

Projet de Fin d’Etudes

Pour l’obtention du diplôme

Master en Internet des Objets et Systèmes Mobiles

Intitulé

Réalisation d’une solution IOT pour le Smart Home.

Réalisé par : El Azzaoui M’hamed

Soutenance le 22/06/2023

Soutenu devant le jury:

Pr. Said MAZER (President)

Pr. Younes BALBOUL (Examinateur)

Pr. Ouafae EL HAMDANI (Examinateur)

Mme Dounia DAGHOUJ (Examinateur)

Encadré par :

Pr. Younes BALBOUL

Mme Dounia DAGHOUJ

List of Codes

[Code III. 1: Converting the ‘ts’ column to datetime format. 6](#_Toc138179349)

[Code III. 2: Histogram of the distribution of ts. 6](#_Toc138179350)

[Code III. 3: Scatter plot of ts and duration. 6](#_Toc138179351)

[Code III. 4: Plot of ts aggregation by minute. 7](#_Toc138179352)

[Code III. 5: Plot of the total duration of network traffic by hour. 7](#_Toc138179353)

[Code III. 6: Bar Plot showing the 20 top source IP addresses with the highest number of connections. 8](#_Toc138179354)

[Code III. 7: Scatter Plot of the relationship between the source IP address and the amount of data transferred. 8](#_Toc138179355)

[Code III. 8: Histogram of the distribution of traffic across different source ports. 9](#_Toc138179356)

[Code III. 9: Scatter the most common 20 source ports used in the network traffic. 9](#_Toc138179357)

[Code III. 10: Scatter All the source ports with the largest durations (53671). 10](#_Toc138179358)

[Code III. 11: Scatter the usage of different source ports by different destination IPs for the first 10000 sample. 10](#_Toc138179359)

[Code III. 12: Histogram of the frequency of each destination IP address. 10](#_Toc138179360)

[Code III. 13: Bar Plot showing the 20 top destination IP addresses with the highest number of connections. 11](#_Toc138179361)

[Code III. 14: Scatter the relationship between the destination IP address and the amount of data transferred from the destination. 11](#_Toc138179362)

[Code III. 15: Histogram of distribution of traffic across different destination ports. 12](#_Toc138179363)

[Code III. 16: Scatter the most common 20 destination ports used in the network traffic. 12](#_Toc138179364)

[Code III. 17: Bar of a small range of the most common 20 destination ports used in the network traffic (between 0 and 450). 12](#_Toc138179365)

[Code III. 18: Scatter All the destination ports with the largest durations (2666). 13](#_Toc138179366)

[Code III. 19: Scatter the usage of different destination ports by different source IPs for the first 100000 sample. 13](#_Toc138179367)

[Code III. 20: Scatter the frequency of network connections for each combination of source and destination port. 14](#_Toc138179368)

[Code III. 21: Bar of the distribution of protocols. 14](#_Toc138179369)

[Code III. 22: Pie Chart of the proportion of each protocol. 14](#_Toc138179370)

[Code III. 23: Stacked Bar chart showing the distribution of protocols by connection state. 15](#_Toc138179371)

[Code III. 24: Countplot the distribution of connection states by protocol. 15](#_Toc138179372)

[Code III. 25: Heatmap of the frequency of each combination of protocol and service. 15](#_Toc138179373)

[Code III. 26: Stacked Bar Plot to show the distribution of attack categories within each protocol type. 16](#_Toc138179374)

[Code III. 27: Count plot of the percentage of each service category in the dataset. 16](#_Toc138179375)

[Code III. 28: Count plot showing the percentage of the service categories for each protocol value. 16](#_Toc138179376)

[Code III. 29: Count plot showing the percentage of the protocol categories for each service value. 16](#_Toc138179377)

[Code III. 30: Stacked bar plot to show the distribution of attack categories within each service type. 17](#_Toc138179378)

[Code III. 31: Heatmap showing the correlation between service type and other features. 17](#_Toc138179379)

[Code III. 32: Boxplot of src\_pkts grouped by the service variable. 17](#_Toc138179380)

[Code III. 33: Boxplot of src\_bytes grouped by the service variable. 18](#_Toc138179381)

[Code III. 34: Bar plot of src\_pkts sum grouped by the service variable. 18](#_Toc138179382)

[Code III. 35: Histogram of the distribution of values for the duration feature. 18](#_Toc138179383)

[Code III. 36: Histogram of the distribution of values for the duration feature for the smallest values. 19](#_Toc138179384)

[Code III. 37: Boxplot of the distribution of values and any potential outliers. 19](#_Toc138179385)

[Code III. 38: Scatter Plot of the duration & src\_bytes. 19](#_Toc138179386)

[Code III. 39: General Histogram of src\_bytes. 20](#_Toc138179387)

[Code III. 40: Histogram of src\_bytes for the very smallest values. 20](#_Toc138179388)

[Code III. 41: Boxplot of src\_bytes. 20](#_Toc138179389)

[Code III. 42: Scatter Plot of src\_bytes and dst\_bytes. 20](#_Toc138179390)

[Code III. 43: Box plot of src\_bytes for each attack category. 21](#_Toc138179391)

[Code III. 44: Box plot of src\_bytes for each protocol. 21](#_Toc138179392)

[Code III. 45: Histogram of dst\_bytes. 21](#_Toc138179393)

[Code III. 46: Histogram of dst\_bytes for the smallest values. 21](#_Toc138179394)

[Code III. 47: Box plot of dst\_bytes. 22](#_Toc138179395)

[Code III. 48: Bar plot of dst\_bytes average for each protocol. 22](#_Toc138179396)

[Code III. 49: Bar plot of dst\_bytes average for each protocol for small values. 22](#_Toc138179397)

[Code III. 50: Heatmap of the correlation between dst\_bytes and other numerical features in the dataset. 22](#_Toc138179398)

[Code III. 51: Bar plot showing the frequency of each connection state. 23](#_Toc138179399)

[Code III. 52: Count plot the frequency of each connection state for each attack category. 23](#_Toc138179400)

[Code III. 53: Violin plot the distribution of the duration for each connection state. 23](#_Toc138179401)

[Code III. 54: Histogram of missed\_bytes. 24](#_Toc138179402)

[Code III. 55: Histogram of missed\_bytes for the small values. 24](#_Toc138179403)

[Code III. 56: Boxplot of missed\_bytes. 24](#_Toc138179404)

[Code III. 57: Scatter Plot of missed\_bytes and duration. 24](#_Toc138179405)

[Code III. 58: Histogram of src\_pkts. 25](#_Toc138179406)

[Code III. 59: Histogram of src\_pkts for the small values. 25](#_Toc138179407)

[Code III. 60: Box plot of src\_pkts and conn\_state. 25](#_Toc138179408)

[Code III. 61: Box plot of src\_pkts and proto. 25](#_Toc138179409)

[Code III. 62: Box plot of src\_pkts and attack category. 26](#_Toc138179410)

[Code III. 63: Box plot of src\_pkts. 26](#_Toc138179411)

[Code III. 64: Histogram of src\_ip\_bytes. 26](#_Toc138179412)

[Code III. 65: Histogram of src\_ip\_bytes of the small values. 26](#_Toc138179413)

[Code III. 66: Box plot of src\_ip\_bytes. 27](#_Toc138179414)

[Code III. 67: Box plot of src\_ip\_bytes and type. 27](#_Toc138179415)

[Code III. 68: Histogram of dst\_pkts. 27](#_Toc138179416)

[Code III. 69: Histogram of dst\_pkts for the small values. 28](#_Toc138179417)

[Code III. 70: Box plot of dst\_pkts and conn\_state. 28](#_Toc138179418)

[Code III. 71: Box plot of dst\_pkts and proto. 28](#_Toc138179419)

[Code III. 72: Box plot of dst\_pkts and type. 28](#_Toc138179420)

[Code III. 73: Box plot of dst\_pkts. 29](#_Toc138179421)

[Code III. 74: Histogram of dst\_ip\_bytes. 29](#_Toc138179422)

[Code III. 75: Histogram of dst\_ip\_bytes for the small values. 29](#_Toc138179423)

[Code III. 76: Box plot of dst\_ip\_bytes. 29](#_Toc138179424)

[Code III. 77: Box plot of dst\_ip\_bytes and type. 30](#_Toc138179425)

[Code III. 78: Bar chart of the number of DNS queries accepted and those rejected. 30](#_Toc138179426)

[Code III. 79: Bar chart of the number of the top 20 DNS queries that have been rejected. 31](#_Toc138179427)

[Code III. 80: Bar chart of the distribution of DNS query types. 31](#_Toc138179428)

[Code III. 81: Bar chart of the distribution of DNS query types of a sample between 0 and 50. 31](#_Toc138179429)

[Code III. 82: Bar chart of the distribution of DNS query types for the small values. 32](#_Toc138179430)

[Code III. 83: Histogram of the distribution of dns\_qclass. 32](#_Toc138179431)

[Code III. 84: Histogram of the distribution of dns\_qclass for the small values. 32](#_Toc138179432)

[Code III. 85: Histogram of the distribution of dns\_qtype. 32](#_Toc138179433)

[Code III. 86: Histogram of the distribution of dns\_qtype for the small values. 33](#_Toc138179434)

[Code III. 87: Histogram of the distribution of dns\_rcode. 33](#_Toc138179435)

[Code III. 88: Bar plot of the distribution of the NON using dns, dns accepted and those rejected queries. 33](#_Toc138179436)

[Code III. 89: Bar chart of dns\_AA. 34](#_Toc138179437)

[Code III. 90: Bar chart of dns\_RD. 34](#_Toc138179438)

[Code III. 91: Bar chart of dns\_RA. 34](#_Toc138179439)

[Code III. 92: Scatter Plot of 400 samples randomly of dns\_query and duration. 35](#_Toc138179440)

[Code III. 93: Stacked Bar Chart showing the number of DNS rejected for each protocol. 35](#_Toc138179441)

[Code III. 94: Bar Chart of ssl\_version. 36](#_Toc138179442)

[Code III. 95: Bar Chart of ssl\_cipher. 36](#_Toc138179443)

[Code III. 96: Bar Chart of ssl\_resumed. 36](#_Toc138179444)

[Code III. 97: Bar Chart of ssl\_established. 37](#_Toc138179445)

[Code III. 98: Bar Chart of ssl\_subject. 37](#_Toc138179446)

[Code III. 99: Bar Chart of ssl\_issuer. 37](#_Toc138179447)

[Code III. 100: Stacked bar chart of ssl\_cipher and proto. 38](#_Toc138179448)

[Code III. 101: Stacked bar chart of ssl\_resumed and proto. 38](#_Toc138179449)

[Code III. 102: Stacked bar chart of ssl\_established and proto. 39](#_Toc138179450)

[Code III. 103: Bar chart of http\_method. 39](#_Toc138179451)

[Code III. 104: Bar chart of http\_orig\_mime\_types. 39](#_Toc138179452)

[Code III. 105: Bar chart of http\_resp\_mime\_types. 40](#_Toc138179453)

[Code III. 106: Stacked Bar chart of http\_trans\_depth & http\_version. 40](#_Toc138179454)

[Code III. 107: Stacked Bar chart of http\_trans\_depth & http\_version for smallest values. 41](#_Toc138179455)

[Code III. 108: Bar plot of the sum of http\_trans\_depth for each HTTP version. 41](#_Toc138179456)

[Code III. 109: Bar Chart of the 50/74 most URIs used in the HTTP request. 42](#_Toc138179457)

