

1_Data_Analysis_IDS2017

June 16, 2024

1 Analyzing the IDS 2017 dataset

After processing and cleaning the files, the full dataset is loaded to analyze basic information about the traffic before applying machine learning algorithms.

```
[1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import glob
import seaborn as sns
import os

file_path = 
    ↪r"CIC-IDS-2017\CSVs\GeneratedLabelledFlows\TrafficLabelling\processed\ids2017_processed.
    ↪csv"
df = pd.read_csv(file_path)
convert_dict = {'label': 'category'}
df = df.astype(convert_dict)
df.info()
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2830743 entries, 0 to 2830742
Data columns (total 96 columns):
#   Column                                Dtype
---  -
0   destination_port                      int64
1   protocol                              int64
2   flow_duration                         int64
3   total_fwd_packets                     int64
4   total_backward_packets                int64
5   total_length_of_fwd_packets           float64
6   total_length_of_bwd_packets           float64
7   fwd_packet_length_max                  float64
8   fwd_packet_length_min                  float64
9   fwd_packet_length_mean                 float64
10  fwd_packet_length_std                   float64
11  bwd_packet_length_max                   float64
```

12	bwd_packet_length_min	float64
13	bwd_packet_length_mean	float64
14	bwd_packet_length_std	float64
15	flow_bytes_s	float64
16	flow_packets_s	float64
17	flow_iat_mean	float64
18	flow_iat_std	float64
19	flow_iat_max	float64
20	flow_iat_min	float64
21	fwd_iat_total	float64
22	fwd_iat_mean	float64
23	fwd_iat_std	float64
24	fwd_iat_max	float64
25	fwd_iat_min	float64
26	bwd_iat_total	float64
27	bwd_iat_mean	float64
28	bwd_iat_std	float64
29	bwd_iat_max	float64
30	bwd_iat_min	float64
31	fwd_psh_flags	int64
32	bwd_psh_flags	int64
33	fwd_urg_flags	int64
34	bwd_urg_flags	int64
35	fwd_header_length	int64
36	bwd_header_length	int64
37	fwd_packets_s	float64
38	bwd_packets_s	float64
39	min_packet_length	float64
40	max_packet_length	float64
41	packet_length_mean	float64
42	packet_length_std	float64
43	packet_length_variance	float64
44	fin_flag_count	int64
45	syn_flag_count	int64
46	rst_flag_count	int64
47	psh_flag_count	int64
48	ack_flag_count	int64
49	urg_flag_count	int64
50	cwe_flag_count	int64
51	ece_flag_count	int64
52	down_up_ratio	float64
53	average_packet_size	float64
54	avg_fwd_segment_size	float64
55	avg_bwd_segment_size	float64
56	fwd_header_length_1	int64
57	fwd_avg_bytes_bulk	int64
58	fwd_avg_packets_bulk	int64
59	fwd_avg_bulk_rate	int64

```

60 bwd_avg_bytes_bulk          int64
61 bwd_avg_packets_bulk       int64
62 bwd_avg_bulk_rate          int64
63 subflow_fwd_packets         int64
64 subflow_fwd_bytes          int64
65 subflow_bwd_packets         int64
66 subflow_bwd_bytes          int64
67 init_win_bytes_forward      int64
68 init_win_bytes_backward     int64
69 act_data_pkt_fwd            int64
70 min_seg_size_forward        int64
71 active_mean                 float64
72 active_std                  float64
73 active_max                  float64
74 active_min                  float64
75 idle_mean                   float64
76 idle_std                    float64
77 idle_max                    float64
78 idle_min                    float64
79 label                       category
80 is_attack                   int64
81 label_code                  int64
82 is_dos_hulk                 int64
83 is_portscan                 int64
84 is_ddos                     int64
85 is_dos_goldeneye            int64
86 is_ftppatator               int64
87 is_sshpatator               int64
88 is_dos_slowloris            int64
89 is_dos_slowhttptest         int64
90 is_bot                      int64
91 is_web_attack_brute_force   int64
92 is_web_attack_xss           int64
93 is_infiltration             int64
94 is_web_attack_sql_injection int64
95 is_heartbleed               int64
dtypes: category(1), float64(45), int64(50)
memory usage: 2.0 GB

