In [1]:
```
import pandas as pd
```

In [2]:
```
import pandas as pd
mssql = pd.read_csv(r"C:\Users\Nitin Agarwal\Downloads\dataset_modified_ddos
bios = pd.read_csv(r"C:\Users\Nitin Agarwal\Downloads\dataset_modified_ddos\
portmap = pd.read_csv(r"C:\Users\Nitin Agarwal\Downloads\dataset_modified_dd
```

```
C:\Users\Nitin Agarwal\AppData\Local\Temp\ipykernel_16088\3849838795.py:2: D
typeWarning: Columns (86) have mixed types. Specify dtype option on import o
r set low_memory=False.
  mssql = pd.read_csv(r"C:\Users\Nitin Agarwal\Downloads\dataset_modified_dd
os\mssql7.csv")
C:\Users\Nitin Agarwal\AppData\Local\Temp\ipykernel_16088\3849838795.py:4: D
typeWarning: Columns (86) have mixed types. Specify dtype option on import o
r set low_memory=False.
  portmap = pd.read_csv(r"C:\Users\Nitin Agarwal\Downloads\dataset_modified_
ddos\portmap7.csv")
```

In [3]:
```
ds = pd.concat([mssql, bios, portmap])
```

In [4]:
```
ds.to_csv(r'C:\Users\Nitin Agarwal\Downloads\dataset_modified_ddos\complete_
```

In [5]:
```
df = pd.read_csv(r'C:\Users\Nitin Agarwal\Downloads\dataset_modified_ddos\cc
```

```
C:\Users\Nitin Agarwal\AppData\Local\Temp\ipykernel_16088\1575314885.py:1: D
typeWarning: Columns (87) have mixed types. Specify dtype option on import o
r set low_memory=False.
  df = pd.read_csv(r'C:\Users\Nitin Agarwal\Downloads\dataset_modified_ddos
\complete_dataset.csv')
```

In [6]:
```
print("columns: ", len(df.columns))
df.columns
```

```
columns:  90
```

Out[6]:
```
Index(['Unnamed: 0.2', 'Unnamed: 0.1', 'Unnamed: 0', 'Flow ID', ' Source I
P',
       ' Source Port', ' Destination IP', ' Destination Port', ' Protocol',
       ' Timestamp', ' 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/Bul
k',
       ' Fwd Avg Bulk Rate', ' Bwd Avg Bytes/Bulk', ' Bwd Avg Packets/Bul
k',
       'Bwd Avg Bulk Rate', 'Subflow Fwd Packets', ' Subflow Fwd Bytes',
       ' Subflow Bwd Packets', ' Subflow Bwd Bytes', 'Init_Win_bytes_forwar
d',
       ' Init_Win_bytes_backward', ' act_data_pkt_fwd',
       ' min_seg_size_forward', 'Active Mean', ' Active Std', ' Active Ma
x',
       ' Active Min', 'Idle Mean', ' Idle Std', ' Idle Max', ' Idle Min',
       'SimillarHTTP', ' Inbound', ' Label'],
      dtype='object')
```

In [7]:
```
df.describe()
```

Out[7]:

	Unnamed: 0.2	Unnamed: 0.1	Unnamed: 0	Source Port	Destina
count	575082.000000	575082.000000	575082.000000	575082.000000	575082.00
mean	95846.500000	95846.500000	111499.386915	15817.463598	32566.64
std	55337.339363	55337.339363	83501.242736	23902.539721	19072.18
min	0.000000	0.000000	0.000000	0.000000	0.00
25%	47923.000000	47923.000000	47195.000000	648.000000	16073.00
50%	95846.500000	95846.500000	94216.000000	864.000000	32565.50
75%	143770.000000	143770.000000	154472.750000	31245.000000	49116.00
max	191693.000000	191693.000000	347713.000000	65532.000000	65535.00

8 rows × 84 columns

In [8]:
```
df=df.drop(' Bwd PSH Flags',axis=1)
df=df.drop(' Fwd URG Flags',axis=1)
df=df.drop(' Bwd URG Flags',axis=1)
df=df.drop('FIN Flag Count',axis=1)
df=df.drop(' PSH Flag Count',axis=1)
df=df.drop(' ECE Flag Count',axis=1)
df=df.drop('Fwd Avg Bytes/Bulk',axis=1)
df=df.drop(' Fwd Avg Packets/Bulk',axis=1)
df=df.drop(' Fwd Avg Bulk Rate',axis=1)
df=df.drop(' Bwd Avg Bytes/Bulk',axis=1)
df=df.drop(' Bwd Avg Packets/Bulk',axis=1)
df=df.drop('Bwd Avg Bulk Rate',axis=1)
df=df.drop('Unnamed: 0',axis=1)
df=df.drop('Unnamed: 0.1',axis=1)
df=df.drop('SimillarHTTP', axis=1)
df=df.drop('Flow ID', axis=1)
df=df.drop(' Timestamp', axis=1)
```

In [9]:
```
df.to_csv(r'C:\Users\Nitin Agarwal\Downloads\dataset_modified_ddos\clean_2.c
```