[Code III. 110: Scatter of the frequency of each http status code. 42](#_Toc138179458)

[Code III. 111: Bar chart of the frequency of each http user agent. 42](#_Toc138179459)

[Code III. 112: scatter of frequency http\_request\_body\_len. 43](#_Toc138179460)

[Code III. 113: scatter of frequency http\_response\_body\_len. 43](#_Toc138179461)

[Code III. 114: Stacked bar plot of http\_method & http\_status\_code. 44](#_Toc138179462)

[Code III. 115: Bar plot showing the distribution of weird\_name. 44](#_Toc138179463)

[Code III. 116: Bar plot showing the distribution of weird\_addl. 44](#_Toc138179464)

[Code III. 117: Bar plot showing the distribution of weird\_notice. 45](#_Toc138179465)

[Code III. 118: Bar chart showing the distribution of label. 45](#_Toc138179466)

[Code III. 119: Bar chart showing the distribution of attack types. 45](#_Toc138179467)

[Code III. 120: Bar chart of the missed\_bytes for each attack type. 46](#_Toc138179468)

[Code III. 121: Stacked bar plot to show the distribution of service types within each attack category. 46](#_Toc138179469)

[Code III. 122: Stacked bar plot to show the distribution of protocol types within each attack category. 46](#_Toc138179470)

[Code IV. 1: Data Preprocessing for all the Categorical Features. 49](#_Toc138179471)

[Code IV. 2: Data Preprocessing for all the Numerical Features. 51](#_Toc138179472)

[Code IV. 3: Data Preprocessing for all the Boolean Features. 51](#_Toc138179473)

[Code IV. 4: Data Preprocessing for the Timestamp Feature. 51](#_Toc138179474)

[Code IV. 5: Mapping of labels to encoded values of the type feature. 52](#_Toc138179475)

[Code IV. 6: Splitting the Data based on the type column. 52](#_Toc138179476)

[Code IV. 7: The application of the Decision Tree Classifier Model. 53](#_Toc138179477)

[Code IV. 8: The application of the Gaussian Naive Bayes Model. 53](#_Toc138179478)

[Code IV. 9: The application of the K-Nearest Neighbors Classifier Model. 54](#_Toc138179479)

[Code IV. 10: The application of the Random Forest Classifier Model. 55](#_Toc138179480)

[Code IV. 11: The application of the Logistic Regression Classifier Model. 55](#_Toc138179481)

*Appendix A*

Visualization Code

**In this section of appendix, you will find the code of all the visualizations performed on the ToN-IoT dataset. The organization of this section is the same as the chapter 3 for easy access.**

* 1. **The Connection Activity Features Visualization**
  + **ts Feature:**
* **ts: Histogram**

# Convert the 'ts' column to datetime format

df\_modified['ts'] = pd.to\_datetime(df\_modified['ts'], unit='s' )

Code III. 1: Converting the ‘ts’ column to datetime format.

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['ts'])

mpt.xlabel('Time')

mpt.ylabel('Number of connections')

mpt.title('Distribution of network traffic over time')

Code III. 2: Histogram of the distribution of ts.

* **ts & duration: Scatter Plot**

# Create a scatter plot of 'ts' vs 'duration'

fig, ax = mpt.subplots(figsize=(12, 6))

ax.scatter(df\_modified['ts'], df\_modified['duration'], s=2, alpha=0.5)

ax.set\_xlabel('Time')

ax.set\_ylabel('Duration')

ax.set\_title('Scatter Plot of Time vs Duration')

ax.set\_yticks(range(0,100000,10000))

mpt.show()

Code III. 3: Scatter plot of ts and duration.

* **ts aggregation by minute: Plot**

# Set the 'ts' column as the index

df\_modified.set\_index('ts', inplace=True)

# Resample the data to 1-minute based frequency and calculate the size of each resulting resampled group

ts\_df = df\_modified.resample('T').size()

mpt.figure(figsize=(12,6))

# Plot the time-series

mpt.plot(ts\_df.index, ts\_df.values)

mpt.title('Number of network connections over time')

mpt.xlabel('Time')

mpt.ylabel('Number of connections')

mpt.show()

Code III. 4: Plot of ts aggregation by minute.

* ts & duration: Plot of the total duration of network traffic by hour

# Resample the data to hourly frequency and calculate the sum of 'duration'

hourly\_traffic = df\_modified['duration'].resample('H').sum()

# Create a time-series plot of the hourly traffic

fig, ax = mpt.subplots(figsize=(12, 6))

ax.plot(hourly\_traffic.index, hourly\_traffic.values, linewidth=2, color='blue')

ax.set\_xlabel('Time')

ax.set\_ylabel('Total Duration')

ax.set\_title('Hourly Network Traffic')

mpt.show()

Code III. 5: Plot of the total duration of network traffic by hour.

* + src\_ip Feature
* src\_ip: Bar Plot to show the 20 top source IP addresses with the highest number of connections.

top\_src\_ips = df\_modified['src\_ip'].value\_counts().head(20)

top\_src\_ips

# Plot a bar chart of the top source IP addresses

mpt.figure(figsize=(12,6))

mpt.bar(top\_src\_ips.index, top\_src\_ips.values)

mpt.xticks(rotation=90)

mpt.grid(axis='y')

mpt.yticks(range(0,100000,10000))

top\_src\_ips\_list= top\_src\_ips.tolist()

for i in range(20):

    mpt.text(i-0.5, top\_src\_ips\_list[i]+1000,top\_src\_ips\_list[i])

mpt.xlabel('Source IP Addresses')

mpt.ylabel('Number of Connections')

mpt.title('Top 20 Source IP Addresses')

mpt.show()

Code III. 6: Bar Plot showing the 20 top source IP addresses with the highest number of connections.

* src\_ip & src\_bytes: Scatter Plot of the relationship between the source IP address and the amount of data transferred

# Plot a scatter plot of the source IP address and the amount of data transferred

mpt.figure(figsize=(24,12))

mpt.xticks(rotation=90)

mpt.scatter(df\_modified['src\_ip'].head(200000), df\_modified['src\_bytes'].head(200000), alpha=0.5 )

mpt.xlabel('Source IP Address')

mpt.ylabel('Amount of Data Transferred')

mpt.title('Amount of Data Transferred by Source IP Address')

mpt.show()

Code III. 7: Scatter Plot of the relationship between the source IP address and the amount of data transferred.

* + src\_port Feature
* src\_port: Histogram of the distribution of traffic across different source ports

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['src\_port'], bins=50)

mpt.title('Distribution of Source Ports')

mpt.xlabel('Source Port')

mpt.ylabel('Frequency')

mpt.show()

Code III. 8: Histogram of the distribution of traffic across different source ports.

* src\_port: Scatter the most common 20 source ports used in the network traffic

top\_ports = df\_modified['src\_port'].value\_counts().head(20)

#top\_ports

mpt.figure(figsize=(12,6))

mpt.scatter( top\_ports.index, top\_ports.values)

mpt.title('Top 20 Source Ports')

mpt.xlabel('Source Port')

mpt.ylabel('Frequency')

mpt.xticks(range(0,70000,5000) ,rotation=90)

mpt.grid(axis='y')

for i in range(20):

    mpt.text(top\_ports.index[i],top\_ports.values[i],top\_ports.values[i], ha='center', va='bottom',color='purple')

#mpt.xlim(40000,60000)

#mpt.yticks(range(0,20000,1000))

mpt.show()

Code III. 9: Scatter the most common 20 source ports used in the network traffic.

**src\_port & duration:** **Scatter All the source ports with the largest durations (53671)**

ports\_duration = df\_modified['src\_port'].value\_counts('duration').head(53671)

#ports\_duration

# total number of ports 53671

mpt.figure(figsize=(12,6))

mpt.scatter( ports\_duration.index, ports\_duration.values)

mpt.title('Top 20 Source Ports in duration')

mpt.xlabel('Source Port')

mpt.ylabel('Duration')

mpt.grid(axis='y')

mpt.show()

Code III. : Scatter All the source ports with the largest durations (53671)**.**

* **src\_port & dst\_ip:** **Scatter the usage of different source ports by different destination IPs for the first 10000 sample**

ports\_duration = df\_modified['src\_port'].value\_counts('duration').head(53671)

#ports\_duration

# total number of ports 53671

mpt.figure(figsize=(12,6))

mpt.scatter( ports\_duration.index, ports\_duration.values)

mpt.title('Top 20 Source Ports in duration')

mpt.xlabel('Source Port')

mpt.ylabel('Duration')

mpt.grid(axis='y')

mpt.show()

Code III. 11: Scatter the usage of different source ports by different destination IPs for the first 10000 sample.

* + dst\_ip Feature
* dst\_ip: Histogram of the frequency of each destination IP address. This can provide insights into the most frequently visited destinations.

import plotly.express as px

# Create a bar chart visualization

fig = px.histogram(df\_modified, x="dst\_ip", title="Destination IP Addresses", labels={"dst\_ip": "Destination IP Address", "count": "Frequency"})

fig.update\_layout(

    width=1000,  # specify the width in pixels

    height=600  # specify the height in pixels

)

# Show the visualization

fig.show()

Code III. 12: Histogram of the frequency of each destination IP address.

* **dst\_ip:** **Bar Plot to show the 20 top destination IP addresses with the highest number of connections.**

top\_dst\_ips = df\_modified['dst\_ip'].value\_counts().head(20)

top\_dst\_ips

# Plot a bar chart of the 20 top source IP addresses

mpt.figure(figsize=(12,6))

mpt.bar(top\_dst\_ips.index, top\_dst\_ips.values)

mpt.xticks(rotation=90)

mpt.grid(axis='y')

mpt.yticks(range(0,100000,10000))

top\_dst\_ips\_list= top\_dst\_ips.tolist()

for i in range(20):

    mpt.text(i-0.5, top\_dst\_ips\_list[i]+1000, top\_dst\_ips\_list[i])

mpt.xlabel('Destination IP Addresses')

mpt.ylabel('Number of Connections')

mpt.title('Top 20 Destination IP Addresses')

mpt.show()

Code III. 13: Bar Plot showing the 20 top destination IP addresses with the highest number of connections.

* dst\_ip & dst\_bytes: Scatter the relationship between the destination IP address and the amount of data transferred from the destination

# Plot a scatter plot of the destination IP address and the amount of data transferred

mpt.figure(figsize=(12,6))

mpt.xticks(rotation=90)

mpt.scatter(df\_modified['dst\_ip'].head(10000), df\_modified['dst\_bytes'].head(10000), alpha=0.5 )

mpt.xlabel('Destination IP Address')

mpt.ylabel('Amount of Data Transferred')

mpt.title('Amount of Data Transferred by Destinatiion IP Address')

mpt.show()

Code III. 14: Scatter the relationship between the destination IP address and the amount of data transferred from the destination.

* + dst\_port Feature
* dst\_port: Histogram of distribution of traffic across different destination ports

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['dst\_port'], bins=50)

mpt.title('Distribution of Destination Ports')

mpt.xlabel('Destination Port')

mpt.ylabel('Frequency')

mpt.show()

Code III. 15: Histogram of distribution of traffic across different destination ports.

