

Project assiment

March 1, 2026

```
[1]: import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
[2]: df = pd.read_csv('embedded_system_network_security_dataset.csv')
```

```
[3]: df.dtypes
```

```
[3]: packet_size          float64
inter_arrival_time      float64
src_port                int64
dst_port                int64
packet_count_5s         float64
mean_packet_size        float64
spectral_entropy        float64
frequency_band_energy   float64
label                   float64
protocol_type_TCP        bool
protocol_type_UDP        bool
src_ip_192.168.1.2       bool
src_ip_192.168.1.3       bool
dst_ip_192.168.1.5       bool
dst_ip_192.168.1.6       bool
tcp_flags_FIN            bool
tcp_flags_SYN            bool
tcp_flags_SYN-ACK        bool
dtype: object
```

```
[4]: df.isnull().sum()
```

```
[4]: packet_size          0
inter_arrival_time      0
src_port                0
dst_port                0
packet_count_5s         0
```

```

mean_packet_size      0
spectral_entropy      0
frequency_band_energy 0
label                 0
protocol_type_TCP      0
protocol_type_UDP      0
src_ip_192.168.1.2    0
src_ip_192.168.1.3    0
dst_ip_192.168.1.5    0
dst_ip_192.168.1.6    0
tcp_flags_FIN         0
tcp_flags_SYN         0
tcp_flags_SYN-ACK     0
dtype: int64

```

```

[5]: missing_percentage = df.isnull().mean() * 100
threshold = 30
columns_to_drop = missing_percentage[missing_percentage > threshold].index
df = df.drop(columns=columns_to_drop)
print(f"Columns dropped: {list(columns_to_drop)}")
print(f"Remaining columns: {list(df.columns)}")

```

```

Columns dropped: []
Remaining columns: ['packet_size', 'inter_arrival_time', 'src_port', 'dst_port',
'packet_count_5s', 'mean_packet_size', 'spectral_entropy',
'frequency_band_energy', 'label', 'protocol_type_TCP', 'protocol_type_UDP',
'src_ip_192.168.1.2', 'src_ip_192.168.1.3', 'dst_ip_192.168.1.5',
'dst_ip_192.168.1.6', 'tcp_flags_FIN', 'tcp_flags_SYN', 'tcp_flags_SYN-ACK']

```

```

[8]: from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
for column in df.select_dtypes(include=['object']).columns:
    df[column] = label_encoder.fit_transform(df[column])
df.head()

```

```

[8]: packet_size  inter_arrival_time  src_port  dst_port  packet_count_5s  \
0      0.405154          0.620362    62569    443          0.857143
1      0.527559          0.741288    59382    443          0.785714
2      0.226199          0.485116    65484     80          0.285714
3      0.573372          0.450965    51707     53          0.142857
4      0.651396          0.888740    26915     53          0.714286

```

```

mean_packet_size  spectral_entropy  frequency_band_energy  label  \
0              0.0          0.834066          0.534891    0.0
1              0.0          0.147196          0.990757    0.0
2              0.0          0.855192          0.031781    0.0
3              0.0          0.153220          0.169958    0.0
4              0.0          0.923916          0.552053    0.0

```

	protocol_type_TCP	protocol_type_UDP	src_ip_192.168.1.2	\
0	False	True	True	
1	False	True	False	
2	False	True	False	
3	False	False	False	
4	True	False	False	

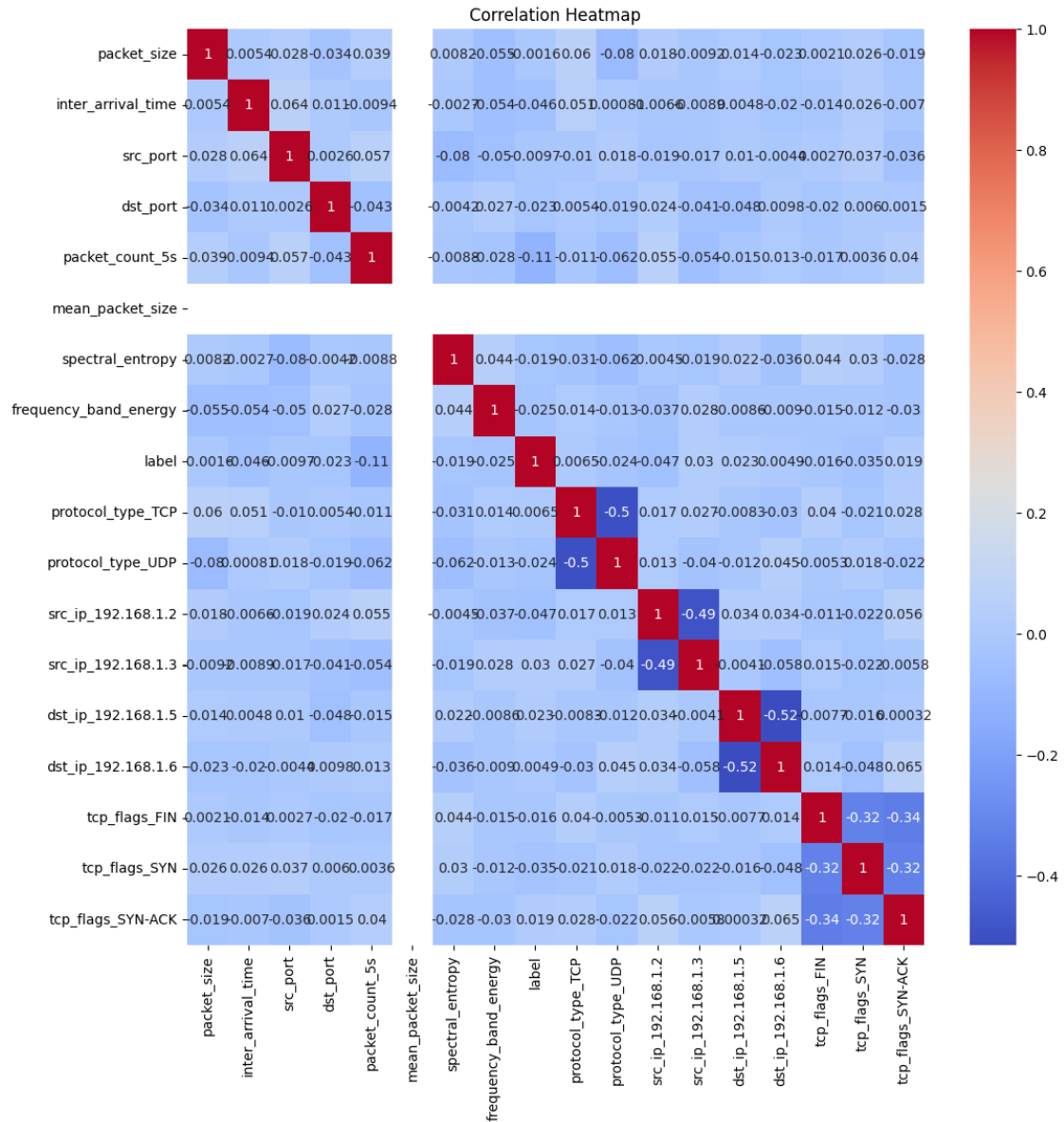
	src_ip_192.168.1.3	dst_ip_192.168.1.5	dst_ip_192.168.1.6	tcp_flags_FIN	\
0	False	False	False	False	
1	False	False	True	False	
2	False	True	False	False	
3	True	False	False	False	
4	True	False	False	False	

	tcp_flags_SYN	tcp_flags_SYN-ACK
0	False	False
1	True	False
2	False	False
3	False	False
4	True	False

```
[9]: for column in df.select_dtypes(include=['float64']).columns:
      mean_value = df[column].mean() # Get the mode (most frequent value)
      df[column] = df[column].fillna(mean_value) # Fill missing values in
      ↪categorical columns with mode
```

```
[10]: for column in df.select_dtypes(include=['object']).columns:
       mode_value = df[column].mode()[0]
       df[column] = df[column].fillna(mode_value)
```

```
[11]: plt.figure(figsize=(12, 12))
      sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
      plt.title('Correlation Heatmap')
      plt.show()
```

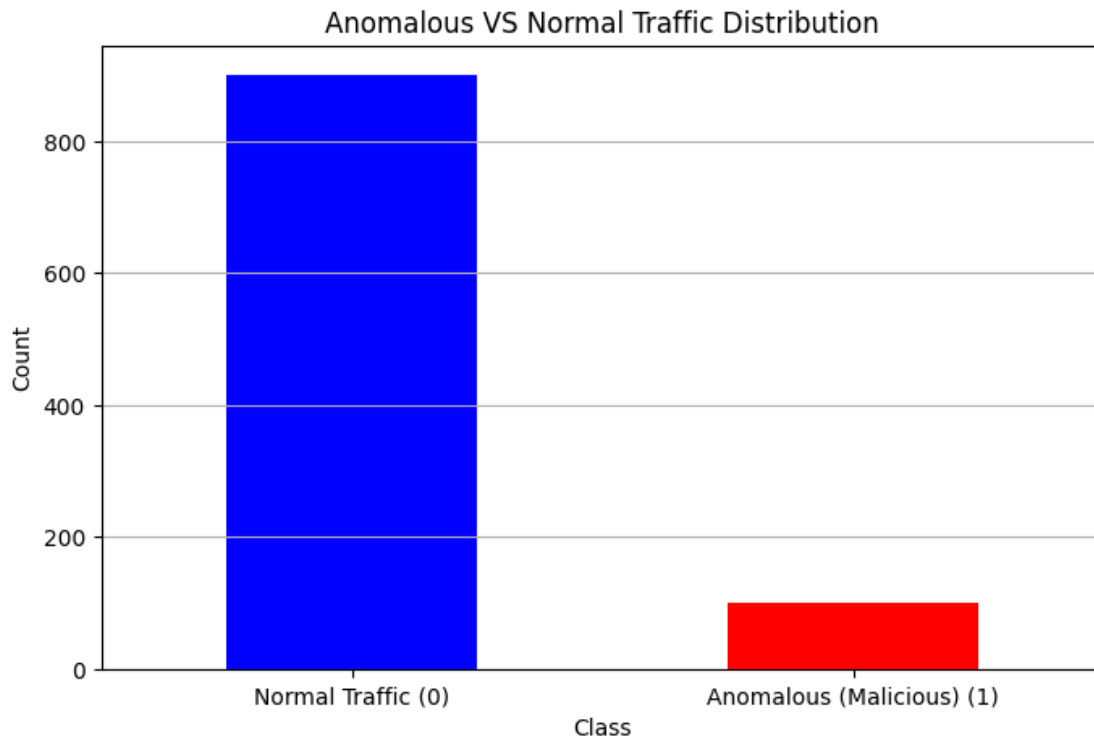


```
[12]: df['label'].value_counts()
```

```
[12]: label
0.0    900
1.0    100
Name: count, dtype: int64
```

```
[16]: plt.figure(figsize=(8, 5))
df['label'].value_counts().plot(kind='bar', color=['Blue', 'Red'])
plt.xticks([0, 1], ['Normal Traffic (0)', 'Anomalous (Malicious) (1)'],
           rotation=0)
```

```
plt.title('Anomalous VS Normal Traffic Distribution')
plt.xlabel('Class')
plt.ylabel('Count')
plt.grid(axis='y')
plt.show()
```



```
[22]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
colors = ['Blue', 'Red']
labels = ['Normal Traffic', 'Anomalous ']
order = df['label'].value_counts().index
plt.figure(figsize=(16, 8))
plt.suptitle('Anomalous VS Normal Traffic Distribution', fontsize=16,
            fontweight='bold')
plt.subplot(1, 2, 1)
plt.title('Pie Chart', fontsize=14)
plt.pie(df['label'].value_counts(), labels=labels, colors=colors,
        autopct='%0.2f%%', startangle=90)
centre = plt.Circle((0, 0), 0.80, color='white')
plt.gca().add_artist(centre)
plt.subplot(1, 2, 2)
```

```

plt.title('Bar chart', fontsize=14)
ax = sns.countplot(x='label', data=df, palette=colors, order=order)
for bar in ax.patches:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width()/2, height + 1, int(height),
            ha='center', fontsize=10)
plt.xlabel('Traffic')
plt.ylabel('Total')
plt.xticks([0, 1], labels)
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()

```

