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

```
data = pd.read_csv('/content/cicddos2019_dataset.csv')
```

```
pd.set_option('display.max_columns',None)
```

```
data = data.drop('Unnamed: 0',axis=1)
```

```
data.head()
```

	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	
0	17	216631	6	0	2088.0	0.0	393.0	321.0	348.0	35
1	17	2	2	0	802.0	0.0	401.0	401.0	401.0	(
2	17	48	2	0	766.0	0.0	383.0	383.0	383.0	(
3	17	107319	4	0	1398.0	0.0	369.0	330.0	349.5	22
4	17	107271	4	0	1438.0	0.0	389.0	330.0	359.5	34

```
data['Protocol'].unique()
```

```
array([17,  6,  0])
```

```
data.isnull().sum()
```

```
Protocol                   0
Flow Duration              0
Total Fwd Packets          0
Total Backward Packets     0
Fwd Packets Length Total   0
                          ..
Idle Std                   0
Idle Max                   0
Idle Min                   0
Label                      0
Class                      0
Length: 79, dtype: int64
```

```
data.describe()
```

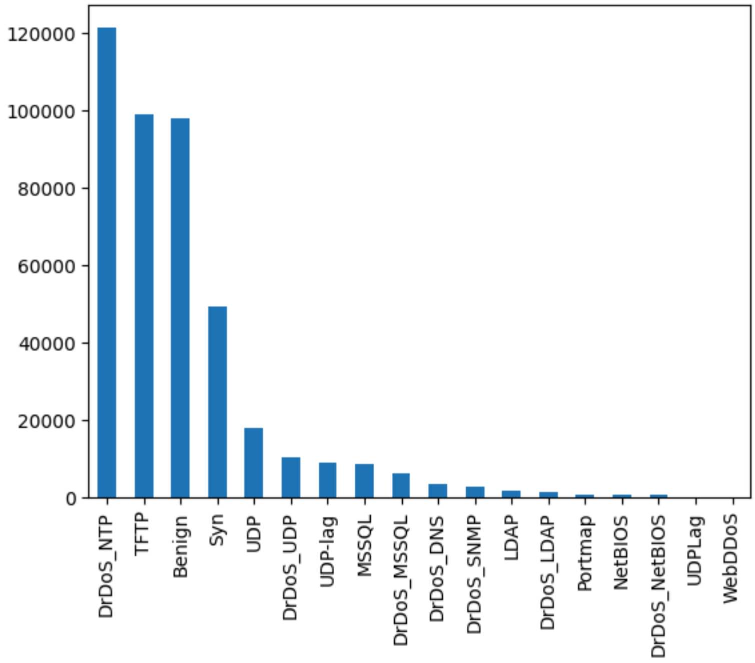
	Protocol	Flow Duration	Total Fwd Packets	Total Backward Packets	Fwd Packets Length Total	Bwd Pa L
count	431371.000000	4.313710e+05	431371.000000	431371.000000	4.313710e+05	4.31371
mean	13.948694	8.404856e+06	24.139117	2.472021	9.416956e+03	1.63289
std	4.966712	2.126596e+07	195.888896	56.370208	3.445253e+04	1.06405
min	0.000000	1.000000e+00	1.000000	0.000000	0.000000e+00	0.00000
25%	6.000000	7.870000e+02	4.000000	0.000000	7.800000e+01	0.00000
50%	17.000000	4.480400e+04	4.000000	0.000000	2.064000e+03	0.00000
75%	17.000000	3.002508e+06	16.000000	2.000000	5.160000e+03	0.00000
max	17.000000	1.199987e+08	86666.000000	31700.000000	1.526642e+07	5.84295