* dst\_port: Scatter the most common 20 destination ports used in the network traffic & Bar of a small range of them (between 0 and 450)

top\_ports\_dst = df\_modified['dst\_port'].value\_counts().head(20)

#top\_ports\_dst

mpt.figure(figsize=(12,6))

mpt.scatter( top\_ports\_dst.index, top\_ports\_dst.values)

mpt.title('Top 20 Destination Ports')

mpt.xlabel('Destination Port')

mpt.ylabel('Frequency')

mpt.grid(axis='y')

for i in range(20):

    mpt.text(top\_ports\_dst.index[i],top\_ports\_dst.values[i],top\_ports\_dst.values[i], ha='center', va='bottom',color='purple')

#mpt.xlim(0,450)

mpt.show()

Code III. 16: Scatter the most common 20 destination ports used in the network traffic.

top\_ports\_dst = df\_modified['dst\_port'].value\_counts().head(20)

#top\_ports

mpt.figure(figsize=(12,6))

mpt.bar( top\_ports\_dst.index, top\_ports\_dst.values)

mpt.title('Top 10 Destination Ports')

mpt.xlabel('Destination Port')

mpt.ylabel('Frequency')

# mpt.xticks(range(0,70000,5000) ,rotation=90)

mpt.xlim(0,450)

mpt.grid(axis='y')

mpt.yticks(range(0,120000,10000))

mpt.show()

Code III. 17: Bar of a small range of the most common 20 destination ports used in the network traffic (between 0 and 450).

* dst\_port & duration: Scatter All the destination ports with the largest durations (2666)

ports\_duration\_dst = df\_modified['dst\_port'].value\_counts('duration').head(2666)

ports\_duration\_dst

# total number of destination ports 2666

mpt.figure(figsize=(12,6))

mpt.scatter( ports\_duration\_dst.index, ports\_duration\_dst.values)

mpt.title('Top Destination Ports in duration')

mpt.xlabel('Destination Port')

mpt.ylabel('Duration')

mpt.grid(axis='y')

mpt.show()

Code III. 18: Scatter All the destination ports with the largest durations (2666).

* dst\_port & src\_ip: Scatter the usage of different destination ports by different source IPs for the first 100000 sample

mpt.figure(figsize=(12,6))

mpt.scatter(df\_modified['src\_ip'].head(100000), df\_modified['dst\_port'].head(100000))

mpt.xticks(rotation=90)

mpt.title('Port Usage by Source IP')

mpt.xlabel('Source IP')

mpt.ylabel('Destination Port')

mpt.show()

Code III. 19: Scatter the usage of different destination ports by different source IPs for the first 100000 sample.

* dst\_port & src\_port: Scatter the frequency of network connections for each combination of source and destination port. This can help identify any patterns or trends in the traffic flow.

# create a scatter

mpt.figure(figsize=(12,6))

mpt.scatter(df\_modified['src\_port'],df\_modified['dst\_port'])

mpt.title('Traffic Scatter')

mpt.xlabel('Destination Port')

mpt.ylabel('Source Port')

mpt.show()

Code III. 20: Scatter the frequency of network connections for each combination of source and destination port.

* + proto Feature
* **proto: Bar of the distribution of protocols**

protocol\_counts = df\_modified['proto'].value\_counts()

mpt.figure(figsize=(12,6))

mpt.bar(protocol\_counts.index, protocol\_counts.values)

mpt.title('Distribution of Network Protocols')

mpt.xlabel('Protocol')

mpt.ylabel('Number of Connections')

proto\_list= protocol\_counts.tolist()

for i in range(3):

    mpt.text(i-0.1, proto\_list[i], proto\_list[i])

mpt.show()

Code III. 21: Bar of the distribution of protocols.

* **proto:** **Pie Chart of the proportion of each protocol**

protocol\_counts = df\_modified['proto'].value\_counts()

mpt.figure(figsize=(12,6))

mpt.pie(protocol\_counts.values, labels=protocol\_counts.index, autopct='%1.1f%%')

mpt.title('Proportion of Network Protocols')

mpt.show()

Code III. 22: Pie Chart of the proportion of each protocol.

* **proto & conn\_state:** **Stacked Bar chart showing the distribution of protocols by connection state**

counts\_by\_state = df\_modified.groupby(['conn\_state', 'proto']).size().unstack(fill\_value=0)

fig, ax = mpt.subplots(figsize=(10, 6))

counts\_by\_state.plot(kind='bar', stacked=True, ax=ax)

mpt.grid(axis='y')

mpt.title('Distribution of Network Protocols by Connection State')

mpt.xlabel('Connection State')

mpt.ylabel('Number of Connections')

mpt.show()

Code III. 23: Stacked Bar chart showing the distribution of protocols by connection state.

* **proto & conn\_state: Countplot the distribution of connection states by protocol**

sns.set(style="whitegrid")

mpt.figure(figsize=(12,6))

sns.countplot(x="proto", hue="conn\_state", data=df\_modified)

# Move the legend to the top right

mpt.legend(loc='upper right', bbox\_to\_anchor=(1.02, 1))

mpt.show()

Code III. 24: Countplot the distribution of connection states by protocol.

* **proto & service:** **Heatmap of the frequency of each combination of protocol and service.**

counts\_by\_proto\_service = df\_modified.groupby(['proto', 'service']).size().unstack(fill\_value=0)

mpt.figure(figsize=(12,6))

sns.heatmap(counts\_by\_proto\_service, cmap='Blues')

mpt.title('Frequency of Network Protocols and Services')

mpt.xlabel('Service')

mpt.ylabel('Protocol')

mpt.show()

Code III. 25: Heatmap of the frequency of each combination of protocol and service.

* **proto & type:** **Stacked Bar Plot to show the distribution of attack categories within each protocol type**

mpt.figure(figsize=(12, 6))

sns.countplot(x='proto', hue='type', data=df\_modified)

mpt.legend(loc='upper right', bbox\_to\_anchor=(1.02, 1))

mpt.xlabel('Protocol Type')

mpt.ylabel('Count')

mpt.show()

Code III. 26: Stacked Bar Plot to show the distribution of attack categories within each protocol type.

* + service Feature
* **service: Bar plot (****Countplot function) of the count or percentage of each service category in the dataset**

sns.set(style="darkgrid")

mpt.figure(figsize=(12,6))

sns.countplot(x="service", data=df\_modified)

Code III. 27: Count plot of the percentage of each service category in the dataset.

* service & proto : Barplot (Countplot) to show the count or percentage of the service categories for each protocol value

mpt.figure(figsize=(12,6))

#sns.set(style="darkgrid")

sns.countplot(x="proto", hue="service", data=df\_modified)

mpt.legend(loc='upper right', bbox\_to\_anchor=(1.02, 1))

Code III. 28: Count plot showing the percentage of the service categories for each protocol value.

* service & proto: Barplot (Countplot) to show the count or percentage of the protocol categories for each service value

mpt.figure(figsize=(12,6))

#sns.set(style="darkgrid")

sns.countplot(x="service", hue="proto", data=df\_modified)

mpt.legend(loc='upper right', bbox\_to\_anchor=(1.02, 1))

Code III. 29: Count plot showing the percentage of the protocol categories for each service value.

* service & type: stacked bar plot to show the distribution of attack categories within each service type

mpt.figure(figsize=(12, 6))

sns.countplot(x='service', hue='type', data=df\_modified)

mpt.legend(loc='upper right', bbox\_to\_anchor=(1.02, 1))

mpt.xlabel('Service Type')

mpt.ylabel('Count')

mpt.show()

Code III. 30: Stacked bar plot to show the distribution of attack categories within each service type.

* service & other features : Heatmap showing the correlation between service type and other features

corr = df\_modified.corr()

sns.heatmap(corr, xticklabels=corr.columns, yticklabels=['service'], cmap='coolwarm')

mpt.show()

Code III. 31: Heatmap showing the correlation between service type and other features.

* service & src\_pkts: Boxplot of src\_pkts grouped by the service variable. It can help to identify any potential differences in the distribution of packet counts across different services.

mpt.figure(figsize=(12,6))

sns.boxplot(x='service', y='src\_pkts', data=df\_modified)

mpt.xticks(rotation=90)

mpt.show()

Code III. 32: Boxplot of src\_pkts grouped by the service variable.

* service & src\_bytes: Boxplot of src\_bytes grouped by the service variable. It can help to identify any potential differences in the distribution of source bytes count across different services.

mpt.figure(figsize=(12,6))

sns.boxplot(x='service', y='src\_bytes', data=df\_modified)

mpt.xticks(rotation=90)

mpt.show()

Code III. 33: Boxplot of src\_bytes grouped by the service variable.

service & src\_pkts: Bar plot of src\_pkts sum grouped by the service variable

# Group by 'service' and calculate the sum of 'src\_pkts' for each service

packets\_counts = df\_modified.groupby('service')['src\_pkts'].sum()

# packets\_count

mpt.figure(figsize=(12,6))

mpt.bar(packets\_counts.index, packets\_counts.values)

mpt.title('Sum of Source Packets by service')

mpt.xlabel('Service')

mpt.ylabel('Source Packets Sum')

mpt.show()

Code III. 34: Bar plot of src\_pkts sum grouped by the service variable.

* + duration Feature
* duration: Histogram of the distribution of values for the duration feature in general and for the smallest values (Their count is very small)

# plot a histogram of the duration feature

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['duration'], bins=50)

mpt.title('Distribution of Duration')

mpt.xlabel('Duration (seconds)')

#mpt.ylim(0, 100)  # Set the y-axis limits according to our data

mpt.ylabel('Count')

mpt.grid(axis='x')

mpt.show()

Code III. 35: Histogram of the distribution of values for the duration feature.

# plot a histogram of the duration feature

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['duration'], bins=50)

mpt.title('Distribution of Duration')

mpt.xlabel('Duration (seconds)')

mpt.ylim(0, 90)  # Set the y-axis limits according to our data

mpt.ylabel('Count')

mpt.yticks(range(0,100,5))

mpt.grid(axis='y', c='blue')

mpt.show()

Code III. 36: Histogram of the distribution of values for the duration feature for the smallest values.

* duration: Boxplot of the distribution of values and any potential outliers

# plot a box plot of the duration feature

mpt.figure(figsize=(12,6))

sns.boxplot(x=df\_modified['duration'])

mpt.title('Box Plot of Duration')

mpt.xlabel('Duration (seconds)')

mpt.show()

Code III. 37: Boxplot of the distribution of values and any potential outliers.

* duration & src\_bytes: Scatter Plot

# plot a scatter plot of the duration feature against the source bytes feature

mpt.figure(figsize=(12,6))

mpt.scatter(df['duration'], df['src\_bytes'])

mpt.title('Duration vs Source Bytes')

mpt.xlabel('Duration (seconds)')

mpt.ylabel('Source Bytes')

mpt.show()

Code III. 38: Scatter Plot of the duration & src\_bytes.

* + src\_bytes Feature
* src\_bytes: General Histogram and another for the very small values

# Create a histogram of the src\_bytes feature

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['src\_bytes'], bins=50)

mpt.xlabel('src\_bytes')

mpt.ylabel('Frequency')

#mpt.ylim(0,20)

mpt.title('Histogram of src\_bytes')

mpt.show()

Code III. 39: General Histogram of src\_bytes.

# Create a histogram of the src\_bytes feature

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['src\_bytes'], bins=50)

mpt.xlabel('src\_bytes')

mpt.ylabel('Frequency')

mpt.ylim(0,20)

mpt.title('Histogram of src\_bytes')

mpt.show()

Code III. 40: Histogram of src\_bytes for the very smallest values.

