from google.colab import drive

drive.mount('/content/drive')

# This Python 3 environment comes with many helpful analytics libraries installed

# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python

# For example, here's several helpful packages to load

import numpy as np # linear algebra

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

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

import matplotlib.pyplot as plt

import seaborn as sns

import time

from sklearn.preprocessing import LabelEncoder

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.cluster import KMeans

from sklearn.metrics import accuracy\_score, confusion\_matrix, ConfusionMatrixDisplay, RocCurveDisplay, auc

from sklearn.metrics import precision\_score, recall\_score, f1\_score, roc\_curve

from sklearn.feature\_selection import RFE

from sklearn.tree import DecisionTreeClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.feature\_selection import mutual\_info\_classif

import xgboost as xgb

from lightgbm import LGBMClassifier

import keras

from numpy import array

# Input data files are available in the read-only "../input/" directory

# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os

**Concatinate train and test to make some data precessing and cleaning before splitting**

df\_train = pd.read\_csv("/content/drive/MyDrive/code/UNSW\_NB15\_testing-set.csv")

df\_test = pd.read\_csv("/content/drive/MyDrive/code/UNSW\_NB15\_training-set.csv")

df = pd.concat([df\_train, df\_test])

# information about the dataset

pd.set\_option('display.max\_columns', None)

df

**Get all Data with "Backdoor Attack" and get 2329 Normal data as the same number as the Backdoor Attack**

# Filter the DataFrame to include only "Backdoor" and "Normal" rows

backdoor\_data = df[df['attack\_cat'] == 'Backdoor']

normal\_data = df[df['attack\_cat'] == 'Normal']

# Randomly sample 2329 rows from the "Normal" category

sampled\_normal\_data = normal\_data.sample(n=2329, random\_state=42)  # You can adjust the random\_state if needed

# Combine the "Backdoor" and sampled "Normal" data

balanced\_data = pd.concat([backdoor\_data, sampled\_normal\_data])

# Shuffle the balanced data

balanced\_data\_shuffled = balanced\_data.sample(frac=1, random\_state=42)  # Shuffle with random\_state for reproducibility

unique\_sttl\_values = balanced\_data\_shuffled['state'].unique()

unique\_sttl\_values

**Check Any missing Value**

# Check for missing values in the dataset

missing\_values = balanced\_data\_shuffled.isnull().sum()

print("Missing values in the dataset:")

print(missing\_values)

target\_variable = balanced\_data\_shuffled['label']

plt.figure(figsize=(12, 8))

sns.boxplot(x=target\_variable, y=balanced\_data\_shuffled['sttl'], data=balanced\_data\_shuffled)

plt.title('Box Plot for sttl')

plt.show()

plt.hist(backdoor\_data['sttl'], bins=20, color='red', alpha=0.7, label='Backdoor', edgecolor='black')

plt.hist(normal\_data['sttl'], bins=20, color='blue', alpha=0.5, label='Normal', edgecolor='black')

plt.title('Histogram of sttl')

plt.xlabel('Total TTL')

plt.ylabel('Frequency')

plt.legend()

plt.tight\_layout()

plt.show()

normal\_data = balanced\_data\_shuffled[balanced\_data\_shuffled['label'] == 0]

backdoor\_data = balanced\_data\_shuffled[balanced\_data\_shuffled['label'] == 1]

# Scatter plot for Duration vs Packet Count

plt.figure(figsize=(10, 6))

# Backdoor traffic

plt.scatter(backdoor\_data['dur'], backdoor\_data['sbytes'], color='red', label='Backdoor', alpha=0.7)

# Normal traffic

plt.scatter(normal\_data['dur'], normal\_data['sbytes'], color='blue', label='Normal', alpha=0.5)

plt.title('Scatter Plot - Duration vs sbytes')

plt.xlabel('Duration')

plt.ylabel('Packet Count')

plt.legend()

plt.grid(True)

plt.show()

normal\_data = balanced\_data\_shuffled[balanced\_data\_shuffled['label'] == 0]['proto'].value\_counts()

print ( balanced\_data\_shuffled[balanced\_data\_shuffled['label'] == 1]['proto'].unique())

backdoor\_data = balanced\_data\_shuffled[balanced\_data\_shuffled['label'] == 1]['proto'].value\_counts()

# Calculate percentages

total\_normal = normal\_data.sum()

normal\_percentages = [(count / total\_normal) \* 100 for count in normal\_data]

total\_backdoor = backdoor\_data.sum()

backdoor\_percentages = [(count / total\_backdoor) \* 100 for count in backdoor\_data]

# Create Pie Charts

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 5))

# Pie chart for Normal Traffic

axes[0].pie(normal\_percentages, labels=normal\_data.index, autopct='%1.1f%%', startangle=90)

axes[0].set\_title('Normal Traffic Distribution')

# Pie chart for Backdoor Traffic

axes[1].pie(backdoor\_percentages, labels=backdoor\_data.index, autopct='%1.1f%%', startangle=90)

axes[1].set\_title('Backdoor Traffic Distribution')

plt.show()

target\_variable = balanced\_data\_shuffled['label']

plt.figure(figsize=(12, 8))

sns.violinplot(x=target\_variable, y=balanced\_data\_shuffled['sttl'], data=balanced\_data\_shuffled)

plt.title('Box Plot for sttl')

plt.show()

unique\_sttl\_values = balanced\_data\_shuffled['ackdat'].unique()

unique\_sttl\_values

target\_variable = balanced\_data\_shuffled['label']

plt.figure(figsize=(12, 8))

sns.boxplot(x=target\_variable, y=balanced\_data\_shuffled['ct\_state\_ttl'], data=balanced\_data\_shuffled)

plt.title('Box Plot for ct\_state\_ttl')

plt.show()

unique\_sttl\_values = balanced\_data\_shuffled['state'].unique()

unique\_sttl\_values

import pandas as pd

import statsmodels.api as sm

# Assuming your DataFrame is named 'df'

# Replace 'YourColumnName' with the actual column names in your DataFrame

# Select the columns of interest

selected\_columns = ['state', 'sttl', 'dttl', 'ct\_state\_ttl']

# Create a new DataFrame with only the selected columns

selected\_df = balanced\_data\_shuffled[selected\_columns]

# Convert 'state' column to numeric

selected\_df['state'] = pd.to\_numeric(selected\_df['state'], errors='coerce')

# Encode categorical variable 'state' into numerical values

selected\_df = pd.get\_dummies(selected\_df, columns=['state'], prefix='state', drop\_first=True)

# Convert all columns to numeric, handle any non-numeric values

selected\_df = selected\_df.apply(pd.to\_numeric, errors='coerce')

# Drop rows with missing values

selected\_df = selected\_df.dropna()

# Add a constant term for the intercept

selected\_df = sm.add\_constant(selected\_df)

# Separate independent variables (X) and dependent variable (y)

X = selected\_df.drop('ct\_state\_ttl', axis=1)  # Exclude the dependent variable

y = selected\_df['ct\_state\_ttl']