aizabayo-pda-analyticassignmentii

November 30, 2023

Name: Ange Izabayo

Program: MSECE

Andrew ID: aizabayo@andrew.cmu.edu

Date: 30/11/2022

PDA Analytic Assignment II

[1]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler, LabelEncoder
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import make_scorer, precision_score, recall_score, f1_score
from sklearn.metrics import accuracy_score
from sklearn.datasets import make_classification
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.tree import DecisionTreeClassifier
from sklearn.decomposition import PCA
from sklearn.utils import class_weight

import warnings
warnings.filterwarnings("ignore")
```

0.0.1 Data Exploration

Loading dataset and then merge them into one dataframe

[2]:
```
Monday_df = pd.read_csv('Monday-WorkingHours.pcap_ISCX.csv')
Tuesday_df = pd.read_csv('Tuesday-WorkingHours.pcap_ISCX.csv')
Wednesday_df = pd.read_csv('Wednesday-workingHours.pcap_ISCX.csv')
Thursday1_df = pd.read_csv('Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.
↪csv')
```

```
Thursday_df = pd.read_csv('Thursday-WorkingHours-Afternoon-Infilteration.
 ↪pcap_ISCX.csv')
Friday_df = pd.read_csv('Friday-WorkingHours-Morning.pcap_ISCX.csv')
Friday1_df = pd.read_csv('Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv')
Friday2_df = pd.read_csv('Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv')
# Merge dataframe
Data_df = pd.concat([Monday_df, Tuesday_df, Wednesday_df,␣
 ↪Thursday_df,Thursday1_df ,Friday_df, Friday1_df, Friday2_df])
display(Data_df.head(10))
```

```
    Destination Port   Flow Duration   Total Fwd Packets  \
0              49188               4                   2
1              49188               1                   2
2              49188               1                   2
3              49188               1                   2
4              49486               3                   2
5              49486               1                   2
6              49486               1                   2
7              49486               1                   2
8                 88             609                   7
9                 88             879                   9

    Total Backward Packets  Total Length of Fwd Packets  \
0                        0                           12
1                        0                           12
2                        0                           12
3                        0                           12
4                        0                           12
5                        0                           12
6                        0                           12
7                        0                           12
8                        4                          484
9                        4                          656

    Total Length of Bwd Packets   Fwd Packet Length Max  \
0                             0                       6
1                             0                       6
2                             0                       6
3                             0                       6
4                             0                       6
5                             0                       6
6                             0                       6
7                             0                       6
8                           414                     233
9                          3064                     313

    Fwd Packet Length Min   Fwd Packet Length Mean   Fwd Packet Length Std  \
```

```
0                   6              6.000000              0.000000
1                   6              6.000000              0.000000
2                   6              6.000000              0.000000
3                   6              6.000000              0.000000
4                   6              6.000000              0.000000
5                   6              6.000000              0.000000
6                   6              6.000000              0.000000
7                   6              6.000000              0.000000
8                   0             69.142857            111.967895
9                   0             72.888889            136.153814

   …  min_seg_size_forward  Active Mean  Active Std  Active Max  \
0  …                    20          0.0         0.0           0
1  …                    20          0.0         0.0           0
2  …                    20          0.0         0.0           0
3  …                    20          0.0         0.0           0
4  …                    20          0.0         0.0           0
5  …                    20          0.0         0.0           0
6  …                    20          0.0         0.0           0
7  …                    20          0.0         0.0           0
8  …                    20          0.0         0.0           0
9  …                    20          0.0         0.0           0

    Active Min  Idle Mean  Idle Std  Idle Max  Idle Min  Label
0            0        0.0       0.0         0         0  BENIGN
1            0        0.0       0.0         0         0  BENIGN
2            0        0.0       0.0         0         0  BENIGN
3            0        0.0       0.0         0         0  BENIGN
4            0        0.0       0.0         0         0  BENIGN
5            0        0.0       0.0         0         0  BENIGN
6            0        0.0       0.0         0         0  BENIGN
7            0        0.0       0.0         0         0  BENIGN
8            0        0.0       0.0         0         0  BENIGN
9            0        0.0       0.0         0         0  BENIGN

[10 rows x 79 columns]
```

Loads datasets individually of each day of the week from Monday to Friday separately, and then combines or merges using pd.concat, into a single DataFrame named Data_df. This approach allows for a more comprehensive analysis of the dataset as it combines data from different days into one unified structure, enabling cross-day insights and facilitating machine learning model training on a larger dataset.

Displayed 10 rows of the merged dataset

[3]:
```
Data_df
```

[3]:

```
        Destination Port   Flow Duration   Total Fwd Packets  \
0                  49188               4                   2
1                  49188               1                   2
2                  49188               1                   2
3                  49188               1                   2
4                  49486               3                   2
…                    …             …                 …
225740             61374              61                   1
225741             61378              72                   1
225742             61375              75                   1
225743             61323              48                   2
225744             61326              68                   1

        Total Backward Packets  Total Length of Fwd Packets  \
0                            0                           12
1                            0                           12
2                            0                           12
3                            0                           12
4                            0                           12
…                          …                           …
225740                       1                            6
225741                       1                            6
225742                       1                            6
225743                       0                           12
225744                       1                            6

        Total Length of Bwd Packets   Fwd Packet Length Max  \
0                                 0                       6
1                                 0                       6
2                                 0                       6
3                                 0                       6
4                                 0                       6
…                               …                     …
225740                            6                       6
225741                            6                       6
225742                            6                       6
225743                            0                       6
225744                            6                       6

        Fwd Packet Length Min   Fwd Packet Length Mean  \
0                           6                      6.0
1                           6                      6.0
2                           6                      6.0
3                           6                      6.0
4                           6                      6.0
…                         …                      …
225740                      6                      6.0
```

```
225741                    6                    6.0
225742                    6                    6.0
225743                    6                    6.0
225744                    6                    6.0

        Fwd Packet Length Std  …  min_seg_size_forward  Active Mean  \
0                         0.0  …                    20          0.0
1                         0.0  …                    20          0.0
2                         0.0  …                    20          0.0
3                         0.0  …                    20          0.0
4                         0.0  …                    20          0.0
…                         …  …                    …          …
225740                    0.0  …                    20          0.0
225741                    0.0  …                    20          0.0
225742                    0.0  …                    20          0.0
225743                    0.0  …                    20          0.0
225744                    0.0  …                    20          0.0

        Active Std  Active Max  Active Min  Idle Mean  Idle Std  \
0              0.0           0           0        0.0       0.0
1              0.0           0           0        0.0       0.0
2              0.0           0           0        0.0       0.0
3              0.0           0           0        0.0       0.0
4              0.0           0           0        0.0       0.0
…              …           …           …        …       …
225740         0.0           0           0        0.0       0.0
225741         0.0           0           0        0.0       0.0
225742         0.0           0           0        0.0       0.0
225743         0.0           0           0        0.0       0.0
225744         0.0           0           0        0.0       0.0

        Idle Max  Idle Min  Label
0              0         0  BENIGN
1              0         0  BENIGN
2              0         0  BENIGN
3              0         0  BENIGN
4              0         0  BENIGN
…              …         …     …
225740         0         0  BENIGN
225741         0         0  BENIGN
225742         0         0  BENIGN
225743         0         0  BENIGN
225744         0         0  BENIGN

[2830743 rows x 79 columns]
```

Count null values and printing the data type of each column

[4]:
```python
print('Printing 10 last row\n')
display(Data_df.tail(10))
counts = Data_df.isnull().sum() #count the number of null values in each column
print(counts)
print(Data_df.dtypes)
```

```
Printing 10 last row


        Destination Port   Flow Duration   Total Fwd Packets  \
225735             61301              28                   1
225736             38130              45                   1
225737             10398               4                   2
225738             61376              44                   1
225739             61377              26                   1
225740             61374              61                   1
225741             61378              72                   1
225742             61375              75                   1
225743             61323              48                   2
225744             61326              68                   1

        Total Backward Packets  Total Length of Fwd Packets  \
225735                       1                            6
225736                       1                            0
225737                       0                          248
225738                       1                            6
225739                       1                            6
225740                       1                            6
225741                       1                            6
225742                       1                            6
225743                       0                           12
225744                       1                            6

        Total Length of Bwd Packets   Fwd Packet Length Max  \
225735                            6                       6
225736                            0                       0
225737                            0                     242
225738                            6                       6
225739                            6                       6
225740                            6                       6
225741                            6                       6
225742                            6                       6
225743                            0                       6
225744                            6                       6

        Fwd Packet Length Min   Fwd Packet Length Mean  \
225735                      6                      6.0
225736                      0                      0.0
```

```
225737                 6                    124.0
225738                 6                      6.0
225739                 6                      6.0
225740                 6                      6.0
225741                 6                      6.0
225742                 6                      6.0
225743                 6                      6.0
225744                 6                      6.0

         Fwd Packet Length Std  …   min_seg_size_forward  Active Mean  \
225735                  0.0000  …                     20          0.0
225736                  0.0000  …                     32          0.0
225737                166.8772  …                     20          0.0
225738                  0.0000  …                     20          0.0
225739                  0.0000  …                     20          0.0
225740                  0.0000  …                     20          0.0
225741                  0.0000  …                     20          0.0
225742                  0.0000  …                     20          0.0
225743                  0.0000  …                     20          0.0
225744                  0.0000  …                     20          0.0

         Active Std   Active Max   Active Min  Idle Mean   Idle Std  \
225735          0.0            0            0        0.0        0.0
225736          0.0            0            0        0.0        0.0
225737          0.0            0            0        0.0        0.0
225738          0.0            0            0        0.0        0.0
225739          0.0            0            0        0.0        0.0
225740          0.0            0            0        0.0        0.0
225741          0.0            0            0        0.0        0.0
225742          0.0            0            0        0.0        0.0
225743          0.0            0            0        0.0        0.0
225744          0.0            0            0        0.0        0.0

         Idle Max   Idle Min   Label
225735          0          0  BENIGN
225736          0          0  BENIGN
225737          0          0  BENIGN
225738          0          0  BENIGN
225739          0          0  BENIGN
225740          0          0  BENIGN
225741          0          0  BENIGN
225742          0          0  BENIGN
225743          0          0  BENIGN
225744          0          0  BENIGN

[10 rows x 79 columns]

 Destination Port              0
```

```
 Flow Duration               0
 Total Fwd Packets           0
 Total Backward Packets      0
Total Length of Fwd Packets  0
                            ..
Idle Mean                    0
 Idle Std                    0
 Idle Max                    0
 Idle Min                    0
 Label                       0
Length: 79, dtype: int64
 Destination Port              int64
 Flow Duration                 int64
 Total Fwd Packets             int64
 Total Backward Packets        int64
Total Length of Fwd Packets    int64
                                …
Idle Mean                    float64
 Idle Std                    float64
 Idle Max                      int64
 Idle Min                      int64
 Label                        object
Length: 79, dtype: object
```

Printing the 10 last rows of the merged dataset, and counting the number of null values in each column. The data type of each column is also displayed.

Basic statistic of the dataset

[5]:
```python
display(Data_df.info())
display(Data_df.describe()) # return the basic statistic of the dataset
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2830743 entries, 0 to 225744
Data columns (total 79 columns):
 #   Column                        Dtype
---  ------                        -----
 0    Destination Port             int64
 1    Flow Duration                int64
 2    Total Fwd Packets            int64
 3    Total Backward Packets       int64
 4   Total Length of Fwd Packets   int64
 5    Total Length of Bwd Packets  int64
 6    Fwd Packet Length Max        int64
 7    Fwd Packet Length Min        int64
 8    Fwd Packet Length Mean       float64
 9    Fwd Packet Length Std        float64
 10  Bwd Packet Length Max         int64
 11   Bwd Packet Length Min        int64
```

```
12   Bwd Packet Length Mean       float64
13   Bwd Packet Length Std        float64
14  Flow Bytes/s                  float64
15   Flow Packets/s               float64
16   Flow IAT Mean                float64
17   Flow IAT Std                 float64
18   Flow IAT Max                 int64
19   Flow IAT Min                 int64
20  Fwd IAT Total                 int64
21   Fwd IAT Mean                 float64
22   Fwd IAT Std                  float64
23   Fwd IAT Max                  int64
24   Fwd IAT Min                  int64
25  Bwd IAT Total                 int64
26   Bwd IAT Mean                 float64
27   Bwd IAT Std                  float64
28   Bwd IAT Max                  int64
29   Bwd IAT Min                  int64
30  Fwd PSH Flags                 int64
31   Bwd PSH Flags                int64
32   Fwd URG Flags                int64
33   Bwd URG Flags                int64
34   Fwd Header Length            int64
35   Bwd Header Length            int64
36  Fwd Packets/s                 float64
37   Bwd Packets/s                float64
38   Min Packet Length            int64
39   Max Packet Length            int64
40   Packet Length Mean           float64
41   Packet Length Std            float64
42   Packet Length Variance       float64
43  FIN Flag Count                int64
44   SYN Flag Count               int64
45   RST Flag Count               int64
46   PSH Flag Count               int64
47   ACK Flag Count               int64
48   URG Flag Count               int64
49   CWE Flag Count               int64
50   ECE Flag Count               int64
51   Down/Up Ratio                int64
52   Average Packet Size          float64
53   Avg Fwd Segment Size         float64
54   Avg Bwd Segment Size         float64
55   Fwd Header Length.1          int64
56  Fwd Avg Bytes/Bulk            int64
57   Fwd Avg Packets/Bulk         int64
58   Fwd Avg Bulk Rate            int64
59   Bwd Avg Bytes/Bulk           int64
```

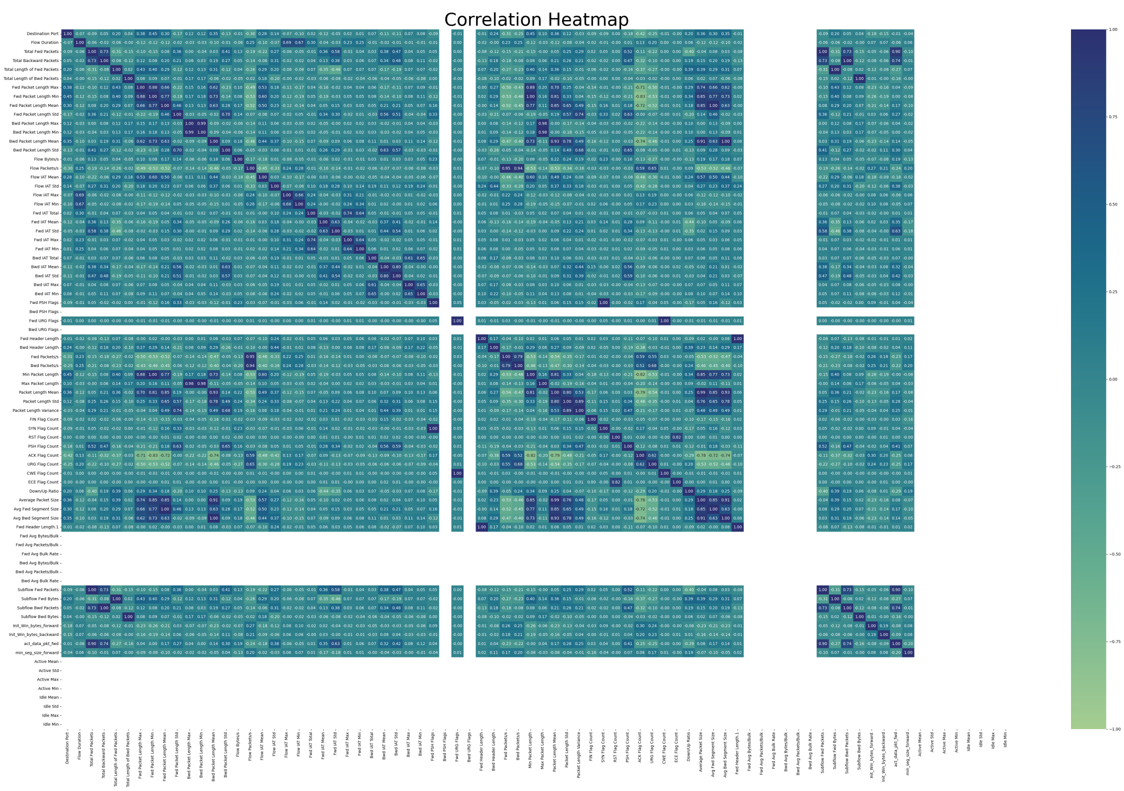
```
 60   Bwd Avg Packets/Bulk        int64
 61  Bwd Avg Bulk Rate            int64
 62  Subflow Fwd Packets          int64
 63   Subflow Fwd Bytes           int64
 64   Subflow Bwd Packets         int64
 65   Subflow Bwd Bytes           int64
 66  Init_Win_bytes_forward       int64
 67   Init_Win_bytes_backward     int64
 68   act_data_pkt_fwd            int64
 69   min_seg_size_forward        int64
 70  Active Mean                  float64
 71   Active Std                  float64
 72   Active Max                  int64
 73   Active Min                  int64
 74  Idle Mean                    float64
 75   Idle Std                    float64
 76   Idle Max                    int64
 77   Idle Min                    int64
 78   Label                       object
dtypes: float64(24), int64(54), object(1)
memory usage: 1.7+ GB

None
        Destination Port   Flow Duration   Total Fwd Packets  \
count       2.830743e+06    2.830743e+06        2.830743e+06
mean        8.071483e+03    1.478566e+07        9.361160e+00
std         1.828363e+04    3.365374e+07        7.496728e+02
min         0.000000e+00   -1.300000e+01        1.000000e+00
25%         5.300000e+01    1.550000e+02        2.000000e+00
50%         8.000000e+01    3.131600e+04        2.000000e+00
75%         4.430000e+02    3.204828e+06        5.000000e+00
max         6.553500e+04    1.200000e+08        2.197590e+05

        Total Backward Packets  Total Length of Fwd Packets  \
count             2.830743e+06                 2.830743e+06
mean              1.039377e+01                 5.493024e+02
std               9.973883e+02                 9.993589e+03
min               0.000000e+00                 0.000000e+00
25%               1.000000e+00                 1.200000e+01
50%               2.000000e+00                 6.200000e+01
75%               4.000000e+00                 1.870000e+02
max               2.919220e+05                 1.290000e+07

        Total Length of Bwd Packets   Fwd Packet Length Max  \
count                  2.830743e+06            2.830743e+06
mean                   1.616264e+04            2.075999e+02
std                    2.263088e+06            7.171848e+02
min                    0.000000e+00            0.000000e+00
```

```
25%                   0.000000e+00            6.000000e+00
50%                   1.230000e+02            3.700000e+01
75%                   4.820000e+02            8.100000e+01
max                   6.554530e+08            2.482000e+04

       Fwd Packet Length Min   Fwd Packet Length Mean  \
count           2.830743e+06             2.830743e+06
mean            1.871366e+01             5.820194e+01
std             6.033935e+01             1.860912e+02
min             0.000000e+00             0.000000e+00
25%             0.000000e+00             6.000000e+00
50%             2.000000e+00             3.400000e+01
75%             3.600000e+01             5.000000e+01
max             2.325000e+03             5.940857e+03

       Fwd Packet Length Std  …   act_data_pkt_fwd   min_seg_size_forward  \
count           2.830743e+06  …       2.830743e+06           2.830743e+06
mean            6.891013e+01  …       5.418218e+00          -2.741688e+03
std             2.811871e+02  …       6.364257e+02           1.084989e+06
min             0.000000e+00  …       0.000000e+00          -5.368707e+08
25%             0.000000e+00  …       0.000000e+00           2.000000e+01
50%             0.000000e+00  …       1.000000e+00           2.400000e+01
75%             2.616295e+01  …       2.000000e+00           3.200000e+01
max             7.125597e+03  …       2.135570e+05           1.380000e+02

       Active Mean    Active Std    Active Max    Active Min     Idle Mean  \
count  2.830743e+06  2.830743e+06  2.830743e+06  2.830743e+06  2.830743e+06
mean   8.155132e+04  4.113412e+04  1.531825e+05  5.829582e+04  8.316037e+06
std    6.485999e+05  3.933815e+05  1.025825e+06  5.770923e+05  2.363008e+07
min    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
25%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
50%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
75%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
max    1.100000e+08  7.420000e+07  1.100000e+08  1.100000e+08  1.200000e+08

          Idle Std      Idle Max      Idle Min
count  2.830743e+06  2.830743e+06  2.830743e+06
mean   5.038439e+05  8.695752e+06  7.920031e+06
std    4.602984e+06  2.436689e+07  2.336342e+07
min    0.000000e+00  0.000000e+00  0.000000e+00
25%    0.000000e+00  0.000000e+00  0.000000e+00
50%    0.000000e+00  0.000000e+00  0.000000e+00
75%    0.000000e+00  0.000000e+00  0.000000e+00
max    7.690000e+07  1.200000e+08  1.200000e+08

[8 rows x 78 columns]
```

Displaying the information of the dataset which include datatype of each column, number of rows and columns and memory usage. Display the basic statistic of the dataset. The basic statistic of the dataset includes the count, mean, standard deviation, minimum, maximum, and quartiles of each column. The quartiles are calculated using the np.percentile function with the 25th and 75th percentiles as the lower and upper bounds, respectively.

Correlation matrix

[39]:
```python
X_ind = Data_df.drop(' Label', axis=1)
y_dep = Data_df[' Label']
df = pd.DataFrame(X_ind)
corr_matrix = df.corr() #calculate the correlation matrix of the dataset independetn variable
plt.figure(figsize=(54, 32))
sns.heatmap(corr_matrix, annot=True, cmap="crest",fmt='.2f',  vmin = -1, vmax = 1)
plt.title("Correlation Heatmap",  fontsize=45)
plt.show()
```

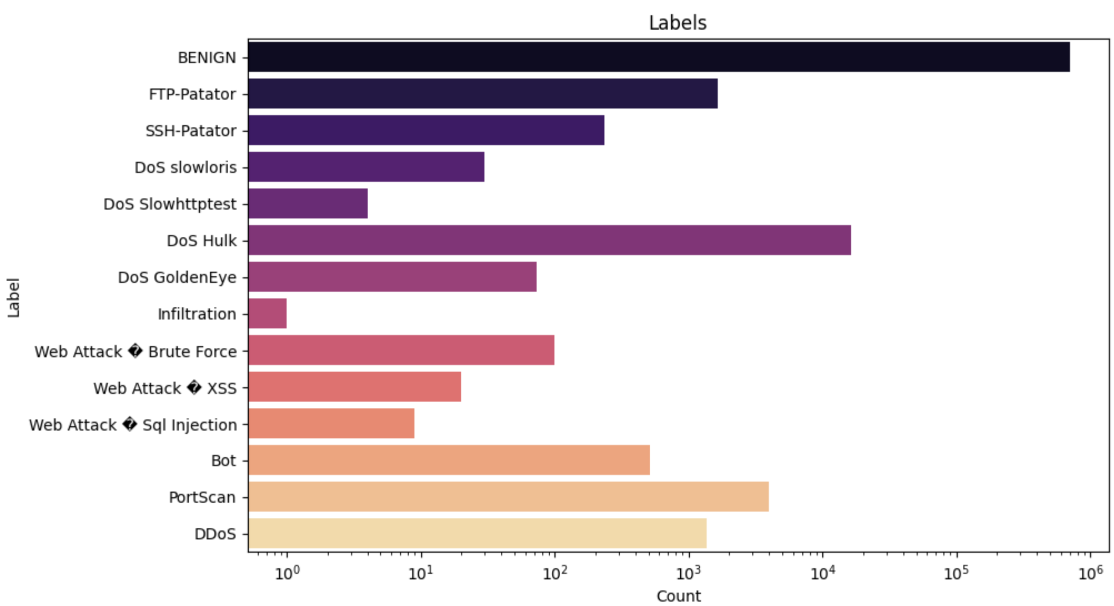


Plotting the correlation heatmap for visualizing and quick examining the relationships and dependencies between features, where warmer colors indicate stronger positive correlations and light cooler indicate stronger negative correlations. where strong correlation mean change of one affect the change of the other. and 0 means there is no correlation between the two features. the correlation coefficient range between -1 and 1

Examine Label Distribution

[29]:
```
label_counts = Data_df[' Label'].value_counts() # count the number of␣
 ↪occurrences of each label
print(label_counts)

plt.figure(figsize=(10, 6))
plot = sns.countplot(data=Data_df, y=' Label', palette='magma')  # plot the␣
 ↪number of occurrences of each label rows
plt.xscale('log')
plt.title('Labels')
plt.xlabel('Count')
plt.ylabel('Label')
plt.show()
```

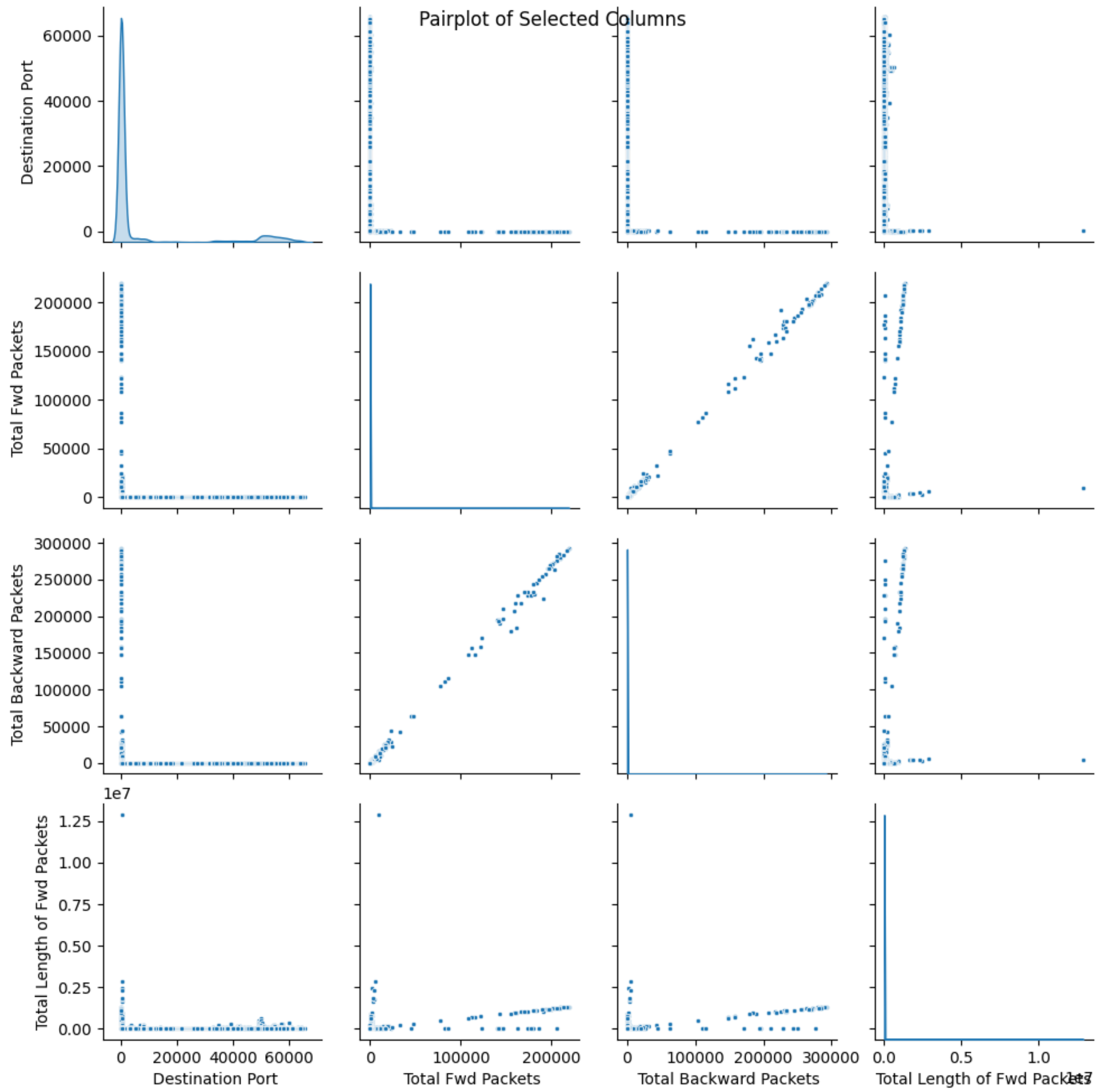
```
 Label
BENIGN                        701324
DoS Hulk                       16424
PortScan                        3937
FTP-Patator                     1638
DDoS                            1355
Bot                              518
SSH-Patator                      237
Web Attack   Brute Force          99
DoS GoldenEye                     73
DoS slowloris                     30
Web Attack   XSS                  20
Web Attack   Sql Injection         9
DoS Slowhttptest                   4
Infiltration                       1
Name: count, dtype: int64
```



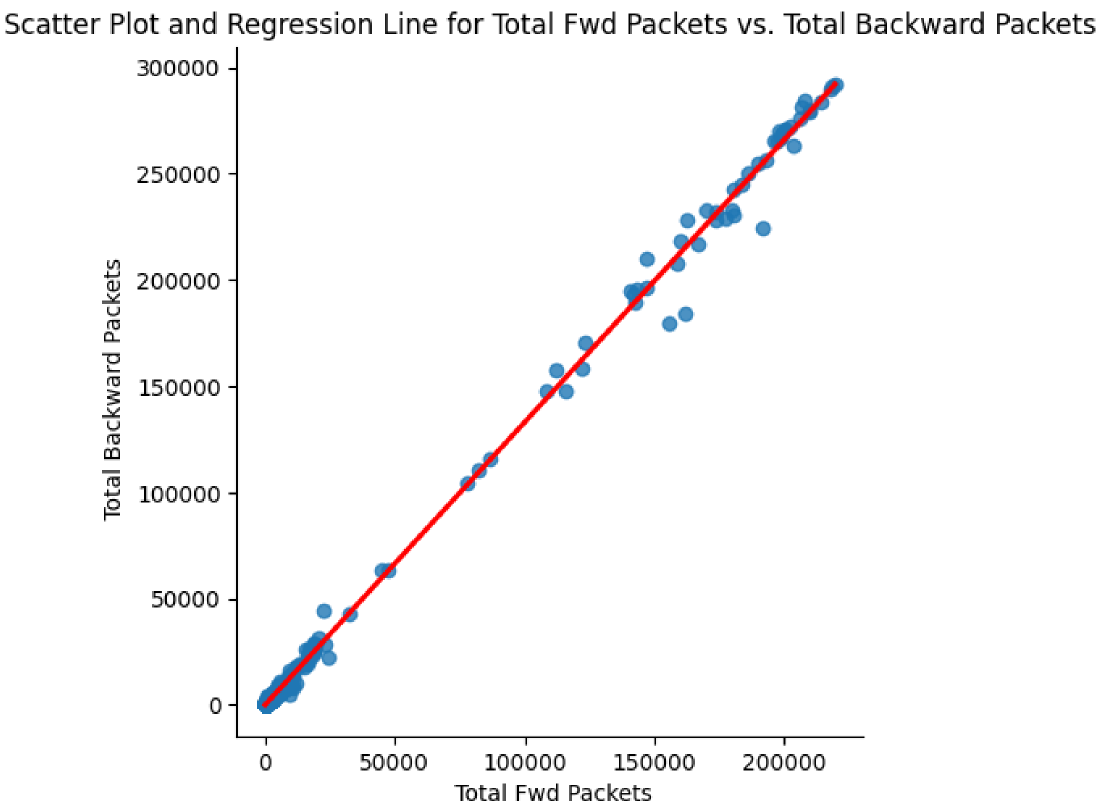
For better understanding the repeated occurrences of the attack to the system I employeed Counterplot for visualization and uses log for scaling since we are dealing with million of data and Benign is the most frequent label, followed by Dos Hulk and portScan. and infiltration is the least frequent attack. Which is important in understanding the frequency class of attack.

Explore correlations between predictors with scatter plots and sns.lmplot() where necessary

[8]:
```
# Select a subset of columns for analysis
Features = [' Destination Port',  ' Total Fwd Packets', ' Total Backward␣
↪Packets', 'Total Length of Fwd Packets']

# Subset the dataframe with selected columns
Features_df = Data_df[Features]
sns.pairplot(Features_df, hue= None, markers='.', diag_kind='kde')
plt.suptitle('Pairplot of Selected Columns')
plt.show()
```



[9]:
```
# Correlation between Total Fwd Packets and Total Backward Packets
sns.lmplot(x=' Total Fwd Packets', y=' Total Backward Packets',␣
 ↪data=Features_df,  line_kws={'color': 'red'})
plt.title('Scatter Plot and Regression Line for Total Fwd Packets vs. Total␣
 ↪Backward Packets')
plt.show()
```

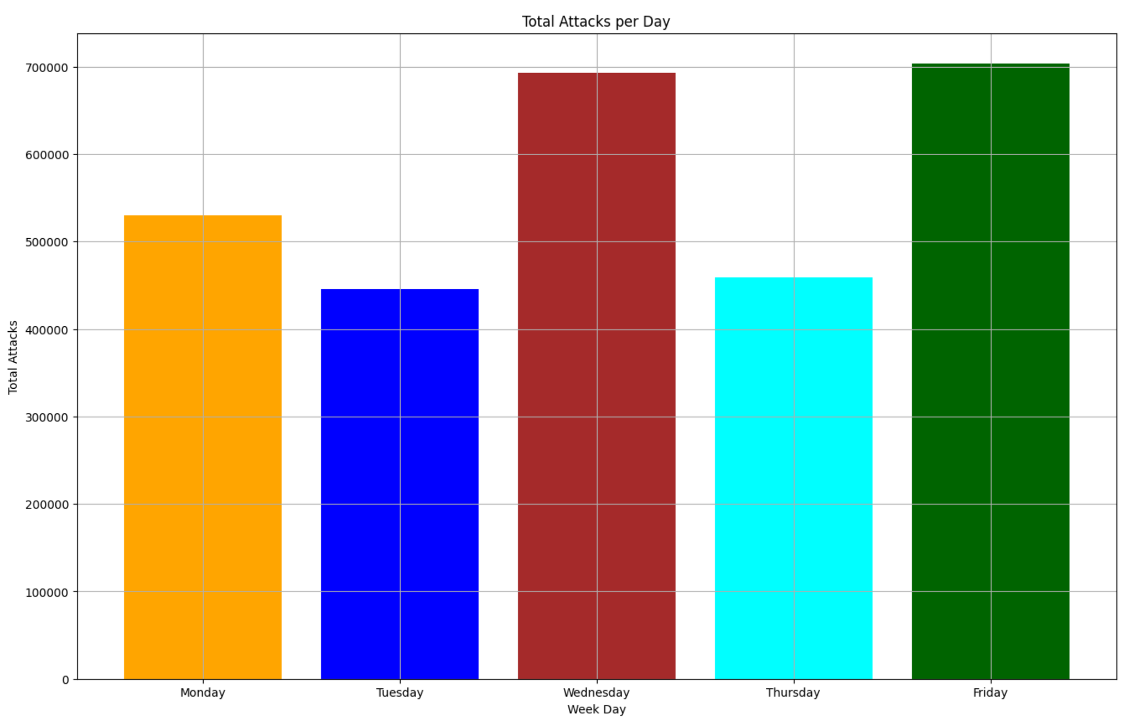


Selected few number of features to explore correlations between predictors and then generates a pair plot to visualize the relationships between these selected columns. Additionally, a scatter plot with a regression line is created to illustrate the correlation between 'Total Fwd Packets' and 'Total Backward Packets.' The pair plot provides a comprehensive overview of the relationships and distributions, while the scatter plot with a regression line emphasizes the linear relationship between the specified columns. Overall, these visualizations aid in understanding the patterns and associations within the samped selected features.

Visualize attacks counts based on day of the week.

[28]:
```
Thursday_merge = pd.concat([Thursday_df,Thursday1_df])     # Merge thursday␣
↪dataframe

Fridays_merge = pd.concat([Friday_df, Friday1_df, Friday2_df]  ) # Merge friday␣
↪dataframe
data = [Monday_df, Tuesday_df, Wednesday_df, Thursday_merge, Fridays_merge]  #␣
↪Merge all dataframes

plt.figure(figsize=(16, 10))

attacks_pday = []     # create an empty list to store the number of attacks per␣
↪day
```

```
colors = {'Monday_df': 'orange', 'Tuesday_df': 'blue', 'Wednesday_df': 'brown',␣
↪'Thursday_merge': 'cyan', 'Fridays_merge': 'darkgreen'}

for day in data:     # loop through the dataframes and count the number of␣
↪attacks per day
    day.columns = day.columns.str.strip()
    total_attacks = day['Label'].count()
    attacks_pday.append(total_attacks)


plt.bar(['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'], attacks_pday,␣
↪color=[colors[day] for day in colors])

plt.title('Total Attacks per Day')
plt.xlabel('Week Day')
plt.ylabel('Total Attacks')
plt.grid()
plt.show()

 # loop through the dataframes and count the number of attacks per day
for day, total_attacks in zip(['Monday', 'Tuesday', 'Wednesday', 'Thursday',␣
↪'Friday'], attacks_pday):
    print(f'Total Attacks on {day}: {total_attacks}')
```



```
Total Attacks on Monday: 529918
Total Attacks on Tuesday: 445909
Total Attacks on Wednesday: 692703
Total Attacks on Thursday: 458968
Total Attacks on Friday: 703245
```

Assessing attacks accur on each Days of the week and visualize them using bargraph as it can seen for thursday datasets are two for morning and afternoon we merged them and the same for friday we have 3 dataset where we combined both. and ploted the bargraph to see the total attacks on each day of the week. Friday experiences the highest number of attacks compared to other days of the week, closely followed by Thursday. In contrast, Tuesday stands out as the day with fewer attacks. The bar graph provides a clear representation of the total attacks on each day of the week, highlighting the varying levels of security incidents across different workdays.

[11]:
```python
display(Data_df.columns) # display the column names
```

```
Index([' Destination Port', ' Flow Duration', ' Total Fwd Packets',
       ' Total Backward Packets', 'Total Length of Fwd Packets',
       ' Total Length of Bwd Packets', ' Fwd Packet Length Max',
       ' Fwd Packet Length Min', ' Fwd Packet Length Mean',
       ' Fwd Packet Length Std', 'Bwd Packet Length Max',
       ' Bwd Packet Length Min', ' Bwd Packet Length Mean',
       ' Bwd Packet Length Std', 'Flow Bytes/s', ' Flow Packets/s',
       ' Flow IAT Mean', ' Flow IAT Std', ' Flow IAT Max', ' Flow IAT Min',
       'Fwd IAT Total', ' Fwd IAT Mean', ' Fwd IAT Std', ' Fwd IAT Max',
       ' Fwd IAT Min', 'Bwd IAT Total', ' Bwd IAT Mean', ' Bwd IAT Std',
       ' Bwd IAT Max', ' Bwd IAT Min', 'Fwd PSH Flags', ' Bwd PSH Flags',
       ' Fwd URG Flags', ' Bwd URG Flags', ' Fwd Header Length',
       ' Bwd Header Length', 'Fwd Packets/s', ' Bwd Packets/s',
       ' Min Packet Length', ' Max Packet Length', ' Packet Length Mean',
       ' Packet Length Std', ' Packet Length Variance', 'FIN Flag Count',
       ' SYN Flag Count', ' RST Flag Count', ' PSH Flag Count',
       ' ACK Flag Count', ' URG Flag Count', ' CWE Flag Count',
       ' ECE Flag Count', ' Down/Up Ratio', ' Average Packet Size',
       ' Avg Fwd Segment Size', ' Avg Bwd Segment Size',
       ' Fwd Header Length.1', 'Fwd Avg Bytes/Bulk', ' Fwd Avg Packets/Bulk',
       ' Fwd Avg Bulk Rate', ' Bwd Avg Bytes/Bulk', ' Bwd Avg Packets/Bulk',
       'Bwd Avg Bulk Rate', 'Subflow Fwd Packets', ' Subflow Fwd Bytes',
       ' Subflow Bwd Packets', ' Subflow Bwd Bytes', 'Init_Win_bytes_forward',
       ' Init_Win_bytes_backward', ' act_data_pkt_fwd',
       ' min_seg_size_forward', 'Active Mean', ' Active Std', ' Active Max',
       ' Active Min', 'Idle Mean', ' Idle Std', ' Idle Max', ' Idle Min',
       ' Label'],
      dtype='object')
```

[12]:
```python
Data_df
```

[12]:

```
        Destination Port   Flow Duration   Total Fwd Packets  \
0                  49188               4                   2
1                  49188               1                   2
2                  49188               1                   2
3                  49188               1                   2
4                  49486               3                   2
…                    …              …                  …
225740             61374              61                   1
225741             61378              72                   1
225742             61375              75                   1
225743             61323              48                   2
225744             61326              68                   1

        Total Backward Packets  Total Length of Fwd Packets  \
0                            0                           12
1                            0                           12
2                            0                           12
3                            0                           12
4                            0                           12
…                          …                           …
225740                       1                            6
225741                       1                            6
225742                       1                            6
225743                       0                           12
225744                       1                            6

        Total Length of Bwd Packets   Fwd Packet Length Max  \
0                                 0                       6
1                                 0                       6
2                                 0                       6
3                                 0                       6
4                                 0                       6
…                               …                       …
225740                            6                       6
225741                            6                       6
225742                            6                       6
225743                            0                       6
225744                            6                       6

        Fwd Packet Length Min   Fwd Packet Length Mean  \
0                           6                      6.0
1                           6                      6.0
2                           6                      6.0
3                           6                      6.0
4                           6                      6.0
…                         …                      …
225740                      6                      6.0
```

```
225741                       6                       6.0
225742                       6                       6.0
225743                       6                       6.0
225744                       6                       6.0

         Fwd Packet Length Std  …   min_seg_size_forward  Active Mean  \
0                          0.0  …                     20          0.0
1                          0.0  …                     20          0.0
2                          0.0  …                     20          0.0
3                          0.0  …                     20          0.0
4                          0.0  …                     20          0.0
…                            …  …                      …           …
225740                     0.0  …                     20          0.0
225741                     0.0  …                     20          0.0
225742                     0.0  …                     20          0.0
225743                     0.0  …                     20          0.0
225744                     0.0  …                     20          0.0

         Active Std   Active Max   Active Min  Idle Mean   Idle Std  \
0               0.0            0            0        0.0        0.0
1               0.0            0            0        0.0        0.0
2               0.0            0            0        0.0        0.0
3               0.0            0            0        0.0        0.0
4               0.0            0            0        0.0        0.0
…                 …            …            …          …          …
225740          0.0            0            0        0.0        0.0
225741          0.0            0            0        0.0        0.0
225742          0.0            0            0        0.0        0.0
225743          0.0            0            0        0.0        0.0
225744          0.0            0            0        0.0        0.0

         Idle Max   Idle Min   Label
0               0          0  BENIGN
1               0          0  BENIGN
2               0          0  BENIGN
3               0          0  BENIGN
4               0          0  BENIGN
…               …          …      …
225740          0          0  BENIGN
225741          0          0  BENIGN
225742          0          0  BENIGN
225743          0          0  BENIGN
225744          0          0  BENIGN

[2830743 rows x 79 columns]
```

0.0.2 Q3 Perform pre-processing on the data using scaling and label encoding as appropriate

Handle Missing Value

[13]:
```
missing_values = Data_df.isnull().sum()        # Checking for missing value
display( missing_values)

Data_df = Data_df.dropna()          # Dropping missing value
```

```
 Destination Port              0
 Flow Duration                 0
 Total Fwd Packets             0
 Total Backward Packets        0
Total Length of Fwd Packets    0
                              ..
Idle Mean                      0
 Idle Std                      0
 Idle Max                      0
 Idle Min                      0
 Label                         0
Length: 79, dtype: int64
```

Handle of duplicate rows

[14]:
```
Data_df = Data_df.drop_duplicates()  # drop duplicate rows
print(Data_df.shape)
```

```
(2522009, 79)
```

For ensuring the uniqueness of the data it is necessary to remove duplicate and ensure that data are well cleaned. Duplicate rows might introduce bias or inaccuracies in analyses, modeling which could result to inaccurate output and by dropping them ensure reliable dataset for modelling.

Converting Data type

[15]:
```
# Convert int64 columns to int8
int_columns = Data_df.select_dtypes(include='int64').columns
Data_df[int_columns] = Data_df[int_columns].astype('int8')

# Convert float64 columns to float16
float_columns = Data_df.select_dtypes(include='float64').columns
Data_df[float_columns] = Data_df[float_columns].astype('float16')
```

[16]:
```
Data_df.dtypes    # printing datatype
```

[16]:
```
 Destination Port                 int8
 Flow Duration                    int8
 Total Fwd Packets                int8
 Total Backward Packets           int8
```

```
Total Length of Fwd Packets       int8
                                ...
Idle Mean                      float16
 Idle Std                      float16
 Idle Max                         int8
 Idle Min                         int8
 Label                          object
Length: 79, dtype: object
```

Conversion of column's datatype where there is float 64 to float16 and int64 to int8 it is necessary for optimizing the memory usage by data especially for large dataset like this reduce the size and complexity of the dataset.

Scaling and label encoder

[17]:
```python
# Extracting the independent and dependent variables
Data_df.replace([np.inf, -np.inf], np.nan, inplace=True)
Data_df.dropna(inplace=True)

X = Data_df.drop(' Label', axis=1)
y = Data_df[' Label']
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the data into training and testing
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y[:
 ↪len(X_scaled)], test_size=0.2, random_state=40)
class_weights = class_weight.compute_class_weight('balanced', classes=np.
 ↪unique(y_train), y=y_train)
```

For performing preprocessing of the dataset, I started by addressing data quality issues by replacing infinite values with NaN and subsequently dropping those rows with NaN accordingly which is crucial step considering the inability of certain classification models like Decision Trees and Logistic Regression to handle NaN values.

[1]https://towardsdatascience.com/pros-and-cons-of-various-classification-ml-algorithms-3b5bfb3c87d6

Standard scaling was then applied to the features, ensuring consistent scale, followed by the division of the dataset into training and testing sets.

Class weights were computed to handle the imbalanced nature of the dataset, a vital consideration for training models to account for minority classes.

Finally, categorical labels were encoded using Label Encoder to facilitate compatibility with algorithms that require numerical input.

[3]https://www.sciencedirect.com/topics/computer-science/class-imbalance-problem#:~:text=Many%20practical%20classification%20problems%20are,and%20ignore%20the%20small%20one

0.0.3 Q4. Create models based on three different machine learning algorithms and compare their performance.

Random Forest

[18]:
```
# Initialize the random forest classifier
random_forest = RandomForestClassifier()

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=40)

class_weights = 'balanced'  # Set class weights to balanced

# Initialize the random forest classifier with class weights
random_forest = RandomForestClassifier(class_weight=class_weights)

# Fit the model on the training set
random_forest.fit(X_train, y_train)

# Make predictions on the test set
y_pred_rf = random_forest.predict(X_test)

# Calculate the specified metrics
avg_recall_rf = recall_score(y_test, y_pred_rf, average='weighted')
avg_precision_rf = precision_score(y_test, y_pred_rf, average='weighted')
avg_f1_rf = f1_score(y_test, y_pred_rf, average='weighted')

# Display results
print("\nRandom Forest Classifier:")
print(f"Average Recall: {avg_recall_rf}")
print(f"Average Precision: {avg_precision_rf}")
print(f"Average F1-score: {avg_f1_rf}")
```

```
Random Forest Classifier:
Average Recall: 0.9983808067027712
Average Precision: 0.9983301096183572
Average F1-score: 0.9983482310573907
```

Logistic Regression

[19]:
```
logistic_regression = LogisticRegression()

# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

class_weights = 'balanced'

# Initialize the random forest classifier with class weights
logistic_regression = LogisticRegression(class_weight=class_weights)


# Fit the model on the training set
logistic_regression.fit(X_train, y_train)

# Make predictions on the test set
y_pred_lr = logistic_regression.predict(X_test)

# Calculate the specified metrics
avg_recall_lr = recall_score(y_test, y_pred_lr, average = 'weighted')
avg_precision_lr = precision_score(y_test, y_pred_lr, average = 'weighted')
avg_f1_lr = f1_score(y_test, y_pred_lr, average = 'weighted')

# Display results
print("\nLogistic Regression Classifier:")
print(f"Average Recall: {avg_recall_lr}")
print(f"Average Precision: {avg_precision_lr}")
print(f"Average F1-score: {avg_f1_lr}")
```

```
Logistic Regression Classifier:
Average Recall: 0.6936555183485608
Average Precision: 0.9772215928941822
Average F1-score: 0.8059028973611762
```

Decision Tree

[20]:
```
# Initialize the decision tree classifier
decision_tree = DecisionTreeClassifier()

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

class_weights = 'balanced'

# Initialize the decision tree classifier with class weights
decision_tree = DecisionTreeClassifier(class_weight=class_weights)  # Set class␣
↪weights to balanced

# Fit the model on the training set
```

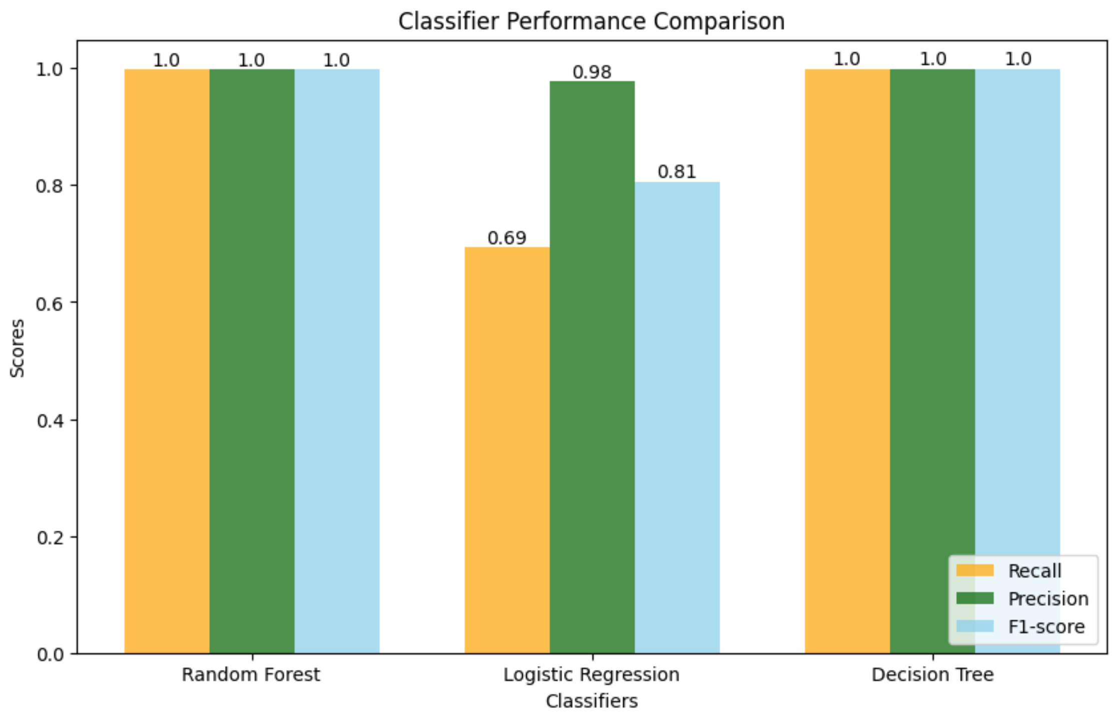
```python
decision_tree.fit(X_train, y_train)

# Make predictions on the test set
y_pred_dt = decision_tree.predict(X_test)

# Calculate the specified metrics
avg_recall_dt = recall_score(y_test, y_pred_dt, average='weighted')
avg_precision_dt = precision_score(y_test, y_pred_dt,  average='weighted')
avg_f1_dt = f1_score(y_test, y_pred_dt, average='weighted')

# Display results
print("\nDecision Tree Classifier:")
print(f"Average Recall: {avg_recall_dt}")
print(f"Average Precision: {avg_precision_dt}")
print(f"Average F1-score: {avg_f1_dt}")
```

```
Decision Tree Classifier:
Average Recall: 0.9990078134689322
Average Precision: 0.9990180295685162
Average F1-score: 0.9990106546131458
```

Visualization of performance of different Classiffication models

[21]:
```python
# Visualization
classifier_names = ["Random Forest", "Logistic Regression", "Decision Tree"]
avg_recalls = [avg_recall_rf, avg_recall_lr, avg_recall_dt]
avg_precisions = [avg_precision_rf, avg_precision_lr, avg_precision_dt]
avg_f1_scores = [avg_f1_rf, avg_f1_lr, avg_f1_dt]

plt.figure(figsize=(10, 6))
bar_width = 0.25
# Set the position of the bars on the x-axis
bar1 = plt.bar([i - bar_width for i in range(len(classifier_names))],␣
 ↪avg_recalls, width=bar_width, label='Recall', alpha=0.7, color = 'orange')
bar2 = plt.bar([i for i in range(len(classifier_names))], avg_precisions,␣
 ↪width=bar_width, label='Precision', alpha=0.7, color = 'darkgreen')
bar3 = plt.bar([i + bar_width for i in range(len(classifier_names))],␣
 ↪avg_f1_scores, width=bar_width, label='F1-score', alpha=0.7,  color =␣
 ↪'skyblue')

plt.xlabel('Classifiers')
plt.ylabel('Scores')
plt.title('Classifier Performance Comparison')
plt.xticks([i for i in range(len(classifier_names))], classifier_names)
plt.legend(loc = 'lower right')
```

```
# Annotate the bars with scores
for bars in [bar1, bar2, bar3]:
    for bar in bars:
        yval = bar.get_height()
        plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2),␣
 ↪ha='center', va='bottom')

plt.show()
```



i) Comparing the performance of three classification model (Random forest, Rogistic Regression, Decision tree) using three metric Recall, Precision, F1-score. I have trained the model each individually using Train test split where the test data were 20% and rest 80% by training set. due to class imbalance between predictor where it is Benign has more data than than others which can impact the perfomance by training on imbalance class. I have used class weights to balance the data. I have also used random state to get same result for each time I run the code. I have also used weighted average to calculate the scores.

benefit of handling class imbalance is that it will reduce the bias and variance of the model. It will also reduce the variance of the model. It will also reduce the variance of the model. [3]https://www.sciencedirect.com/topics/computer-science/class-imbalance-problem#:~:text=Many%20practical%20classification%20problems%20are,and%20ignore%20the%20small%20one

The output shows that for - Random Forest Classifier: Recall: 0.9983, Precision: 0.9982, F1-score: 0.99823 - Logistic Regression Classifier: Recall: 0.6936 Precision: 0.9772, F1-score: 0.8059 - Decision Tree Classifier: Recall: 0.9989, Precision: 0.9989, F1-score: 0.9989

This result show the Random Forest classifier exhibited exceptional performance across all metrics, showcasing robustness and accuracy This makes it particularly well-suited for intrusion detection tasks, where the timely and accurate identification of potential threats is paramount Despite displaying high precision, the Logistic Regression model faced challenges in recall, leading to a comparatively lower F1-score. In the context of intrusion detection, where the focus is on identifying actual threats recall, this model might have limitations. However, in other contexts, such as monitoring, where the goal is to detect anomalous behavior precision, the Logistic Regression model may be employeed for that purpose.

For decision Tree the metric shows the result which is nearly close to that of Random Forest classifier which means for detecting intrusion, decision tree also can be trusted to deliver the desired output.

Inconclusion, an intrusion detection system serves as a crucial defense tool by identifying potential attacks and providing insights into preventive measures for safeguarding the system. It proves valuable not only in detecting attacks but also in monitoring the system for anomalous behavior and pinpointing emerging threats. To ensure effective defense against potential risks, it is imperative to select a model that accurately captures and addresses threats. In this regard, the Random Forest model emerges as the optimal candidate, offering reliability and trustworthiness in delivering accurate results for potential system threats.

ii) Recommendation Based on the result of the metric on each classifier it is recommendend to use Random Forest which is the one that give the better result and decision tree compare to logistic regression. To enhance the models further, addressing class imbalance through advanced sampling techniques and fine-tuning hyperparameters could be considered. Ensemble methods involve combining predictions from multiple models to make more accurate predictions. This can be achieved through techniques like bagging, boosting, or stacking.

Reference [1]https://jmlr.org/papers/volume20/18-444/18-444.pdf

[2]https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10294237/

[22]:
```
Data_df
```

[22]:
```
        Destination Port  Flow Duration  Total Fwd Packets  \
29                    80            -61                  3
122                 -123             37                  5
123                  -60            123                  5
125                 -123            100                 17
197                 -117           -103                 49
…                    …              …                  …
225727               -10             64                  1
225728               -98             79                  2
225729                14            -34                  2
225730                12            -38                  2
225736               -14             45                  1

        Total Backward Packets  Total Length of Fwd Packets  \
29                           4                          103
122                          2                            0
```

```
123                           2                             0
125                           8                            28
197                          30                           -72
...                         ...                           ...
225727                        1                             0
225728                        0                            12
225729                        0                             0
225730                        0                             0
225736                        1                             0

         Total Length of Bwd Packets   Fwd Packet Length Max  \
29                               -65                      97
122                                0                       0
123                                0                       0
125                              -54                      64
197                               54                     -62
...                              ...                     ...
225727                             0                       0
225728                             0                       6
225729                             0                       0
225730                             0                       0
225736                             0                       0

         Fwd Packet Length Min   Fwd Packet Length Mean  \
29                           0                34.343750
122                          0                 0.000000
123                          0                 0.000000
125                          0                16.703125
197                          0                50.781250
...                        ...                      ...
225727                       0                 0.000000
225728                       6                 6.000000
225729                       0                 0.000000
225730                       0                 0.000000
225736                       0                 0.000000

         Fwd Packet Length Std  ...   min_seg_size_forward  Active Mean  \
29                   54.343750  ...                     20          0.0
122                   0.000000  ...                     32          0.0
123                   0.000000  ...                     32          0.0
125                  25.265625  ...                     32          0.0
197                  66.000000  ...                     32          0.0
...                        ...  ...                    ...          ...
225727                0.000000  ...                     32          0.0
225728                0.000000  ...                     20          0.0
225729                0.000000  ...                     32          0.0
225730                0.000000  ...                     32          0.0
```

```
225736                0.000000  …                    32          0.0

         Active Std   Active Max   Active Min  Idle Mean   Idle Std  \
29              0.0            0            0        0.0        0.0
122             0.0            0            0        0.0        0.0
123             0.0            0            0        0.0        0.0
125             0.0            0            0        0.0        0.0
197             0.0            0            0        0.0        0.0
…             …          …          …        …        …
225727          0.0            0            0        0.0        0.0
225728          0.0            0            0        0.0        0.0
225729          0.0            0            0        0.0        0.0
225730          0.0            0            0        0.0        0.0
225736          0.0            0            0        0.0        0.0

         Idle Max   Idle Min   Label
29              0          0  BENIGN
122             0          0  BENIGN
123             0          0  BENIGN
125             0          0  BENIGN
197             0          0  BENIGN
…            …        …      …
225727          0          0  BENIGN
225728          0          0  BENIGN
225729          0          0  BENIGN
225730          0          0  BENIGN
225736          0          0  BENIGN

[725669 rows x 79 columns]
```

0.0.4 Q5 Apply feature selection approaches on the dataset and build ML models for the same algorithms

Random forest classifier with PCA

[23]:
```python
selector = SelectKBest(score_func=f_classif, k=10)
X_selected = selector.fit_transform(X, y)

# Perform PCA
pca = PCA(n_components=10)
X_pca = pca.fit_transform(X_selected)

print(f"Explained Variance Ratio: {sum(pca.explained_variance_ratio_):.2f}")

# Split the data into training and testing sets
X_train_pca, X_test_pca, y_train, y_test = train_test_split(X_pca, y,
  test_size=0.2, random_state=42)
```

```
# Initialize the random forest classifier
random_forest = RandomForestClassifier()

# Fit the model on the training set
random_forest.fit(X_train_pca, y_train)

# Make predictions on the test set
y_pred_rf = random_forest.predict(X_test_pca)

# Calculate the specified metrics
recall_rf = recall_score(y_test, y_pred_rf, average= 'weighted')
precision_rf = precision_score(y_test, y_pred_rf, average = 'weighted')
f1_rf = f1_score(y_test, y_pred_rf, average= 'weighted')

# Display results
print("\nRandom Forest Classifier with PCA:")
print(f"Average Recall: {recall_rf}")
print(f"Average Precision: {precision_rf}")
print(f"Average F1-score: {f1_rf}")
```

```
Explained Variance Ratio: 1.00

Random Forest Classifier with PCA:
Average Recall: 0.9879421775738283
Average Precision: 0.9876162561110196
Average F1-score: 0.9863037514181103
```

Logistic Regression with PCA

[24]:
```
selector = SelectKBest(score_func=f_classif, k=10)
X_selected = selector.fit_transform(X, y)

# Perform PCA
pca = PCA(n_components=10)
X_pca = pca.fit_transform(X_selected)

print(f"Explained Variance Ratio: {sum(pca.explained_variance_ratio_):.2f}")

# Split the data into training and testing sets
X_train_pca, X_test_pca, y_train, y_test = train_test_split(X_pca, y,
  ↪test_size=0.2, random_state=42)

# Initialize the logistic regression classifier
logistic_regression = LogisticRegression()

# Fit the model on the training set
logistic_regression.fit(X_train_pca, y_train)
```

```python
# Make predictions on the test set
y_pred_lr = logistic_regression.predict(X_test_pca)

# Calculate the specified metrics
recall_lr = recall_score(y_test, y_pred_lr, average = 'weighted')
precision_lr = precision_score(y_test, y_pred_lr, average = 'weighted')
f1_lr = f1_score(y_test, y_pred_lr, average = 'weighted')

# Display results
print("\nLogistic Regression Classifier with PCA:")
print(f"Average Recall: {recall_lr}")
print(f"Average Precision: {precision_lr}")
print(f"Average F1-score: {f1_lr}")
```

```
Explained Variance Ratio: 1.00

Logistic Regression Classifier with PCA:
Average Recall: 0.9370237160141662
Average Precision: 0.956467568130246
Average F1-score: 0.9445321477848511
```

Decision Tree

[25]:
```python
selector = SelectKBest(score_func=f_classif, k=10)
X_selected = selector.fit_transform(X, y)

# Perform PCA
pca = PCA(n_components=10)
X_pca = pca.fit_transform(X_selected)

print(f"Explained Variance Ratio: {sum(pca.explained_variance_ratio_):.4f}")

# Split the data into training and testing sets
X_train_pca, X_test_pca, y_train, y_test = train_test_split(X_pca, y,␣
↪test_size=0.2, random_state=42)

# Initialize the decision tree classifier
decision_tree = DecisionTreeClassifier()

# Fit the model on the training set
decision_tree.fit(X_train_pca, y_train)

# Make predictions on the test set
y_pred_dt = decision_tree.predict(X_test_pca)

# Calculate the specified metrics
recall_dt = recall_score(y_test, y_pred_dt, average = 'weighted')
precision_dt = precision_score(y_test, y_pred_dt, average = 'weighted')
```

```
f1_dt = f1_score(y_test, y_pred_dt, average = 'weighted')

# Display results
print("\nDecision Tree Classifier with PCA:")
print(f"Average Recall: {recall_dt}")
print(f"Average Precision: {precision_dt}")
print(f"Average F1-score: {f1_dt}")
```

```
Explained Variance Ratio: 1.0000

Decision Tree Classifier with PCA:
Average Recall: 0.9878801659156365
Average Precision: 0.9874621484542625
Average F1-score: 0.9862404983965508
```

Visualization of the Classification Model using Performance metric

[26]:
```
# Visualization
classifier_names = ["Random Forest PCA", "Logistic Regression PCA", "Decision␣
↪Tree PCA"]
avg_recalls = [recall_rf, recall_lr, recall_dt]
avg_precisions = [precision_rf, precision_lr, precision_dt]
avg_f1_scores = [f1_rf, f1_lr, f1_dt]

plt.figure(figsize=(10, 6))
bar_width = 0.25

bar1 = plt.bar([i - bar_width for i in range(len(classifier_names))],␣
↪avg_recalls, width=bar_width, label='Recall', alpha=0.7, color  = 'orange')
bar2 = plt.bar([i for i in range(len(classifier_names))], avg_precisions,␣
↪width=bar_width, label='Precision', alpha=0.7, color = 'darkgreen')
bar3 = plt.bar([i + bar_width for i in range(len(classifier_names))],␣
↪avg_f1_scores, width=bar_width, label='F1-score', alpha=0.7, color =␣
↪'skyblue')

plt.xlabel('Classifiers')
plt.ylabel('Scores')
plt.title('Classifier Performance Comparison with PCA')
plt.xticks([i for i in range(len(classifier_names))], classifier_names)
plt.legend(loc = 'lower right')

# Annotate the bars with scores
for bars in [bar1, bar2, bar3]:
    for bar in bars:
        yval = bar.get_height()
        plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2),␣
↪ha='center', va='bottom')
```

```
plt.grid()
plt.show()
```

Q5. i) Apply feature selection approaches on the dataset and build ML models Finding the performance of each classification model using principal component analysis (PCA) and feature selection. I used Kbest for features selection and then employee PCA for dimensional reduction based on large dataset it was the better approach for model analysis. then I used the following classification models: - Random Forest - Logistic Regression - Decision Tree

Train the model on the features selected and then the evaluate the performance of every model using the following metrics: Recall, Precision, F1-score and then return the output to determine the better classifier that could be used for intrusion detection system.

The output shows for - Random Forest classifier: Recall: 0.9879, Precision: 0.9876, F1-score: 0.9863 - Logistic Regression classifier: Recall: 0.9370, Precision: 0.9564, F1-score: 0.9445 - Decision Tree classifier: Recall: 0.9878, Precision: 0.9874, F1-score: 0.9862

This result show the Random Forest classifier exhibited exceptional performance across all metrics, where recall is near to 1 suggests that the Random Forest classifier is effective in capturing most instances of intrusions, for precision which is 0.98 precision score indicates that when the model predicts an intrusion, it is likely to be correct or the accuracy of the predicted intrusion. and F1 score shows how model is perform well.

For logistic regression classifier demonstrates slightly lower recall compared to Random Forest, suggesting it may result to miss of instances of intrusions in system. However in term of precision it increases and F1 score too but still less than that of Decision Tree.

For decision Tree the metric shows the result which is nearly close to that of Random Forest classifier which means for detecting intrusion, decision tree also can be trusted to deliver the desired output.

Inconclusion, All three models demonstrate solid performance, with Random Forest standing out for its balanced and robust results across recall, precision, and F1-score. This suggests that Random Forest is a promising choice for an Intrusion Detection System, offering a strong compromise between identifying intrusions and minimizing false alarms.

ii)Recommendations on the algorithm For the increase the performance of the model it it recommendend to assess accuracy of prediction and incorporate advanced techniques like Hyperparameter Tuning using methods such as grid search, random search, and Bayesian optimization. The implementation of robust cross-validation strategies is crucial to ensure consistent performance across diverse subsets of the data, revealing any signs of overfitting or underfitting even if we dropped it for the purpose of getting the result but for the sake of the system it is better to employee it. Regularly monitoring and updating the model with new data is essential, considering the evolving nature of intrusion patterns over time. It's pivotal to subject the model to testing on various data subsets, including training, validation, and testing sets, to validate that it effectively generalizes without overfitting or underfitting the data.

[2]https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10294237/

0.0.5 6 Comparing the result for 3 different classification models and after different feature selection approaches

The Random Forest classifier continues to exhibit superior performance, maintaining its dominance even after feature selection. And the Decision Tree classifier closely follows Random Forest in terms of performance. However, the standout improvement is observed in the Logistic Regression classifier after features selection. Specifically, the recall score has significantly increased from 0.64 to an impressive 0.94. This substantial enhancement indicates a heightened capability of the model to detect instances of intrusion within the system. The results suggest that feature selection has a particularly positive impact on the Logistic Regression model, enhancing its overall effectiveness in identifying potential threats.

[ ]:

```
```