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 431371 entries, 0 to 431370
Data columns (total 79 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   Protocol                  431371 non-null  int64  
 1   Flow Duration             431371 non-null  int64  
 2   Total Fwd Packets         431371 non-null  int64  
 3   Total Backward Packets     431371 non-null  int64
 4   Fwd Packets Length Total   431371 non-null  float64
 5   Bwd Packets Length Total   431371 non-null  float64
 6   Fwd Packet Length Max      431371 non-null  float64
 7   Fwd Packet Length Min      431371 non-null  float64
 8   Fwd Packet Length Mean     431371 non-null  float64
 9   Fwd Packet Length Std      431371 non-null  float64
 10  Bwd Packet Length Max      431371 non-null  float64
 11  Bwd Packet Length Min      431371 non-null  float64
 12  Bwd Packet Length Mean     431371 non-null  float64
 13  Bwd Packet Length Std      431371 non-null  float64
 14  Flow Bytes/s               431371 non-null  float64
 15  Flow Packets/s             431371 non-null  float64
 16  Flow IAT Mean              431371 non-null  float64
 17  Flow IAT Std               431371 non-null  float64
 18  Flow IAT Max               431371 non-null  float64
 19  Flow IAT Min               431371 non-null  float64
 20  Fwd IAT Total              431371 non-null  float64
 21  Fwd IAT Mean               431371 non-null  float64
 22  Fwd IAT Std                431371 non-null  float64
 23  Fwd IAT Max                431371 non-null  float64
 24  Fwd IAT Min                431371 non-null  float64
 25  Bwd IAT Total              431371 non-null  float64
 26  Bwd IAT Mean               431371 non-null  float64
 27  Bwd IAT Std                431371 non-null  float64
 28  Bwd IAT Max                431371 non-null  float64
 29  Bwd IAT Min                431371 non-null  float64
 30  Fwd PSH Flags              431371 non-null  int64
 31  Bwd PSH Flags              431371 non-null  int64
 32  Fwd URG Flags              431371 non-null  int64
 33  Bwd URG Flags              431371 non-null  int64
 34  Fwd Header Length          431371 non-null  int64
 35  Bwd Header Length          431371 non-null  int64
 36  Fwd Packets/s              431371 non-null  float64
 37  Bwd Packets/s              431371 non-null  float64
 38  Packet Length Min          431371 non-null  float64
 39  Packet Length Max          431371 non-null  float64
 40  Packet Length Mean         431371 non-null  float64
 41  Packet Length Std          431371 non-null  float64
 42  Packet Length Variance     431371 non-null  float64
 43  FIN Flag Count             431371 non-null  int64
 44  SYN Flag Count             431371 non-null  int64
 45  RST Flag Count             431371 non-null  int64
 46  PSH Flag Count             431371 non-null  int64
 47  ACK Flag Count             431371 non-null  int64
 48  URG Flag Count             431371 non-null  int64
 49  CWE Flag Count             431371 non-null  int64
 50  ECE Flag Count             431371 non-null  int64
 51  Down/Up Ratio              431371 non-null  float64
 52  Avg Packet Size            431371 non-null  float64
```

```
data['Label'].value_counts().plot(kind='bar')
```

```
<Axes: >
```



```
ind = data['Label'].value_counts(normalize=True).index
value = data['Label'].value_counts(normalize=True).values
```

```
data['Label'].value_counts(normalize=True)[:9]
```

```
DrDoS_NTP      0.281354
TFTP           0.229308
Benign         0.226791
Syn            0.114456
UDP            0.041936
DrDoS_UDP      0.024156
UDP-lag        0.020567
MSSQL          0.019758
DrDoS_MSSQL    0.014401
Name: Label, dtype: float64
```

```
(len(data[data['Label']=='DrDoS_NTP'])/len(data))*100
```

```
28.135410122609077
```

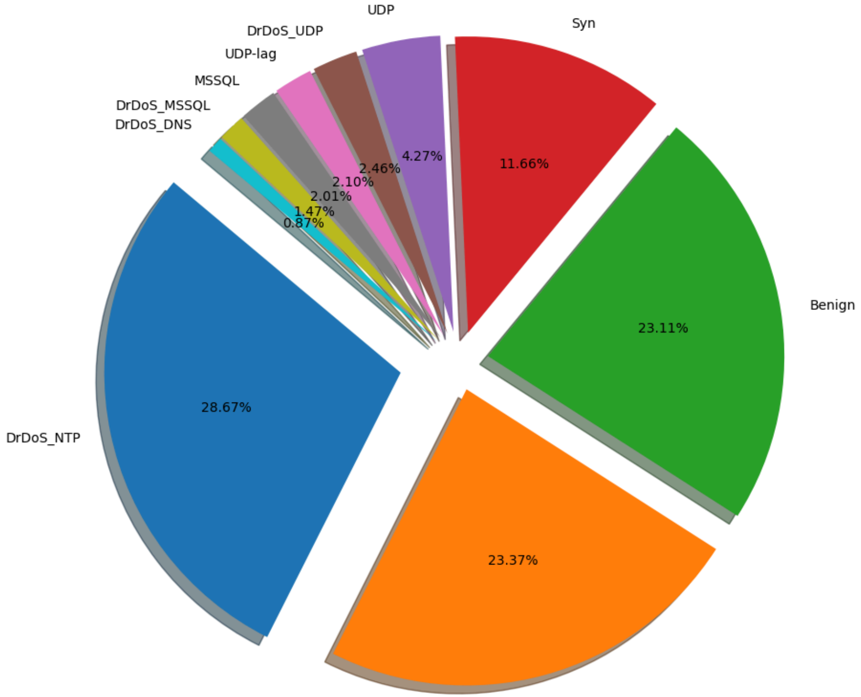
```
data['Label'].value_counts()
```