* src\_bytes: Boxplot

# Create a boxplot of the src\_bytes feature

mpt.figure(figsize=(12,6))

sns.boxplot(x=df\_modified['src\_bytes'])

mpt.xlabel('src\_bytes')

mpt.title('Boxplot of src\_bytes')

mpt.show()

Code III. 41: Boxplot of src\_bytes.

* src\_bytes & dst\_bytes: Scatter Plot

# Create a scatter plot of src\_bytes vs dst\_bytes

mpt.figure(figsize=(12,6))

mpt.scatter(df\_modified['src\_bytes'], df\_modified['dst\_bytes'])

mpt.xlabel('src\_bytes')

mpt.ylabel('dst\_bytes')

mpt.title('Scatter plot of src\_bytes vs dst\_bytes')

mpt.show()

Code III. 42: Scatter Plot of src\_bytes and dst\_bytes.

* src\_bytes & type: Box plot of src\_bytes for each attack category

mpt.figure(figsize=(12,6))

sns.boxplot(x='type', y='src\_bytes', data=df\_modified)

mpt.xticks(rotation=45)

mpt.xlabel('Attack Category')

mpt.ylabel('Source Bytes')

mpt.show()

Code III. 43: Box plot of src\_bytes for each attack category.

* src\_bytes & proto: Box plot of src\_bytes for each protocol

mpt.figure(figsize=(12,6))

sns.boxplot(x='proto', y='src\_bytes', data=df\_modified)

mpt.xticks(rotation=45)

mpt.xlabel('Protocol')

mpt.ylabel('Source Bytes')

mpt.show()

Code III. 44: Box plot of src\_bytes for each protocol.

* + dst\_bytes Feature
* dst\_bytes: Histogram

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['dst\_bytes'], bins=100)

mpt.xlabel('dst\_bytes')

mpt.ylabel('Count')

# mpt.ylim(0,15)  #add to see the other counts (small  values)

mpt.show()

Code III. 45: Histogram of dst\_bytes.

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['dst\_bytes'], bins=100)

mpt.xlabel('dst\_bytes')

mpt.ylabel('Count')

mpt.ylim(0,15)  #add to see the other counts (small  values)

mpt.show()

Code III. 46: Histogram of dst\_bytes for the smallest values.

* dst\_bytes: Box plot

mpt.figure(figsize=(12,6))

sns.boxplot(x='dst\_bytes', data=df\_modified)

mpt.show()

Code III. 47: Box plot of dst\_bytes.

* dst\_bytes & proto: Bar plot of dst\_bytes average for each protocol

mpt.figure(figsize=(12,6))

df\_modified.groupby('proto')['dst\_bytes'].mean().plot(kind='bar')

mpt.ylabel('Average dst\_bytes')

#mpt.ylim(0,2000)  #add to see the udp and icmp dst\_bytes(small  value)

mpt.show()

Code III. 48: Bar plot of dst\_bytes average for each protocol.

mpt.figure(figsize=(12,6))

df\_modified.groupby('proto')['dst\_bytes'].mean().plot(kind='bar')

mpt.ylabel('Average dst\_bytes')

mpt.ylim(0,2000)  #add to see the udp and icmp dst\_bytes(small  value)

mpt.show()

Code III. 49: Bar plot of dst\_bytes average for each protocol for small values.

* dst\_bytes & src\_bytes & duration: Heatmap of the correlation between dst\_bytes and Those other numerical features in the dataset

mpt.figure(figsize=(12,6))

sns.heatmap(df\_modified[['duration', 'src\_bytes', 'dst\_bytes']].corr(), annot=True)

mpt.show()

Code III. 50: Heatmap of the correlation between dst\_bytes and other numerical features in the dataset.

* + conn\_state Feature
* conn\_state : Bar plot showing the frequency of each connection state

# Create a bar plot showing the frequency of each connection state

conn\_state\_counts = df['conn\_state'].value\_counts()

mpt.figure(figsize=(12,6))

mpt.bar(conn\_state\_counts.index, conn\_state\_counts.values)

mpt.xticks(rotation=45)

mpt.xlabel('Connection State')

mpt.ylabel('Frequency')

mpt.title('Frequency of Each Connection State')

conn\_list= conn\_state\_counts.tolist()

for i in range(13):

    mpt.text(i-0.3, conn\_list[i], conn\_list[i])

mpt.show()

Code III. 51: Bar plot showing the frequency of each connection state.

* conn\_state & type: Count plot the frequency of each connection state for each attack category

mpt.figure(figsize=(12,6))

sns.countplot(x='conn\_state', hue='type', data=df\_modified)

mpt.xticks(rotation=45)

mpt.xlabel('Connection State')

mpt.ylabel('Frequency')

mpt.title('Frequency of Each Connection State for Each Attack Category')

mpt.legend(loc='upper right', bbox\_to\_anchor=(1.02, 1))

mpt.show()

Code III. 52: Count plot the frequency of each connection state for each attack category.

* conn\_state & duration: Violin plot the distribution of the duration for each connection state

# Create a violin plot showing the distribution of the duration for each connection state

mpt.figure(figsize=(12,6))

sns.violinplot(x='conn\_state', y='duration', data=df\_modified)

mpt.xticks(rotation=45)

mpt.xlabel('Connection State')

mpt.ylabel('Duration')

mpt.title('Distribution of Duration for Each Connection State')

mpt.show()

Code III. 53: Violin plot the distribution of the duration for each connection state.

* + missed\_bytes Feature
* missed\_bytes: Histogram

mpt.figure(figsize=(12,6))

mpt.hist(data=df\_modified, x="missed\_bytes", )

mpt.title("Distribution of Missed Bytes")

mpt.xlabel("Missed Bytes")

mpt.ylabel("Count")

#mpt.ylim(0,15)     #to show the smallest values

mpt.show()

Code III. 54: Histogram of missed\_bytes.

mpt.figure(figsize=(12,6))

mpt.hist(data=df\_modified, x="missed\_bytes", )

mpt.title("Distribution of Missed Bytes")

mpt.xlabel("Missed Bytes")

mpt.ylabel("Count")

mpt.ylim(0,15)     #to show the smallest values

mpt.show()

Code III. 55: Histogram of missed\_bytes for the small values.

* missed\_bytes: Boxplot

mpt.figure(figsize=(12,6))

sns.boxplot(data=df\_modified, y="missed\_bytes")

mpt.title("Distribution of Missed Bytes")

mpt.ylabel("Missed Bytes")

mpt.show()

Code III. 56: Boxplot of missed\_bytes.

* missed\_bytes & duration: Scatter Plot

mpt.figure(figsize=(12,6))

sns.scatterplot(data=df\_modified, x="duration", y="missed\_bytes")

mpt.title("Missed Bytes vs. Duration")

mpt.xlabel("Duration (seconds)")

mpt.ylabel("Missed Bytes")

mpt.show()

Code III. 57: Scatter Plot of missed\_bytes and duration.

* 1. **The Statistical Activity Features Visualization**
  + src\_pkts Feature
* src\_pkts: Histogram

# Plot histogram

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['src\_pkts'], bins=50)

mpt.title('Distribution of Source Packets')

mpt.xlabel('Source Packets')

mpt.ylabel('Frequency')

#mpt.ylim(0,30)

mpt.show()

Code III. 58: Histogram of src\_pkts.

# Plot histogram

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['src\_pkts'], bins=50)

mpt.title('Distribution of Source Packets')

mpt.xlabel('Source Packets')

mpt.ylabel('Frequency')

mpt.ylim(0,30)

mpt.show()

Code III. 59: Histogram of src\_pkts for the small values.

src\_pkts & conn\_state: Box plot

# Plot box plot

mpt.figure(figsize=(12,6))

sns.boxplot(data=df\_modified, y='src\_pkts', x='conn\_state')

mpt.xticks(rotation=90)

mpt.title('Distribution of Source Packets by Connection State')

mpt.xlabel('Connection State')

mpt.ylabel('Source Packets')

mpt.show()

Code III. 60: Box plot of src\_pkts and conn\_state.

* src\_pkts & proto: Box plot

# Plot box plot

mpt.figure(figsize=(12,6))

sns.boxplot(data=df\_modified, y='src\_pkts', x='proto')

#mpt.xticks(rotation=90)

mpt.title('Distribution of Source Packets by Protocol')

mpt.xlabel('Protocol')

mpt.ylabel('Source Packets')

mpt.show()

Code III. 61: Box plot of src\_pkts and proto.

* src\_pkts & type : Box plot btw src\_pkts and attack category

# Plot box plot

mpt.figure(figsize=(12,6))

sns.boxplot(data=df\_modified, y='src\_pkts', x='type')

#mpt.xticks(rotation=90)

mpt.title('Distribution of Source Packets by attack category')

mpt.xlabel('Attack category')

mpt.ylabel('Source Packets')

mpt.show()

Code III. : Box plot of src\_pkts and attack category.

* src\_pkts : Box plot

mpt.figure(figsize=(12,6))

sns.boxplot(data=df\_modified, y="src\_pkts")

mpt.title("Distribution of Source Packets")

mpt.ylabel("Source packets")

mpt.show()

Code III. 63: Box plot of src\_pkts.

* + src\_ip\_bytes Feature
* src\_ip\_bytes: Histogram

# Plot histogram

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['src\_ip\_bytes'], bins=50)

mpt.title('Distribution of Source IP bytes')

mpt.xlabel('Source IP Bytes')

mpt.ylabel('Frequency')

#mpt.ylim(0,35)

mpt.show()

Code III. 64: Histogram of src\_ip\_bytes.

# Plot histogram

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['src\_ip\_bytes'], bins=50)

mpt.title('Distribution of Source IP bytes')

mpt.xlabel('Source IP Bytes')

mpt.ylabel('Frequency')

mpt.ylim(0,35)

mpt.show()

Code III. 65: Histogram of src\_ip\_bytes of the small values.

* src\_ip\_bytes: Box plot

mpt.figure(figsize=(12,6))

sns.boxplot(data=df\_modified, y="src\_ip\_bytes")

mpt.title("Distribution of Source IP Bytes")

mpt.ylabel("Source IP Bytes")

mpt.show()

Code III. 66: Box plot of src\_ip\_bytes.

* src\_ip\_bytes & type: Box plot

# Plot box plot

mpt.figure(figsize=(12,6))

sns.boxplot(data=df\_modified, y='src\_ip\_bytes', x='type')

#mpt.xticks(rotation=90)

mpt.title('Distribution of Source IP bytes by attack category')

mpt.xlabel('Attack category')

mpt.ylabel('Source IP bytes')

mpt.show()

Code III. 67: Box plot of src\_ip\_bytes and type.

* + dst\_pkts Feature
* dst\_pkts: Histogram

# Plot histogram

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['dst\_pkts'], bins=50)

mpt.title('Distribution of Destination Packets')

mpt.xlabel('Destination Packets')

mpt.ylabel('Frequency')

#mpt.ylim(0,40)

mpt.show()

Code III. 68: Histogram of dst\_pkts.

# Plot histogram

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['dst\_pkts'], bins=50)

mpt.title('Distribution of Destination Packets')

mpt.xlabel('Destination Packets')

mpt.ylabel('Frequency')

mpt.ylim(0,40)

mpt.show()

Code III. 69: Histogram of dst\_pkts for the small values.

dst\_pkts & conn\_state: Box plot

# Plot box plot

mpt.figure(figsize=(12,6))

sns.boxplot(data=df\_modified, y='dst\_pkts', x='conn\_state')

mpt.xticks(rotation=90)

mpt.title('Distribution of Destination Packets by Connection State')

mpt.xlabel('Connection State')

mpt.ylabel('Destination Packets')

mpt.show()

Code III. 70: Box plot of dst\_pkts and conn\_state.