In [10]:
```
df = df.rename(columns={" Label": "Label"})
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 575082 entries, 0 to 575081
Data columns (total 73 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   Unnamed: 0.2                   575082 non-null  int64
 1    Source IP                     575082 non-null  object
 2    Source Port                   575082 non-null  int64
 3    Destination IP                575082 non-null  object
 4    Destination Port              575082 non-null  int64
 5    Protocol                      575082 non-null  int64
 6    Flow Duration                 575082 non-null  int64
 7    Total Fwd Packets             575082 non-null  int64
 8    Total Backward Packets        575082 non-null  int64
 9   Total Length of Fwd Packets    575082 non-null  float64
 10   Total Length of Bwd Packets   575082 non-null  float64
 11   Fwd Packet Length Max         575082 non-null  float64
 12   Fwd Packet Length Min         575082 non-null  float64
 13   Fwd Packet Length Mean        575082 non-null  float64
 14   Fwd Packet Length Std         575082 non-null  float64
 15  Bwd Packet Length Max          575082 non-null  float64
 16   Bwd Packet Length Min         575082 non-null  float64
 17   Bwd Packet Length Mean        575082 non-null  float64
 18   Bwd Packet Length Std         575082 non-null  float64
 19  Flow Bytes/s                   575081 non-null  float64
 20   Flow Packets/s                575082 non-null  float64
 21   Flow IAT Mean                 575082 non-null  float64
 22   Flow IAT Std                  575082 non-null  float64
 23   Flow IAT Max                  575082 non-null  float64
 24   Flow IAT Min                  575082 non-null  float64
 25  Fwd IAT Total                  575082 non-null  float64
 26   Fwd IAT Mean                  575082 non-null  float64
 27   Fwd IAT Std                   575082 non-null  float64
 28   Fwd IAT Max                   575082 non-null  float64
 29   Fwd IAT Min                   575082 non-null  float64
 30  Bwd IAT Total                  575082 non-null  float64
 31   Bwd IAT Mean                  575082 non-null  float64
 32   Bwd IAT Std                   575082 non-null  float64
 33   Bwd IAT Max                   575082 non-null  float64
 34   Bwd IAT Min                   575082 non-null  float64
 35  Fwd PSH Flags                  575082 non-null  int64
 36   Fwd Header Length             575082 non-null  int64
 37   Bwd Header Length             575082 non-null  int64
 38  Fwd Packets/s                  575082 non-null  float64
 39   Bwd Packets/s                 575082 non-null  float64
 40   Min Packet Length             575082 non-null  float64
 41   Max Packet Length             575082 non-null  float64
 42   Packet Length Mean            575082 non-null  float64
 43   Packet Length Std             575082 non-null  float64
 44   Packet Length Variance        575082 non-null  float64
 45   SYN Flag Count                575082 non-null  int64
 46   RST Flag Count                575082 non-null  int64
 47   ACK Flag Count                575082 non-null  int64
 48   URG Flag Count                575082 non-null  int64
 49   CWE Flag Count                575082 non-null  int64
 50   Down/Up Ratio                 575082 non-null  float64
```

```
 51   Average Packet Size          575082 non-null  float64
 52   Avg Fwd Segment Size         575082 non-null  float64
 53   Avg Bwd Segment Size         575082 non-null  float64
 54   Fwd Header Length.1          575082 non-null  int64
 55  Subflow Fwd Packets           575082 non-null  int64
 56   Subflow Fwd Bytes            575082 non-null  int64
 57   Subflow Bwd Packets          575082 non-null  int64
 58   Subflow Bwd Bytes            575082 non-null  int64
 59  Init_Win_bytes_forward        575082 non-null  int64
 60   Init_Win_bytes_backward      575082 non-null  int64
 61   act_data_pkt_fwd             575082 non-null  int64
 62   min_seg_size_forward         575082 non-null  int64
 63  Active Mean                   575082 non-null  float64
 64   Active Std                   575082 non-null  float64
 65   Active Max                   575082 non-null  float64
 66   Active Min                   575082 non-null  float64
 67  Idle Mean                     575082 non-null  float64
 68   Idle Std                     575082 non-null  float64
 69   Idle Max                     575082 non-null  float64
 70   Idle Min                     575082 non-null  float64
 71   Inbound                      575082 non-null  int64
 72  Label                         575082 non-null  object
dtypes: float64(45), int64(25), object(3)
memory usage: 320.3+ MB
```

In [11]:
```
df.Label.unique()
df['Label'] = df['Label'].replace('BENIGN', '0')
df['Label'] = df['Label'].replace('NetBIOS', '1')
df['Label'] = df['Label'].replace('MSSQL', '3')
df['Label'] = df['Label'].replace('LDAP', '2')
df['Label'] = df['Label'].replace('Portmap', '4')
df.Label.unique()
```

Out[11]:
```
array(['2', '0', '3', '1', '4'], dtype=object)
```

In [12]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
a = df
```

In [13]:
```
import netaddr
#print(int(netaddr.IPAddress('192.168.4.54')))
ips = df[' Destination IP']
converted=[]
for i in range(len(ips)):
    converted.append(int(netaddr.IPAddress(ips[i])))
print("done")
```