```
DrDoS_NTP        121368
TFTP              98917
Benign            97831
Syn               49373
UDP               18090
DrDoS_UDP         10420
UDP-lag            8872
MSSQL              8523
DrDoS_MSSQL        6212
DrDoS_DNS          3669
DrDoS_SNMP         2717
LDAP               1906
DrDoS_LDAP         1440
Portmap             685
NetBIOS             644
DrDoS_NetBIOS       598
UDPLag               55
WebDDoS              51
Name: Label, dtype: int64
```

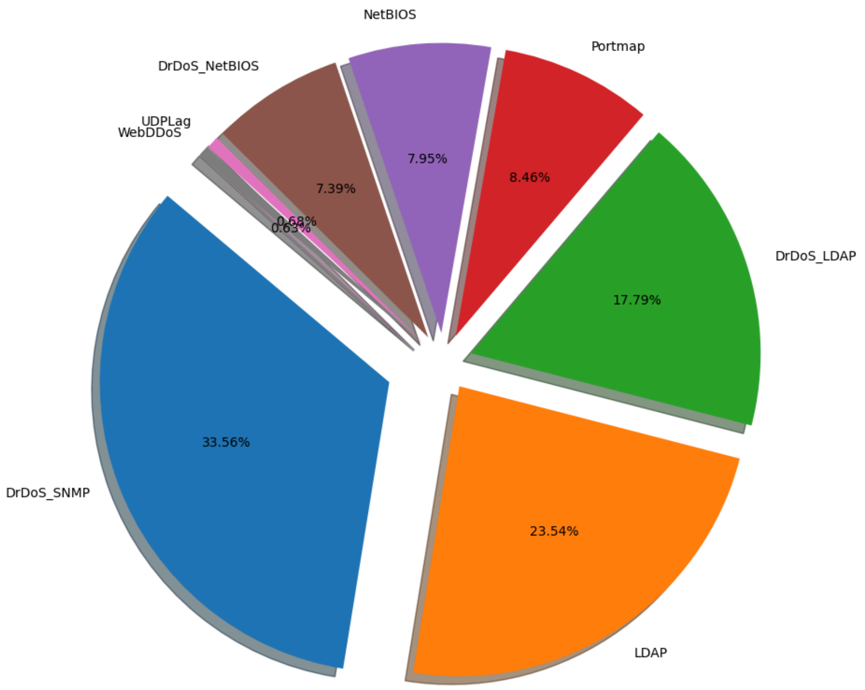
```
value[:9]
```

```
array([0.2813541 , 0.22930841, 0.22679086, 0.114456  , 0.04193606,
       0.02415554, 0.02056698, 0.01975793, 0.0144006 ])
```

```
plt.figure(figsize=(10,15))
plt.pie(value[:10]*100,autopct='%1.2f%%',explode=[0.2,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1],
        labels=ind[:10],shadow=True,startangle=140)
plt.show()
```

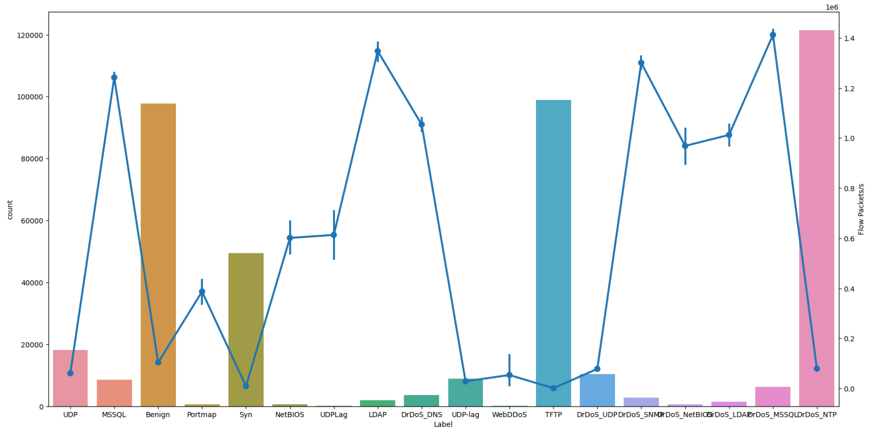


```
plt.figure(figsize=(10,15))
plt.pie(value[10:]*100,autopct='%1.2f%%',explode=[0.2,0.1,0.1,0.1,0.1,0.1,0.1,0.1],
        labels=ind[10:],shadow=True,startangle=140)
plt.show()
```



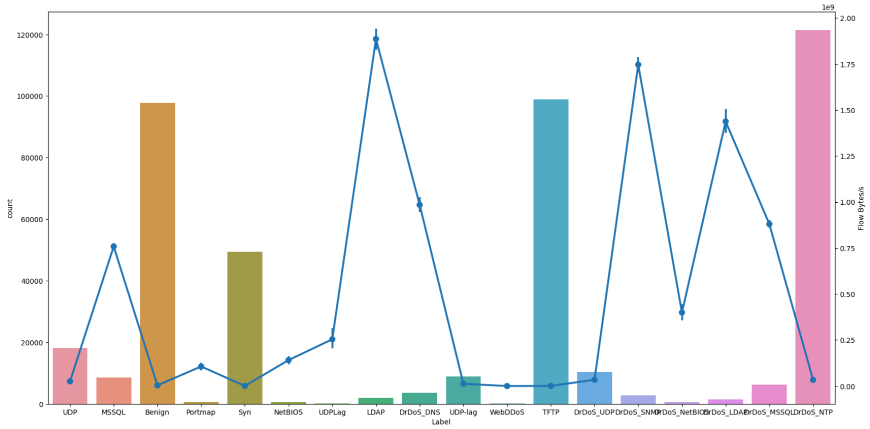
```
import seaborn as sns
plt.figure(figsize=(20,10))
ax1 = sns.countplot(data=data,x='Label')
ax2 = ax1.twinx()
ax2 = sns.pointplot(data=data,x='Label',y='Flow Packets/s')
ax2.set_ylabel('Flow Packets/s')
```

```
Text(0, 0.5, 'Flow Packets/s')
```

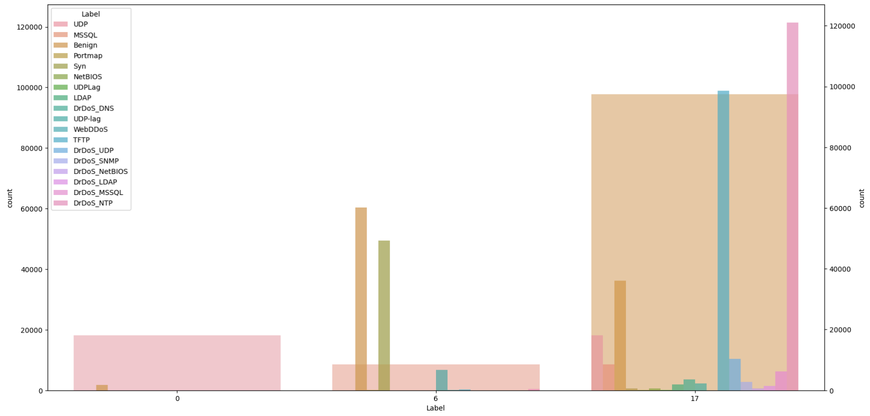


```
import seaborn as sns
plt.figure(figsize=(20,10))
ax1 = sns.countplot(data=data,x='Label')
ax2 = ax1.twinx()
ax2 = sns.pointplot(data=data,x='Label',y='Flow Bytes/s')
ax2.set_ylabel('Flow Bytes/s')
```

```
Text(0, 0.5, 'Flow Bytes/s')
```



```
plt.figure(figsize=(20,10))
ax1 = sns.countplot(data=data,x = 'Label',alpha=0.5)
ax2 = ax1.twinx()
ax2 = sns.countplot(data=data,x='Protocol',alpha=0.7,hue='Label')
```



```
data.head()
```

	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	
0	17	216631	6	0	2088.0	0.0	393.0	321.0	348.0	35
1	17	2	2	0	802.0	0.0	401.0	401.0	401.0	(
2	17	48	2	0	766.0	0.0	383.0	383.0	383.0	(
3	17	107319	4	0	1398.0	0.0	369.0	330.0	349.5	22
4	17	107271	4	0	1438.0	0.0	389.0	330.0	359.5	34