```

```

[1]:  destination_port  protocol  flow_duration  total_fwd_packets  \
0           49188         6           4           2
1           49188         6           1           2
2           49188         6           1           2
3           49188         6           1           2
4           49486         6           3           2

total_backward_packets  total_length_of_fwd_packets  \

```

0	0	12.0
1	0	12.0
2	0	12.0
3	0	12.0
4	0	12.0

	total_length_of_bwd_packets	fwd_packet_length_max	fwd_packet_length_min	\
0	0.0	6.0	6.0	
1	0.0	6.0	6.0	
2	0.0	6.0	6.0	
3	0.0	6.0	6.0	
4	0.0	6.0	6.0	

	fwd_packet_length_mean	...	is_ftppatator	is_sshpatator	\
0	6.0	...	0	0	
1	6.0	...	0	0	
2	6.0	...	0	0	
3	6.0	...	0	0	
4	6.0	...	0	0	

	is_dos_slowloris	is_dos_slowhttpstest	is_bot	is_web_attack_brute_force	\
0	0	0	0	0	
1	0	0	0	0	
2	0	0	0	0	
3	0	0	0	0	
4	0	0	0	0	

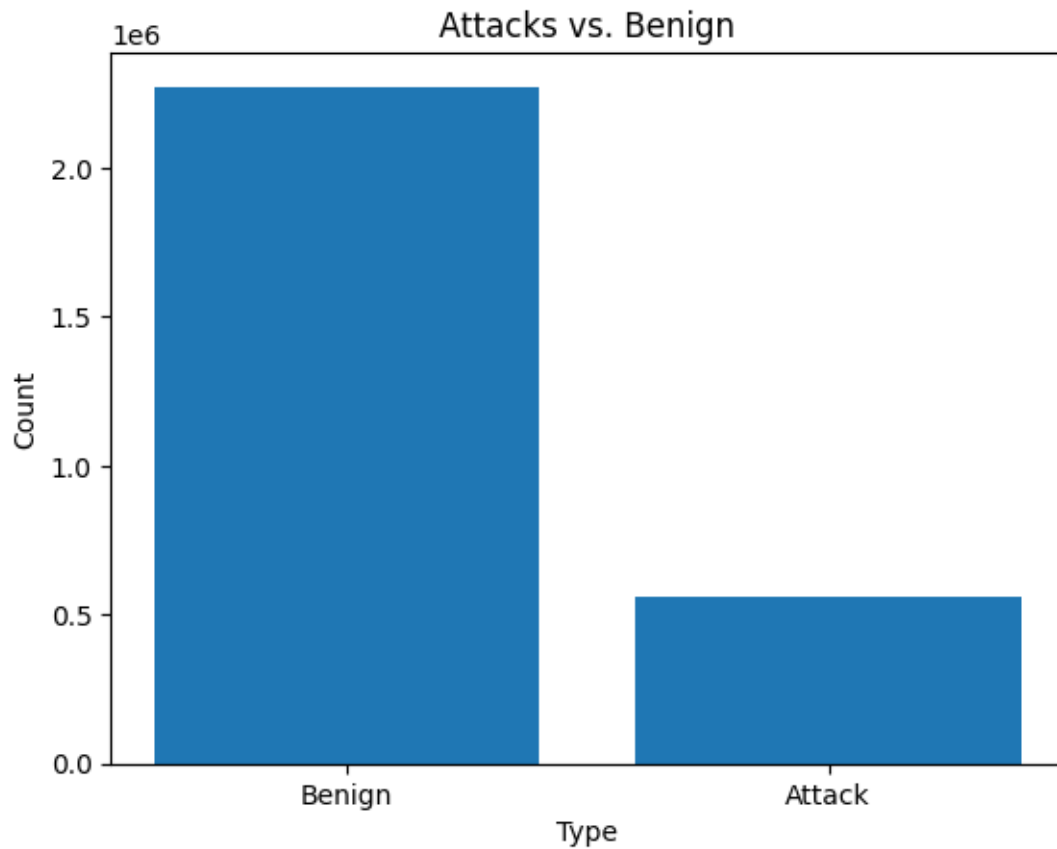
	is_web_attack_xss	is_infiltration	is_web_attack_sql_injection	\
0	0	0	0	
1	0	0	0	
2	0	0	0	
3	0	0	0	
4	0	0	0	

	is_heartbleed
0	0
1	0
2	0
3	0
4	0

[5 rows x 96 columns]