```
done
```

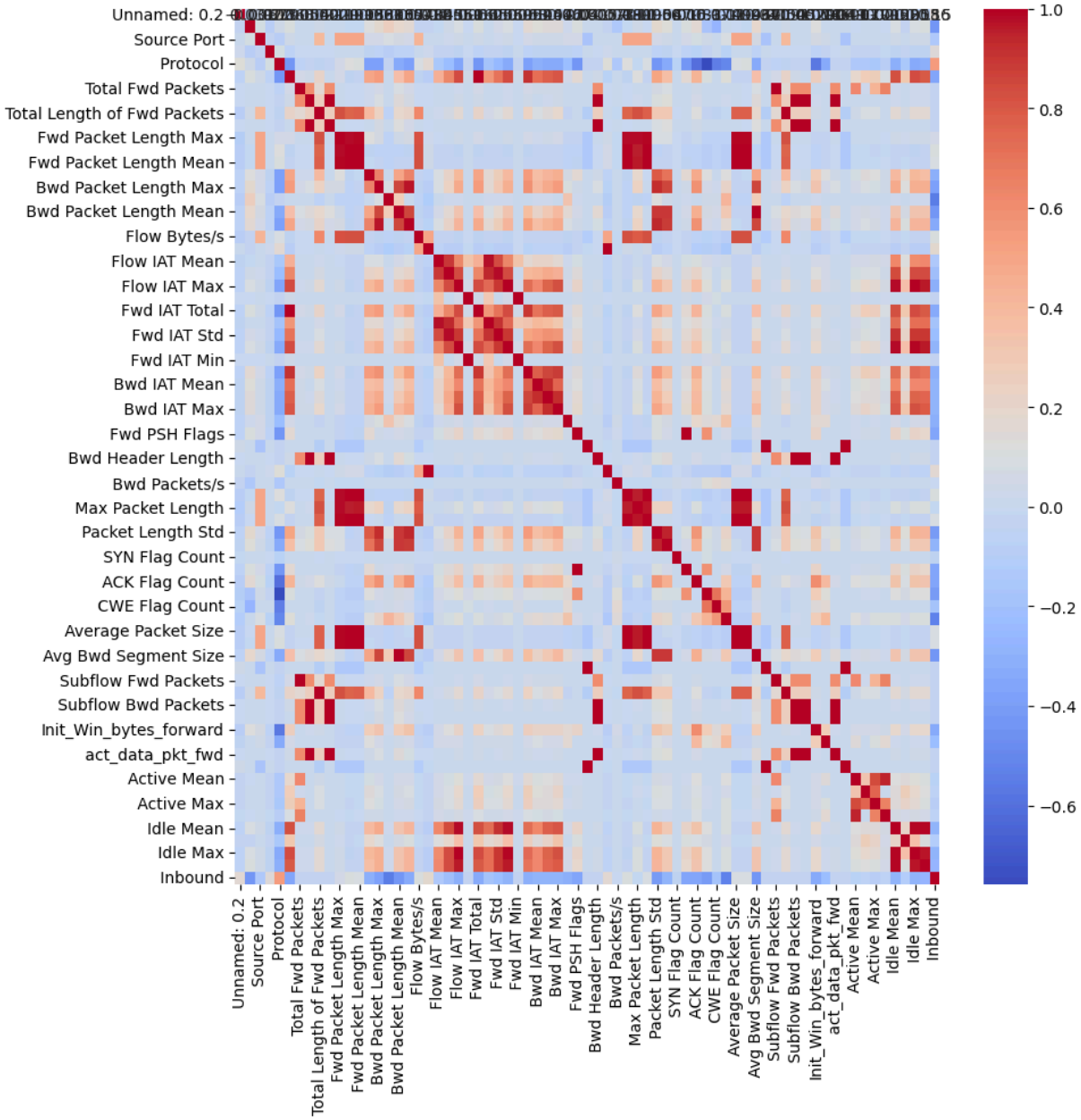
In [14]:
```
import netaddr
# Get unique IP addresses from the 'Source IP' column
ips = df[' Source IP'].unique()
l = len(ips)
print('Starting loop, length is', l)
# Loop over each unique IP address and replace with its integer representati
for i in range(l):
```

```
    try:
        # Check if IP is already a string representation
        ip = str(ips[i])
        # Convert IP address to integer using netaddr
        df[' Source IP'] = df[' Source IP'].replace(ip, int(netaddr.IPAddres
    except netaddr.core.AddrFormatError:
        # If it's an integer, convert it back to string format IP
        ip_int = int(ips[i])
        df[' Source IP'] = df[' Source IP'].replace(ip_int, int(netaddr.IPAc
print('Loop over')
```

```
Starting loop, length is 222
Loop over
```

In [15]:
```
df['Flow Bytes/s'] = df['Flow Bytes/s'].astype('float')
```

In [16]:
```
import seaborn as sn
import matplotlib.pyplot as plt
# Select only numeric columns for correlation
numeric_df = df.select_dtypes(include=['float64', 'int64'])
# Compute the correlation matrix
corr_mat = numeric_df.corr()
# Plot heatmap
plt.figure(figsize=(10,10)) # Set figure size for better readability
sn.heatmap(corr_mat, annot=True, cmap="coolwarm")
plt.show()
```



In [17]:
```
df.columns
```

Out[17]:
```
Index(['Unnamed: 0.2', ' Source IP', ' Source Port', ' Destination IP',
       ' Destination Port', ' Protocol', ' 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 Ma
x',
       ' 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',
       ' 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 Varianc
e',
       ' SYN Flag Count', ' RST Flag Count', ' ACK Flag Count',
       ' URG Flag Count', ' CWE Flag Count', ' Down/Up Ratio',
       ' Average Packet Size', ' Avg Fwd Segment Size',
       ' Avg Bwd Segment Size', ' Fwd Header Length.1', 'Subflow Fwd Packet
s',
       ' 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 St
d',
       ' Idle Max', ' Idle Min', ' Inbound', 'Label'],
      dtype='object')
```

In [18]:
```
df=df.drop(' Source IP',axis=1)
df=df.drop(' Flow Duration',axis=1)
df=df.drop(' Total Fwd Packets',axis=1)
df=df.drop(' Total Backward Packets',axis=1)
df=df.drop(' Total Length of Bwd Packets',axis=1)
df=df.drop(' Fwd Packet Length Std',axis=1)
df=df.drop(' Flow IAT Max',axis=1)
df=df.drop(' Flow IAT Min',axis=1)
df=df.drop('Fwd IAT Total',axis=1)
df=df.drop(' Fwd IAT Max',axis=1)
df=df.drop(' Fwd IAT Min',axis=1)
df=df.drop('Bwd IAT Total',axis=1)
df=df.drop(' Bwd IAT Mean',axis=1)
df=df.drop(' Bwd IAT Std',axis=1)
df=df.drop(' Bwd IAT Max',axis=1)
df=df.drop(' Bwd IAT Min',axis=1)
df=df.drop(' Fwd Header Length',axis=1)
df=df.drop(' Bwd Header Length',axis=1)
df=df.drop(' Bwd Packets/s',axis=1)
df=df.drop(' SYN Flag Count',axis=1)
df=df.drop(' Down/Up Ratio',axis=1)
df=df.drop(' Fwd Header Length.1',axis=1)
df=df.drop('Subflow Fwd Packets',axis=1)
df=df.drop(' Subflow Bwd Packets',axis=1)
df=df.drop(' Subflow Bwd Bytes',axis=1)
df=df.drop(' act_data_pkt_fwd',axis=1)
```