```
X = data.drop(['Label','Class'],axis=1)
```

```
y = data['Label']
```

```
from sklearn.preprocessing import LabelEncoder
LE = LabelEncoder()
y_trans = LE.fit_transform(y)
```

```
y_trans
```

```
array([14, 14, 14, ...,  0,  0,  0])
```

```
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
X_std = ss.fit_transform(X)
model = ExtraTreesClassifier(random_state=42)
model.fit(X,y_trans)
```

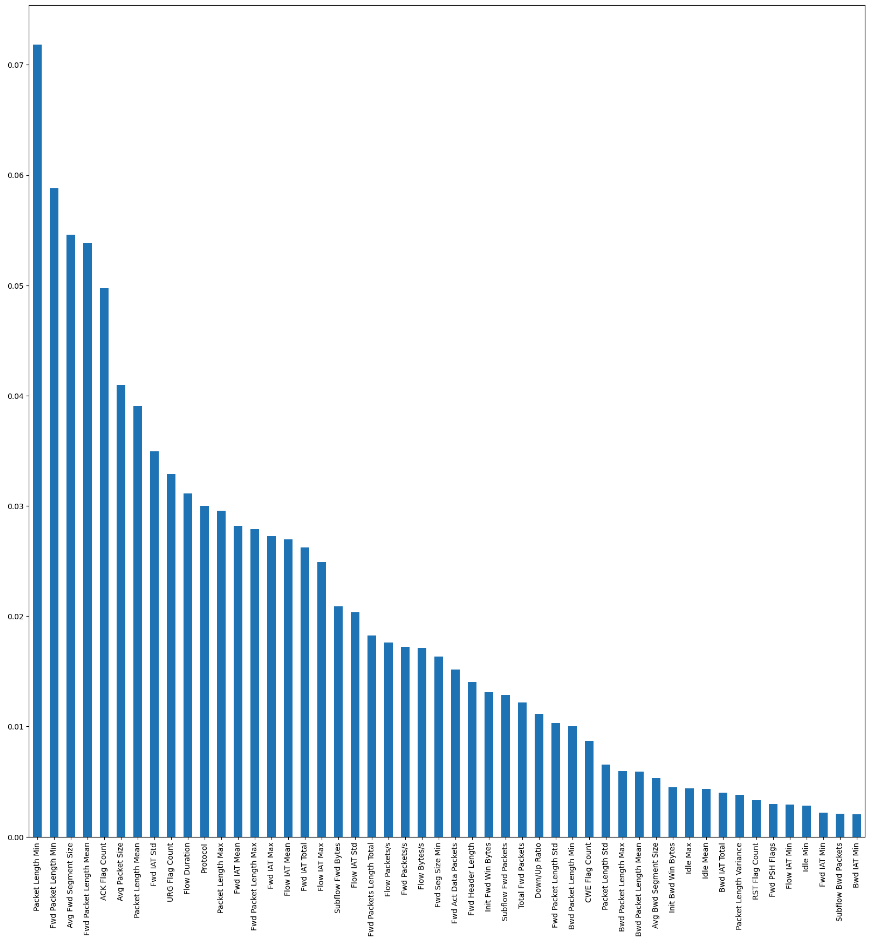
```
▾        ExtraTreesClassifier
ExtraTreesClassifier(random_state=42)
```

```
model.feature_importances_
```

```
array([3.00175972e-02, 3.11416554e-02, 1.21552825e-02, 6.70355969e-04,
       1.82498898e-02, 7.00422560e-04, 2.79204594e-02, 5.88147882e-02,
       5.38558262e-02, 1.03063875e-02, 5.97902717e-03, 1.00317014e-02,
       5.90178231e-03, 1.72691644e-03, 1.71252044e-02, 1.76158790e-02,
       2.69674978e-02, 2.03519305e-02, 2.48911914e-02, 2.92809779e-03,
       2.62325864e-02, 2.82067346e-02, 3.49551649e-02, 2.72769196e-02,
       2.17606862e-03, 3.98081110e-03, 1.70801573e-03, 6.63744696e-04,
       1.76144072e-03, 2.04203052e-03, 2.96972413e-03, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 1.40234716e-02, 5.32362297e-04,
       1.72230829e-02, 9.42116706e-04, 7.18291811e-02, 2.95504999e-02,
       3.90587947e-02, 6.52550598e-03, 3.79138438e-03, 0.00000000e+00,
       4.68994002e-05, 3.30941855e-03, 0.00000000e+00, 4.97423982e-02,
       3.29114234e-02, 8.72322170e-03, 0.00000000e+00, 1.11678374e-02,
       4.09932490e-02, 5.45811686e-02, 5.30426958e-03, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 1.28456771e-02, 2.09154062e-02, 2.08677780e-03,
       1.30278027e-03, 1.31242945e-02, 4.46808839e-03, 1.51779561e-02,
       1.63317203e-02, 3.12469899e-04, 4.57544518e-05, 3.01534697e-04,
       3.00820654e-04, 4.32579192e-03, 1.70141089e-03, 4.37151753e-03,
       2.80658002e-03])
```

```
plt.figure(figsize=(20,20))
feature_importance_std = pd.Series(model.feature_importances_,index=X.columns)
feature_importance_std.nlargest(50).plot(kind='bar')
```

```
<Axes: >
```



```
X_from_tree = data[['Packet Length Min','Fwd Packet Length Min','Avg Fwd Segment Size','Fwd Packet Length Mean','ACK Flag Count','Avg
        'Packet Length Mean','Flow IAT Std','URG Flag Count','Flow Duration','Protocol','Packet Length Max','Flow IAT Mean','Fwd Packet
        'Fwd IAT Max','Flow IAT Max','Fwd IAT Total','Flow IAT Max','Subflow Bwd Bytes','Flow IAT Std','Fwd Packets Length Total',
        'Flow Packets/s','Fwd Packets/s','Flow Bytes/s','Fwd Seg Size Min','Fwd Act Data Packets','Fwd Header Length']]
```

```
from imblearn.over_sampling import SMOTE
smote = SMOTE(sampling_strategy='auto', random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_from_tree, y_trans)
```

```
X_resampled.shape
```

```
(2184624, 27)
```

```
from sklearn.model_selection import train_test_split
X_train1,X_test1,y_train1,y_test1 = train_test_split(X_from_tree, y_trans,test_size=0.20,random_state=42)
```

```
from sklearn.preprocessing import StandardScaler
SC = StandardScaler()
X_train_std_tr = SC.fit_transform(X_train1)
X_test_std_tr = SC.transform(X_test1)
```

```
# Decision Tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
dt = DecisionTreeClassifier()
dt.fit(X_train_std_tr,y_train1)
dt_y_pred1 = dt.predict(X_test_std_tr)
print("Classification Report for Decision Tree: \n", classification_report(LE.inverse_transform(y_test1), LE.inverse_transform(dt_y_
```

```
Classification Report for Decision Tree: 
               precision    recall  f1-score   support

       Benign       0.99      0.99      0.99     24190
    DrDoS_DNS       0.74      0.62      0.68     24492
   DrDoS_LDAP       0.51      0.52      0.52     24224
  DrDoS_MSSQL       0.62      0.70      0.66     24214
    DrDoS_NTP       1.00      1.00      1.00     24284
DrDoS_NetBIOS       0.69      0.46      0.55     24503
   DrDoS_SNMP       0.81      0.68      0.74     24193
    DrDoS_UDP       0.68      0.77      0.72     24232
         LDAP       0.51      0.68      0.58     24342
        MSSQL       0.68      0.60      0.64     24160
      NetBIOS       0.59      0.87      0.70     24338
      Portmap       0.83      0.72      0.77     24411
          Syn       0.99      0.99      0.99     24143
         TFTP       1.00      1.00      1.00     24237
          UDP       0.73      0.68      0.70     24338
      UDP-lag       0.93      0.85      0.89     24084
       UDPLag       0.96      0.97      0.97     24311
      WebDDoS       1.00      1.00      1.00     24229

     accuracy                           0.78    436925
    macro avg       0.79      0.78      0.78    436925
 weighted avg       0.79      0.78      0.78    436925
```

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_train_std_tr,y_train1)
rfc_y_pred = dt.predict(X_test_std_tr)
print("Classification Report for Decision Tree: \n", classification_report(LE.inverse_transform(y_test1), LE.inverse_transform(rfc_y_
```

```
Classification Report for Decision Tree: 
               precision    recall  f1-score   support

       Benign       0.99      0.99      0.99     24190
    DrDoS_DNS       0.74      0.62      0.68     24492
   DrDoS_LDAP       0.51      0.52      0.52     24224
  DrDoS_MSSQL       0.62      0.70      0.66     24214
    DrDoS_NTP       1.00      1.00      1.00     24284
DrDoS_NetBIOS       0.69      0.46      0.55     24503
   DrDoS_SNMP       0.81      0.68      0.74     24193
    DrDoS_UDP       0.68      0.77      0.72     24232
         LDAP       0.51      0.68      0.58     24342
        MSSQL       0.68      0.60      0.64     24160
      NetBIOS       0.59      0.87      0.70     24338
      Portmap       0.83      0.72      0.77     24411
          Syn       0.99      0.99      0.99     24143
         TFTP       1.00      1.00      1.00     24237
          UDP       0.73      0.68      0.70     24338
      UDP-lag       0.93      0.85      0.89     24084
       UDPLag       0.96      0.97      0.97     24311
      WebDDoS       1.00      1.00      1.00     24229

     accuracy                           0.78    436925
    macro avg       0.79      0.78      0.78    436925
 weighted avg       0.79      0.78      0.78    436925
```

```
from sklearn.svm import LinearSVC
from sklearn.metrics import classification_report
svm = LinearSVC(multi_class='ovr')
svm.fit(X_train_std_tr,y_train1)
y_pred_svm = svm.predict(X_test_std_tr)
print("Classification Report for Decision Tree: \n", classification_report(LE.inverse_transform(y_test1), LE.inverse_transform(y_pre
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244: ConvergenceWarning: Liblinear failed to converge, increase the nu
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
Classification Report for Decision Tree:
                 precision    recall  f1-score   support

        Benign       0.98      0.97      0.97     19619
     DrDoS_DNS       0.38      0.26      0.31       748
    DrDoS_LDAP       0.00      0.00      0.00       263
   DrDoS_MSSQL       0.33      0.00      0.00      1215
     DrDoS_NTP       0.98      0.95      0.97     24300
 DrDoS_NetBIOS       0.00      0.00      0.00       112
    DrDoS_SNMP       0.36      0.75      0.49       557
     DrDoS_UDP       0.65      0.02      0.03      2085
          LDAP       0.00      0.00      0.00       375
         MSSQL       0.44      0.95      0.60      1726
       NetBIOS       0.00      0.00      0.00       112
       Portmap       0.00      0.00      0.00       124
           Syn       0.90      0.99      0.94      9847
          TFTP       0.95      0.99      0.97     19869
           UDP       0.57      0.87      0.69      3545
       UDP-lag       0.88      0.45      0.59      1747
        UDPLag       0.00      0.00      0.00        16
       WebDDoS       0.00      0.00      0.00        15

      accuracy                           0.90     86275
     macro avg       0.41      0.40      0.36     86275
  weighted avg       0.90      0.90      0.88     86275

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train_std_tr,y_train1)
y_pred_knn = knn.predict(X_test_std_tr)
print("Classification Report for Decision Tree: \n", classification_report(LE.inverse_transform(y_test1), LE.inverse_transform(y_pre
```

```
Classification Report for Decision Tree:
                 precision    recall  f1-score   support

        Benign       0.99      1.00      1.00     19619
     DrDoS_DNS       0.41      0.51      0.45       748
    DrDoS_LDAP       0.28      0.29      0.28       263
   DrDoS_MSSQL       0.30      0.31      0.30      1215
     DrDoS_NTP       1.00      1.00      1.00     24300
 DrDoS_NetBIOS       0.18      0.21      0.20       112
    DrDoS_SNMP       0.71      0.60      0.65       557
     DrDoS_UDP       0.35      0.35      0.35      2085
          LDAP       0.40      0.31      0.35       375
         MSSQL       0.50      0.50      0.50      1726
       NetBIOS       0.22      0.18      0.20       112
       Portmap       0.41      0.29      0.34       124
           Syn       0.98      0.99      0.99      9847
          TFTP       1.00      1.00      1.00     19869
           UDP       0.59      0.63      0.61      3545
       UDP-lag       0.86      0.67      0.76      1747
        UDPLag       0.50      0.25      0.33        16
       WebDDoS       0.00      0.00      0.00        15

      accuracy                           0.92     86275
     macro avg       0.54      0.51      0.52     86275
  weighted avg       0.93      0.92      0.92     86275
```

```
from sklearn.metrics import classification_report
print("Classification Report for Decision Tree: \n", classification_report(LE.inverse_transform(y_test1), LE.inverse_transform(y_pre
```

```
Classification Report for Decision Tree:
                 precision    recall  f1-score   support

        Benign       0.99      1.00      1.00     19619
     DrDoS_DNS       0.41      0.51      0.46       748
    DrDoS_LDAP       0.26      0.28      0.27       263
   DrDoS_MSSQL       0.30      0.31      0.30      1215
     DrDoS_NTP       1.00      1.00      1.00     24300
 DrDoS_NetBIOS       0.19      0.22      0.21       112
    DrDoS_SNMP       0.71      0.60      0.65       557
     DrDoS_UDP       0.35      0.35      0.35      2085
          LDAP       0.41      0.32      0.36       375
         MSSQL       0.50      0.50      0.50      1726
       NetBIOS       0.21      0.18      0.19       112
       Portmap       0.42      0.29      0.34       124
           Syn       0.98      0.99      0.99      9847
          TFTP       1.00      1.00      1.00     19869
           UDP       0.59      0.63      0.61      3545
       UDP-lag       0.86      0.67      0.76      1747
        UDPLag       0.50      0.25      0.33        16
       WebDDoS       0.00      0.00      0.00        15

      accuracy                           0.92     86275
     macro avg       0.54      0.51      0.52     86275
  weighted avg       0.93      0.92      0.92     86275
```

```python
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Embedding
from keras.layers import LSTM, SimpleRNN, GRU
from keras.datasets import imdb
from keras.utils import to_categorical
from sklearn.metrics import (precision_score, recall_score,f1_score, accuracy_score,mean_squared_error,mean_absolute_error)
from sklearn import metrics
from sklearn.preprocessing import Normalizer
from keras import callbacks
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau, CSVLogger
```

```python
from sklearn.preprocessing import StandardScaler,LabelEncoder
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
X_train, X_test, y_train, y_test = train_test_split(X_from_tree, y_encoded, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train_std = scaler.fit_transform(X_train)
X_test_std = scaler.transform(X_test)
```

```python
model = Sequential()
model.add(Dense(units=128, input_dim=X_train_std.shape[1], activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(units=64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(units=len(np.unique(y_encoded)), activation='softmax'))
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```python
history = model.fit(X_train_std, y_train, epochs=10, batch_size=32, validation_split=0.2)
```

```
Epoch 1/10
8628/8628 [==============================] - 39s 4ms/step - loss: 0.3768 - accuracy: 0.8804 - val_loss: 0.2453 - val_accuracy: 0.907
Epoch 2/10
8628/8628 [==============================] - 36s 4ms/step - loss: 0.2732 - accuracy: 0.9034 - val_loss: 0.2348 - val_accuracy: 0.909
Epoch 3/10
8628/8628 [==============================] - 32s 4ms/step - loss: 0.2677 - accuracy: 0.9059 - val_loss: 0.2331 - val_accuracy: 0.909
Epoch 4/10
8628/8628 [==============================] - 32s 4ms/step - loss: 0.2552 - accuracy: 0.9070 - val_loss: 0.2296 - val_accuracy: 0.913
Epoch 5/10
8628/8628 [==============================] - 30s 3ms/step - loss: 0.2510 - accuracy: 0.9074 - val_loss: 0.2266 - val_accuracy: 0.911
Epoch 6/10
8628/8628 [==============================] - 34s 4ms/step - loss: 0.2487 - accuracy: 0.9088 - val_loss: 0.2237 - val_accuracy: 0.912
Epoch 7/10
8628/8628 [==============================] - 35s 4ms/step - loss: 0.2478 - accuracy: 0.9090 - val_loss: 0.2211 - val_accuracy: 0.912
Epoch 8/10
8628/8628 [==============================] - 33s 4ms/step - loss: 0.2437 - accuracy: 0.9111 - val_loss: 0.2152 - val_accuracy: 0.917
Epoch 9/10
8628/8628 [==============================] - 31s 4ms/step - loss: 0.2400 - accuracy: 0.9128 - val_loss: 0.2094 - val_accuracy: 0.919
Epoch 10/10
8628/8628 [==============================] - 31s 4ms/step - loss: 0.2385 - accuracy: 0.9143 - val_loss: 0.2046 - val_accuracy: 0.921
```

```python
accuracy = model.evaluate(X_test_std, y_test)[1]
print(f"Test Accuracy: {accuracy:.2%}")
```

```
2697/2697 [==============================] - 7s 2ms/step - loss: 0.1979 - accuracy: 0.9243
Test Accuracy: 92.43%
```

```
X_train_lstm = X_train_std.reshape((X_train_std.shape[0], 1, X_train_std.shape[1]))
```