* dst\_pkts & proto: Box plot

# Plot box plot

mpt.figure(figsize=(12,6))

sns.boxplot(data=df\_modified, y='dst\_pkts', x='proto')

#mpt.xticks(rotation=90)

mpt.title('Distribution of Destination Packets by Protocol')

mpt.xlabel('Protocol')

mpt.ylabel('Destination Packets')

mpt.show()

Code III. 71: Box plot of dst\_pkts and proto.

* dst\_pkts & type : Box plot btw dst\_pkts and attack category

# Plot box plot

mpt.figure(figsize=(12,6))

sns.boxplot(data=df\_modified, y='dst\_pkts', x='type')

#mpt.xticks(rotation=90)

mpt.title('Distribution of Destination Packets by attack category')

mpt.xlabel('Attack category')

mpt.ylabel('Destination Packets')

mpt.show()

Code III. 72: Box plot of dst\_pkts and type.

* dst\_pkts : Box plot

mpt.figure(figsize=(12,6))

sns.boxplot(data=df\_modified, y="dst\_pkts")

mpt.title("Distribution of Destination Packets")

mpt.ylabel("Destination packets")

mpt.show()

Code III. 73: Box plot of dst\_pkts.

* + dst\_ip\_bytes Feature
* dst\_ip\_bytes: Histogram

# Plot histogram

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['dst\_ip\_bytes'], bins=50)

mpt.title('Distribution of Destination IP bytes')

mpt.xlabel('Destination IP Bytes')

mpt.ylabel('Frequency')

#mpt.ylim(0,30)

mpt.show()

Code III. 74: Histogram of dst\_ip\_bytes.

# Plot histogram

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['dst\_ip\_bytes'], bins=50)

mpt.title('Distribution of Destination IP bytes')

mpt.xlabel('Destination IP Bytes')

mpt.ylabel('Frequency')

mpt.ylim(0,30)

mpt.show()

Code III. 75: Histogram of dst\_ip\_bytes for the small values.

* dst\_ip\_bytes: Box plot

mpt.figure(figsize=(12,6))

sns.boxplot(data=df\_modified, y="dst\_ip\_bytes")

mpt.title("Distribution of Destination IP Bytes")

mpt.ylabel("Destination IP Bytes")

mpt.show()

Code III. 76: Box plot of dst\_ip\_bytes.

* dst\_ip\_bytes & type: Box plot

# Plot box plot

mpt.figure(figsize=(12,6))

sns.boxplot(data=df\_modified, y='dst\_ip\_bytes', x='type')

mpt.title('Distribution of Destination IP bytes by attack category')

mpt.xlabel('Attack category')

mpt.ylabel('Destination IP bytes')

mpt.show()

Code III. 77: Box plot of dst\_ip\_bytes and type.

* 1. **The DNS Activity Features Visualization**
* dns\_query & dns\_rejected: Bar chart of the number of DNS queries accepted and those rejected

import random

# Group the data by dns\_query and dns\_rejected, and count the occurrences

dns\_counts = df\_modified.groupby(['dns\_query', 'dns\_rejected']).size().unstack().sample(n=20, random\_state=random.seed()) #each time different sample of data

dns\_counts

# Plot the stacked bar chart

dns\_counts.plot(kind='bar', stacked=True, figsize=(12, 6))

# Set the labels and title

mpt.xlabel('DNS Query')

mpt.ylabel('Count')

mpt.title('Stacked Bar Chart of DNS Rejected by DNS Query')

# Show the plot

mpt.show()

Code III. 78: Bar chart of the number of DNS queries accepted and those rejected.

* dns\_query & dns\_rejected: Bar chart of the number of the top 20 DNS queries that have been rejected

# Specify the condition for filtering

condition = "dns\_rejected == 'T'"

# Group the data by dns\_query and dns\_rejected, apply the condition, count the occurrences, and sample from sorted values

dns\_counts = df\_modified.query(condition).groupby(['dns\_query', 'dns\_rejected']).size().unstack().sort\_values(ascending=False, by='T').head(20)

# Plot the stacked bar chart

dns\_counts.plot(kind='bar', stacked=True, figsize=(12, 6))

# Set the labels and title

mpt.xlabel('DNS Query')

mpt.ylabel('Count')

mpt.title('Stacked Bar Chart of DNS Rejected by DNS Query')

# Show the plot

mpt.show()

Code III. 79: Bar chart of the number of the top 20 DNS queries that have been rejected.

* dns\_qtype: Bar chart of the distribution of DNS query types

dns\_qtype\_count = df\_modified['dns\_qtype'].value\_counts()

dns\_qtype\_count

# Plot a bar chart of the count of the DNS query type

mpt.figure(figsize=(12,6))

mpt.grid(axis='y')

#mpt.ylim(0,40000) # change it also for the range to 40000 and also for values smaller then that

mpt.xlabel('DNS Query Type')

mpt.ylabel('Number of Queries')

mpt.title('Number of Queries by each Query type')

mpt.show()

Code III. 80: Bar chart of the distribution of DNS query types.

dns\_qtype\_count = df\_modified['dns\_qtype'].value\_counts()

dns\_qtype\_count

# Plot a bar chart of the count of the DNS query type

mpt.figure(figsize=(12,6))

mpt.bar(dns\_qtype\_count.index, dns\_qtype\_count.values)

mpt.grid(axis='y')

#mpt.ylim(0,40000) # change it also for the range to 40000 and also for values smaller then that

mpt.xlim(0,50)

mpt.xlabel('DNS Query Type')

mpt.ylabel('Number of Queries')

mpt.title('Number of Queries by each Query type')

mpt.show()

Code III. 81: Bar chart of the distribution of DNS query types of a sample between 0 and 50.

dns\_qtype\_count = df\_modified['dns\_qtype'].value\_counts()

dns\_qtype\_count

# Plot a bar chart of the count of the DNS query type

mpt.figure(figsize=(12,6))

mpt.bar(dns\_qtype\_count.index, dns\_qtype\_count.values)

#mpt.xticks(rotation=90)

mpt.grid(axis='x')

mpt.ylim(0,40000) # change it also for the range to 40000

mpt.xlim(240,260)

mpt.xlabel('DNS Query Type')

mpt.ylabel('Number of Queries')

mpt.title('Number of Queries by each Query type')

mpt.show()

Code III. 82: Bar chart of the distribution of DNS query types for the small values.

* dns\_qclass: Histogram

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['dns\_qclass'], bins=50)

mpt.title('Distribution of DNS Query Classes')

mpt.xlabel('DNS Query Classes')

mpt.ylabel('Frequency')

mpt.show()

Code III. 83: Histogram of the distribution of dns\_qclass.

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['dns\_qclass'], bins=50)

mpt.title('Distribution of DNS Query Classes')

mpt.xlabel('DNS Query Classes')

mpt.ylabel('Frequency')

mpt.ylim(0,2500)

mpt.show()

Code III. 84: Histogram of the distribution of dns\_qclass for the small values.

* dns\_qtype : Histogram

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['dns\_qtype'], bins=50)

mpt.title('Distribution of DNS Query types')

mpt.xlabel('DNS Query types')

mpt.ylabel('Frequency')

mpt.show()

Code III. 85: Histogram of the distribution of dns\_qtype.

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['dns\_qtype'], bins=50)

mpt.title('Distribution of DNS Query types')

mpt.xlabel('DNS Query types')

mpt.ylabel('Frequency')

mpt.ylim(0,30000)

mpt.show()

Code III. 86: Histogram of the distribution of dns\_qtype for the small values.

* dns\_rcode : Histogram

mpt.figure(figsize=(12,6))

mpt.hist(df\_modified['dns\_rcode'])

mpt.title('Distribution of DNS Query Response Code')

mpt.xlabel('DNS Query Response Code')

mpt.ylabel('Frequency')

mpt.show()

Code III. 87: Histogram of the distribution of dns\_rcode.

* dns\_rejected: Bar plot of the distribution of the NON using dns, dns accepted and those rejected queries

dns\_query\_count=df\_modified['dns\_rejected'].value\_counts()

dns\_query\_count

mpt.figure(figsize=(12,6))

mpt.bar(dns\_query\_count.index,dns\_query\_count.values)

mpt.xlabel('DNS Query Rejected by the server ( -, F: Not Rejected, T:Rejected)')

mpt.ylabel('Count')

mpt.title('Bar Chart of the DNS Rejected/Accepted/NotUsing by the server Queries')

dns\_query\_count\_list= dns\_query\_count.tolist()

for i in range(3):

    mpt.text(i-0.1, dns\_query\_count\_list[i]+1000,dns\_query\_count\_list[i])

Code III. 88: Bar plot of the distribution of the NON using dns, dns accepted and those rejected queries.

* dns\_AA: Bar chart

dns\_AA=df\_modified['dns\_AA'].value\_counts()

mpt.figure(figsize=(12,6))

mpt.bar(dns\_AA.index,dns\_AA.values)

mpt.xlabel('DNS Authoritative answers , where T denotes server is authoritative for query')

mpt.ylabel('Count')

mpt.title(' DNS Authoritative answers')

dns\_AA\_list= dns\_AA.tolist()

for i in range(3):

    mpt.text(i-0.1, dns\_AA\_list[i]+1000,dns\_AA\_list[i])

Code III. 89: Bar chart of dns\_AA.

* dns\_RD: Bar Chart

dns\_RD=df\_modified['dns\_RD'].value\_counts()

dns\_RD

mpt.figure(figsize=(12,6))

mpt.bar(dns\_RD.index,dns\_RD.values)

mpt.xlabel('DNS Recursion desired, where T denotes request recursive lookup of query')

mpt.ylabel('Count')

mpt.title(' DNS Recursion desired')

dns\_RD\_list= dns\_RD.tolist()

for i in range(3):

    mpt.text(i-0.1, dns\_RD\_list[i]+1000,dns\_RD\_list[i])

Code III. 90: Bar chart of dns\_RD.

* dns\_RA: Bar Chart

dns\_RA=df\_modified['dns\_RA'].value\_counts()

dns\_RA

mpt.figure(figsize=(12,6))

mpt.bar(dns\_RA.index,dns\_RA.values)

mpt.xlabel('DNS Recursion available, where T denotes server supports recursive queries')

mpt.ylabel('Count')

mpt.title(' DNS Recursion available')

dns\_RA\_list= dns\_RA.tolist()

for i in range(3):

    mpt.text(i-0.1, dns\_RA\_list[i]+1000,dns\_RA\_list[i])

Code III. 91: Bar chart of dns\_RA.

* dns\_query and duration: Scatter Plot of 400 samples randomly

# Create the scatter plot

fig, ax = mpt.subplots(figsize=(12,6))

ax.scatter(df\_modified['dns\_query'].sample(n=400), df\_modified['duration'].sample(n=400))

mpt.xticks(rotation=90)

ax.set\_xlabel('DNS Query')

ax.set\_ylabel('Duration')

ax.set\_title('Relationship Between DNS Query and Connection Duration')

mpt.show()

Code III. 92: Scatter Plot of 400 samples randomly of dns\_query and duration.

* dns\_rejected & proto: Stacked Bar Chart showing the number of DNS rejected for each protocol

# Group the data by proto and dns\_rejected, and count the occurrences

proto\_counts = df\_modified.groupby(['proto', 'dns\_rejected']).size().unstack()

proto\_counts

# Plot the stacked bar chart

proto\_counts.plot(kind='bar', stacked=True, figsize=(12, 6))

mpt.grid(axis='y')

mpt.xticks(rotation=0)

# Set the labels and title

mpt.xlabel('Protocol')

mpt.ylabel('Number of DNS queries rejected or not')

mpt.title('Number of DNS Queries rejected by Protocol')

# Show the plot

mpt.show()

Code III. 93: Stacked Bar Chart showing the number of DNS rejected for each protocol.

* 1. **The SSL Activity Features Visualization**
* **ssl\_version: Bar chart**

ssl\_version\_count=df\_modified['ssl\_version'].value\_counts()

ssl\_version\_count

mpt.figure(figsize=(12,6))

mpt.bar(ssl\_version\_count.index,ssl\_version\_count.values)

#mpt.ylim(0,1000)# show the smallest versions

mpt.xlabel('SSL Version')

mpt.ylabel('Frequency')

mpt.title(' SSL Version Frequency')

ssl\_version\_list= ssl\_version\_count.tolist()

for i in range(4):

    mpt.text(i-0.1, ssl\_version\_list[i]+1000,ssl\_version\_list[i])

Code III. 94: Bar Chart of ssl\_version.

* **ssl\_cipher: Bar chart**

ssl\_cipher\_count=df\_modified['ssl\_cipher'].value\_counts()

ssl\_cipher\_count

mpt.figure(figsize=(12,6))

mpt.bar(ssl\_cipher\_count.index,ssl\_cipher\_count.values)

mpt.xticks(rotation=90)

#mpt.ylim(0,1000)# show the smallest versions

mpt.xlabel('SSL Cipher')

mpt.ylabel('Frequency')

mpt.title(' SSL Cipher Frequency')

ssl\_cipher\_list= ssl\_cipher\_count.tolist()

for i in range(6):

    mpt.text(i-0.1, ssl\_cipher\_list[i]+1000,ssl\_cipher\_list[i])

Code III. 95: Bar Chart of ssl\_cipher.

* **ssl\_resumed: Bar chart**

ssl\_resumed\_count=df\_modified['ssl\_resumed'].value\_counts()

ssl\_resumed\_count

mpt.figure(figsize=(12,6))

mpt.bar(ssl\_resumed\_count.index,ssl\_resumed\_count.values)

#mpt.ylim(0,1000)# show the smallest versions

mpt.xlabel('SSL Resumed')

mpt.ylabel('Frequency')

mpt.title(' SSL Resumed Frequency')

ssl\_resumed\_list= ssl\_resumed\_count.tolist()

for i in range(3):

    mpt.text(i-0.1, ssl\_resumed\_list[i]+1000,ssl\_resumed\_list[i])

Code III. 96: Bar Chart of ssl\_resumed.

* **ssl\_established: Bar chart**

ssl\_established\_count=df\_modified['ssl\_established'].value\_counts()

ssl\_established\_count

mpt.figure(figsize=(12,6))

mpt.bar(ssl\_established\_count.index,ssl\_established\_count.values)

#mpt.ylim(0,1000)# show the smallest versions

mpt.xlabel('SSL Established')

mpt.ylabel('Frequency')

mpt.title(' SSL Established Frequency')

ssl\_established\_list= ssl\_established\_count.tolist()

for i in range(3):

    mpt.text(i-0.1, ssl\_established\_list[i]+1000,ssl\_established\_list[i])

Code III. 97: Bar Chart of ssl\_established.

* **ssl\_subject: Bar chart**

ssl\_subject\_count=df\_modified['ssl\_subject'].value\_counts()

ssl\_subject\_count

mpt.figure(figsize=(12,6))

mpt.bar(ssl\_subject\_count.index,ssl\_subject\_count.values)

mpt.xticks(rotation=90)

mpt.xlabel('SSL Subject')

mpt.ylabel('Frequency')

mpt.title(' SSL Subject Frequency')

ssl\_subject\_list= ssl\_subject\_count.tolist()

for i in range(6):

    mpt.text(i-0.1, ssl\_subject\_list[i]+1000,ssl\_subject\_list[i])

Code III. 98: Bar Chart of ssl\_subject.

* **ssl\_issuer: Bar chart**

ssl\_issuer\_count=df\_modified['ssl\_issuer'].value\_counts()

ssl\_issuer\_count

mpt.figure(figsize=(12,6))

mpt.bar(ssl\_issuer\_count.index,ssl\_issuer\_count.values)

mpt.xticks(rotation=90)

mpt.xlabel('SSL Issuer')

mpt.ylabel('Frequency')

mpt.title(' SSL Issuer Frequency')

ssl\_issuer\_list= ssl\_issuer\_count.tolist()

for i in range(5):

    mpt.text(i-0.1, ssl\_issuer\_list[i]+1000,ssl\_issuer\_list[i])

Code III. 99: Bar Chart of ssl\_issuer.

* ssl\_cipher & proto: Stacked bar chart

# Group the data by ssl\_cipher and proto, and count the occurrences

cipher\_counts = df\_modified.groupby(['ssl\_cipher', 'proto']).size().unstack()

cipher\_counts

# Plot the stacked bar chart

cipher\_counts.plot(kind='bar', stacked=True, figsize=(12, 6))

mpt.grid(axis='y')

# Set the labels and title

mpt.xlabel('SSL Cipher')

mpt.ylabel('Protocol Frequency')

mpt.title('Protocol Frequency  by SSL Cipher ')

# Show the plot

mpt.show()

Code III. 100: Stacked bar chart of ssl\_cipher and proto.

* ssl\_resumed & proto: Stacked bar chart

# Group the data by ssl\_cipher and proto, and count the occurrences

resumed\_counts = df\_modified.groupby(['ssl\_resumed', 'proto']).size().unstack()

resumed\_counts

# Plot the stacked bar chart

resumed\_counts.plot(kind='bar', stacked=True, figsize=(12, 6))

mpt.grid(axis='y')

mpt.xticks(rotation=0)

# Set the labels and title

mpt.xlabel('SSL Resumed')

mpt.ylabel('Protocol Frequency')

mpt.title('Protocol Frequency  by SSL Resumed')

# Show the plot

mpt.show()

Code III. 101: Stacked bar chart of ssl\_resumed and proto.

* ssl\_established & proto: Stacked bar chart

# Group the data by ssl\_cipher and proto, and count the occurrences

established\_counts = df\_modified.groupby(['ssl\_established', 'proto']).size().unstack()

established\_counts

# Plot the stacked bar chart

established\_counts.plot(kind='bar', stacked=True, figsize=(12, 6))

mpt.grid(axis='y')

mpt.xticks(rotation=0)

# Set the labels and title

mpt.xlabel('SSL Established')

mpt.ylabel('Protocol Frequency')

mpt.title('Protocol Frequency  by SSL Established')

# Show the plot

mpt.show()

Code III. 102: Stacked bar chart of ssl\_established and proto.

* 1. **The HTTP Activity Features Visualization**
* **http\_method: Bar chart**

http\_method\_count=df\_modified['http\_method'].value\_counts()

http\_method\_count

mpt.figure(figsize=(12,6))

mpt.bar(http\_method\_count.index,http\_method\_count.values)

mpt.xlabel('HTTP Method')

mpt.ylabel('Frequency')

mpt.title(' HTTP Method Frequency')

http\_method\_list= http\_method\_count.tolist()

for i in range(4):

    mpt.text(i-0.1, http\_method\_list[i]+1000,http\_method\_list[i])

Code III. 103: Bar chart of http\_method.

* http\_orig\_mime\_types: Bar chart

http\_orig\_mime\_count=df\_modified['http\_orig\_mime\_types'].value\_counts()

http\_orig\_mime\_count

mpt.figure(figsize=(12,6))

mpt.bar(http\_orig\_mime\_count.index,http\_orig\_mime\_count.values)

#mpt.ylim(0,1000)# show the smallest versions

mpt.xlabel('HTTP mime types from source system')

mpt.ylabel('Frequency')

mpt.title(' HTTP mime types from source system Frequency')

http\_orig\_mime\_list= http\_orig\_mime\_count.tolist()

for i in range(3):

    mpt.text(i-0.1, http\_orig\_mime\_list[i]+1000,http\_orig\_mime\_list[i])

Code III. 104: Bar chart of http\_orig\_mime\_types.

* http\_resp\_mime\_types: Bar chart

http\_resp\_mime\_count=df\_modified['http\_resp\_mime\_types'].value\_counts()

http\_resp\_mime\_count

mpt.figure(figsize=(12,6))

mpt.bar(http\_resp\_mime\_count.index,http\_resp\_mime\_count.values)

mpt.xticks(rotation=90)

#mpt.ylim(0,1000)# show the smallest versions

mpt.xlabel('HTTP mime types from destination system')

mpt.ylabel('Frequency')

mpt.title(' HTTP mime types from destination system Frequency')

http\_resp\_mime\_list= http\_resp\_mime\_count.tolist()

for i in range(10):

    mpt.text(i-0.1, http\_resp\_mime\_list[i]+1000,http\_resp\_mime\_list[i])

Code III. 105: Bar chart of http\_resp\_mime\_types.

* http\_trans\_depth & http\_version: Stacked Bar Plot

# Group the data by proto and dns\_rejected, and count the occurrences

http\_version\_counts = df\_modified.groupby(['http\_version', 'http\_trans\_depth']).size().unstack()

http\_version\_counts

# Plot the stacked bar chart

http\_version\_counts.plot(kind='bar', stacked=True, figsize=(12, 6))

mpt.grid(axis='y')

mpt.xticks(rotation=0)

#mpt.ylim(0,250)  # to add

# Set the labels and title

mpt.xlabel('HTTP Versions')

mpt.ylabel('Pipelined Depth into the HTTP connection')

mpt.title('Pipelined Depth into the HTTP connection of each HTTP Version')

# Show the plot

mpt.show()

Code III. 106: Stacked Bar chart of http\_trans\_depth & http\_version.

# Group the data by proto and dns\_rejected, and count the occurrences

http\_version\_counts = df\_modified.groupby(['http\_version', 'http\_trans\_depth']).size().unstack()

# Plot the stacked bar chart

http\_version\_counts.plot(kind='bar', stacked=True, figsize=(12, 6))

mpt.grid(axis='y')

mpt.xticks(rotation=0)

mpt.ylim(0,250)  # to add

# Set the labels and title

mpt.xlabel('HTTP Versions')

mpt.ylabel('Pipelined Depth into the HTTP connection')

mpt.title('Pipelined Depth into the HTTP connection of each HTTP Version')

# Show the plot

mpt.show()

Code III. 107: Stacked Bar chart of http\_trans\_depth & http\_version for smallest values.

* http\_trans\_depth & http\_version: Bar plot of the sum of http\_trans\_depth for each HTTP version

df\_test['http\_trans\_depth'] = pd.to\_numeric(df\_test['http\_trans\_depth'], errors='coerce')   #df\_test is the same as df\_modified

# Group the data by http\_version and calculate the sum of http\_trans\_depth

http\_data = df\_test[df\_test['service']=='http'].groupby('http\_version')['http\_trans\_depth'].sum()

http\_data

# Create a bar plot

fig, ax = mpt.subplots(figsize=(12, 6))

http\_data.plot(kind='bar', ax=ax)

mpt.xticks(rotation=0)

# Set the labels and title

ax.set\_xlabel('HTTP Version')

ax.set\_ylabel('Sum of HTTP Transaction Depth')

ax.set\_title('Sum of HTTP Transaction Depth by HTTP Version')

http\_data\_list= http\_data.tolist()

for i in range(2):

    mpt.text(i-0.1, http\_data\_list[i],http\_data\_list[i])

# Show the plot

mpt.show()

Code III. 108: Bar plot of the sum of http\_trans\_depth for each HTTP version.

http\_uri: Bar Chart of the 50/74 most URIs used in the HTTP request

http\_uri\_count=df\_modified['http\_uri'].value\_counts(ascending=False).head(50)

http\_uri\_count

mpt.figure(figsize=(12,6))

mpt.bar(http\_uri\_count.index,http\_uri\_count.values)

mpt.xticks(rotation=90)

#mpt.ylim(0,1000)# show the smallest versions

mpt.xlabel('HTTP URI')

mpt.ylabel('Frequency')

mpt.title(' HTTP URI Frequency')

http\_uri\_list= http\_uri\_count.tolist()

for i in range(50):

    mpt.text(i-0.5, http\_uri\_list[i],http\_uri\_list[i])

Code III. 109: Bar Chart of the 50/74 most URIs used in the HTTP request.

* http\_status\_code: Scatter of the frequency of each http status code

http\_status\_count=df\_modified['http\_status\_code'].value\_counts(ascending=False)

http\_status\_count

mpt.figure(figsize=(12,6))

mpt.scatter(http\_status\_count.index,http\_status\_count.values, s=15)

mpt.xticks(rotation=90)

#mpt.ylim(0,70)

mpt.xlabel('HTTP URI')

mpt.ylabel('Frequency')

mpt.title(' HTTP URI Frequency')

Code III. 110: Scatter of the frequency of each http status code.

* http\_user\_agent: Bar chart of the frequency of each http user agent

http\_agent\_count=df\_modified['http\_user\_agent'].value\_counts(ascending=False)

http\_agent\_count

mpt.figure(figsize=(12,6))

mpt.bar(http\_agent\_count.index,http\_agent\_count.values)

mpt.xticks(rotation=90)

#mpt.ylim(0,70) # show the smallest versions

mpt.xlabel('HTTP User Agent ')

mpt.ylabel('Frequency')

mpt.title(' HTTP User Agent Frequency')

Code III. 111: Bar chart of the frequency of each http user agent.

* http\_request\_body\_len: scatter of frequency

http\_request\_count=df\_modified['http\_request\_body\_len'].value\_counts(ascending=False)

http\_request\_count

mpt.figure(figsize=(12,6))

mpt.scatter(http\_request\_count.index,http\_request\_count.values, color='black',s=10)

#mpt.ylim(0,70) # show the smallest versions

mpt.xlabel('HTTP Request Body Length')

mpt.ylabel('Frequency')

mpt.title(' HTTP Request Body Length Frequency')

Code III. 112: scatter of frequency http\_request\_body\_len.

* http\_response\_body\_len: scatter of frequency

http\_response\_count=df\_modified['http\_response\_body\_len'].value\_counts(ascending=False)

http\_response\_count

mpt.figure(figsize=(12,6))

mpt.scatter(http\_response\_count.index,http\_response\_count.values, color='black',s=10)

#mpt.ylim(0,70) # show the smallest versions

mpt.xlabel('HTTP Response Body Length')

mpt.ylabel('Frequency')

mpt.title(' HTTP Response Body Length Frequency')

Code III. 113: scatter of frequency http\_response\_body\_len.

* http\_method & http\_status\_code: Stacked bar plot

# Group the data by proto and dns\_rejected, and count the occurrences

http\_method\_counts = df\_modified.groupby(['http\_method', 'http\_status\_code']).size().unstack()

# Plot the stacked bar chart

http\_method\_counts.plot(kind='bar', stacked=True, figsize=(12, 6))

mpt.grid(axis='y')

mpt.xticks(rotation=0)

#mpt.ylim(0,250)  # to add

# Set the labels and title

mpt.xlabel('HTTP Methods')

mpt.ylabel('HTTP Status Codes')

mpt.title('HTTP Status Codes of each HTTP Method')

mpt.show()

Code III. 114: Stacked bar plot of http\_method & http\_status\_code.

* 1. **The Violation Activity Features Visualization**
* **weird\_name: Bar chart**

weird\_name\_count=df\_modified['weird\_name'].value\_counts()

weird\_name\_count

mpt.figure(figsize=(12,6))

mpt.bar(weird\_name\_count.index,weird\_name\_count.values)

mpt.xticks(rotation=90)

#mpt.ylim(0,800)# show the smallest versions

mpt.xlabel('Weird Name')

mpt.ylabel('Frequency')

mpt.title(' Weird Name Frequency')

weird\_name\_list= weird\_name\_count.tolist()

for i in range(12):

    mpt.text(i-0.2, weird\_name\_list[i], weird\_name\_list[i])

Code III. 115: Bar plot showing the distribution of weird\_name.

* weird\_addl: Bar chart

weird\_addl\_count=df\_modified['weird\_addl'].value\_counts()

weird\_addl\_count

mpt.figure(figsize=(12,6))

mpt.bar(weird\_addl\_count.index,weird\_addl\_count.values)

#mpt.ylim(0,800) # show the smallest values

mpt.xlabel('Weird Additional Information ')

mpt.ylabel('Frequency')

mpt.title(' Weird Additional Information Frequency')

weird\_addl\_list= weird\_addl\_count.tolist()

for i in range(4):

    mpt.text(i-0.1, weird\_addl\_list[i], weird\_addl\_list[i])

Code III. 116: Bar plot showing the distribution of weird\_addl.

* weird\_notice: Bar Chart

weird\_notice\_count=df\_modified['weird\_notice'].value\_counts()

weird\_notice\_count

mpt.figure(figsize=(12,6))

mpt.bar(weird\_notice\_count.index,weird\_notice\_count.values)

mpt.xlabel('Weird Notice ')

mpt.ylabel('Frequency')

mpt.title(' Weird  Notice Frequency')

weird\_notice\_list= weird\_notice\_count.tolist()

for i in range(2):

    mpt.text(i-0.1, weird\_notice\_list[i], weird\_notice\_list[i])

Code III. 117: Bar plot showing the distribution of weird\_notice.

* 1. **The Data Labelling Features Visualization**
* **label: Bar chart**

label\_count=df\_modified['label'].value\_counts()

mpt.figure(figsize=(12,6))

mpt.bar(label\_count.index,label\_count.values)

mpt.xlabel('Attack Label ')

mpt.ylabel('Frequency')

mpt.title(' Attack Label Frequency')

label\_list= label\_count.tolist()

for i in range(2):

    mpt.text(i-0.1, label\_list[i], label\_list[i])

Code III. 118: Bar chart showing the distribution of label.

* type: Bar Chart

type\_count=df\_modified['type'].value\_counts()

type\_count

mpt.figure(figsize=(12,6))

mpt.bar(type\_count.index,type\_count.values)

mpt.xlabel('Attack Category ')

mpt.ylabel('Frequency')

mpt.title(' Attack Category Frequency')

type\_list= type\_count.tolist()

for i in range(10):

    mpt.text(i-0.3, type\_list[i], type\_list[i])

Code III. 119: Bar chart showing the distribution of attack types.

* type & missed\_bytes: Bar chart of the missed\_bytes for each attack type

type\_count\_sorted=df\_modified.groupby('type')['missed\_bytes'].sum().sort\_values(ascending=False)

type\_count\_sorted

mpt.figure(figsize=(12,6))

mpt.bar(type\_count\_sorted.index,type\_count\_sorted.values)

for i in range(10):

    mpt.text(i-0.5,type\_count\_sorted.values[i],type\_count\_sorted.values[i], c='blue')

mpt.xlabel('Attack Category ')

mpt.ylabel('Sum of Missed Bytes')

mpt.title('Sum of Missed Bytes by Attack Category')

Code III. 120: Bar chart of the missed\_bytes for each attack type.

* type & service: Stacked bar plot to show the distribution of service types within each attack category

mpt.figure(figsize=(12, 6))

sns.countplot(x='type', hue='service', data=df\_modified)

mpt.legend(loc='upper right', bbox\_to\_anchor=(1.02, 1))

mpt.xlabel('Attack Category')

mpt.ylabel('Count')

mpt.show()

Code III. 121: Stacked bar plot to show the distribution of service types within each attack category.

* type & proto: Stacked bar plot to show the distribution of protocol types within each attack category

mpt.figure(figsize=(12, 6))

sns.countplot(x='type', hue='proto', data=df\_modified)

mpt.legend(loc='upper right', bbox\_to\_anchor=(1.02, 1))

mpt.xlabel('Attack Category')

mpt.ylabel('Count')

mpt.show()

Code III. 122: Stacked bar plot to show the distribution of protocol types within each attack category.

*Appendix B*

The application of ML Models Code

* Categorical Features Treatment

selected\_rows\_normal = df[df['type'] == 'normal']

# Use only a portion of the train\_test\_df\_normal for demonstration purposes

subset\_size = 0.1   # 10% of the normal flow from the original dataframe

subset\_df = selected\_rows\_normal.sample(frac=subset\_size, random\_state=42)

selected\_rows\_ddos = df[df['type'] == 'ddos']

selected\_rows\_ransomware = df[df['type'] == 'ransomware']

selected\_rows\_injection = df[df['type'] == 'injection']

subset\_df = pd.concat([subset\_df, selected\_rows\_ddos])

subset\_df = pd.concat([subset\_df, selected\_rows\_ransomware])

subset\_df = pd.concat([subset\_df, selected\_rows\_injection])

# To keep only the specified features from our dataframe df

selected\_features = ['ts', 'src\_ip', 'src\_port', 'dst\_ip', 'dst\_port', 'proto', 'service','duration', 'conn\_state', 'missed\_bytes', 'src\_pkts', 'dst\_pkts','dns\_rcode', 'dns\_rejected', 'ssl\_resumed', 'ssl\_established', 'http\_status\_code', 'http\_user\_agent', 'http\_method', 'weird\_name', 'label', 'type']

# Filter the dataframe to keep only the selected features

df\_selected = subset\_df[selected\_features]

# Encoding 'src\_ip' which is the column containing the Source IP addresses

# Perform one-hot encoding

one\_hot\_encoded = pd.get\_dummies(df\_selected['src\_ip'], prefix='src\_ip')

# Concatenate the encoded columns with the original dataframe

df\_encoded = pd.concat([df\_selected, one\_hot\_encoded], axis=1)

# Remove the original 'src\_ip' column if desired

df\_encoded.drop('src\_ip', axis=1, inplace=True)

# Encoding 'dst\_ip' which is the column containing the Destination IP addresses

# Perform one-hot encoding

one\_hot\_encoded\_dst\_ip = pd.get\_dummies(df\_selected['dst\_ip'], prefix='dst\_ip')

# Concatenate the encoded columns with the original dataframe

df\_encoded = pd.concat([df\_encoded, one\_hot\_encoded\_dst\_ip], axis=1)

# Remove the original 'dst\_ip' column if desired

df\_encoded.drop('dst\_ip', axis=1, inplace=True)

# Encoding 'proto'

# Perform one-hot encoding

one\_hot\_encoded\_proto = pd.get\_dummies(df\_selected['proto'], prefix='proto')

# Concatenate the encoded columns with the original dataframe

df\_encoded = pd.concat([df\_encoded, one\_hot\_encoded\_proto], axis=1)

# Remove the original 'src\_ip' column if desired

df\_encoded.drop('proto', axis=1, inplace=True)

# Encoding 'service'

# Perform one-hot encoding

one\_hot\_encoded\_service = pd.get\_dummies(df\_selected['service'], prefix='service')

# Concatenate the encoded columns with the original dataframe

df\_encoded = pd.concat([df\_encoded, one\_hot\_encoded\_service], axis=1)

df\_encoded.drop('service', axis=1, inplace=True)

# Encoding 'conn\_state'

# Perform one-hot encoding

one\_hot\_encoded\_conn\_state = pd.get\_dummies(df\_selected['conn\_state'], prefix='conn\_state')

# Concatenate the encoded columns with the original dataframe

df\_encoded = pd.concat([df\_encoded, one\_hot\_encoded\_conn\_state], axis=1)

df\_encoded.drop('conn\_state', axis=1, inplace=True)

# Encoding 'http\_method'

# Perform one-hot encoding

one\_hot\_encoded\_http\_method = pd.get\_dummies(df\_selected['http\_method'], prefix='http\_method')

# Concatenate the encoded columns with the original dataframe

df\_encoded = pd.concat([df\_encoded, one\_hot\_encoded\_http\_method], axis=1)

df\_encoded.drop('http\_method', axis=1, inplace=True)

# Encoding 'http\_user\_agent'

# Perform one-hot encoding

one\_hot\_encoded\_http\_user\_agent = pd.get\_dummies(df\_selected['http\_user\_agent'], prefix='http\_user\_agent')

# Concatenate the encoded columns with the original dataframe

df\_encoded = pd.concat([df\_encoded, one\_hot\_encoded\_http\_user\_agent], axis=1)

df\_encoded.drop('http\_user\_agent', axis=1, inplace=True)

# Encoding 'weird\_name'

# Perform one-hot encoding

one\_hot\_encoded\_weird\_name = pd.get\_dummies(df\_selected['weird\_name'], prefix='weird\_name')

# Concatenate the encoded columns with the original dataframe

df\_encoded = pd.concat([df\_encoded, one\_hot\_encoded\_weird\_name], axis=1)

df\_encoded.drop('weird\_name', axis=1, inplace=True)

from sklearn.preprocessing import LabelEncoder

# Label Encoding of the type

label\_encoder = LabelEncoder()

df\_encoded['type'] = label\_encoder.fit\_transform(df\_encoded['type'])

Code IV. 1: Data Preprocessing for all the Categorical Features.

* Numerical Features Treatment

from sklearn.preprocessing import MinMaxScaler

from scipy.stats.mstats import winsorize

# Apply winsorization to handle outliers

df\_encoded['duration'] = winsorize(df\_encoded['duration'], limits=[0.05,0.05])

# Create an instance of MinMaxScaler

scaler = MinMaxScaler()

# Reshape the feature values into a 2D array

duration\_values = df\_selected['duration'].values.reshape(-1, 1)

# Normalize and scale the values

normalized\_duration = scaler.fit\_transform(duration\_values)

normalized\_duration

# Update the original dataframe with the normalized values

df\_encoded['duration'] = normalized\_duration

# Apply winsorization to handle outliers

df\_encoded['missed\_bytes'] = winsorize(df\_encoded['missed\_bytes'], limits=[0.05, 0.05])

# Create an instance of MinMaxScaler

scaler = MinMaxScaler()

# Reshape the feature values into a 2D array

missed\_bytes\_values = df\_selected['missed\_bytes'].values.reshape(-1, 1)

# Normalize and scale the values

normalized\_missed\_bytes = scaler.fit\_transform(missed\_bytes\_values)

# Update the original dataframe with the normalized values

df\_encoded['missed\_bytes'] = normalized\_missed\_bytes

# Apply winsorization to handle outliers

df\_encoded['src\_pkts'] = winsorize(df\_encoded['src\_pkts'], limits=[0.05, 0.05])

# Create an instance of MinMaxScaler

scaler = MinMaxScaler()

# Reshape the feature values into a 2D array

src\_pkts\_values = df\_selected['src\_pkts'].values.reshape(-1, 1)

# Normalize and scale the values

normalized\_src\_pkts = scaler.fit\_transform(src\_pkts\_values)

# Update the original dataframe with the normalized values

df\_encoded['src\_pkts'] = normalized\_src\_pkts

# Apply winsorization to handle outliers

df\_encoded['dst\_pkts'] = winsorize(df\_encoded['dst\_pkts'], limits=[0.05, 0.05])

# Create an instance of MinMaxScaler

scaler = MinMaxScaler()

# Reshape the feature values into a 2D array

dst\_pkts\_values = df\_selected['dst\_pkts'].values.reshape(-1, 1)

# Normalize and scale the values

normalized\_dst\_pkts = scaler.fit\_transform(dst\_pkts\_values)

# Update the original dataframe with the normalized values

df\_encoded['dst\_pkts'] = normalized\_dst\_pkts

Code IV. 2: Data Preprocessing for all the Numerical Features.

* Boolean Features Treatment

# Perform binary encoding

df\_encoded['dns\_rejected'] = df\_encoded['dns\_rejected'].map({'-': 0, 'T': 1, 'F': 0}).astype(int)

# Perform binary encoding

df\_encoded['ssl\_resumed'] = df\_encoded['ssl\_resumed'].map({'-': 0, 'T': 1, 'F': 0}).astype(int)

# Perform binary encoding

df\_encoded['ssl\_established'] = df\_encoded['ssl\_established'].map({'-': 0, 'T': 1, 'F': 0}).astype(int)

Code IV. 3: Data Preprocessing for all the Boolean Features.

* Timestamp Treatment

# Timestamp transformation

df\_encoded['ts'] = pd.to\_datetime(df\_encoded['ts'], unit='s')

# Extracting useful information from the timestamp

df\_encoded['year'] = df\_encoded['ts'].dt.year

df\_encoded['month'] = df\_encoded['ts'].dt.month

df\_encoded['day'] = df\_encoded['ts'].dt.day

df\_encoded['hour'] = df\_encoded['ts'].dt.hour

df\_encoded['minute'] = df\_encoded['ts'].dt.minute

df\_encoded['second'] = df\_encoded['ts'].dt.second

df\_encoded = df\_encoded.drop('ts', axis=1)

Code IV. 4: Data Preprocessing for the Timestamp Feature.

* Mapping of labels to encoded values of the type feature

# Get the mapping of labels to encoded values

label\_mapping = {label: encoded\_value for label, encoded\_value in zip(label\_encoder.classes\_, label\_encoder.transform(label\_encoder.classes\_))}

# Print the mapping

for label, encoded\_value in label\_mapping.items():

    print(f"Label: {label}, Encoded Value: {encoded\_value}")

Code IV. 5: Mapping of labels to encoded values of the type feature.

* Applying ML Models

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, mean\_squared\_error

# Splitting the Data

y = df\_encoded['type']  # Target variable

X = df\_encoded.drop('type', axis=1)  # Input features (remove the target variable column)

# Train-Test Split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

Code IV. 6: Splitting the Data based on the type column.

from sklearn.tree import DecisionTreeClassifier

# Decision Tree Classifier Model

model = DecisionTreeClassifier()

# Model Training

model.fit(X\_train, y\_train)

# Model Prediction

y\_pred = model.predict(X\_test)

# Model Evaluation

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average=None)

#precision = precision\_score(y\_test, y\_pred, labels=[0, 1, 2])

recall = recall\_score(y\_test, y\_pred, average=None)

f1 = f1\_score(y\_test, y\_pred, average=None)

mse = mean\_squared\_error(y\_test, y\_pred)

# Print Performance Metrics

print("Decision Tree Classifier Model Performance:")

print(f"Accuracy: {accuracy}")

print(f"Precision: {precision}")

print(f"Recall: {recall}")

print(f"F1-Score: {f1}")

print(f"Mean Squared Error: {mse}")

Code IV. 7: The application of the Decision Tree Classifier Model.

from sklearn.naive\_bayes import GaussianNB

# Gaussian Naive Bayes Model

model = GaussianNB()

# Model Training

model.fit(X\_train, y\_train)

# Model Prediction

y\_pred = model.predict(X\_test)

# Model Evaluation

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average=None)

#precision = precision\_score(y\_test, y\_pred, labels=[0, 1, 2])

recall = recall\_score(y\_test, y\_pred, average=None)

f1 = f1\_score(y\_test, y\_pred, average=None)

mse = mean\_squared\_error(y\_test, y\_pred)

# Print Performance Metrics

print("Gaussian Naive Bayes Model Performance:")

print(f"Accuracy: {accuracy}")

print(f"Precision: {precision}")

print(f"Recall: {recall}")

print(f"F1-Score: {f1}")

print(f"Mean Squared Error: {mse}")

Code IV. 8: The application of the Gaussian Naive Bayes Model.

from sklearn.neighbors import KNeighborsClassifier

# K-Nearest Neighbors Classifier Model

model = KNeighborsClassifier()

# Model Training

model.fit(X\_train, y\_train)

# Model Prediction

y\_pred = model.predict(X\_test)

# Model Evaluation

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average=None)

#precision = precision\_score(y\_test, y\_pred, labels=[0, 1, 2])

recall = recall\_score(y\_test, y\_pred, average=None)

f1 = f1\_score(y\_test, y\_pred, average=None)

mse = mean\_squared\_error(y\_test, y\_pred)

# Print Performance Metrics

print("K-Nearest Neighbors Classifier Model Performance:")

print(f"Accuracy: {accuracy}")

print(f"Precision: {precision}")

print(f"Recall: {recall}")

print(f"F1-Score: {f1}")

print(f"Mean Squared Error: {mse}")

Code IV. 9: The application of the K-Nearest Neighbors Classifier Model.

from sklearn.ensemble import RandomForestClassifier

# Random Forest Classifier Model

model = RandomForestClassifier()

# Model Training

model.fit(X\_train, y\_train)

# Model Prediction

y\_pred = model.predict(X\_test)

# Model Evaluation

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average=None)

#precision = precision\_score(y\_test, y\_pred, labels=[0, 1, 2])

recall = recall\_score(y\_test, y\_pred, average=None)

f1 = f1\_score(y\_test, y\_pred, average=None)

mse = mean\_squared\_error(y\_test, y\_pred)

# Print Performance Metrics

print("Random Forest Classifier Model Performance:")

print(f"Accuracy: {accuracy}")

print(f"Precision: {precision}")

print(f"Recall: {recall}")

print(f"F1-Score: {f1}")

print(f"Mean Squared Error: {mse}")

Code IV. 10: The application of the Random Forest Classifier Model.

from sklearn.linear\_model import LogisticRegression

# Logistic Regression Classifier Model

model = LogisticRegression(max\_iter=1000)

# Model Training

model.fit(X\_train, y\_train)

# Model Prediction

y\_pred = model.predict(X\_test)

# Model Evaluation

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average=None)

#precision = precision\_score(y\_test, y\_pred, labels=[0, 1, 2])

recall = recall\_score(y\_test, y\_pred, average=None)

f1 = f1\_score(y\_test, y\_pred, average=None)

mse = mean\_squared\_error(y\_test, y\_pred)

# Print Performance Metrics

print("Logistic Regression Classifier Model Performance:")

print(f"Accuracy: {accuracy}")

print(f"Precision: {precision}")

print(f"Recall: {recall}")

print(f"F1-Score: {f1}")

print(f"Mean Squared Error: {mse}")

Code IV. 11: The application of the Logistic Regression Classifier Model.