```
df=df.drop(' min_seg_size_forward',axis=1)
df=df.drop('Active Mean',axis=1)
df=df.drop(' Active Std',axis=1)
df=df.drop(' Active Max',axis=1)
df=df.drop(' Active Min',axis=1)
df=df.drop('Idle Mean',axis=1)
df=df.drop(' Idle Max',axis=1)
df=df.drop(' Idle Min',axis=1)
df = df.drop(' Packet Length Std',axis=1)
```

In [19]:
```
df.to_csv(r'C:\Users\Nitin Agarwal\Downloads\dataset_modified_ddos\clean_fir
```

In [20]:
```
df.head()
```

Out[20]:

	Unnamed: 0.2	Source Port	Destination IP	Destination Port	Protocol	Total Length of Fwd Packets	Fwd Packet Length Max	P Le
0	0	615	192.168.50.4	28754	17	2944.0	1472.0	1
1	1	0	192.168.50.4	0	0	0.0	0.0	
2	2	900	192.168.50.4	42364	17	2944.0	1472.0	1
3	3	616	192.168.50.4	10537	17	2944.0	1472.0	1
4	4	617	192.168.50.4	14928	17	2944.0	1472.0	1

5 rows × 38 columns

In [21]:
```
df.describe()
```

Out[21]:

	Unnamed: 0.2	Source Port	Destination Port	Protocol	Total Le of Pac
count	575082.000000	575082.000000	575082.000000	575082.000000	575082.00
mean	95846.500000	15817.463598	32566.649758	16.926605	714.60
std	55337.339363	23902.539721	19072.188581	0.905467	632.99
min	0.000000	0.000000	0.000000	0.000000	0.00
25%	47923.000000	648.000000	16073.000000	17.000000	458.00
50%	95846.500000	864.000000	32565.500000	17.000000	458.00
75%	143770.000000	31245.000000	49116.000000	17.000000	878.00
max	191693.000000	65532.000000	65535.000000	17.000000	150726.00

8 rows × 36 columns

In [55]:
```
import pandas as pd
df= pd.read_csv(r"C:\Users\Nitin Agarwal\Downloads\clean_final.csv")
#df = dff.iloc[:191694,]
```

```
x=df.iloc[:,df.columns != 'Label']
y=df.iloc[:,-1]
print("x\n",x.info())
y = pd.DataFrame(y)
print('y\n',y.info())
```

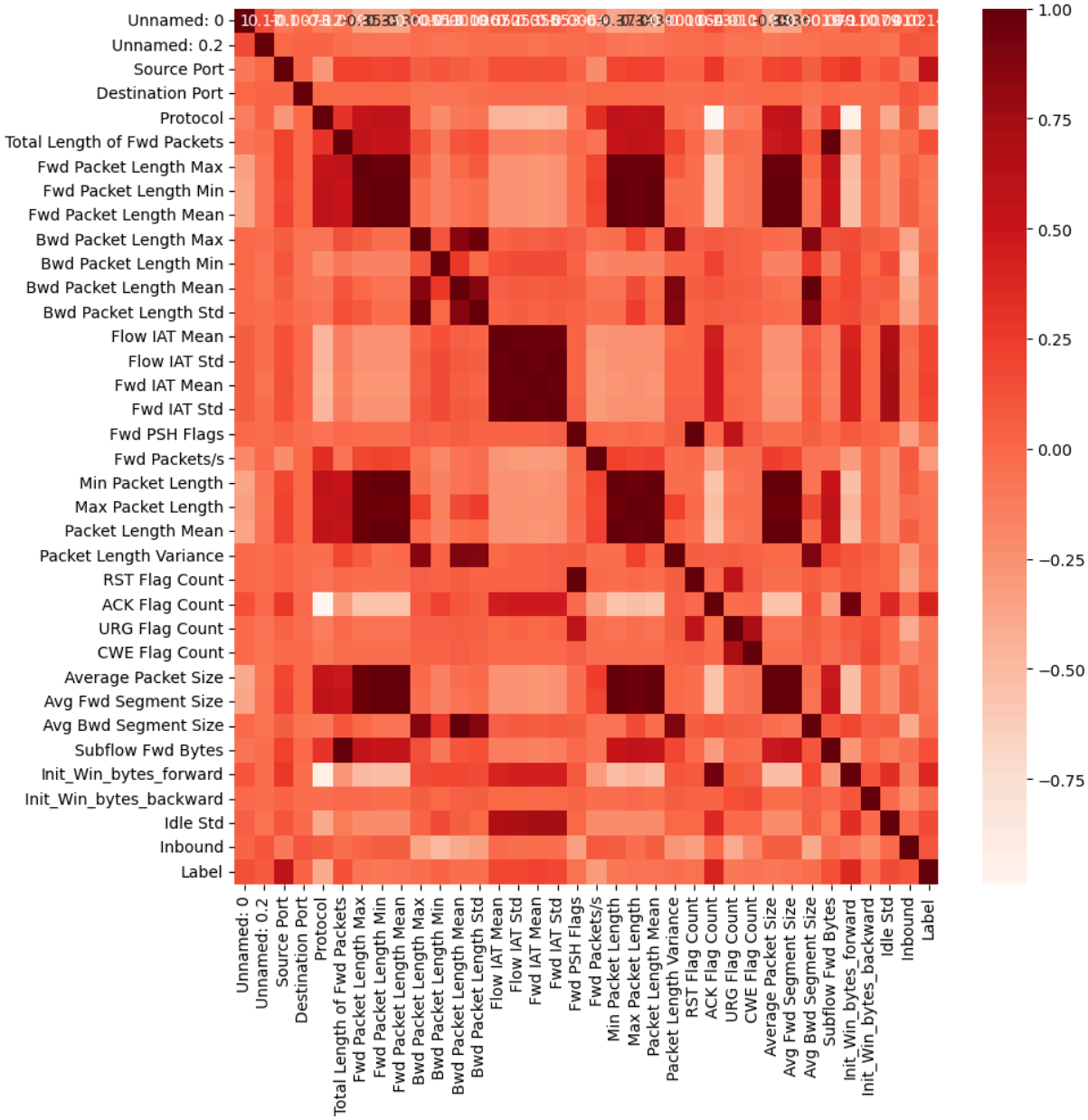
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1150164 entries, 0 to 1150163
Data columns (total 38 columns):
 #   Column                       Non-Null Count    Dtype  
---  ------                       --------------    -----  
 0   Unnamed: 0                   1150164 non-null  int64  
 1   Unnamed: 0.2                 1150164 non-null  int64  
 2    Source Port                 1150164 non-null  int64  
 3    Destination IP              1150164 non-null  object 
 4    Destination Port            1150164 non-null  int64  
 5    Protocol                    1150164 non-null  int64  
 6   Total Length of Fwd Packets  1150164 non-null  float64
 7    Fwd Packet Length Max       1150164 non-null  float64
 8    Fwd Packet Length Min       1150164 non-null  float64
 9    Fwd Packet Length Mean      1150164 non-null  float64
 10  Bwd Packet Length Max        1150164 non-null  float64
 11   Bwd Packet Length Min       1150164 non-null  float64
 12   Bwd Packet Length Mean      1150164 non-null  float64
 13   Bwd Packet Length Std       1150164 non-null  float64
 14  Flow Bytes/s                 1150154 non-null  float64
 15   Flow Packets/s              1150164 non-null  float64
 16   Flow IAT Mean               1150164 non-null  float64
 17   Flow IAT Std                1150164 non-null  float64
 18   Fwd IAT Mean                1150164 non-null  float64
 19   Fwd IAT Std                 1150164 non-null  float64
 20  Fwd PSH Flags                1150164 non-null  int64  
 21  Fwd Packets/s                1150164 non-null  float64
 22   Min Packet Length           1150164 non-null  float64
 23   Max Packet Length           1150164 non-null  float64
 24   Packet Length Mean          1150164 non-null  float64
 25   Packet Length Variance      1150164 non-null  float64
 26   RST Flag Count              1150164 non-null  int64  
 27   ACK Flag Count              1150164 non-null  int64  
 28   URG Flag Count              1150164 non-null  int64  
 29   CWE Flag Count              1150164 non-null  int64  
 30   Average Packet Size         1150164 non-null  float64
 31   Avg Fwd Segment Size        1150164 non-null  float64
 32   Avg Bwd Segment Size        1150164 non-null  float64
 33   Subflow Fwd Bytes           1150164 non-null  int64  
 34  Init_Win_bytes_forward       1150164 non-null  int64  
 35   Init_Win_bytes_backward     1150164 non-null  int64  
 36   Idle Std                    1150164 non-null  float64
 37   Inbound                     1150164 non-null  int64  
dtypes: float64(23), int64(14), object(1)
memory usage: 333.5+ MB
x
 None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1150164 entries, 0 to 1150163
Data columns (total 1 columns):
 #   Column  Non-Null Count    Dtype
---  ------  --------------    -----
 0   Label   1150164 non-null  int64
dtypes: int64(1)
memory usage: 8.8 MB
```

```
y
 None
```

In [56]:
```
# Select only numeric columns for normalization
numeric_df = df.select_dtypes(include=['float64', 'int64'])
# Normalize the numeric data
normalized_df = (numeric_df - numeric_df.mean()) / numeric_df.std()
# Drop specific columns if needed (ensure they exist in numeric_df)
columns_to_drop = [' Flow Packets/s', 'Flow Bytes/s']
normalized_df = normalized_df.drop(columns=[col for col in columns_to_drop i
print("Normalization complete")
```

```
Normalization complete
```

In [57]:
```
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure(figsize=(10,10))
cor = normalized_df.corr()
sn.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.show()
```



In [58]:
```
normalized_df.head()
```

Out[58]:

	Unnamed: 0	Unnamed: 0.2	Source Port	Destination Port	Protocol	Total Length of Fwd Packets	Fw Packe Lengt Ma
0	-1.732049	-1.732041	-0.868731	-0.209151	0.454516	2.527913	5.27438
1	-1.732046	-1.732023	-0.894156	-1.722085	-3.645767	-0.746297	-1.26122
2	-1.732043	-1.732005	-0.856949	0.506959	0.454516	2.527913	5.27438
3	-1.732040	-1.731987	-0.868690	-1.167665	0.454516	2.527913	5.27438
4	-1.732037	-1.731969	-0.868648	-0.936626	0.454516	2.527913	5.27438

5 rows × 36 columns

In [59]:
```
normalized_df=normalized_df.drop('Unnamed: 0', axis=1)
normalized_df=normalized_df.drop('Unnamed: 0.2', axis=1)
```

In [60]:
```
normalized_df.describe()
```

Out[60]:

	Source Port	Destination Port	Protocol	Total Length of Fwd Packets	Fwd Pack Length M
count	1.150164e+06	1.150164e+06	1.150164e+06	1.150164e+06	1.150164e+
mean	5.139889e-18	5.310395e-17	-5.197219e-16	1.027978e-16	1.089261e-
std	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+
min	-8.941557e-01	-1.722085e+00	-3.645767e+00	-7.462968e-01	-1.261222e+
25%	-8.647625e-01	-8.647522e-01	4.545160e-01	-2.369258e-01	-2.444735e-
50%	-8.522362e-01	6.836382e-04	4.545160e-01	-2.369258e-01	-2.444735e-
75%	9.825486e-01	8.681715e-01	4.545160e-01	1.679017e-01	4.659182e-
max	1.814986e+00	1.726136e+00	4.545160e-01	1.671170e+02	1.464711e+

8 rows × 34 columns

In [61]:
```
normalized_x=normalized_df.iloc[:,normalized_df.columns != 'Label']
```

In [62]:
```
normalized_x.describe()
```

Out[62]:

	Source Port	Destination Port	Protocol	Total Length of Fwd Packets	Fwd Pack Length M
count	1.150164e+06	1.150164e+06	1.150164e+06	1.150164e+06	1.150164e+
mean	5.139889e-18	5.310395e-17	-5.197219e-16	1.027978e-16	1.089261e-
std	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+
min	-8.941557e-01	-1.722085e+00	-3.645767e+00	-7.462968e-01	-1.261222e+
25%	-8.647625e-01	-8.647522e-01	4.545160e-01	-2.369258e-01	-2.444735e-
50%	-8.522362e-01	6.836382e-04	4.545160e-01	-2.369258e-01	-2.444735e-
75%	9.825486e-01	8.681715e-01	4.545160e-01	1.679017e-01	4.659182e-
max	1.814986e+00	1.726136e+00	4.545160e-01	1.671170e+02	1.464711e+

8 rows × 33 columns

In [63]:
```
from sklearn.model_selection import train_test_split
import numpy as np
X_train, X_test, y_train, y_test = train_test_split(normalized_x, y,test_siz
y_train = np.array(y_train)
```

In [64]:
```
normalized_x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1150164 entries, 0 to 1150163
Data columns (total 33 columns):
 #   Column                        Non-Null Count    Dtype  
---  ------                        --------------    -----  
 0    Source Port                  1150164 non-null  float64
 1    Destination Port             1150164 non-null  float64
 2    Protocol                     1150164 non-null  float64
 3   Total Length of Fwd Packets   1150164 non-null  float64
 4    Fwd Packet Length Max        1150164 non-null  float64
 5    Fwd Packet Length Min        1150164 non-null  float64
 6    Fwd Packet Length Mean       1150164 non-null  float64
 7   Bwd Packet Length Max         1150164 non-null  float64
 8    Bwd Packet Length Min        1150164 non-null  float64
 9    Bwd Packet Length Mean       1150164 non-null  float64
 10   Bwd Packet Length Std        1150164 non-null  float64
 11   Flow IAT Mean                1150164 non-null  float64
 12   Flow IAT Std                 1150164 non-null  float64
 13   Fwd IAT Mean                 1150164 non-null  float64
 14   Fwd IAT Std                  1150164 non-null  float64
 15  Fwd PSH Flags                 1150164 non-null  float64
 16  Fwd Packets/s                 1150164 non-null  float64
 17   Min Packet Length            1150164 non-null  float64
 18   Max Packet Length            1150164 non-null  float64
 19   Packet Length Mean           1150164 non-null  float64
 20   Packet Length Variance       1150164 non-null  float64
 21   RST Flag Count               1150164 non-null  float64
 22   ACK Flag Count               1150164 non-null  float64
 23   URG Flag Count               1150164 non-null  float64
 24   CWE Flag Count               1150164 non-null  float64
 25   Average Packet Size          1150164 non-null  float64
 26   Avg Fwd Segment Size         1150164 non-null  float64
 27   Avg Bwd Segment Size         1150164 non-null  float64
 28   Subflow Fwd Bytes            1150164 non-null  float64
 29  Init_Win_bytes_forward        1150164 non-null  float64
 30   Init_Win_bytes_backward      1150164 non-null  float64
 31   Idle Std                     1150164 non-null  float64
 32   Inbound                      1150164 non-null  float64
dtypes: float64(33)
memory usage: 289.6 MB
```

In [65]:
```
y.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1150164 entries, 0 to 1150163
Data columns (total 1 columns):
 #   Column  Non-Null Count    Dtype
---  ------  --------------    -----
 0   Label   1150164 non-null  int64
dtypes: int64(1)
memory usage: 8.8 MB
```

In [66]:
```
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
```

```
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (862623, 33)
X_test shape: (287541, 33)
y_train shape: (862623, 1)
y_test shape: (287541, 1)
```

In [67]:
```
print("First few entries of y_train:", y_train[:5])
print("First few entries of y_test:", y_test[:5])
```

```
First few entries of y_train: [[5]
 [1]
 [6]
 [1]
 [5]]
First few entries of y_test:         Label
720827      5
644278      5
919597      6
681471      5
402562      4
```

In [68]:
```
# Convert y_train to int type if necessary
y_train = y_train.astype(int)

# Print the updated data type and first few entries
print("Data types in y_train:", set(type(x) for x in y_train))
print("First few entries of y_train:", y_train[:5])
```

```
Data types in y_train: {<class 'numpy.ndarray'>}
First few entries of y_train: [[5]
 [1]
 [6]
 [1]
 [5]]
```

In [69]:
```
y_train = y_train.astype(str)
y_test = y_test.astype(str)
```

In [70]:
```
from sklearn.preprocessing import LabelEncoder

# Convert y_train and y_test to 1D arrays if they are DataFrames
if isinstance(y_train, pd.DataFrame):  # Check if y_train is a DataFrame
    y_train = y_train.values.ravel()  # Convert to NumPy array and flatten

if isinstance(y_test, pd.DataFrame):  # Check if y_test is a DataFrame
    y_test = y_test.values.ravel()  # Convert to NumPy array and flatten

# Initialize and fit the label encoder
label_encoder = LabelEncoder()

# Fit and transform y_train
y_train = label_encoder.fit_transform(y_train)

# Transform y_test using the same encoder
y_test = label_encoder.transform(y_test)
```

```
# Check the transformed arrays
print("Transformed y_train:", y_train)
print("Transformed y_test:", y_test)
```

```
C:\Users\Nitin Agarwal\anaconda3\Lib\site-packages\sklearn\preprocessing\_la
bel.py:114: DataConversionWarning: A column-vector y was passed when a 1d ar
ray was expected. Please change the shape of y to (n_samples, ), for example
using ravel().
  y = column_or_1d(y, warn=True)
Transformed y_train: [5 1 6 ... 1 3 1]
Transformed y_test: [5 5 6 ... 3 6 1]
```

In [71]:
```
from sklearn.neighbors import KNeighborsClassifier
from tqdm import tqdm  # Import tqdm for the progress bar

# Initialize the model
model = KNeighborsClassifier(n_neighbors=8)

# Fit the model with a progress bar (for demonstration)
# Here we are simulating a fitting process
n_iterations = 1  # You can change this to fit over multiple iterations if c
for _ in tqdm(range(n_iterations), desc="Fitting model"):
    model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Print predictions
print("Predictions:", y_pred)
```

```
Fitting model: 100%|
| 1/1 [00:00<00:00,  2.29it/s]
Predictions: [5 5 6 ... 3 6 1]
```

In [72]:
```
from sklearn import metrics
from sklearn.metrics import f1_score
print('K Nearest Neighbour Classifier')
# Accuracy
accuracy = metrics.accuracy_score(y_test, y_pred) * 100
print('Accuracy = {:.2f}%'.format(accuracy))
# Confusion Matrix
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
print("Confusion Matrix =\n", conf_matrix)
# Recall
recall = metrics.recall_score(y_test, y_pred, average='weighted')
print("Recall =", recall)
# Classification Report
class_report = metrics.classification_report(y_test, y_pred, digits=2)
print("Classification Report =\n", class_report)
# F1 Score
f1 = f1_score(y_test, y_pred, average='macro')
print("F1 Score = ", f1)
```

```
K Nearest Neighbour Classifier
Accuracy = 82.60%
Confusion Matrix =
 [[ 1744     0     0     0     2     4     0]
 [    1 84460     0    28 10917     0     0]
 [    1     0  2405    23     6     2     0]
 [    3     4    40 50086    35     1  1072]
 [    6 36713    18   111 10259     1     2]
 [    2     0     0     7     2 47811     0]
 [    0     0     0  1039     2     0 40734]]
Recall = 0.8259656883713975
Classification Report =
               precision    recall  f1-score   support

           0       0.99      1.00      0.99      1750
           1       0.70      0.89      0.78     95406
           2       0.98      0.99      0.98      2437
           3       0.98      0.98      0.98     51241
           4       0.48      0.22      0.30     47110
           5       1.00      1.00      1.00     47822
           6       0.97      0.98      0.97     41775

    accuracy                           0.83    287541
   macro avg       0.87      0.86      0.86    287541
weighted avg       0.81      0.83      0.80    287541

F1 Score =  0.8582661574187904
```

In [73]:
```
normalized_df.head()
```

Out[73]:

	Source Port	Destination Port	Protocol	Total Length of Fwd Packets	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean
0	-0.868731	-0.209151	0.454516	2.527913	5.274382	5.368796	5.350017
1	-0.894156	-1.722085	-3.645767	-0.746297	-1.261222	-1.255619	-1.262493
2	-0.856949	0.506959	0.454516	2.527913	5.274382	5.368796	5.350017
3	-0.868690	-1.167665	0.454516	2.527913	5.274382	5.368796	5.350017
4	-0.868648	-0.936626	0.454516	2.527913	5.274382	5.368796	5.350017

5 rows × 34 columns

In [74]:
```
print(df.columns)
```

```
Index(['Unnamed: 0', 'Unnamed: 0.2', ' Source Port', ' Destination IP',
       ' Destination Port', ' Protocol', 'Total Length of Fwd Packets',
       ' Fwd Packet Length Max', ' Fwd Packet Length Min',
       ' Fwd Packet Length Mean', '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', ' Fwd IAT Mean', ' Fwd IAT Std',
       'Fwd PSH Flags', 'Fwd Packets/s', ' Min Packet Length',
       ' Max Packet Length', ' Packet Length Mean', ' Packet Length Varianc
e',
       ' RST Flag Count', ' ACK Flag Count', ' URG Flag Count',
       ' CWE Flag Count', ' Average Packet Size', ' Avg Fwd Segment Size',
       ' Avg Bwd Segment Size', ' Subflow Fwd Bytes', 'Init_Win_bytes_forwar
d',
       ' Init_Win_bytes_backward', ' Idle Std', ' Inbound', 'Label'],
      dtype='object')
```

In [126…
```
y = pd.get_dummies(df['Label'], prefix='Label')
y = y.astype(int)
y.head()
```

Out[126…

	Label_0	Label_1	Label_2	Label_3	Label_4	Label_5	Label_6
0	0	0	1	0	0	0	0
1	0	0	1	0	0	0	0
2	0	0	1	0	0	0	0
3	0	0	1	0	0	0	0
4	0	0	1	0	0	0	0

In [127…
```
print(y.shape)  # What is the shape of y before the train_test_split?
```

```
(1150164, 7)
```

In [129…
```
sample_fraction = 1  # Use 60% of the data
normalized_df_sample = normalized_df.sample(frac=sample_fraction, random_sta
y_sample = y.sample(frac=sample_fraction, random_state=0)

X_train, X_test, y_train, y_test = train_test_split(normalized_df_sample, y_
```

In [130…
```
# Reshape if needed (for CNN or models expecting 3D input)
X_train = np.array(X_train)
X_test = np.array(X_test)

# Assuming 1D input per feature, reshape as (samples, timesteps, features)
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

# Convert y to numpy arrays as well
y_train = np.array(y_train)
y_test = np.array(y_test)

print('xtrain={}, ytrain={}, xtest={}, ytest={}'.format(X_train.shape, y_tra
```

```
xtrain=(862623, 34, 1), ytrain=(862623, 7), xtest=(287541, 34, 1), ytest=(28
7541, 7)
```

In [131…
```
print(y_train.shape)
print(y_test.shape)
```

```
(862623, 7)
(287541, 7)
```

In [132…
```
from keras import optimizers
sgd = optimizers.SGD(learning_rate=0.009, decay=1e-6, momentum=0.9, nesterov
```

In [133…
```
from keras.models import Sequential
from keras.layers import Convolution1D, MaxPooling1D, Flatten, Dense
# Define your model
model = Sequential()
model.add(Convolution1D(64, 3, activation="relu", input_shape=(34, 1)))
model.add(Convolution1D(32, 3, activation="relu"))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(7))
```

In [134…
```
model.compile(loss="mean_absolute_error", optimizer="adam", metrics=['accura
model.fit(X_train, y_train, epochs=1, batch_size=10, validation_data=(X_test
```

```
86263/86263 [==============================] - 538s 6ms/step - loss: 0.0048
- accuracy: 0.9964 - val_loss: 0.0021 - val_accuracy: 0.9986
```

Out[134… <keras.callbacks.History at 0x24738688810>

In [120…
```
# Save the model to a file
model.save(r"C:\Users\Nitin Agarwal\Downloads\dataset_modified_ddos\modelcnn
```

In [121…
```
y_pred = model.predict(X_test)
```

```
899/899 [==============================] - 4s 4ms/step
```

In [122…
```
a = (y_pred > 0.5)
b = (y_test > 0.5)
```

In [123…
```
b=np.argmax(y_test, axis=1)
```

In [124…
```
a=np.argmax(y_pred, axis=1)
a[2]
```

Out[124… 6

In [125…
```
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score

print('Convolution Neural Network')

print('Accuracy: %f' % (accuracy_score(a, b)*100))
print("Confusion Matrix =\n", metrics.confusion_matrix(b, a, labels=None,
                                                       sample_weight=None))
```

```
print("Recall =", metrics.recall_score(b, a, labels=None,
                                        pos_label=1, average='weighted'
                                        sample_weight=None))
print("Classification Report =\n", metrics.classification_report(b, a,
                                                                 labels=None
                                                                 target_name
                                                                 sample_weig
                                                                 digits=2,
                                                                 output_dict

print("F1 Score = ",f1_score(a, b, average='macro'))
```

```
Convolution Neural Network
Accuracy: 99.770467
Confusion Matrix =
 [[ 170    4    0    0    0    3    0]
 [   0 9635    0    0    0    0    0]
 [   1    0  245    4    0    0    0]
 [   0    0    2 5136    1    0    0]
 [   7    0    0   31 4519    8    0]
 [   1    0    0    0    0 4858    0]
 [   0    0    0    4    0    0 4125]]
Recall = 0.9977046671767407
Classification Report =
               precision    recall  f1-score   support

           0       0.95      0.96      0.96       177
           1       1.00      1.00      1.00      9635
           2       0.99      0.98      0.99       250
           3       0.99      1.00      1.00      5139
           4       1.00      0.99      0.99      4565
           5       1.00      1.00      1.00      4859
           6       1.00      1.00      1.00      4129

    accuracy                           1.00     28754
   macro avg       0.99      0.99      0.99     28754
weighted avg       1.00      1.00      1.00     28754

F1 Score =  0.9899714964954326
```

In [ ]: