import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import matplotlib.pyplot as plt # for data visualization

import seaborn as sns # for statistical data visualization

import socket

import struct

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import matplotlib.pyplot as plt # for data visualization

import seaborn as sns # for statistical data visualization

import socket

import struct

import warnings

​

df = pd.read\_csv(data)

data = 'C:\\Users\\udayk\\Desktop\\mmj\\fdata.csv'

​

df = pd.read\_csv(data)

df.shape

(59999, 81)

df.head(20)

Src\_IP Src Port Dst\_IP Dst Port Protocol Flow Duration Tot Fwd Pkts Tot Bwd Pkts TotLen Fwd Pkts TotLen Bwd Pkts ... Fwd Seg Size Min Active Mean Active Std Active Max Active Min Idle Mean Idle Std Idle Max Idle Min target\_class

0 3232236662 4504 3410565195 80 6 3974862 29 44 86 59811 ... 0 0.0 0.0 0 0 0.0 0.0 0 0 0

1 3232236662 4504 3410565195 80 6 63 1 1 0 0 ... 0 0.0 0.0 0 0 0.0 0.0 0 0 0

2 3232236662 4505 3410565195 80 6 476078 2 6 86 3037 ... 0 0.0 0.0 0 0 0.0 0.0 0 0 0

3 3232236662 4505 3410565195 80 6 151 2 1 0 0 ... 0 0.0 0.0 0 0 0.0 0.0 0 0 0

4 3232236662 4506 3410565195 80 6 472507 2 5 73 1050 ... 0 0.0 0.0 0 0 0.0 0.0 0 0 0

5 3232236662 4506 3410565195 80 6 138 1 1 0 0 ... 0 0.0 0.0 0 0 0.0 0.0 0 0 0

6 3232236662 4507 3410565195 80 6 469596 2 5 69 890 ... 0 0.0 0.0 0 0 0.0 0.0 0 0 0

7 3232236662 4507 3410565195 80 6 1978 1 1 0 0 ... 0 0.0 0.0 0 0 0.0 0.0 0 0 0

8 3232236662 4508 3410565195 80 6 468629 2 5 70 669 ... 0 0.0 0.0 0 0 0.0 0.0 0 0 0

9 3232236662 4508 3410565195 80 6 102 1 1 0 0 ... 0 0.0 0.0 0 0 0.0 0.0 0 0 0

10 3232236662 4509 3410565195 80 6 1434885 14 25 63 31473 ... 0 0.0 0.0 0 0 0.0 0.0 0 0 0

11 3232236662 4509 3410565195 80 6 39 1 1 0 0 ... 0 0.0 0.0 0 0 0.0 0.0 0 0 0

12 3232236662 4510 3410565195 80 6 966411 6 12 63 11803 ... 0 0.0 0.0 0 0 0.0 0.0 0 0 0

13 3232236662 4510 3410565195 80 6 6 1 1 0 0 ... 0 0.0 0.0 0 0 0.0 0.0 0 0 0

14 3232236662 4511 3410565195 80 6 510476 6 11 70 11403 ... 0 0.0 0.0 0 0 0.0 0.0 0 0 0

15 3232236662 4511 3410565195 80 6 6 1 1 0 0 ... 0 0.0 0.0 0 0 0.0 0.0 0 0 0

16 3232236662 4512 3410565195 80 6 513876 6 11 70 11403 ... 0 0.0 0.0 0 0 0.0 0.0 0 0 0

17 3232236662 4512 3410565195 80 6 6 1 1 0 0 ... 0 0.0 0.0 0 0 0.0 0.0 0 0 0

18 3232236662 4513 3410565195 80 6 506932 6 11 70 11403 ... 0 0.0 0.0 0 0 0.0 0.0 0 0 0

19 3232236662 4513 3410565195 80 6 6 1 1 0 0 ... 0 0.0 0.0 0 0 0.0 0.0 0 0 0

20 rows × 81 columns

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 59999 entries, 0 to 59998

Data columns (total 81 columns):

Src\_IP 59999 non-null int64

Src Port 59999 non-null int64

Dst\_IP 59999 non-null int64

Dst Port 59999 non-null int64

Protocol 59999 non-null int64

Flow Duration 59999 non-null int64

Tot Fwd Pkts 59999 non-null int64

Tot Bwd Pkts 59999 non-null int64

TotLen Fwd Pkts 59999 non-null int64

TotLen Bwd Pkts 59999 non-null int64

Fwd Pkt Len Max 59999 non-null int64

Fwd Pkt Len Min 59999 non-null int64

Fwd Pkt Len Mean 59999 non-null float64

Fwd Pkt Len Std 59999 non-null float64

Bwd Pkt Len Max 59999 non-null int64

Bwd Pkt Len Min 59999 non-null int64

Bwd Pkt Len Mean 59999 non-null float64

Bwd Pkt Len Std 59999 non-null float64

Flow Pkts/s 59999 non-null float64

Flow IAT Mean 59999 non-null float64

Flow IAT Std 59999 non-null float64

Flow IAT Max 59999 non-null int64

Flow IAT Min 59999 non-null int64

Fwd IAT Tot 59999 non-null int64

Fwd IAT Mean 59999 non-null float64

Fwd IAT Std 59999 non-null float64

Fwd IAT Max 59999 non-null int64

Fwd IAT Min 59999 non-null int64

Bwd IAT Tot 59999 non-null int64

Bwd IAT Mean 59999 non-null float64

Bwd IAT Std 59999 non-null float64

Bwd IAT Max 59999 non-null int64

Bwd IAT Min 59999 non-null int64

Fwd PSH Flags 59999 non-null int64

Bwd PSH Flags 59999 non-null int64

Fwd URG Flags 59999 non-null int64

Bwd URG Flags 59999 non-null int64

Fwd Header Len 59999 non-null int64

Bwd Header Len 59999 non-null int64

Fwd Pkts/s 59999 non-null float64

Bwd Pkts/s 59999 non-null float64

Pkt Len Min 59999 non-null int64

Pkt Len Max 59999 non-null int64

Pkt Len Mean 59999 non-null float64

Pkt Len Std 59999 non-null float64

Pkt Len Var 59999 non-null float64

FIN Flag Cnt 59999 non-null int64

SYN Flag Cnt 59999 non-null int64

RST Flag Cnt 59999 non-null int64

PSH Flag Cnt 59999 non-null int64

ACK Flag Cnt 59999 non-null int64

URG Flag Cnt 59999 non-null int64

CWE Flag Count 59999 non-null int64

ECE Flag Cnt 59999 non-null int64

Down/Up Ratio 59999 non-null int64

Pkt Size Avg 59999 non-null float64

Fwd Seg Size Avg 59999 non-null float64

Bwd Seg Size Avg 59999 non-null float64

Fwd Byts/b Avg 59999 non-null int64

Fwd Pkts/b Avg 59999 non-null int64

Fwd Blk Rate Avg 59999 non-null int64

Bwd Byts/b Avg 59999 non-null int64

Bwd Pkts/b Avg 59999 non-null int64

Bwd Blk Rate Avg 59999 non-null int64

Subflow Fwd Pkts 59999 non-null int64

Subflow Fwd Byts 59999 non-null int64

Subflow Bwd Pkts 59999 non-null int64

Subflow Bwd Byts 59999 non-null int64

Init Fwd Win Byts 59999 non-null int64

Init Bwd Win Byts 59999 non-null int64

Fwd Act Data Pkts 59999 non-null int64

Fwd Seg Size Min 59999 non-null int64

Active Mean 59999 non-null float64

Active Std 59999 non-null float64

Active Max 59999 non-null int64

Active Min 59999 non-null int64

Idle Mean 59999 non-null float64

Idle Std 59999 non-null float64

Idle Max 59999 non-null int64

Idle Min 59999 non-null int64

target\_class 59999 non-null int64

dtypes: float64(23), int64(58)

memory usage: 37.1 MB

def ip2int(addr):

return struct.unpack("!I", socket.inet\_aton(addr))[0]

#def ip2int(ip):

# o = map(int, ip.split('.'))

# res = (16777216 \* o[0]) + (65536 \* o[1]) + (256 \* o[2]) + o[3]

# return res

​

def Int2IP(ipnum):

o1 = int(ipnum / 16777216) % 256

o2 = int(ipnum / 65536) % 256

o3 = int(ipnum / 256) % 256

o4 = int(ipnum) % 256

return '%(o1)s.%(o2)s.%(o3)s.%(o4)s' % locals()

df.Src\_IP.apply(lambda x: ip2int(x))

df.Src\_IP.apply(lambda x: ip2int(x))

0 3232236662

1 3232236662

2 3232236662

3 3232236662

4 3232236662

...

59995 2887729751

59996 2887729163

59997 2887730455

59998 3232238097

59999 2887729690

Name: Src\_IP, Length: 60000, dtype: int64

df.Dst\_IP.apply(lambda x: ip2int(x))

0 3410565195

1 3410565195

2 3410565195

3 3410565195

4 3410565195

...

59995 2887712770

59996 2852039166

59997 2887712770

59998 918562044

59999 2887712770

Name: Dst\_IP, Length: 60000, dtype: int64

df['target\_class'].value\_counts()

0 30000

1 29999

Name: target\_class, dtype: int64

round(df.describe(),2)

Src\_IP Src Port Dst\_IP Dst Port Protocol Flow Duration Tot Fwd Pkts Tot Bwd Pkts TotLen Fwd Pkts TotLen Bwd Pkts ... Fwd Seg Size Min Active Mean Active Std Active Max Active Min Idle Mean Idle Std Idle Max Idle Min target\_class

count 5.999900e+04 59999.00 5.999900e+04 59999.00 59999.00 5.999900e+04 59999.00 59999.00 59999.00 59999.00 ... 59999.00 5.999900e+04 59999.00 5.999900e+04 5.999900e+04 5.999900e+04 59999.00 5.999900e+04 5.999900e+04 59999.0

mean 2.928749e+09 29338.85 2.528640e+09 4098.37 7.87 1.440959e+07 7.90 11.35 342.71 11032.83 ... 7.27 2.697059e+05 197856.48 5.113795e+05 1.426696e+05 3.826779e+06 347656.06 4.201537e+06 3.507851e+06 0.5

std 6.486651e+08 25482.71 9.103448e+08 13409.31 4.24 3.095443e+07 98.17 194.64 1210.39 272056.85 ... 8.91 2.472502e+06 1573565.65 3.448329e+06 2.066965e+06 1.198692e+07 2156348.14 1.271172e+07 1.170960e+07 0.5

min 1.718418e+07 0.00 2.042831e+07 0.00 0.00 0.000000e+00 0.00 0.00 0.00 0.00 ... 0.00 0.000000e+00 0.00 0.000000e+00 0.000000e+00 0.000000e+00 0.00 0.000000e+00 0.000000e+00 0.0

25% 2.887730e+09 2726.50 1.632277e+09 80.00 6.00 4.130000e+02 1.00 1.00 0.00 0.00 ... 0.00 0.000000e+00 0.00 0.000000e+00 0.000000e+00 0.000000e+00 0.00 0.000000e+00 0.000000e+00 0.0

50% 3.232236e+09 36936.00 2.887713e+09 80.00 6.00 4.688340e+05 2.00 2.00 58.00 118.00 ... 0.00 0.000000e+00 0.00 0.000000e+00 0.000000e+00 0.000000e+00 0.00 0.000000e+00 0.000000e+00 0.0

75% 3.232236e+09 53403.50 3.410565e+09 443.00 6.00 6.017230e+06 6.00 7.00 229.00 1581.00 ... 20.00 0.000000e+00 0.00 0.000000e+00 0.000000e+00 0.000000e+00 0.00 0.000000e+00 0.000000e+00 1.0

max 3.755864e+09 65532.00 4.026532e+09 65518.00 17.00 1.199996e+08 10105.00 20861.00 54342.00 30365784.00 ... 44.00 1.059147e+08 65906755.16 1.059147e+08 1.059147e+08 1.199339e+08 64689315.27 1.199339e+08 1.199339e+08 1.0

8 rows × 81 columns

X = df.drop(['target\_class'], axis=1)

​

y = df['target\_class']

# split X and y into training and testing sets

​

from sklearn.model\_selection import train\_test\_split

​

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

# check the shape of X\_train and X\_test

​

X\_train.shape, X\_test.shape

((47999, 80), (12000, 80))

cols = X\_train.columns

'C:\\Users\\udayk\\Desktop\\mmj\\fdata.csv'

df\_selected\_ddos =df.fillna(df.mean())

df\_selected\_ddos.to\_csv('C:\\Users\\udayk\\Desktop\\mmj\\fdata.csv')

np.any(np.isnan(df))

False

np.any(np.isnan(X\_test))

False

from sklearn.preprocessing import StandardScaler

​

scaler = StandardScaler()

​

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

X\_train = pd.DataFrame(X\_train, columns=[cols])

X\_test = pd.DataFrame(X\_test, columns=[cols])

# check for missing values in variables

​

df.isnull().sum()

​

Src\_IP 0

Src Port 0

Dst\_IP 0

Dst Port 0

Protocol 0

..

Idle Mean 0

Idle Std 0

Idle Max 0

Idle Min 0

target\_class 0

Length: 81, dtype: int64

# import SVC classifier

from sklearn.svm import SVC

​

​

# import metrics to compute accuracy

from sklearn.metrics import accuracy\_score

​

​

# instantiate classifier with default hyperparameters

svc=SVC()

​

​

# fit classifier to training set

svc.fit(X\_train,y\_train)

​

​

# make predictions on test set

y\_pred=svc.predict(X\_test)

​

​

# compute and print accuracy score

print('Model accuracy score with default hyperparameters: {0:0.4f}'. format(accuracy\_score(y\_test, y\_pred)))

---------------------------------------------------------------------------

Model accuracy score with default hyperparameters: 0.9827