In [1]:

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

In [2]:

```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:

```
dfps_tr = []
dfps_ts = []
for dirname, _, filenames in os.walk('CIC-DDoS-2019/'):
    for filename in filenames:
        if filename.endswith('-training.parquet'):
            dfp = os.path.join(dirname, filename)
            dfps_tr.append(dfp)
            print(dfp)
        elif filename.endswith('-testing.parquet'):
            dfp = os.path.join(dirname, filename)
            dfps_ts.append(dfp)
            print(dfp)
```

```
CIC-DDoS-2019/DNS-testing.parquet
CIC-DDoS-2019/LDAP-testing.parquet
CIC-DDoS-2019/LDAP-training.parquet
CIC-DDoS-2019/MSSQL-testing.parquet
CIC-DDoS-2019/MSSQL-training.parquet
CIC-DDoS-2019/NetBIOS-testing.parquet
CIC-DDoS-2019/NetBIOS-training.parquet
CIC-DDoS-2019/NTP-testing.parquet
CIC-DDoS-2019/Portmap-training.parquet
CIC-DDoS-2019/SNMP-testing.parquet
CIC-DDoS-2019/Syn-testing.parquet
CIC-DDoS-2019/Syn-training.parquet
CIC-DDoS-2019/TFTP-testing.parquet
CIC-DDoS-2019/UDP-testing.parquet
CIC-DDoS-2019/UDP-training.parquet
CIC-DDoS-2019/UDPLag-testing.parquet
CIC-DDoS-2019/UDPLag-training.parquet
```

In [4]:

```
data = pd.concat([pd.read_parquet(dfp) for dfp in dfps_tr], ignore_index=True)
```

In [5]:

```
null_counts = data.isnull().sum()
# Print the number of null values
print(f"{null_counts.sum()} null entries have been found in the dataset\n")
# Drop null values
data.dropna(inplace=True)          # or df_data = df_data.dropna()

# Find and handle duplicates
duplicate_count = data.duplicated().sum()
# Print the number of duplicate entries
print(f"{duplicate_count} duplicate entries have been found in the dataset\n")
# Remove duplicates
data.drop_duplicates(inplace=True)  # or df_data = df_data.drop_duplicates()
# Display relative message
print(f"All duplicates have been removed\n")

# Reset the indexes
data.reset_index(drop=True, inplace=True)

# Inspect the dataset for categorical columns
print("Categorical columns:",data.select_dtypes(include=['object']).columns.tolist(),'\n')

# Print the first 5 lines
data.head()
```

```
0 null entries have been found in the dataset

3494 duplicate entries have been found in the dataset

All duplicates have been removed

Categorical columns: ['Label']
```

Out[5]:

	Protocol	Flow Duration	Total Fwd Packets	Total Backward Packets	Fwd Packets Length Total	Bwd Packets Length Total	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean	Fwd Packet Length Std	...	Fwd Seg Size Min	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
0	17	49	2	0	458.0	0.0	229.0	229.0	229.0	0.0	...	8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	NetBIOS
1	17	1	2	0	2944.0	0.0	1472.0	1472.0	1472.0	0.0	...	1480	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	LDAP
2	17	1	2	0	458.0	0.0	229.0	229.0	229.0	0.0	...	14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	NetBIOS
3	17	1	2	0	2944.0	0.0	1472.0	1472.0	1472.0	0.0	...	14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	LDAP
4	17	1	2	0	2944.0	0.0	1472.0	1472.0	1472.0	0.0	...	32	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	LDAP

5 rows × 78 columns

In [6]:

```
data.columns
```

Out[6]:

```
Index(['Protocol', 'Flow Duration', 'Total Fwd Packets',
       'Total Backward Packets', 'Fwd Packets Length Total',
       'Bwd Packets Length Total', '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',
       'Packet Length Min', 'Packet Length Max', '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',
       'Avg Packet Size', 'Avg Fwd Segment Size', 'Avg Bwd Segment Size',
       '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 Fwd Win Bytes', 'Init Bwd Win Bytes',
       'Fwd Act Data Packets', 'Fwd Seg Size Min', 'Active Mean', 'Active Std',
       'Active Max', 'Active Min', 'Idle Mean', 'Idle Std', 'Idle Max',
       'Idle Min', 'Label'],
      dtype='object')
```

In [7]:

```
data['Label'].value_counts()
```

Out[7]:

```
Syn        47246
Benign     45101
UDP        17795
MSSQL       8434
LDAP        1885
Portmap      685
NetBIOS      475
UDPLag        55
Name: Label, dtype: int64
```

In [8]:

```
# changing attack labels to their respective attack class
def change_label(df):
    df['Label'].replace(['Syn','UDP','MSSQL','LDAP','Portmap','NetBIOS','UDPLag'],'Attack',inplace=True)
    df['Label'].replace(['Benign'],'Normal',inplace=True)
```

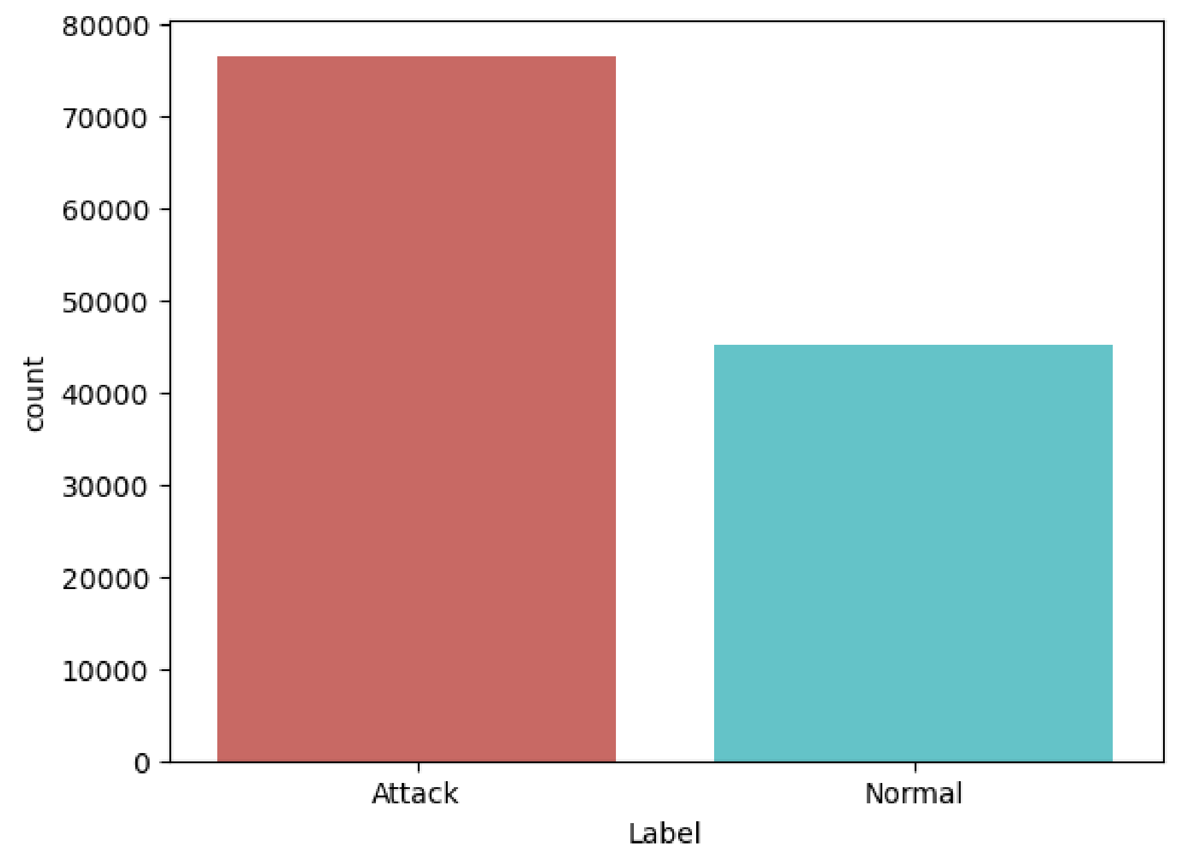
In [9]:

```
change_label(data)
```

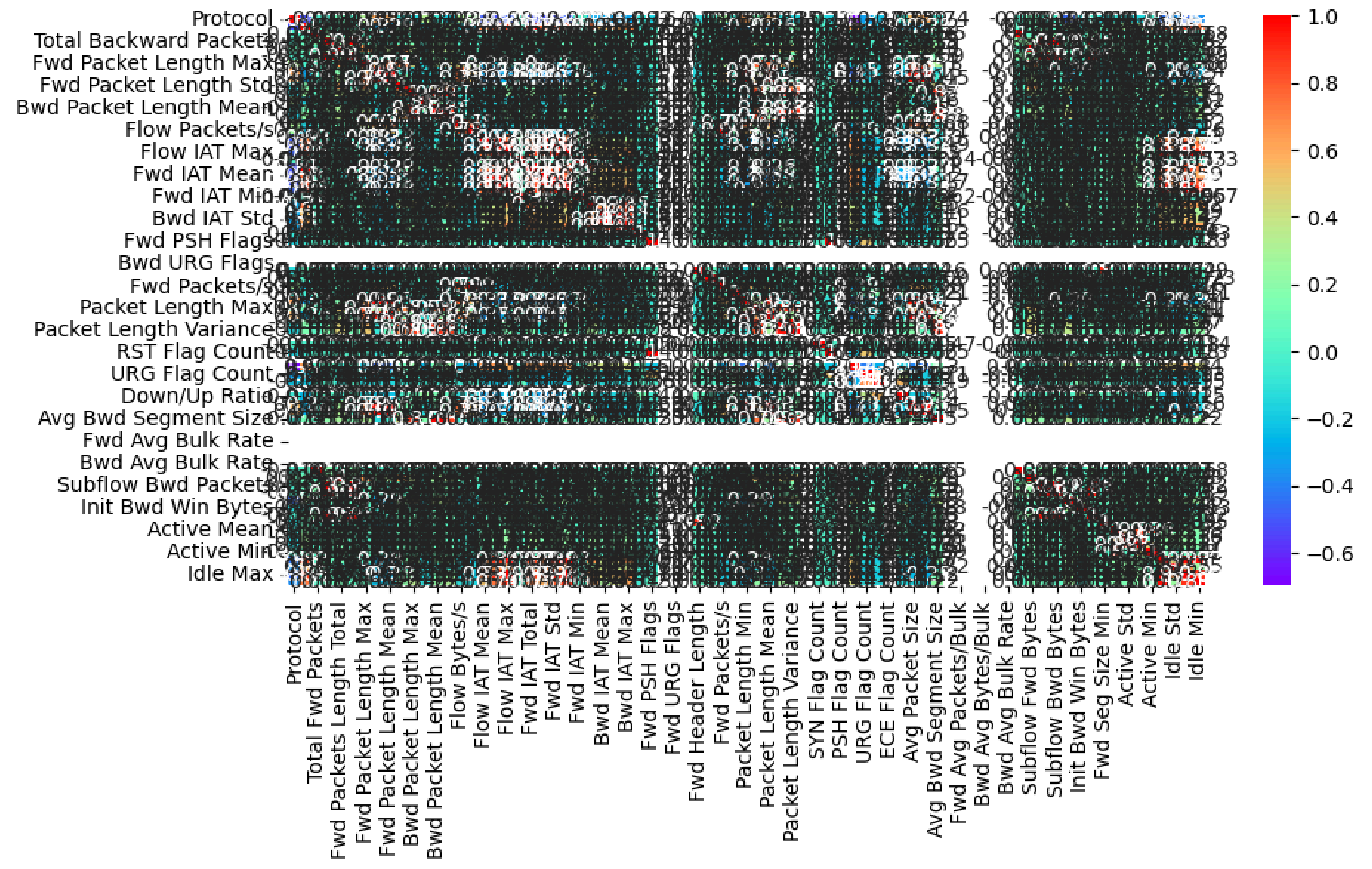
In [10]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 121676 entries, 0 to 121675
Data columns (total 78 columns):
 #   Column                   Non-Null Count   Dtype  
---  ------                   --------------   -----  
 0   Protocol                 121676 non-null  int8   
 1   Flow Duration            121676 non-null  int32  
 2   Total Fwd Packets        121676 non-null  int32  
 3   Total Backward Packets   121676 non-null  int16  
 4   Fwd Packets Length Total 121676 non-null  float32
 5   Bwd Packets Length Total 121676 non-null  float32
 6   Fwd Packet Length Max    121676 non-null  float32
 7   Fwd Packet Length Min    121676 non-null  float32
 8   Fwd Packet Length Mean   121676 non-null  float32
 9   Fwd Packet Length Std    121676 non-null  float32
 10  Bwd Packet Length Max    121676 non-null  float32
 11  Bwd Packet Length Min    121676 non-null  float32
 12  Bwd Packet Length Mean   121676 non-null  float32
 13  Bwd Packet Length Std    121676 non-null  float32
 14  Flow Bytes/s             121676 non-null  float64
 15  Flow Packets/s           121676 non-null  float64
 16  Flow IAT Mean            121676 non-null  float32
 17  Flow IAT Std             121676 non-null  float32
 18  Flow IAT Max             121676 non-null  float32
 19  Flow IAT Min             121676 non-null  float32
 20  Fwd IAT Total            121676 non-null  float32
 21  Fwd IAT Mean             121676 non-null  float32
 22  Fwd IAT Std              121676 non-null  float32
 23  Fwd IAT Max              121676 non-null  float32
 24  Fwd IAT Min              121676 non-null  float32
 25  Bwd IAT Total            121676 non-null  float32
 26  Bwd IAT Mean             121676 non-null  float32
 27  Bwd IAT Std              121676 non-null  float32
 28  Bwd IAT Max              121676 non-null  float32
 29  Bwd IAT Min              121676 non-null  float32
 30  Fwd PSH Flags            121676 non-null  int8   
 31  Bwd PSH Flags            121676 non-null  int8   
 32  Fwd URG Flags            121676 non-null  int8   
 33  Bwd URG Flags            121676 non-null  int8   
 34  Fwd Header Length        121676 non-null  int64  
 35  Bwd Header Length        121676 non-null  int64  
 36  Fwd Packets/s            121676 non-null  float32
 37  Bwd Packets/s            121676 non-null  float32
 38  Packet Length Min        121676 non-null  float32
 39  Packet Length Max        121676 non-null  float32
 40  Packet Length Mean       121676 non-null  float32
 41  Packet Length Std        121676 non-null  float32
 42  Packet Length Variance   121676 non-null  float32
 43  FIN Flag Count           121676 non-null  int8   
 44  SYN Flag Count           121676 non-null  int8   
 45  RST Flag Count           121676 non-null  int8   
 46  PSH Flag Count           121676 non-null  int8   
 47  ACK Flag Count           121676 non-null  int8   
 48  URG Flag Count           121676 non-null  int8   
 49  CWE Flag Count           121676 non-null  int8   
 50  ECE Flag Count           121676 non-null  int8   
 51  Down/Up Ratio            121676 non-null  float32
 52  Avg Packet Size          121676 non-null  float32
 53  Avg Fwd Segment Size     121676 non-null  float32
 54  Avg Bwd Segment Size     121676 non-null  float32
 55  Fwd Avg Bytes/Bulk       121676 non-null  int8   
 56  Fwd Avg Packets/Bulk     121676 non-null  int8   
 57  Fwd Avg Bulk Rate        121676 non-null  int8   
 58  Bwd Avg Bytes/Bulk       121676 non-null  int8   
 59  Bwd Avg Packets/Bulk     121676 non-null  int8   
 60  Bwd Avg Bulk Rate        121676 non-null  int8   
 61  Subflow Fwd Packets      121676 non-null  int32  
 62  Subflow Fwd Bytes        121676 non-null  int32  
 63  Subflow Bwd Packets      121676 non-null  int16  
 64  Subflow Bwd Bytes        121676 non-null  int32  
 65  Init Fwd Win Bytes       121676 non-null  int32  
 66  Init Bwd Win Bytes       121676 non-null  int32  
 67  Fwd Act Data Packets     121676 non-null  int16  
 68  Fwd Seg Size Min         121676 non-null  int32  
 69  Active Mean              121676 non-null  float32
 70  Active Std               121676 non-null  float32
 71  Active Max               121676 non-null  float32
 72  Active Min               121676 non-null  float32
 73  Idle Mean                121676 non-null  float32
 74  Idle Std                 121676 non-null  float32
 75  Idle Max                 121676 non-null  float32
 76  Idle Min                 121676 non-null  float32
 77  Label                    121676 non-null  object 
dtypes: float32(43), float64(2), int16(3), int32(8), int64(2), int8(19), object(1)
memory usage: 31.2+ MB
```

In [11]:
```
sns.countplot(x='Label',data=data, palette='hls')
plt.show()
#plt.savefig('count_plot') mal: the nodule malignancy, 0: benign, 1: malignant
```



In [12]:
```
plt.figure(figsize = (10,5))
sns.heatmap(data.corr(), annot = True, cmap="rainbow")
plt.show()
```



In [13]:
```
data['Label'].value_counts()
```

Out[13]:
```
Attack    76575
Normal    45101
Name: Label, dtype: int64
```

In [15]:
```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows
# how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'species'.
data['Label']= label_encoder.fit_transform(data['Label'])
```

In [16]:
```
X = data.drop(["Label"],axis =1)
y = data["Label"]
```

FS

In [17]:
```
from sklearn.feature_selection import SelectKBest, SelectPercentile, mutual_info_classif
```

In [20]:
```
selector = SelectPercentile(mutual_info_classif, percentile=15)
X_reduced = selector.fit_transform(X, y)
#X_reduced.shape
```

In [21]:
```
cols = selector.get_support(indices=True)
selected_columns = X.iloc[:,cols].columns.tolist()
selected_columns
```

Out[21]:
```
['Fwd Packets Length Total',
 'Fwd Packet Length Mean',
 'Flow Bytes/s',
 'Flow IAT Max',
 'Fwd IAT Max',
 'Packet Length Min',
 'Packet Length Max',
 'Packet Length Mean',
 'Avg Packet Size',
 'Avg Fwd Segment Size',
 'Subflow Fwd Bytes',
 'Init Fwd Win Bytes']
```

In [22]:
```
len(selected_columns)
```

Out[22]:
```
12
```

In [23]:
```
df = data[['Fwd Packets Length Total',
             'Fwd Packet Length Mean',
             'Flow Bytes/s',
             'Flow IAT Max',
             'Fwd IAT Max',
             'Packet Length Min',
             'Packet Length Max',
             'Packet Length Mean',
             'Avg Packet Size',
             'Avg Fwd Segment Size',
             'Subflow Fwd Bytes',
             'Init Fwd Win Bytes','Label']]
```

In [24]:
```
df.columns
```

Out[24]:
```
Index(['Fwd Packets Length Total', 'Fwd Packet Length Mean', 'Flow Bytes/s',
       'Flow IAT Max', 'Fwd IAT Max', 'Packet Length Min', 'Packet Length Max',
       'Packet Length Mean', 'Avg Packet Size', 'Avg Fwd Segment Size',
       'Subflow Fwd Bytes', 'Init Fwd Win Bytes', 'Label'],
      dtype='object')
```

In [25]:
```
X = df.drop(["Label"],axis =1)
y = df["Label"]
```

In [26]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 42)
#X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

In [27]:
```
from sklearn.metrics import accuracy_score # for calculating accuracy of model
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

In [28]:
```
ML_Model = []
accuracy = []
precision = []
recall = []
f1score = []


#function to call for storing the results
def storeResults(model, a,b,c,d):
    ML_Model.append(model)
    accuracy.append(round(a, 3))
    precision.append(round(b, 3))
    recall.append(round(c, 3))
    f1score.append(round(d, 3))
```

BernoulliNB

In [29]:
```
from sklearn.naive_bayes import BernoulliNB

bnb = BernoulliNB(alpha=1.0, binarize=0.0, fit_prior=True, class_prior=None)

bnb.fit(X_train, y_train)

y_pred = bnb.predict(X_test)

bnb_acc = accuracy_score(y_pred, y_test)
bnb_prec = precision_score(y_pred, y_test,average='weighted')
bnb_rec = recall_score(y_pred, y_test,average='weighted')
bnb_f1 = f1_score(y_pred, y_test,average='weighted')
```

In [30]:
```
storeResults('BernoulliNB',bnb_acc,bnb_prec,bnb_rec,bnb_f1)
```

Passive Aggressive

In [31]:
```
from sklearn.linear_model import PassiveAggressiveClassifier

pa = PassiveAggressiveClassifier(C=1.0, fit_intercept=True, max_iter=1000, tol=0.001, early_stopping=False,
                                 validation_fraction=0.1, n_iter_no_change=5, shuffle=True, verbose=0,
                                 loss='hinge', n_jobs=None, random_state=None, warm_start=False,
                                 class_weight=None, average=False)

pa.fit(X_train, y_train)

y_pred = pa.predict(X_test)

pa_acc = accuracy_score(y_pred, y_test)
pa_prec = precision_score(y_pred, y_test,average='weighted')
pa_rec = recall_score(y_pred, y_test,average='weighted')
pa_f1 = f1_score(y_pred, y_test,average='weighted')
```

In [32]:
```
storeResults('PassiveAggressive',pa_acc,pa_prec,pa_rec,pa_f1)
```

SGDClassifier

In [33]:
```
from sklearn.linear_model import SGDClassifier

sgd = SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True,
                   max_iter=1000, tol=0.001, shuffle=True, verbose=0, epsilon=0.1, n_jobs=None,
                   random_state=None, learning_rate='optimal', eta0=0.0, power_t=0.5, early_stopping=False,
                   validation_fraction=0.1, n_iter_no_change=5, class_weight=None, warm_start=False, average=False)

sgd.fit(X_train, y_train)

y_pred = sgd.predict(X_test)

sgd_acc = accuracy_score(y_pred, y_test)
sgd_prec = precision_score(y_pred, y_test,average='weighted')
sgd_rec = recall_score(y_pred, y_test,average='weighted')
sgd_f1 = f1_score(y_pred, y_test,average='weighted')
```

In [34]:
```
storeResults('SGDClassifier',sgd_acc,sgd_prec,sgd_rec,sgd_f1)
```

MLP Classifier

In [35]:
```
from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier(hidden_layer_sizes=(100,), activation='relu', solver='adam', alpha=0.0001, batch_size='auto',
                    learning_rate='constant', learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True,
                    random_state=None, tol=0.0001, verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True,
                    early_stopping=False, validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08,
                    n_iter_no_change=10, max_fun=15000)

mlp.fit(X_train, y_train)

y_pred = mlp.predict(X_test)

mlp_acc = accuracy_score(y_pred, y_test)
mlp_prec = precision_score(y_pred, y_test,average='weighted')
mlp_rec = recall_score(y_pred, y_test,average='weighted')
mlp_f1 = f1_score(y_pred, y_test,average='weighted')
```

In [36]:
```
storeResults('MLPClassifier',mlp_acc,mlp_prec,mlp_rec,mlp_f1)
```

Ensemble

In [44]:

```
from sklearn.ensemble import VotingClassifier

eclf1 = VotingClassifier(estimators=[('BNB', bnb),('PA', pa),('SGD', sgd),('MLP', mlp)], voting='hard')


eclf1.fit(X_train, y_train)

y_pred = eclf1.predict(X_test)

stac_acc = accuracy_score(y_pred, y_test)
stac_prec = precision_score(y_pred, y_test,average='weighted')
stac_rec = recall_score(y_pred, y_test,average='weighted')
stac_f1 = f1_score(y_pred, y_test,average='weighted')
```

In [39]:

```
storeResults('Ensemble',stac_acc,stac_prec,stac_rec,stac_f1)
```

Extension

In [40]:

```
from sklearn.ensemble import VotingClassifier, AdaBoostClassifier, RandomForestClassifier, BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

brf = BaggingClassifier(RandomForestClassifier(),n_estimators=10, random_state=0,max_samples=1.0,max_features=1.0)

bdt = AdaBoostClassifier(
    DecisionTreeClassifier(max_depth=1), algorithm="SAMME", n_estimators=200
)

ext = VotingClassifier(estimators=[('BoostDT', bdt),('BagRF', brf)], voting='soft')
ext.fit(X_train, y_train)

y_pred    = ext.predict(X_test)

ml_acc = accuracy_score(y_pred, y_test)
ml_prec = precision_score(y_pred, y_test,average='weighted')
ml_rec = recall_score(y_pred, y_test,average='weighted')
ml_f1 = f1_score(y_pred, y_test,average='weighted')
```

In [41]:

```
storeResults('Extension',ml_acc,ml_prec,ml_rec,ml_f1)
```

Comparison

In [42]:

```
#creating dataframe
result = pd.DataFrame({ 'ML Model' : ML_Model,
                        'Accuracy' : accuracy,
                       'Precision': precision,
                       'Recall'   : recall,
                       'F1_score' : f1score
                      })
```

In [43]:

```
result
```

Out[43]:

	ML Model	Accuracy	Precision	Recall	F1_score
0	BernoulliNB	0.763	0.866	0.763	0.776
1	PassiveAggressive	0.802	0.827	0.802	0.804
2	SGDClassifier	0.409	0.454	0.409	0.422
3	MLPClassifier	0.572	0.716	0.572	0.604
4	Ensemble	0.750	0.858	0.750	0.765
5	Extension	0.998	0.998	0.998	0.998

Modelling

In [47]:

```
import joblib
filename = 'models/model_cic19.sav'
joblib.dump(ext, filename)
```

Out[47]:

