```
#Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

```
#Load the data
df1 = pd.read_csv('/content/Tuesday-WorkingHours.pcap_ISCX.csv')
df2 = pd.read_csv('/content/Wednesday-workingHours.pcap_ISCX.csv')
df3 = pd.read_csv('/content/Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv')
```

Reading the data

```
#Combining of three Dataframes into one Dataframe
frames = [df1,df2,df3]
df = pd.concat(frames)
```

```
#It gives an overview about a Dataframe columns
df.info()
```

```
 23   Fwd IAT Max                  70208 non-null  float64
 24   Fwd IAT Min                  70208 non-null  float64
 25  Bwd IAT Total                 70208 non-null  float64
 26   Bwd IAT Mean                 70208 non-null  float64
 27   Bwd IAT Std                  70208 non-null  float64
 28   Bwd IAT Max                  70208 non-null  float64
 29   Bwd IAT Min                  70208 non-null  float64
 30  Fwd PSH Flags                 70208 non-null  float64
 31   Bwd PSH Flags                70207 non-null  float64
 32   Fwd URG Flags                70207 non-null  float64
 33   Bwd URG Flags                70207 non-null  float64
 34   Fwd Header Length            70207 non-null  float64
 35   Bwd Header Length            70207 non-null  float64
 36  Fwd Packets/s                 70207 non-null  float64
 37   Bwd Packets/s                70207 non-null  float64
 38   Min Packet Length            70207 non-null  float64
 39   Max Packet Length            70207 non-null  float64
 40   Packet Length Mean           70207 non-null  float64
 41   Packet Length Std            70207 non-null  float64
 42   Packet Length Variance       70207 non-null  float64
 43  FIN Flag Count                70207 non-null  float64
 44   SYN Flag Count               70207 non-null  float64
 45   RST Flag Count               70207 non-null  float64
 46   PSH Flag Count               70207 non-null  float64
 47   ACK Flag Count               70207 non-null  float64
 48   URG Flag Count               70207 non-null  float64
 49   CWE Flag Count               70207 non-null  float64
 50   ECE Flag Count               70207 non-null  float64
 51   Down/Up Ratio                70207 non-null  float64
 51   Down/Up Ratio                70207 non-null  float64
 52   Average Packet Size          70207 non-null  float64
 53   Avg Fwd Segment Size         70207 non-null  float64
 54   Avg Bwd Segment Size         70207 non-null  float64
 55   Fwd Header Length.1          70207 non-null  float64
 56  Fwd Avg Bytes/Bulk            70207 non-null  float64
 57   Fwd Avg Packets/Bulk         70207 non-null  float64
 58   Fwd Avg Bulk Rate            70207 non-null  float64
 59   Bwd Avg Bytes/Bulk           70207 non-null  float64
 60   Bwd Avg Packets/Bulk         70207 non-null  float64
 61  Bwd Avg Bulk Rate             70207 non-null  float64
 62  Subflow Fwd Packets           70207 non-null  float64
 63   Subflow Fwd Bytes            70207 non-null  float64
 64   Subflow Bwd Packets          70207 non-null  float64
 65   Subflow Bwd Bytes            70207 non-null  float64
 66  Init_Win_bytes_forward        70207 non-null  float64
 67   Init_Win_bytes_backward      70207 non-null  float64
 68   act_data_pkt_fwd             70207 non-null  float64
 69   min_seg_size_forward         70207 non-null  float64
 70  Active Mean                   70207 non-null  float64
 71   Active Std                   70207 non-null  float64
 72   Active Max                   70207 non-null  float64
 73   Active Min                   70207 non-null  float64
 74  Idle Mean                     70207 non-null  float64
 75   Idle Std                     70207 non-null  float64
 76   Idle Max                     70207 non-null  float64
 77   Idle Min                     70207 non-null  float64
 78   Label                        70207 non-null  object
dtypes: float64(68), int64(10), object(1)
memory usage: 42.9+ MB
```

```
#By default the head function returns the first 5 rows
df.head()
```

```
#By default the tail function returns the first 5 rows
df.tail()
```

```
#Count the number of rows and column in the data set
df.shape
```

```
(70210, 79)
```

```
#Explore the data
df.columns
```

```
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')
```

```
#It calculates some basic statistical details
df.describe()
```

Preprocessing the data

```
#It shows how many unique values are there in every column
df.nunique()
```

```
 Destination Port               5749
 Flow Duration                 46095
 Total Fwd Packets               448
 Total Backward Packets          508
Total Length of Fwd Packets     4821
                               ...  
Idle Mean                       8106
 Idle Std                       9786
 Idle Max                       4473
 Idle Min                      10089
 Label                             4
Length: 79, dtype: int64
```

```
#It will check if any value is NaN in a Dataframe
np.isnan(df.any())
```

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

```
#The function tests element-wise whether it is finite or not and return the result as a
np.isfinite(df.all())
```

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

```
#It will replace NaN values by Zeroes in a column of a Dataframe
df = df.fillna(0)
```

```
#It will remove nan and inf values in a Dataframe
df =df[~df.isin([np.nan, np.inf]).any(1)]
```

```
#Renaming the Dataframe column from Label to Attacks
df.rename({' Label':'Attacks'},axis=1,inplace=True)
```

```
#Returns a list of unique values
print(df['Attacks'].unique())
```

```
['BENIGN' 'FTP-Patator' 0 'DoS slowloris' 'Web Attack � Brute Force']
```

```
#dataframe Visualization
#We can plot histogram only for numerical attributres
df.hist(bins=50, figsize=(30,20))
plt.show()
```

```
sns.countplot(df['Attacks'])
```

```
#Return the data types of each column in the DataFrame.
df.dtypes
```

```
 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                       float64
 Idle Min                       float64
Attacks                          object
Length: 79, dtype: object
```

Feature engineering

```
from sklearn.preprocessing import LabelEncoder
Attacks_encoder = LabelEncoder()
df['Attacks']=Attacks_encoder.fit_transform(df['Attacks'].astype(str))
```

```
sns.distplot(df['Attacks'])
```

Splitting the data

```
#Split the data set into independent (X) and dependent (Y) data sets
# X --> contains the dataframe without the target i.e price
X = df.drop('Attacks',axis=1)
# Y --> contains only the target value
Y = df['Attacks']
```

```
#Split the data set into 75% training and 25% testing
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test= train_test_split(X, Y, test_size = 0.25, random_state
X_train = X_train.copy()
Y_train = Y_train.copy()
X_test = X_test.copy()
Y_test = Y_test.copy()
```

```
X_train.shape,Y_train.shape
```

```
((52597, 78), (52597,))
```

```
X_test.shape,Y_test.shape
```

```
((17533, 78), (17533,))
```

Feature Selection

Variance Threshold

```
# it will remove zero variance features
from sklearn.feature_selection import VarianceThreshold
var_thres = VarianceThreshold(threshold=0)
var_thres.fit(df)
```

```
VarianceThreshold(threshold=0)
```

```
#Getting all number of columns that have constant values
constant_columns = [column for column in df.columns
                    if column not in df.columns[var_thres.get_support()]]

print(len(constant_columns))
```

```
10
```

```
#Dropping constant_columns from the Dataframe
df.drop(constant_columns,axis=1,inplace=True)
```

```
#Returning the columns that are having constant values
for feature in constant_columns:
  print(feature)
```

```
 Bwd PSH Flags
 Fwd URG Flags
 Bwd URG Flags
 CWE Flag Count
Fwd Avg Bytes/Bulk
 Fwd Avg Packets/Bulk
 Fwd Avg Bulk Rate
 Bwd Avg Bytes/Bulk
```

```
  Bwd Avg Packets/Bulk
 Bwd Avg Bulk Rate
```

Feature Importance

```python
# decision tree for feature importance on a classification problem
from sklearn.datasets import make_classification
from sklearn.tree import DecisionTreeClassifier
from matplotlib import pyplot
# define the model
model = DecisionTreeClassifier()
# fit the model
model.fit(X_train, Y_train)
# get importance
importance = pd.Series(model.feature_importances_,index=X.columns)
# plot feature importance
importance.nlargest(10).plot(kind='barh')
pyplot.show()
```

```python
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculatio
# Create Decision Tree classifer object
model = DecisionTreeClassifier()
# Train Decision Tree Classifer
model = model.fit(X_train,Y_train)
#Predict the response for test dataset
y_pred = model.predict(X_test)
# Model Accuracy, how often is the classifier correct?
print("Decision Tree Training Accuracy: ",metrics.accuracy_score(Y_test, y_pred))
```

```
Decision Tree Training Accuracy:  0.9998288940854388
```

```python
# random forest for feature importance on a classification problem
from sklearn.datasets import make_classification
from sklearn.ensemble import RandomForestClassifier
from matplotlib import pyplot
# define the model
```

```
model = RandomForestClassifier()
model.fit(X_train,Y_train)
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_,index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
pyplot.show()
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
# classify using random forest classifier
classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(X_train, Y_train)
y_pred = classifier.predict(X_test)
# print the accuracy
print('Random Forest Training Accuracy : ' + str(accuracy_score(Y_test, y_pred)))
```

```
Random Forest Training Accuracy : 0.9834597615924258
```

Correlation Matrix with Heatmap

```
#get correlations of each features in dataset
corrmat = df.iloc[:,:-1].corr()
top_corr_features = corrmat.index
plt.figure(figsize=(55,40))
#plot heat map
g=sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```

```
# with the following function we can select highly correlated features
# it will remove the first feature that is correlated with anything other feature

def correlation(df, threshold):
    col_corr = set()  # Set of all the names of correlated columns
    corr_matrix = df.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold: # we are interested in absolute
                colname = corr_matrix.columns[i]  # getting the name of column
                col_corr.add(colname)
    return col_corr
```

```
corr_features = correlation(X_train, 0.85)
len(set(corr_features))
```

```
35
```

```
corr_features
```

```
{' Active Min',
 ' Average Packet Size',
 ' Avg Bwd Segment Size',
 ' Avg Fwd Segment Size',
 ' Bwd IAT Max',
 ' Bwd IAT Mean',
 ' Bwd IAT Min',
 ' Bwd Packet Length Mean',
 ' Bwd Packet Length Std',
 ' ECE Flag Count',
 ' Flow IAT Max',
 ' Flow IAT Std',
 ' Fwd Header Length.1',
 ' Fwd IAT Max',
 ' Fwd IAT Mean',
 ' Fwd IAT Min',
 ' Fwd Packet Length Std',
 ' Idle Max',
 ' Idle Min',
 ' Max Packet Length',
 ' Packet Length Mean',
 ' Packet Length Std',
 ' Packet Length Variance',
 ' SYN Flag Count',
 ' Subflow Bwd Bytes',
 ' Subflow Bwd Packets',
 ' Subflow Fwd Bytes',
 ' Total Backward Packets',
 ' Total Length of Bwd Packets',
 ' act_data_pkt_fwd',
 'Bwd IAT Total',
 'Fwd IAT Total',
 'Fwd Packets/s',
 'Idle Mean',
 'Subflow Fwd Packets'}
```

```
X_train.drop(corr_features,axis=1,inplace=True)
X_test.drop(corr_features,axis=1,inplace=True)
```

Feature Extraction

Principal Component Analysis(PCA)

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(df)
```

```
StandardScaler()
```

```
scaled_data=scaler.transform(df)
scaled_data
```

```
array([[-0.38073022, -0.51492432, -0.02174142, ..., -0.38075162,
        -0.36807126, -0.17935819],
       [-0.38073022, -0.51491725, -0.02116492, ..., -0.38075162,
        -0.36807126, -0.17935819],
       [-0.38073022, -0.51490895, -0.02174142, ..., -0.38075162,
        -0.36807126, -0.17935819],
       ...,
       [-0.38123306, -0.3586925 , -0.02289443, ..., -0.38075162,
        -0.36807126, -0.17935819],
       [-0.35841652, -0.35844285, -0.02174142, ..., -0.06353657,
        -0.04174499, -0.17935819],
       [-0.35841652, -0.35392095, -0.02202967, ..., -0.38075162,
        -0.36807126, -2.85846143]])
```

```
from sklearn.decomposition import PCA
pca=PCA(n_components=2)
pca.fit(scaled_data)
```

```
PCA(n_components=2)
```

```
x_pca=pca.transform(scaled_data)
```

```
scaled_data.shape
```

```
(70130, 69)
```

```
x_pca.shape
```

```
(70130, 2)
```

```
x_pca
```

```
array([[-1.43631724,  0.38878127],
       [ 0.19335055,  3.14376053],
       [ 1.35836627,  5.18771646],
       ...,
       [-1.35138189, -0.52696208],
       [-0.99697889, -0.06553593],
       [-1.74096465, -0.30221312]])
```

```
plt.figure(figsize=(8,6))
plt.scatter(x_pca[:,0],x_pca[:,1],c=df['Attacks'])
plt.xlabel('First principle component')
plt.ylabel('Second principle component')
```

```
from sklearn.dummy import DummyClassifier
from sklearn.metrics import accuracy_score
# define model
model = DummyClassifier(strategy='most_frequent')
# fit model
model.fit(x_pca, Y)
# make predictions
y = model.predict(x_pca)
# calculate accuracy
accuracy = accuracy_score(Y, y)
print('Naive Bayes Classifier Training Accuracy: %.3f' % accuracy)
```

```
Naive Bayes Classifier Training Accuracy: 0.968
```

```
from sklearn.linear_model import LogisticRegression
log = LogisticRegression(random_state=0,solver='lbfgs')
log.fit(x_pca, Y)
# print the accuracy
print('Logistic Regression Training Accuracy: ', log.score (x_pca, Y))
```

```
Logistic Regression Training Accuracy:  0.9680878368743762
```

Linear Discriminant Analysis

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

# apply Linear Discriminant Analysis
lda = LinearDiscriminantAnalysis(n_components=2)
X_train = lda.fit_transform(X_train, Y_train)
X_test = lda.transform(X_test)

# plot the scatterplot
plt.scatter(
    X_train[:,0],X_train[:,1],c=Y_train,cmap='rainbow',
  alpha=0.7,edgecolors='b'
)
```

```
from sklearn import metrics
from sklearn.metrics import accuracy_score

from sklearn.linear_model import LogisticRegression
log = LogisticRegression(random_state=0,solver='lbfgs')
log.fit(X_train, Y_train)

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, Y_train)
y_pred = knn.predict(X_test)

# print the accuracy
print('Logistic Regression Training Accuracy: ', log.score (X_train, Y_train))
print('KNN Training Accuracy: ', metrics.accuracy_score(Y_test, y_pred))
```

```
Logistic Regression Training Accuracy:  0.9735155997490351
KNN Training Accuracy:  0.9974334112815832
```

● ×