1.0.1 1. Benign Network flows vs Attacks

```
[2]: attack_counts = df['is_attack'].value_counts()
plt.bar(attack_counts.index, attack_counts.values)
plt.xlabel('Type')
plt.ylabel('Count')
plt.title('Attacks vs. Benign')
plt.xticks(ticks=[0, 1], labels=['Benign', 'Attack']) # Add custom x-axis labels
plt.show()
```



```
[3]: attack_percentages = (attack_counts / attack_counts.sum()) * 100

# Create a new DataFrame for the table
table_data = pd.DataFrame({'Type': attack_counts.index, 'Number of Attacks': attack_counts.values, 'Percentage': attack_percentages.values})

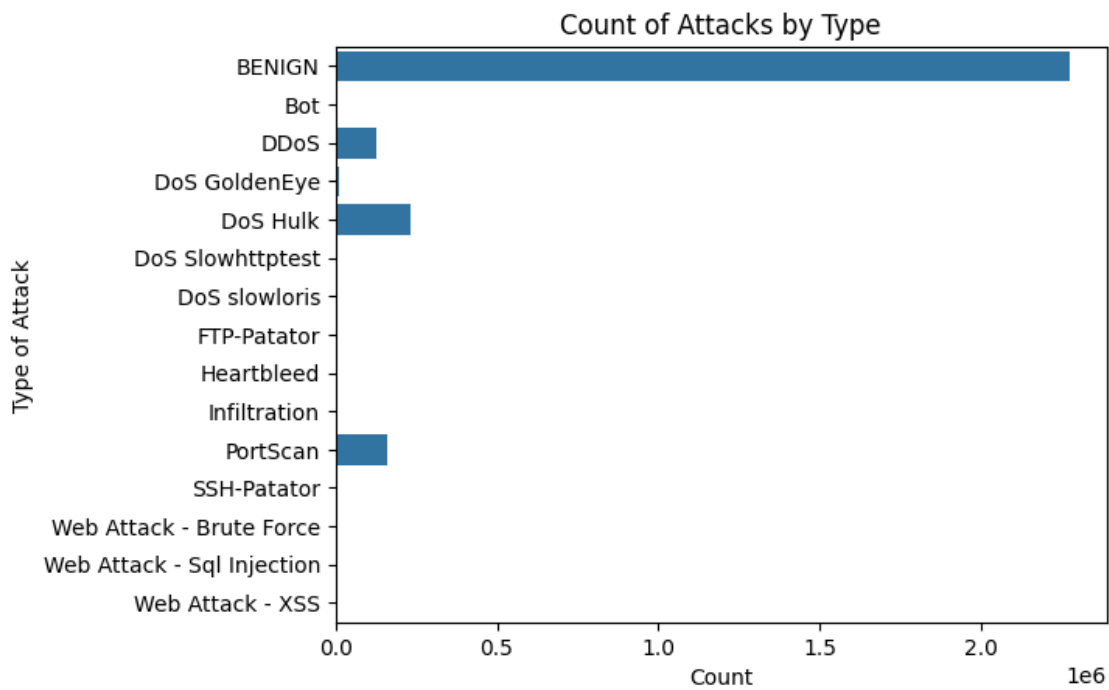
# Display the table
table_data
```

```
[3]:
```

Type	Number of Attacks	Percentage
0	2273097	80.300366
1	557646	19.699634

1.0.2 2. Plot by the Type of Network Traffic

```
[5]: sns.countplot(y='label', data=df)
plt.xlabel('Count')
plt.ylabel('Type of Attack')
plt.title('Count of Attacks by Type')
plt.show()
```



```
[7]: attack_counts = df['label'].value_counts()
table_data = pd.DataFrame({'Type of Attack': attack_counts.index, 'Number of_
↳ Attacks': attack_counts.values})
table_data
```

```
[7]:
```

	Type of Attack	Number of Attacks
0	BENIGN	2273097
1	DoS Hulk	231073
2	PortScan	158930
3	DDoS	128027
4	DoS GoldenEye	10293
5	FTP-Patator	7938