```
['models/model_cic19.sav']
```

Graph

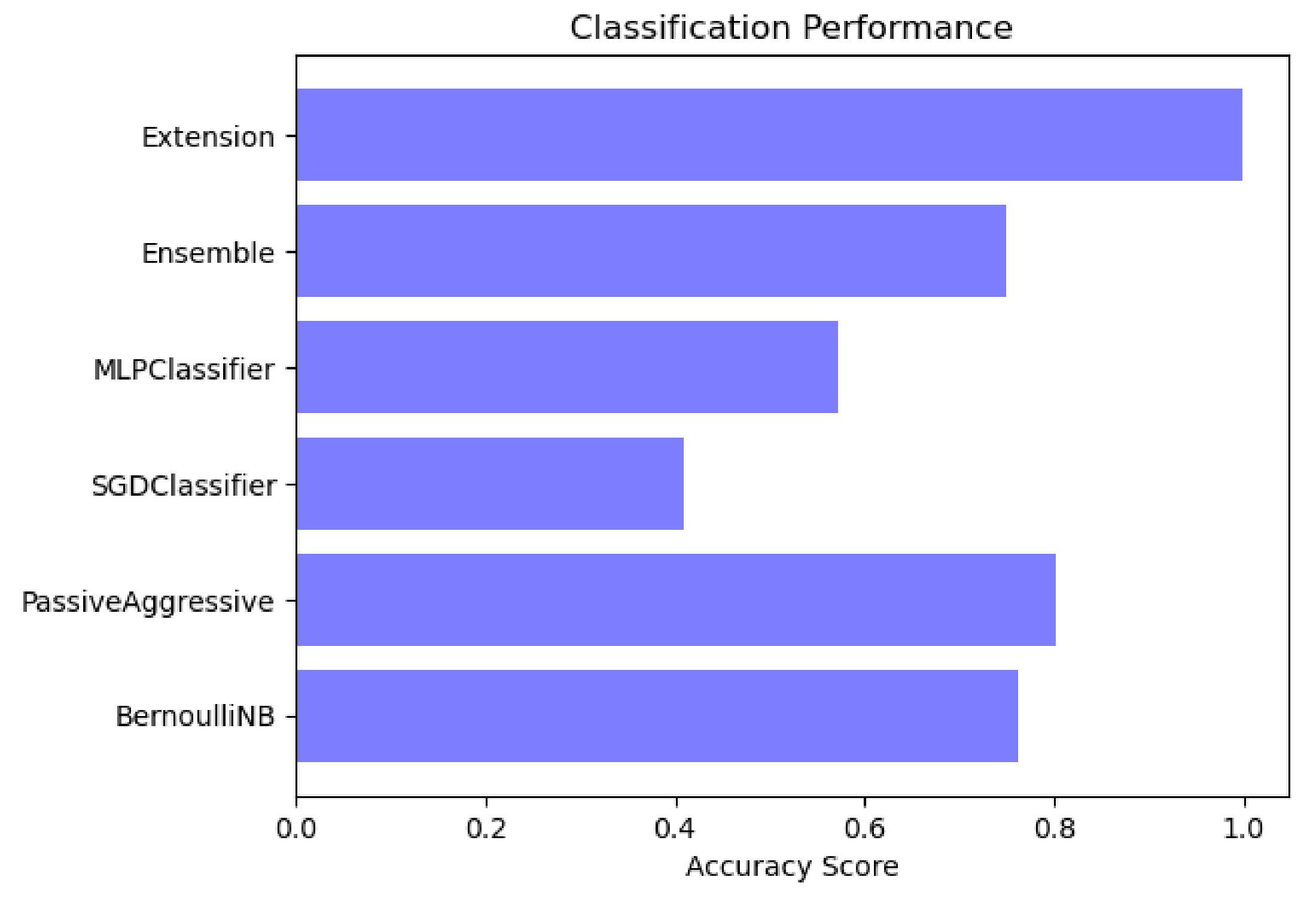
In [48]:

```
classifier = ML_Model
y_pos = np.arange(len(classifier))
```

Accuracy

In [49]:

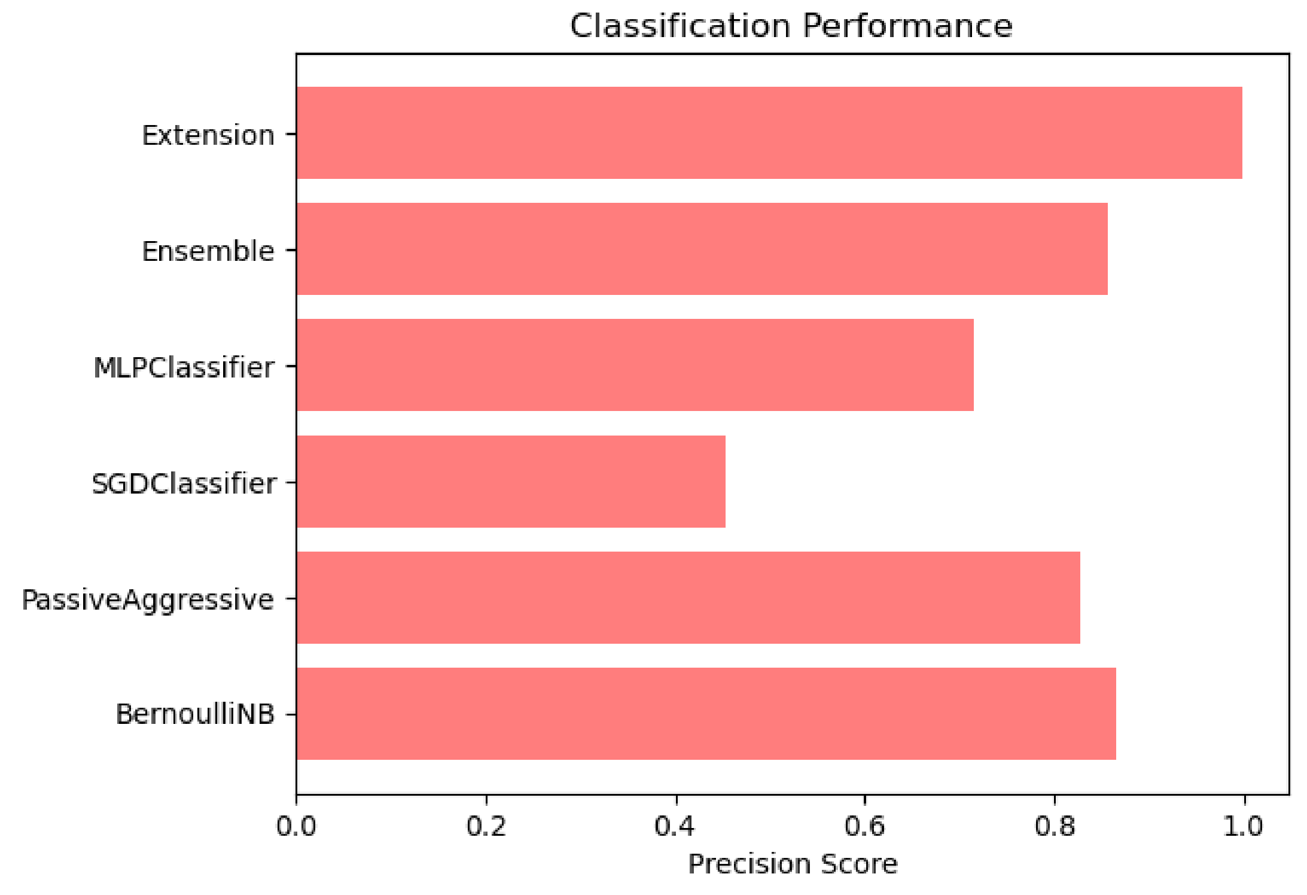
```
import matplotlib.pyplot as plt2
plt2.barh(y_pos, accuracy, align='center', alpha=0.5,color='blue')
plt2.yticks(y_pos, classifier)
plt2.xlabel('Accuracy Score')
plt2.title('Classification Performance')
plt2.show()
```



Precision

In [50]:

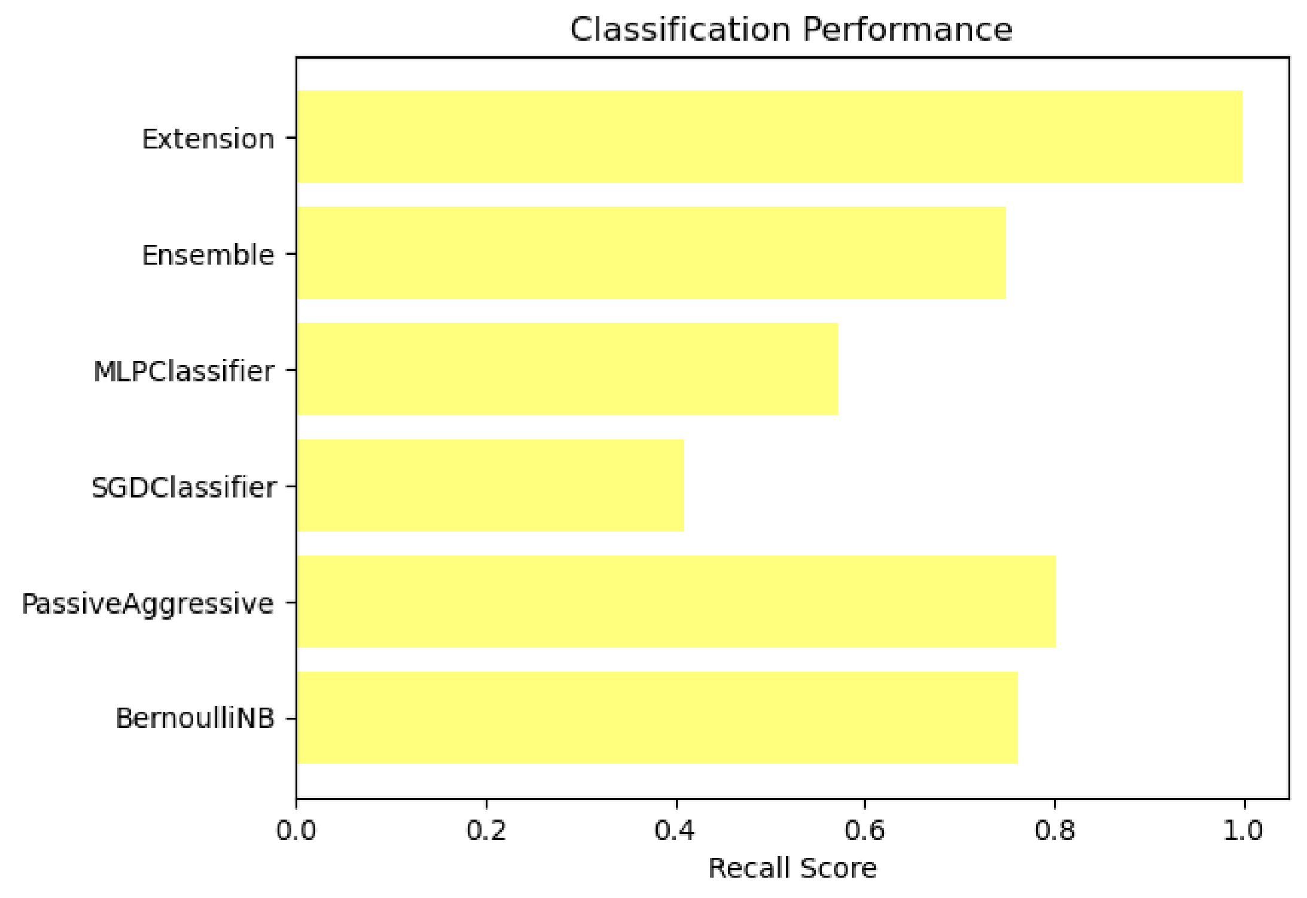
```
plt2.barh(y_pos, precision, align='center', alpha=0.5,color='red')
plt2.yticks(y_pos, classifier)
plt2.xlabel('Precision Score')
plt2.title('Classification Performance')
plt2.show()
```



Recall

In [51]:

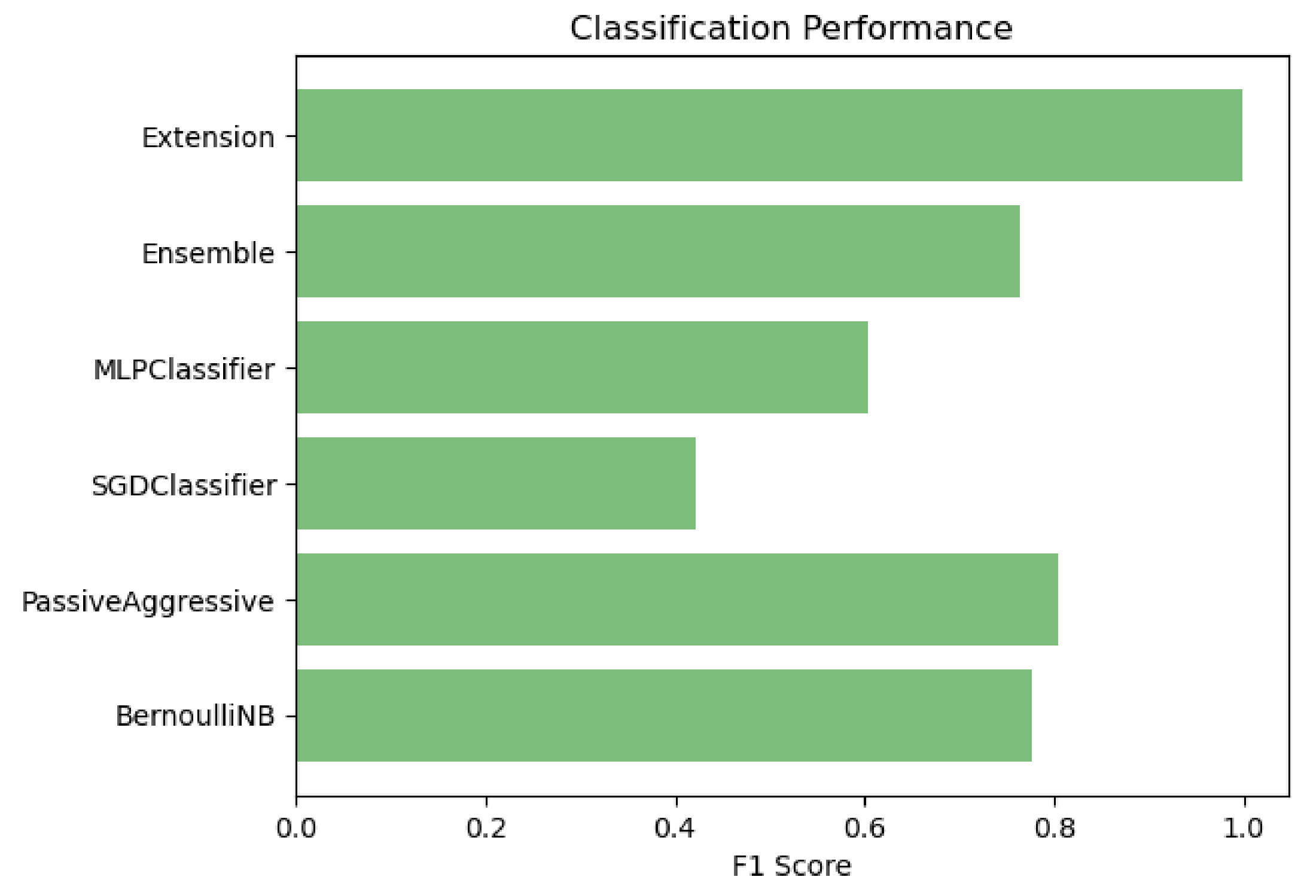
```
plt2.barh(y_pos, recall, align='center', alpha=0.5,color='yellow')
plt2.yticks(y_pos, classifier)
plt2.xlabel('Recall Score')
plt2.title('Classification Performance')
plt2.show()
```



F1 Score

In [52]:

```
plt2.barh(y_pos, f1score, align='center', alpha=0.5,color='green')
plt2.yticks(y_pos, classifier)
plt2.xlabel('F1 Score')
plt2.title('Classification Performance')
plt2.show()
```



In [ ]: