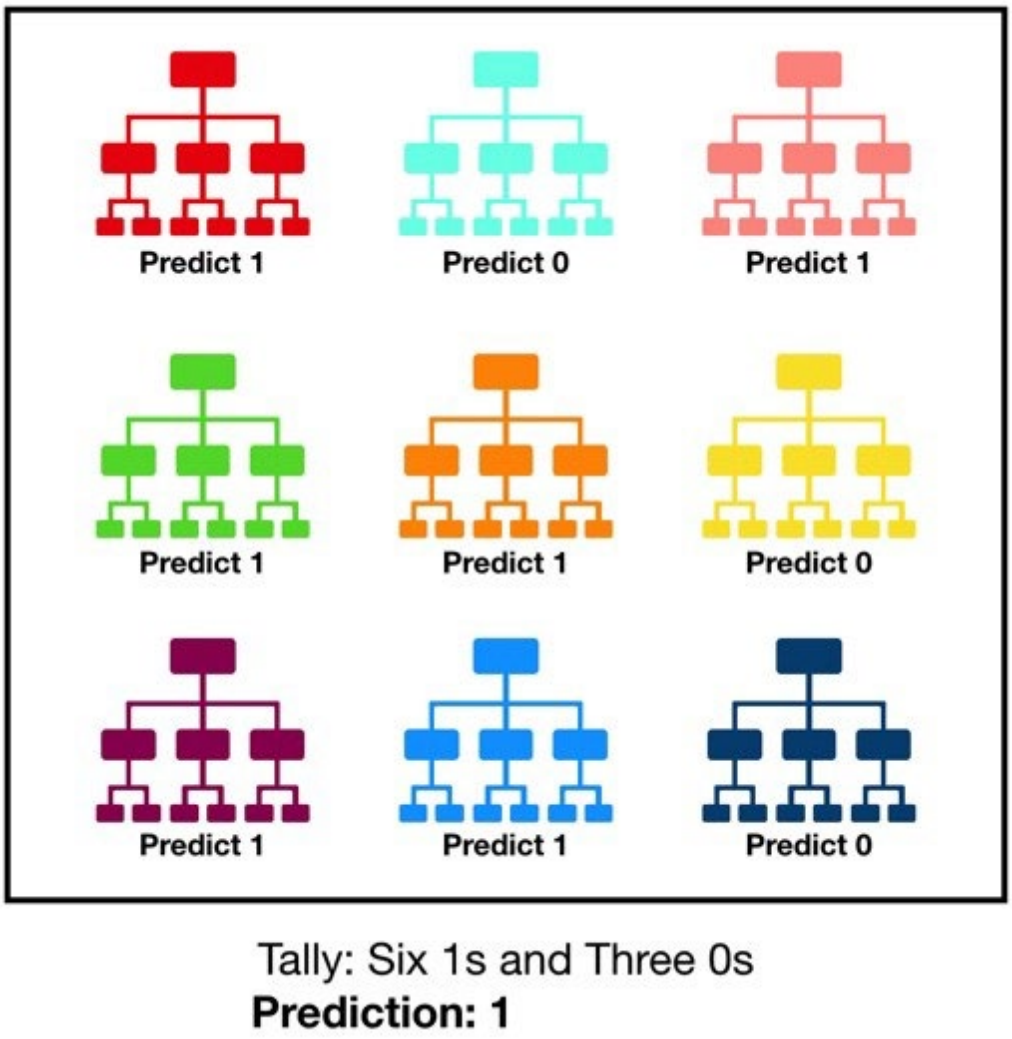


Fig:6.6 Random forest

Visualization of a Random Forest Model Making a Prediction

The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. In data science speak, the reason that the random forest model works so well is:

A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

The low correlation between models is the key. Just like how investments with low correlations come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. The reason for this wonderful effect is that the trees protect each other from their individual errors (as long as they don't constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. So the prerequisites for random forest to perform well are:

There needs to be some actual signal in our features so that models built using those features do better than random guessing.

The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

CHAPTER 7
CONCLUSION

In this project, several experiments were performed and tested to evaluate the efficiency and the performance of the Decision tree and Random Forest classifier. All the tests were based on the generated dataset. The rate of the different type of the attacks in the are approximately 80% of normal traffic and 20% DOS attacks. In the project 251617 instances of records have been extracted as training data to build the training models for the selected machine learning classifiers. The testing phase is implemented based on 60000 random instances of records. Several performance metrics are computed (accuracy rate, precision, false negative , false positive, true negative and true positive). The experiments have demonstrated that there is no single machine learning algorithm which can handle efficiently all the types of attacks. The decision tree classifier achieved accuracy of 0.9998 and false positive and false negative values of 4 and 6 respectively. The Random Forest Classifier gave a accuracy of 1.0 with false positive and false negative value of 1 each.

In contrast, all of the selected machine learning classifiers, were able to build their training models in an acceptable period of time. Furthermore, to save the availability and the confidentiality of the network resources, the true positive and the average accuracy rates alone are not sufficient to detect the intrusion. False negative and false positive rates are also needed to be taken into consideration. The normal traffic and DoS traffic was visualized with matplotlib.

CHAPTER 8

REFERENCES

1. https://www.google.com/search?q=IDS+using+machine+learning&oq=IDS+using+machine+learning&aqs=chrome..69i57.30575j0j1&sourceid=chrome&ie=UTF-8
2. https://medium.com/cuelogic-technologies/evaluation-of-machine-learning-algorithms-for-intrusion-detection-system-6854645f9211
3. https://ieeexplore.ieee.org/document/8681044
4. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
5. https://www.ee.ryerson.ca/~bagheri/papers/cisda.pdf