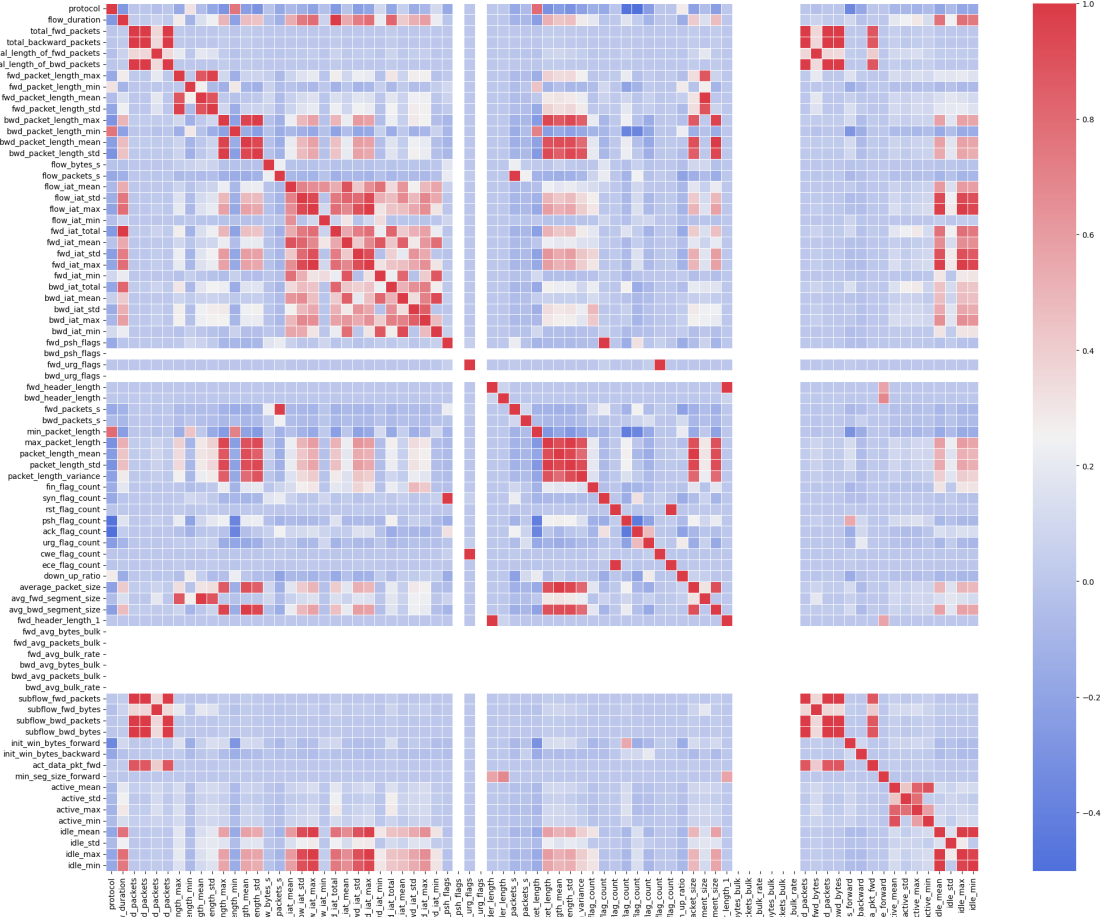
6	SSH-Patator	5897
7	DoS slowloris	5796
8	DoS Slowhttptest	5499
9	Bot	1966
10	Web Attack - Brute Force	1507
11	Web Attack - XSS	652
12	Infiltration	36
13	Web Attack - Sql Injection	21
14	Heartbleed	11

1.0.3 3. Correlation Between Features

A heatmap for the correlation matrix of all relevant features is used to visualize groups of highly correlated features.

```
[16]: new_df = df.iloc[:, 1:79]
      corr = new_df.corr()
      plt.figure(figsize=(25, 20))
      sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns,
                  linewidths=.5, cmap=sns.diverging_palette(620, 10, as_cmap=True))
```

```
[16]: <Axes: >
```



1.0.4 Conclusion

From this preliminary analysis of the dataset, it can be concluded that the number of benign traffics is much higher than the number of attacks, which will make the machine learning models very heavily skewed towards benign traffics. Some attacks are also very underrepresented, meaning that a binary classifier will be more accurate for detecting certain attacks. Finally, the dataset contains a number of highly correlated features that could be redundant for training machine learning models. Feature engineering must be applied before creating the models.