# Network Intrusion Detection System based on Machine Learning

**Introduction:**

In this project, I will make a network intrusion detection system, which is based on machine learning. An intrusion detection system is a device or software application that monitors networks or systems for malicious activity or policy violations. types of IDS range from single computers to large networks. The most common classifications are network intrusion detection systems (NIDS) and host-based intrusion detection systems (HIDS). It is worth noting that intrusion detection systems are fundamentally different from firewalls in that they both relate to network security. A traditional network firewall uses a set of static rules to allow or deny network connections. In other words, a firewall restricts access between networks to prevent intrusion, but does not signal an attack from within the network. In contrast, an IDS describes suspicious intrusions that occur and sends alerts, and monitors attacks that originate from within the system.

A network intrusion detection system (NIDS) is placed at one or more strategic points within a network to monitor traffic to and from all devices on the network. It analyzes the traffic passing through the entire subnet and matches the traffic passing through the subnet with a library of known attacks. Once an attack is identified or anomalous behavior is detected, an alert can be sent to the administrator. an example of a NIDS is to install it on the subnet where the firewall is to see if someone is trying to break into the firewall. Ideally, all inbound and outbound traffic can be scanned, but this can create bottlenecks that can affect the overall speed of the network. NIDS can also be combined with other technologies to improve detection and prediction rates. Because the self-organizing structure allows the IDS to identify intrusion patterns more effectively, machine learning based IDSs can analyze large amounts of data in an intelligent manner.

Machine learning is the study of computer algorithms that are automatically improved through experience and the use of data. It is considered a part of artificial intelligence. Machine learning algorithms build models based on sample data (or "training data") to make predictions or decisions without explicit programming. Machine learning is usually divided into three categories: supervised learning, unsupervised learning, and reinforcement learning. Commonly used models of machine learning algorithms are artificial neural networks, decision trees, support vector machines, Bayesian networks, and random forests.

In this project, I will use different machine learning algorithms to train different models so that I can compare their accuracy. As well as the accuracy and advantages and disadvantages of different models under specific conditions.

**Literature & Software:**

It has been realized for many years that machine learning can bring benefits to cyber security. Many research papers have been published. There are already some very good open-source networks intrusion detection tools available online, such as Snort, Security Onion and Zeek and. Here is my overview of some of them.

Snort can capture and analyze packets on the network, but what distinguishes it from other sniffers is that it can respond and process according to the defined rules. snort can take five kinds of response mechanisms: Activation, Dynamic, Alert, Pass, and Log, according to the rule chain after analyzing the acquired packets and conducting each rule. Snort has various functions such as packet sniffing, packet analysis, packet detection, response processing, etc. Each module implements different functions, and each module is combined with Snort in the form of a plug-in, making it easy to expand the functions.

For example, the pre-processing plug-in runs before the rule matching misuse detection, completes TIP fragmentation, http decoding, telnet decoding and other functions, the processing plug-in completes checking the protocol fields, closing the connection, attacking the response and other functions, and the output plug-in outputs the various situations in the form of logs or warnings after being processed.

Security Onion is a Linux distribution designed for intrusion detection and network security monitoring. Its installation process is simple, and a complete suite of NSM collection, detection and analysis can be deployed in a short time. security Onion may be active, used to identify vulnerabilities or expired SSL certificates. Or it may be reactive, such as incident response and network forensics. Its images can be distributed as sensors in the network to monitor multiple VLANs and subnets.

Zeek is a passive open-source network traffic analyzer that is primarily used as a security monitor that performs in-depth inspection of all traffic on a link and generates many log files to facilitate the search for signs of suspicious activity. The logs obtained after deploying Zeek include not only a comprehensive record of every connection on the network, but also application-level records, such as all HTTP sessions and their request URIs, key headers, MIME types and server responses; DNS requests with replies; SSL certificates; and key elements of SMTP sessions. Zeek is more flexible than traditional IDSs and can be customized and extended. Zeek's scripting language supports different methods to find suspicious activities on UNIX systems, including semantic abuse detection, anomaly detection and behavior analysis.

**Prerequisites:**

Hardware: PC / Laptop

Software: PyCharm.

PyCharm is an integrated development environment (IDE) for computer programming, specifically for the Python language. Upfront, I use PyCharm to process datasets locally so that they can be trained. I also use PyCharm to edit the code for different machine learning algorithms locally.

Other resources: Google Colab

Colab is an online development environment developed by Google Inc. One can write and execute Python code in the browser. The advantages are no configuration, free GPU use and easy sharing. This is certainly a great tool for machine learning where the GPU is a high requirement. I will use Colab to run my locally compiled Python code to derive results for comparison.

**Steps:**

1. Obtaining network traffic datasets

I will collect as much data as possible and make sure they meet the requirements of machine learning. The dataset is the starting point in your journey of building a machine learning model. Simply put, a dataset is essentially an M x N matrix, where M represents the columns (features) and N represents the rows (samples). Columns can be decomposed into X and Y. First, X is synonymous with several similar terms such as features, independent variables, and input variables. Second, Y is also a synonym for several terms, i.e., category labels, dependent variables, and output variables. In the preliminary stage, I will collect relevant datasets of web traffic on the web.

## 2. Using Exploratory Data Analysis (EDA) to analyze the dataset

Exploratory data analysis (EDA) is performed to gain an initial understanding of the data. In a typical data science project, the first thing to do is to "stare at the data" by performing EDA to better understand the data. I might use the following three approaches to data analysis.

Descriptive statistics: mean, median, mode, standard deviation.

Data visualization: heat map (to identify intra-feature correlations), box plot (to visualize group differences), scatter plot (to visualize correlations between features), principal component analysis (to visualize the clustering distribution presented in the data set), etc.

Data shaping: pivoting, grouping, filtering, etc. of data.

## 3. Pre-processing of the dataset

After the data analysis, I will pre-process the data to improve the quality of the data. This will improve the quality of the model and the accuracy of the model. Data pre-processing is the process of performing various checks and reviews on data to correct missing values, spelling errors, normalize/standardize values to make them comparable, transform data, etc. The quality of the data will have a great impact on the quality of the generated model. Therefore, a lot of effort should be spent on the data pre-processing stage to achieve the highest model quality. In general, data pre-processing can easily account for 80% of the time spent on a data science project, while the actual model building phase and subsequent model analysis only account for the remaining 20%. After the data analysis, I would pre-process the data to improve the quality of the data. This will improve the quality of the model and the accuracy of the model.

4. Segmentation of the dataset

After data pre-processing, I will segment the data. 80% of the data set will be used as training set and 20% as test set. During the development of machine learning models, it is expected that the trained model will perform well on new, unseen data. To simulate new, unseen data, the available data is data partitioned so that it is split into 2 parts (sometimes called training-test partition). In particular, the first part is a larger subset of the data used as the training set (e.g., 80% of the original data) and the second part is usually a smaller subset used as the test set (the remaining 20% of the data). It is important to note that this data splitting is performed only once. Next, a prediction model is built using the training set, and this trained model is then applied to the test set (i.e., as new, unseen data) for prediction. The best model is selected based on its performance on the test set, and hyperparameter optimization is also performed to obtain the best model.

5. Coding and modeling locally

After data segmentation, I will edit the code in PyCharm to build the model. Usually, the first step is to learn the algorithm. I will base the code on different machine learning algorithms. Then there is parameter tuning. I will tune the parameters according to different algorithms. The parameters of a machine learning algorithm directly affect the learning process and prediction performance, and there is no one parameter that can be universally applied to all data sets, so hyperparameter optimization is needed. Finally, there is feature selection. Feature selection is the process of selecting a subset of features from an initially large number of features. In addition to achieving highly accurate models, one of the most important aspects of machine learning model building is obtaining actionable insights, and to achieve this goal, it is important to be able to select a significant subset of features from many features.

6. Running the code on Colab

Since I do not have a GPU to run my model, I chose to run my code on Colab to test it.

7. Testing the model for accuracy

Test the model with the 20% of the data that was previously segmented. If there is a model in the process that is particularly inaccurate, consider reprocessing the data or tweaking the parameters.

8. Draw conclusions

By comparing the accuracy of several models, the optimal solution is derived, and the advantages and disadvantages of each model are analyzed. Finally, the characteristics of each model under different conditions are discussed.

**Detail & Results:**

The result of this project is a model trained based on network traffic data and machine learning algorithms. As the algorithms are different, the models are also different. After comparison it can be concluded which algorithm is more suitable for training network intrusion detection system model. Or for some specific network attack methods, which algorithm trained models will be more effective. Theoretically, the trained models can be deployed in servers for real-time monitoring of the network. However, model deployment is more difficult because of the high hardware and environment requirements. Even a platform needs to be built to perform model deployment. The results of this project can give some reference to those who want to use machine learning to enhance network security. It allows them to more clearly select the right algorithm to train the model.

**Benefits & Advantages:**

Ideally, an algorithmic model with high accuracy can be found through the project results. The model trained by machine learning has the following advantages.

1. Automation:

Once a model is trained and updated in time, it can be used continuously on a class of problems and works much better than a human on certain problems, such as large-scale image recognition and language recognition.

2. High efficiency:

If a traditional algorithm is used to evaluate the merits of a chess game, it may be computationally overwhelming and not always accurate. To evaluate it with a trained neural network is a blink of an eye. Therefore, AlphaGo is so powerful, it saves a lot of computation and makes things feasible that would otherwise not be feasible.

3. Plasticity:

If you use a traditional algorithm to solve a problem, the cost of adjusting the model can be to rewrite the code, which makes the cost of improvement huge. Deep learning can change the model by simply adjusting the parameters. This makes it highly flexible and growable. A program can be continuously improved and then reach the optimal state.

4. Universality:

Machine learning models solve problems by learning and can be built automatically based on the problem, so they can be applied to a variety of problems, not just limited to a fixed problem.

**Who will use it?**

The results of this project will be applied to cyber security analysis. Theoretically, the model can be applied to real-time monitoring of network traffic. This project will be used by people or companies who are interested in network security or need to make their networks more secure. A general definition of network security is that the hardware and software of a network information system and the data in its system are protected from accidental or malicious damage, alteration, or leakage, and that the system operates continuously, reliably, and normally without interruption of service. Network security is simply the ability to identify and eliminate insecurities in a networked environment. With the rapid popularity of the network, the network has become more and more influential to the society with its open and shared characteristics. The rise of various new businesses on the network, such as e-commerce, e-government, e-money, internet banking, and the construction of various professional networks, has made the security of various confidential information increasingly important. Computer crime incidents are rising year by year and have become a common international problem. In recent years, computer viruses, Trojan horses, worms, and hacker attacks have become increasingly popular, causing harm to national politics, economy, and society, and posing a serious threat to the Internet and national critical information systems. Most security threats are the use of security vulnerabilities in the system or software to achieve the purpose of damaging the system, theft of confidential information, etc., and the resulting security incidents have been numerous. Therefore, network security has become a hot topic in the society, and more and more people want to understand network security.

Theoretically, the results of this project can be used by individuals or enterprises for real-time monitoring of network traffic. The model has the role of identifying normal traffic and dangerous traffic. Individuals can use it to protect their personal or home LANs from hacking and thus personal information leakage. Companies and businesses can use it to protect their corporate LANs from theft of important information.

In addition, the results of this project can be used by some people who are interested in cyber security knowledge or machine learning knowledge. A lot of knowledge of network security and machine learning is included in this project. For example, there is a lot of data on the characteristics of network attacks included in the source data, which includes phishing, man-in-the-middle attacks, denial-of-service attacks, SQL injection, zero-day exploit attacks and DNS tunneling. Also, because this project will train different models based on different machine learning algorithms, it will also include the knowledge of machine learning algorithms. For example: artificial neural networks, decision trees, support vector machines, Bayesian networks and random forests. Users can compare the accuracy of different models to determine the advantages and disadvantages of each model under different conditions, and which model is more accurate.

**Requirements & Costs & Commitments:**

Although the training methods for machine learning models are now relatively mature, deploying models is not an easy task. Deploying models is very demanding in terms of hardware as well as environment. Currently, the main difficulties in deploying machine learning models are the lack of portability, the lack of scalability and the extremely high requirements for computing power (GPU). Often, deploying a machine learning model requires a server with extremely high computational and data processing power, and other special hardware and software environments. This makes it almost impossible for an individual to complete the deployment. There are only a few large-scale technology companies in the market that can do the deployment of models.

**Problems & Difficulties:**

With the rapid development of Internet technology, the network environment has become more complex, and the means of network attacks have become more diversified and sophisticated. The situation of network security is not optimistic. Some new malwares can generate up to 3 million new samples per hour. Traditional malware detection and malware analysis cannot keep up with new attacks and variants. New attacks and sophisticated malware have been able to bypass network and endpoint detection and launch cyber-attacks at an alarming rate.

Network security is a branch of information security in the computer and network domain. With the popularity of computers and the growth of networks, network security has become a central issue. Currently, network attacks have become very common and diverse. The traditional research methods of analyzing security problems and setting fixed rules have become inefficient. For example, a lot of manpower is needed to identify threats, extract features from threats, and encode these features into software to detect threats; spam detection methods that rely on fixed rules or black and whitelist filtering suffer from low detection efficiency and untimely rule updates. As the cyberspace environment becomes more and more complex and the data dimension keeps increasing, new demands are made on the research of cyberspace security issues.

**Bibliography:**

1. Panda, M., Abraham, A., Das, S., & Patra, M. R. (2011). Network intrusion detection system: A machine learning approach. *Intelligent Decision Technologies*, *5*(4), 347-356.
2. Wikimedia Foundation. (2021, June 25). *PyCharm*. Wikipedia. https://en.wikipedia.org/wiki/PyCharm.
3. Google. (n.d.). *Google Colaboratory*. Google Colab. https://colab.research.google.com/notebooks/intro.ipynb.
4. Wikimedia Foundation. (2021, June 21). *Snort (software)*. Wikipedia. https://en.wikipedia.org/wiki/Snort_(software).
5. *Security onion solutions*. Security Onion Solutions. (n.d.). https://securityonionsolutions.com/.
6. Wikimedia Foundation. (2020, November 13). *Zeek*. Wikipedia. https://en.wikipedia.org/wiki/Zeek.

```
from google.colab import drive
drive.mount('/content/drive')

    Mounted at /content/drive


import pandas as pd
import os
from sklearn import preprocessing
import time

%matplotlib inline
seconds = time.time()
print("This process may take a moment, depending on the performance of your computer.\n\n\n")
number = "0123456789"

# CSV files:
csv_files = ["Friday-02-03-2018_TrafficForML_CICFlowMeter",
                          "Friday-16-02-2018_TrafficForML_CICFlowMeter",
                          "Friday-23-02-2018_TrafficForML_CICFlowMeter",
                          "Thursday-01-03-2018_TrafficForML_CICFlowMeter",
                          "Thursday-15-02-2018_TrafficForML_CICFlowMeter",
                          "Thursday-22-02-2018_TrafficForML_CICFlowMeter",
                          "Wednesday-14-02-2018_TrafficForML_CICFlowMeter",
                          "Wednesday-21-02-2018_TrafficForML_CICFlowMeter",
                          "Wednesday-28-02-2018_TrafficForML_CICFlowMeter", ]

# Headers of columns:
main_labels = ["Dst Port", "Protocol", "Timestamp", "Flow Duration", "Tot Fwd Pkts", "Tot Bwd Pkts", "TotLen Fwd Pkts",
                              "TotLen Bwd Pkts", "Fwd Pkt Len Max", "Fwd Pkt Len Min", "Fwd Pkt Len Mean", "Fwd Pkt Len Std",
                              "Bwd Pkt Len Max", "Bwd Pkt Len Min", "Bwd Pkt Len Mean", "Bwd Pkt Len Std", "Flow Byts/s",
                              "Flow Pkts/s", "Flow IAT Mean", "Flow IAT Std", "Flow IAT Max", "Flow IAT Min", "Fwd IAT Tot",
                              "Fwd IAT Mean", "Fwd IAT Std", "Fwd IAT Max", "Fwd IAT Min", "Bwd IAT Tot", "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 Len", "Bwd Header Len", "Fwd Pkts/s", "Bwd Pkts/s", "Pkt Len Min",
                              "Pkt Len Max", "Pkt Len Mean", "Pkt Len Std", "Pkt Len Var", "FIN Flag Cnt", "SYN Flag Cnt",
                              "RST Flag Cnt", "PSH Flag Cnt", "ACK Flag Cnt", "URG Flag Cnt", "CWE Flag Count", "ECE Flag Cnt",
                              "Down/Up Ratio", "Pkt Size Avg", "Fwd Seg Size Avg", "Bwd Seg Size Avg", "Fwd Byts/b Avg",
                              "Fwd Pkts/b Avg", "Fwd Blk Rate Avg", "Bwd Byts/b Avg", "Bwd Pkts/b Avg", "Bwd Blk Rate Avg",
                              "Subflow Fwd Pkts", "Subflow Fwd Byts", "Subflow Bwd Pkts", "Subflow Bwd Byts", "Init Fwd Win Byts",
                              "Init Bwd Win Byts", "Fwd Act Data Pkts", "Fwd Seg Size Min", "Active Mean", "Active Std", "Active
                              "Active Min", "Idle Mean", "Idle Std", "Idle Max", "Idle Min", "Label"]


main_labels2 = main_labels
main_labels = (",".join(i for i in main_labels))
main_labels = main_labels + "\n"
flag = True

# Read CSV files
for i in range(len(csv_files)):
        ths = open(str(i) + ".csv", "w")
        ths.write(main_labels)
        with open("/content/drive/MyDrive/ISU/CSVs/" + csv_files[i] + ".csv", "r") as file:
                while True:
                        try:
                                line = file.readline()
                                # Get rid of the headers of CSV files and incomplete streams
                                if line[0] in number:
                                        # Change all "–" (Unicode: 8211) to "-" (Unicode: 45)
                                        if " – " in str(line):
                                                line = (str(line).replace(" – ", "-"))
                                        ths.write(str(line))
                                else:
                                        continue
                        except:
                                break
        ths.close()

        # Read CSV files
        df = pd.read_csv(str(i) + ".csv", low_memory=False)
        # Replace NaN with 0
        df.fillna(0, inplace=True)
        string_features = ["Flow Byts/s", "Flow Pkts/s"]

        # Change the data in "Flow Byts/s" and "Flow Pkts/s" to numeric
        for ii in string_features:
                # Replace Infinity with -1, replace NaN with 0
                df[ii].replace('Infinity', -1, inplace=True)
                df[ii].replace('NaN', 0, inplace=True)
                df[ii].replace('NaN', 0, inplace=True)
                number_or_not = []
                for iii in df[ii]:
                        try:
                                k = int(float(iii))
                                number_or_not.append(int(k))
                        except:
                                number_or_not.append(iii)
                df[ii] = number_or_not
        string_features = []

        # Detect non-numeric properties
        for j in main_labels2:
                if df[j].dtype == "object":
                        string_features.append(j)
        try:
                # Remove "Label" property
                string_features.remove('Label')
        except:
                print("error!")
        labelencoder_X = preprocessing.LabelEncoder()

        # Change non-numeric properties to numeric
        for ii in string_features:
                try:
                        df[ii] = labelencoder_X.fit_transform(df[ii])
                except:
                        # Replace Infinity with -1
                        df[ii].replace('Infinity', -1, inplace=True)
                        df[ii].replace('inf', -1, inplace=True)

        # Merge all CSV files into a single file
        if flag:
                df.to_csv('/content/drive/MyDrive/ISU/all_data.csv', index=False)
                flag = False
        else:
                df.to_csv('/content/drive/MyDrive/ISU/all_data.csv', index=False, header=False, mode="a")
        os.remove(str(i) + ".csv")
        print("The pre-processing phase of the ", csv_files[i], " file is completed.\n")

print("Mission accomplished!")
print("Total operation time: = ", time.time() - seconds, "seconds")
```

```
This process may take a moment, depending on the performance of your computer.



The pre-processing phase of the  Friday-02-03-2018_TrafficForML_CICFlowMeter  file is completed.

The pre-processing phase of the  Friday-16-02-2018_TrafficForML_CICFlowMeter  file is completed.

The pre-processing phase of the  Friday-23-02-2018_TrafficForML_CICFlowMeter  file is completed.

The pre-processing phase of the  Thursday-01-03-2018_TrafficForML_CICFlowMeter  file is completed.

The pre-processing phase of the  Thursday-15-02-2018_TrafficForML_CICFlowMeter  file is completed.

The pre-processing phase of the  Thursday-22-02-2018_TrafficForML_CICFlowMeter  file is completed.

The pre-processing phase of the  Wednesday-14-02-2018_TrafficForML_CICFlowMeter  file is completed.

The pre-processing phase of the  Wednesday-21-02-2018_TrafficForML_CICFlowMeter  file is completed.

The pre-processing phase of the  Wednesday-28-02-2018_TrafficForML_CICFlowMeter  file is completed.

Mission accomplished!
Total operation time: =  494.37202739715576 seconds
```

```
import pandas as pd

df = pd.read_csv('/content/drive/MyDrive/ISU/all_data.csv')

print(df.shape[0])
print(df.shape[1])
```

```
8284195
80
```

```
from google.colab import drive
drive.mount('/content/drive')

     Mounted at /content/drive


import matplotlib.pyplot as plt;

plt.rcdefaults()
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import time

%matplotlib inline
seconds = time.time()

# Graph creation
def graph(objects, performance, x_label, y_label):
        y_pos = np.arange(len(objects))
        plt.barh(y_pos, performance, align='center', alpha=0.5)
        plt.yticks(y_pos, objects)
        plt.xlabel(x_label)
        plt.title(y_label)
        plt.show()

# Read "all_data" file
df = pd.read_csv('/content/drive/MyDrive/ISU/all_data.csv', usecols=["Label"])
print(df.iloc[:, 0].value_counts())
a = (df.iloc[:, 0].value_counts())

key = a.keys()
values = a.values
small_labels = []
small_values = []
big_labels = []
big_values = []
medium_labels = []
medium_values = []
attack = 0
benign = 0

# Divide attacks into 3 groups
for i in range(0, len(values)):
        if values[i] > 11000:
                big_labels.append(str(key[i]))
                big_values.append(values[i])
        elif values[i] < 600:
                small_labels.append(str(key[i]))
                small_values.append(values[i])
        else:
                medium_labels.append(str(key[i]))
                medium_values.append(values[i])
        if str(key[i]) == "Benign":
                benign += values[i]
        else:
                attack += values[i]

key = [benign, attack]
labels = ["BENIGN %" + str(round(benign / (benign + attack), 2) * 100),
                   "ATTACK %" + str(round(attack / (benign + attack), 2) * 100)]

# Call the function to create graphes
graph(big_labels, big_values, "Numbers", "Attacks Labels - High-number group")
graph(medium_labels, medium_values, "Numbers", "Attacks Labels - Medium-number group")
graph(small_labels, small_values, "Numbers", "Attacks Labels - Small -number group")
graph(labels, key, "Numbers", "Attack and Benign Percentage")

print("Mission accomplished!")
print("Total operation time: = ", time.time() - seconds, "seconds")
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
     Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import random
import os
import pandas as pd
import time

%matplotlib inline
seconds = time.time()

# Create "attacks" folder
def folder(f_name):
        try:
                if not os.path.exists(f_name):
                        os.makedirs(f_name)
        except OSError:
                print("The folder could not be created!")


print("This process may take a moment, depending on the performance of your computer.\n\n\n")

# Headers of columns:
main_labels = ["Dst Port", "Protocol", "Timestamp", "Flow Duration", "Tot Fwd Pkts", "Tot Bwd Pkts", "TotLen Fwd Pkts",
                              "TotLen Bwd Pkts", "Fwd Pkt Len Max", "Fwd Pkt Len Min", "Fwd Pkt Len Mean", "Fwd Pkt Len Std",
                              "Bwd Pkt Len Max", "Bwd Pkt Len Min", "Bwd Pkt Len Mean", "Bwd Pkt Len Std", "Flow Byts/s",
                              "Flow Pkts/s", "Flow IAT Mean", "Flow IAT Std", "Flow IAT Max", "Flow IAT Min", "Fwd IAT Tot",
                              "Fwd IAT Mean", "Fwd IAT Std", "Fwd IAT Max", "Fwd IAT Min", "Bwd IAT Tot", "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 Len", "Bwd Header Len", "Fwd Pkts/s", "Bwd Pkts/s", "Pkt Len Min",
                              "Pkt Len Max", "Pkt Len Mean", "Pkt Len Std", "Pkt Len Var", "FIN Flag Cnt", "SYN Flag Cnt",
                              "RST Flag Cnt", "PSH Flag Cnt", "ACK Flag Cnt", "URG Flag Cnt", "CWE Flag Count", "ECE Flag Cnt",
                              "Down/Up Ratio", "Pkt Size Avg", "Fwd Seg Size Avg", "Bwd Seg Size Avg", "Fwd Byts/b Avg",
                              "Fwd Pkts/b Avg", "Fwd Blk Rate Avg", "Bwd Byts/b Avg", "Bwd Pkts/b Avg", "Bwd Blk Rate Avg",
                              "Subflow Fwd Pkts", "Subflow Fwd Byts", "Subflow Bwd Pkts", "Subflow Bwd Byts", "Init Fwd Win Byts",
                              "Init Bwd Win Byts", "Fwd Act Data Pkts", "Fwd Seg Size Min", "Active Mean", "Active Std", "Active
                              "Active Min", "Idle Mean", "Idle Std", "Idle Max", "Idle Min", "Label"]

main_labels = (",".join(i for i in main_labels))

# List of attacks:
attacks = ["Benign", "DDOS attack-HOIC", "DoS attacks-Hulk", "Bot", "FTP-BruteForce", "SSH-Bruteforce", "Infilteration",
                      "DoS attacks-SlowHTTPTest", "DoS attacks-GoldenEye", "DoS attacks-Slowloris", "DDOS attack-LOIC-UDP",
                      "Brute Force -Web", "Brute Force -XSS", "SQL Injection"]

# Path
folder("/content/drive/MyDrive/ISU/attacks/")
benign = 6112151

# Number of each attack:
dict_attack = {
        "DDOS attack-HOIC": 686012,
        "DoS attacks-Hulk": 461912,
        "Bot": 286191,
        "FTP-BruteForce": 193360,
        "SSH-Bruteforce": 187589,
        "Infilteration": 161934,
        "DoS attacks-SlowHTTPTest": 139890,
        "DoS attacks-GoldenEye": 41508,
        "DoS attacks-Slowloris": 10990,
        "DDOS attack-LOIC-UDP": 1730,
        "Brute Force -Web": 611,
        "Brute Force -XSS": 230,
        "SQL Injection": 87, }

# In this section, a file is opened for each attack type and is recorded at a random benign flow
for i in dict_attack:
        a, b = 0, 0
        ths = open("/content/drive/MyDrive/ISU/attacks/" + i + ".csv", "w")
        ths.write(str(main_labels) + "\n")
        benign_num = int(benign / (dict_attack[i] * (7 / 3)))
        with open("/content/drive/MyDrive/ISU/all_data.csv", "r") as file:
                while True:
                        try:
                                line = file.readline()
                                line = line[:-1]
                k = line.split(",")
                if k[79] == "Benign":
                    rnd = random.randint(1, benign_num)
                    if rnd == 1:
                        ths.write(str(line) + "\n")
                        b += 1
                if k[79] == i:
                    ths.write(str(line) + "\n")
                    a += 1
                else:
                    continue
            except:
                break
    ths.close()

    print(i, "file is completed\n Attack:%d\n Benign:%d\n\n\n " % (a, b))

# Merge all web attack files into a single file
webs = ["Brute Force -Web", "Brute Force -XSS", "SQL Injection"]
flag = True

for i in webs:
    df = pd.read_csv("/content/drive/MyDrive/ISU/attacks/" + str(i) + ".csv")
    if flag:
        df.to_csv('/content/drive/MyDrive/ISU/attacks/Web Attack.csv', index=False)
        flag = False
    else:
        df.to_csv('/content/drive/MyDrive/ISU/attacks/Web Attack.csv', index=False, header=False, mode="a")
    os.remove("/content/drive/MyDrive/ISU/attacks/" + str(i) + ".csv")

print("Mission accomplished!")
print("Total operation time: = ", time.time() - seconds, "seconds")
```

```
This process may take a moment, depending on the performance of your computer.



DDOS attack-HOIC file is completed
 Attack:686012
 Benign:2037873



DoS attacks-Hulk file is completed
 Attack:461912
 Benign:1222286



Bot file is completed
 Attack:286191
 Benign:679128



FTP-BruteForce file is completed
 Attack:193360
 Benign:470311



SSH-Bruteforce file is completed
 Attack:187589
 Benign:470887



Infilteration file is completed
 Attack:161934
 Benign:382081



DoS attacks-SlowHTTPTest file is completed
 Attack:139890
 Benign:340081



DoS attacks-GoldenEye file is completed
 Attack:41508
 Benign:96931
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
     Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import numpy as np
import os
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import RandomForestRegressor
import sklearn as sk
import time

%matplotlib inline
seconds = time.time()

# Create "feature_pics_attacks" folder
def folder(f_name):
        try:
                if not os.path.exists(f_name):
                        os.makedirs(f_name)
        except OSError:
                print("The folder could not be created!")

# CSV files names:
csv_files = os.listdir("/content/drive/MyDrive/ISU/attacks/")

# Headers of columns:
main_labels = ["Flow Duration", "Tot Fwd Pkts", "Tot Bwd Pkts", "TotLen Fwd Pkts", "TotLen Bwd Pkts", "Fwd Pkt Len Max",
                              "Fwd Pkt Len Min", "Fwd Pkt Len Mean", "Fwd Pkt Len Std", "Bwd Pkt Len Max", "Bwd Pkt Len Min",
                              "Bwd Pkt Len Mean", "Bwd Pkt Len Std", "Flow Byts/s", "Flow Pkts/s", "Flow IAT Mean", "Flow IAT Std
                              "Flow IAT Max", "Flow IAT Min", "Fwd IAT Tot", "Fwd IAT Mean", "Fwd IAT Std", "Fwd IAT Max",
                              "Fwd IAT Min", "Bwd IAT Tot", "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 Len", "Bwd Header L
                              "Fwd Pkts/s", "Bwd Pkts/s", "Pkt Len Min", "Pkt Len Max", "Pkt Len Mean", "Pkt Len Std", "Pkt Len
                              "FIN Flag Cnt", "SYN Flag Cnt", "RST Flag Cnt", "PSH Flag Cnt", "ACK Flag Cnt", "URG Flag Cnt",
                              "CWE Flag Count", "ECE Flag Cnt", "Down/Up Ratio", "Pkt Size Avg", "Fwd Seg Size Avg",
                              "Bwd Seg Size Avg", "Fwd Byts/b Avg", "Fwd Pkts/b Avg", "Fwd Blk Rate Avg", "Bwd Byts/b Avg",
                              "Bwd Pkts/b Avg", "Bwd Blk Rate Avg", "Subflow Fwd Pkts", "Subflow Fwd Byts", "Subflow Bwd Pkts",
                              "Subflow Bwd Byts", "Init Fwd Win Byts", "Init Bwd Win Byts", "Fwd Act Data Pkts", "Fwd Seg Size M
                              "Active Mean", "Active Std", "Active Max", "Active Min", "Idle Mean", "Idle Std", "Idle Max", "Idle M
                              "Label"]

# Path
ths = open("importance_list_for_attack_files.csv", "w")
folder("/content/drive/MyDrive/ISU/feature_pics_attack/")

for j in csv_files:
        # Read CSV files
        df = pd.read_csv("/content/drive/MyDrive/ISU/attacks/" + j, usecols=main_labels)
        # Replace NaN with 0
        df.fillna(0, inplace=True)
        # Replace Infinity with -1
        df.replace([np.inf, -np.inf], -1, inplace=True)
        attack_or_not = []

        # Change "Label" to 0/1 beased on the data is "Benign" or attack
        for i in df["Label"]:
                if i == "Benign":
                        attack_or_not.append(1)
                else:
                        attack_or_not.append(0)

        df["Label"] = attack_or_not
        y = df["Label"].values
        del df["Label"]
        X = df.values
        X = np.float32(X)
        X[np.isnan(X)] = 0
        X[np.isinf(X)] = 0

        # Calculate the feature importances
        forest = sk.ensemble.RandomForestRegressor(n_estimators=250, random_state=0)
        forest.fit(X, y)
        importances = forest.feature_importances_
        std = np.std([tree.feature_importances_ for tree in forest.estimators_], axis=0)
        indices = np.argsort(importances)[::-1]
```

```
    indices = np.argsort(importances)[::-1]
    refclasscol = list(df.columns.values)
    impor_bars = pd.DataFrame({'Features': refclasscol[0:20], 'importance': importances[0:20]})
    impor_bars = impor_bars.sort_values('importance', ascending=False).set_index('Features')
    plt.rcParams['figure.figsize'] = (10, 5)
    impor_bars.plot.bar();

    # Print the feature importances
    count = 0
    fea_ture = j[0:-4] + "=["

    for i in impor_bars.index:
        fea_ture = fea_ture + "\"" + str(i) + "\","
        count += 1
        if count == 5:
            fea_ture = fea_ture[0:-1] + "]"
            break

    print(j[0:-4], "importance list:")
    print(j[0:-4], "\n", impor_bars.head(20), "\n\n\n")
    print(fea_ture)
    # Print charts
    plt.title(j[0:-4] + " Attack - Feature Importance")
    plt.ylabel('Importance')
    plt.savefig("/content/drive/MyDrive/ISU/feature_pics_attack/" + j[0:-4] + ".pdf", bbox_inches='tight',
                papertype='a4', orientation='portrait', format='pdf')
    ths.write((fea_ture))
    plt.tight_layout()
    plt.show()
    print("------------------------------------------------------------------------------------------------\n\n\n\n")

print("Mission accomplished!")
print("Total operation time: = ", time.time() - seconds, "seconds")
ths.close()
```

```
from  google.colab  import  drive
drive.mount('/content/drive')

     Mounted at /content/drive


import  numpy  as  np
import  os
import  pandas  as  pd
import  matplotlib.pyplot  as  plt
from  sklearn.ensemble  import  ExtraTreesClassifier
from  sklearn.ensemble  import  RandomForestRegressor
import  sklearn  as  sk
import  time

%matplotlib  inline
seconds  =  time.time()

#  Create  "feature_pics_alldata"  folder
def  folder(f_name):
        try:
                if  not  os.path.exists(f_name):
                        os.makedirs(f_name)
        except  OSError:
                print("The  folder  could  not  be  created!")

#  CSV  files  names:
csv_files  =  ["/content/drive/MyDrive/ISU/all_data.csv"]

#  Headers  of  Columns:
main_labels  =  ["Flow  Duration",  "Tot  Fwd  Pkts",  "Tot  Bwd  Pkts",  "TotLen  Fwd  Pkts",  "TotLen  Bwd  Pkts",  "Fwd  Pkt  Len  Max",
                              "Fwd  Pkt  Len  Min",  "Fwd  Pkt  Len  Mean",  "Fwd  Pkt  Len  Std",  "Bwd  Pkt  Len  Max",  "Bwd  Pkt  Len  Min",
                              "Bwd  Pkt  Len  Mean",  "Bwd  Pkt  Len  Std",  "Flow  Byts/s",  "Flow  Pkts/s",  "Flow  IAT  Mean",  "Flow  IAT  Std
                              "Flow  IAT  Max",  "Flow  IAT  Min",  "Fwd  IAT  Tot",  "Fwd  IAT  Mean",  "Fwd  IAT  Std",  "Fwd  IAT  Max",
                              "Fwd  IAT  Min",  "Bwd  IAT  Tot",  "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  Len",  "Bwd  Header  L
                              "Fwd  Pkts/s",  "Bwd  Pkts/s",  "Pkt  Len  Min",  "Pkt  Len  Max",  "Pkt  Len  Mean",  "Pkt  Len  Std",  "Pkt  Len
                              "FIN  Flag  Cnt",  "SYN  Flag  Cnt",  "RST  Flag  Cnt",  "PSH  Flag  Cnt",  "ACK  Flag  Cnt",  "URG  Flag  Cnt",
                              "CWE  Flag  Count",  "ECE  Flag  Cnt",  "Down/Up  Ratio",  "Pkt  Size  Avg",  "Fwd  Seg  Size  Avg",
                              "Bwd  Seg  Size  Avg",  "Fwd  Byts/b  Avg",  "Fwd  Pkts/b  Avg",  "Fwd  Blk  Rate  Avg",  "Bwd  Byts/b  Avg",
                              "Bwd  Pkts/b  Avg",  "Bwd  Blk  Rate  Avg",  "Subflow  Fwd  Pkts",  "Subflow  Fwd  Byts",  "Subflow  Bwd  Pkts",
                              "Subflow  Bwd  Byts",  "Init  Fwd  Win  Byts",  "Init  Bwd  Win  Byts",  "Fwd  Act  Data  Pkts",  "Fwd  Seg  Size  M
                              "Active  Mean",  "Active  Std",  "Active  Max",  "Active  Min",  "Idle  Mean",  "Idle  Std",  "Idle  Max",  "Idle  M
                              "Label"]

#  Path
ths  =  open("importance_list_all_data.csv",  "w")
folder("/content/drive/MyDrive/ISU/feature_pics_alldata/")

for  j  in  csv_files:
        #  Read  "all_data"  file
        df  =  pd.read_csv(j,  usecols=main_labels)
        #  Replace  NaN  with  0
        df.fillna(0,  inplace=True)
        #  Replace  Infinity  with  -1
        df.replace([np.inf,  -np.inf],  -1,  inplace=True)
        attack_or_not  =  []

        #  Change  "Label"  to  0/1  beased  on  the  data  is  "Benign"  or  attack
        for  i  in  df["Label"]:
                if  i  ==  "Benign":
                        attack_or_not.append(1)
                else:
                        attack_or_not.append(0)

        df["Label"]  =  attack_or_not
        y  =  df["Label"].values
        del  df["Label"]
        X  =  df.values
        X  =  np.float32(X)
        X[np.isnan(X)]  =  0
        X[np.isinf(X)]  =  0

        #  Calculate  the  feature  importances
        forest  =  sk.ensemble.RandomForestRegressor(n_estimators=250,  random_state=0)
        forest.fit(X,  y)
        importances  =  forest.feature_importances_
        std  =  np.std([tree.feature_importances_  for  tree  in  forest.estimators_],  axis=0)
        indices  =  np.argsort(importances)[::-1]
```

```
    indices = np.argsort(importances)[::-1]
    refclasscol = list(df.columns.values)
    impor_bars = pd.DataFrame({'Features': refclasscol[0:20], 'importance': importances[0:20]})
    impor_bars = impor_bars.sort_values('importance', ascending=False).set_index('Features')
    plt.rcParams['figure.figsize'] = (10, 5)
    impor_bars.plot.bar();

    # Print the feature importances
    count = 0
    fea_ture = j[0:-4] + "=["

    for i in impor_bars.index:
        fea_ture = fea_ture + "\"" + str(i) + "\","
        count += 1
        if count == 5:
            fea_ture = fea_ture[0:-1] + "]"
            break

    print(j[0:-4], "importance list:")
    print(j[0:-4], "\n", impor_bars.head(20), "\n\n\n")
    print(fea_ture)
    # Print charts
    plt.title(j[0:-4] + " Attack - Feature Importance")
    plt.ylabel('Importance')
    plt.savefig("/content/drive/MyDrive/ISU/feature_pics_alldata/" + j[0:-4] + ".pdf", bbox_inches='tight',
                papertype='a4', orientation='portrait', format='pdf')
    ths.write(fea_ture)
    plt.tight_layout()
    plt.show()
    print("-----------------------------------------------------------------------------------------------\n\n\n\n")

print("Mission accomplished!")
print("Total operation time: = ", time.time() - seconds, "seconds")
ths.close()
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
from sklearn import metrics
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.metrics import average_precision_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
import csv
import time
import warnings
import math

%matplotlib inline
warnings.filterwarnings("ignore")

# The name and path of the results
result = "/content/drive/MyDrive/ISU/results/results_1.csv"
csv_files = os.listdir("/content/drive/MyDrive/ISU/attacks")
path = "/content/drive/MyDrive/ISU/attacks/"
repetition = 10

# Create "results" and "result_graph_1" folder
def folder(f_name):
        try:
                if not os.path.exists(f_name):
                        os.makedirs(f_name)
        except OSError:
                print("The folder could not be created!")

# Path:
folder_name = "/content/drive/MyDrive/ISU/results/"
folder(folder_name)
folder_name = "/content/drive/MyDrive/ISU/results/result_graph_1/"
folder(folder_name)

# List of machine learning algorithms:
ml_list = {
        "Naive Bayes": GaussianNB(),
        "QDA": QDA(),
        "Random Forest": RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
        "ID3": DecisionTreeClassifier(max_depth=5, criterion="entropy"),
        "AdaBoost": AdaBoostClassifier(),
        "MLP": MLPClassifier(hidden_layer_sizes=(13, 13, 13, 13, 13), max_iter=500),
        "Nearest Neighbors": KNeighborsClassifier(3)}

# Features used for each attack:
features = {"Bot": ["Bwd Pkt Len Mean", "Flow Pkts/s", "Flow IAT Mean", "Flow IAT Min", "Label"],
                        "DDOS attack-HOIC": ["Flow Duration", "Bwd Pkt Len Std", "Fwd IAT Tot", "Flow IAT Max", "Label"],
                        "DDOS attack-LOIC-UDP": ["Tot Fwd Pkts", "TotLen Fwd Pkts", "Flow Duration", "Bwd Pkt Len Mean", "Label"],
                        "DoS attacks-GoldenEye": ["Fwd Pkt Len Max", "Flow Duration", "Flow IAT Max", "Fwd IAT Tot", "Label"],
                        "DoS attacks-Hulk": ["Flow IAT Min", "Bwd Pkt Len Std", "Fwd Pkt Len Max", "TotLen Bwd Pkts", "Label"],
                        "DoS attacks-SlowHTTPTest": ["Flow Duration", "Flow IAT Mean", "Flow Pkts/s", "Flow IAT Max", "Label"],
                        "DoS attacks-Slowloris": ["Fwd Pkt Len Max", "Fwd Pkt Len Std", "Flow Byts/s", "Fwd IAT Tot", "Label"],
                        "FTP-BruteForce": ["Flow IAT Mean", "Flow Duration", "Flow IAT Max", "Tot Fwd Pkts", "Label"],
                        "Infilteration": ["Flow Byts/s", "Flow IAT Min", "Flow IAT Max", "Flow Duration", "Label"],
                        "SSH-Bruteforce": ["Flow IAT Max", "Flow Duration", "Flow Pkts/s", "Flow IAT Mean", "Label"],
                        "Web Attack": ["Flow IAT Max", "Fwd Pkt Len Mean", "Flow IAT Mean", "Flow Duration", "Label"]}

# Timestamp
seconds = time.time()

# Create a CSV file for results
# Create a csv file for results
with open(result, "w", newline="", encoding="utf-8") as f:
        wrt = csv.writer(f)
        wrt.writerow(["File", "ML algorithm", "accuracy", "Precision", "Recall", "F1-score", "Time"])

# This loop runs on the list containing the filenames. Operations are repeated for all attack files
for j in csv_files:
        print('%-17s %-17s  %-15s %-15s %-15s %-15s %-15s' % (
                "File", "ML algorithm", "accuracy", "Precision", "Recall", "F1-score", "Time"))

        a = []
        feature_list = list(features[j[0:-4]])
        # Read attack files
        df = pd.read_csv(path + j, usecols=feature_list)
        # Replace NaN with 0
        df.fillna(0, inplace=True)
        # Replace Infinity with -1
        df.replace([np.inf, -np.inf], -1, inplace=True)
        attack_or_not = []

        # Change "Label" to 0/1 beased on the data is "Benign" or attack
        for i in df["Label"]:
                if i == "Benign":
                        attack_or_not.append(1)
                else:
                        attack_or_not.append(0)

        df["Label"] = attack_or_not
        # Separate label and data into 2 pieces, data=x Label=y
        y = df["Label"]
        del df["Label"]
        feature_list.remove('Label')
        X = df[feature_list]

        # This loop runs on the list containing the machine learning algorithm names. Operations are repeated for all the 7 al
        for ii in ml_list:
                precision = []
                recall = []
                f1 = []
                accuracy = []
                t_time = []

                # This loop allows cross-validation and machine learning algorithm to be repeated 10 times
                for i in range(repetition):
                        second = time.time()
                        # Cross-validation
                        # Data(X) and Labels(y) are divided into 2 parts to be sent to the machine learning algorithm (80% train
                        # In total there are 4 tracks: training data(X_train), training tag(y_train), test data(X_test) and test ta
                        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=repetition)
                        # Machine learning algorithms are applied in this section
                        # Choose algorithm from ml_list dictionary
                        clf = ml_list[ii]
                        clf.fit(X_train, y_train)
                        predict = clf.predict(X_test)
                        # Make "classification report" and assign the precision, f-measure, and recall values
                        f_1 = f1_score(y_test, predict, average='macro')
                        pr = precision_score(y_test, predict, average='macro')
                        rc = recall_score(y_test, predict, average='macro')
                        precision.append(float(pr))
                        recall.append(float(rc))
                        f1.append(float(f_1))
                        accuracy.append(clf.score(X_test, y_test))
                        t_time.append(float((time.time() - second)))

                # Print the result of ten repetitions
                print('%-17s %-17s  %-15s %-15s %-15s %-15s %-15s' % (
                        j[0:-4], ii, str(round(np.mean(accuracy), 2)), str(round(np.mean(precision), 2)),
                        str(round(np.mean(recall), 2)), str(round(np.mean(f1), 2)), str(round(np.mean(t_time), 4))))

                # Save all the values in the file
                with open(result, "a", newline="", encoding="utf-8") as f:
                        wrt = csv.writer(f)

                        for i in range(0, len(t_time)):
                                # Write file name, algorithm name, precision, recall and f-measure in CSV file
                                wrt.writerow([j[0:-4], ii, accuracy[i], precision[i], recall[i], f1[i], t_time[i]])
                a.append(f1)

        # Create and save the graphics results
        ml = ["Naive Bayes", "QDA", "Random Forest", "ID3", "AdaBoost", "MLP", "Nearest Neighbors"]
        temp = 0
        fig, axes = plt.subplots(nrows=2, ncols=4, figsize=(12, 6), sharey=True)

        for c in range(2):
                for b in range(4):
                        axes[c, b].boxplot(a[temp])
                        axes[c, b].set_title(str(j[0:-4]) + " - " + str(ml[temp]), fontsize=7)
                        axes[c, b].set_ylabel(("F measure"))
                        temp += 1
                        if temp == 7:
                                break
                if temp == 7:
                        break

        plt.savefig(folder_name + j[0:-4] + ".pdf", bbox_inches='tight', papertype='a4', orientation='portrait',
                                format='pdf')
        plt.show()
        print(
                "\n----------------------------------------------------------------------------------------------------------\n\n")

print("Mission accomplished!")
print("Total operation time: = ", time.time() - seconds, "seconds")
```

```
from google.colab import drive
drive.mount('/content/drive')

     Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).


from sklearn import metrics
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.metrics import average_precision_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
import csv
import time
import warnings
import math

%matplotlib inline
warnings.filterwarnings("ignore")

# The name and path of the results
result = "/content/drive/MyDrive/ISU/results/results_2.csv"
csv_files = ["all_data.csv"]
path = "/content/drive/MyDrive/ISU/"
repetition = 10

# Create "results" and "result_graph_2" folder
def folder(f_name):
        try:
                if not os.path.exists(f_name):
                        os.makedirs(f_name)
        except OSError:
                print("The folder could not be created!")

# Path:
folder_name = "/content/drive/MyDrive/ISU/results/"
folder(folder_name)
folder_name = "/content/drive/MyDrive/ISU/results/result_graph_2/"
folder(folder_name)

# List of machine learning algorithms:
ml_list = {
        "Naive Bayes": GaussianNB(),
        "QDA": QDA(),
        "Random Forest": RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
        "ID3": DecisionTreeClassifier(max_depth=5, criterion="entropy"),
        "AdaBoost": AdaBoostClassifier(),
        "MLP": MLPClassifier(hidden_layer_sizes=(13, 13, 13, 13, 13), max_iter=500),
        "Nearest Neighbors": KNeighborsClassifier(3)}

# The features to be used for each attack type is defined in a dictionary(features)
# The first 4 of the features created by the file "1.4.a_SelectFeature_attacks.py" are used here
# The set of features to be used consists of combining the 4 features with the highest importance-weight achieved for each at
# Thus, 4 features are obtained from each of the 11 attack types, resulting in a pool of features consisting of 44 attribute
# After the repetitions are removed, the number of features is 15. The list of these features can be seen in below:
features = {
        "all_data": ["Bwd Pkt Len Mean", "Flow Pkts/s", "Flow IAT Mean", "Flow IAT Min", "Flow Duration", "Bwd Pkt Len Std",
                                  "Fwd IAT Tot", "Flow IAT Max", "Tot Fwd Pkts", "TotLen Fwd Pkts", "Fwd Pkt Len Max", "TotLen Bw
                                  "Fwd Pkt Len Std", "Flow Byts/s", "Fwd Pkt Len Mean", "Label"]}

# Timestamp
seconds = time.time()

# Create a CSV file to save the results
with open(result, "w", newline="", encoding="utf-8") as f:
        wrt = csv.writer(f)
        wrt.writerow(["File", "ML algorithm", "accuracy", "Precision", "Recall", "F1-score", "Time"])
        wrt.writerow(["File", "ML algorithm", "accuracy", "Precision", "Recall", "F1-score", "Time"])

# This loop runs on the list containing the filenames. Operations are repeated for all attack files
for j in csv_files:
        print('%-17s %-17s   %-15s %-15s %-15s %-15s %-15s' % (
                "File", "ML algorithm", "accuracy", "Precision", "Recall", "F1-score", "Time"))
        feature_list = list(features[j[0:-4]])
        # Read ""all_data"" file
        df = pd.read_csv(path + j, usecols=feature_list)
        # Replace NaN with 0
        df.fillna(0, inplace=True)
        # Replace Infinity with -1
        df.replace([np.inf, -np.inf], -1, inplace=True)
        attack_or_not = []

        # Change "Label" to 0/1 beased on the data is "Benign" or attack
        for i in df["Label"]:
                if i == "Benign":
                        attack_or_not.append(1)
                else:
                        attack_or_not.append(0)

        df["Label"] = attack_or_not
        # Separate label and data into 2 pieces, data=x Label=y
        y = df["Label"]
        del df["Label"]
        feature_list.remove('Label')
        X = df[feature_list]

        # This loop runs on the list containing the machine learning algorithm names. Operations are repeated for all the 7 al
        for ii in ml_list:
                precision = []
                recall = []
                f1 = []
                accuracy = []
                t_time = []

                # This loop allows cross-validation and machine learning algorithm to be repeated 10 times
                for i in range(repetition):
                        second = time.time()
                        # Cross-validation
                        # Data(X) and Labels(y) are divided into 2 parts to be sent to the machine learning algorithm (80% train
                        # In total there are 4 tracks: training data(X_train), training tag(y_train), test data(X_test) and test ta
                        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=repetition)
                        # Machine learning algorithms are applied in this section
                        # Choose algorithm from ml_list dictionary
                        clf = ml_list[ii]
                        clf.fit(X_train, y_train)
                        predict = clf.predict(X_test)
                        # Make "classification report" and assign the precision, f-measure, and recall values
                        f_1 = f1_score(y_test, predict, average='macro')
                        pr = precision_score(y_test, predict, average='macro')
                        rc = recall_score(y_test, predict, average='macro')
                        precision.append(float(pr))
                        recall.append(float(rc))
                        f1.append(float(f_1))
                        accuracy.append(clf.score(X_test, y_test))
                        t_time.append(float((time.time() - second)))

                # Print the result of ten repetitions
                print('%-17s %-17s   %-15s %-15s %-15s %-15s %-15s' % (
                        j[0:-4], ii, str(round(np.mean(accuracy), 2)), str(round(np.mean(precision), 2)),
                        str(round(np.mean(recall), 2)), str(round(np.mean(f1), 2)), str(round(np.mean(t_time), 4))))

                # Save all the values in the file
                with open(result, "a", newline="", encoding="utf-8") as f:
                        wrt = csv.writer(f)

                        for i in range(0, len(t_time)):
                                # Write file name, algorithm name, precision, recall and f-measure in CSV file
                                wrt.writerow([j[0:-4], ii, accuracy[i], precision[i], recall[i], f1[i], t_time[i]])

                # Create and save the graphics results
                plt.boxplot(f1)
                plt.title("All Dataset - " + str(ii))
                plt.ylabel('F-measure')
                plt.savefig(folder_name + j[0:-4] + str(ii) + ".pdf", bbox_inches='tight', papertype='a4',
                                        orientation='portrait', format='pdf')
                plt.show()
```

```
print("Mission  accomplished!")
print("Total  operation  time:  =  ",  time.time()  -  seconds,  "seconds")
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
    Mounted at /content/drive
```

```
from sklearn import metrics
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.metrics import average_precision_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
import csv
import time
import warnings
import math

%matplotlib inline
warnings.filterwarnings("ignore")

# The name and path of the results
result = "/content/drive/MyDrive/ISU/results/results_3.csv"
csv_files = ["all_data.csv"]
path = "/content/drive/MyDrive/ISU/"
repetition = 10

# Create "results" and "result_graph_3" folder
def folder(f_name):
        try:
                if not os.path.exists(f_name):
                        os.makedirs(f_name)
        except OSError:
                print("The folder could not be created!")

# Path:
folder_name = "/content/drive/MyDrive/ISU/results/"
folder(folder_name)
folder_name = "/content/drive/MyDrive/ISU/results/result_graph_3/"
folder(folder_name)

# List of machine learning algorithms:
ml_list = {
        "Naive Bayes": GaussianNB(),
        "QDA": QDA(),
        "Random Forest": RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
        "ID3": DecisionTreeClassifier(max_depth=5, criterion="entropy"),
        "AdaBoost": AdaBoostClassifier(),
        "MLP": MLPClassifier(hidden_layer_sizes=(13, 13, 13, 13, 13), max_iter=500),
        "Nearest Neighbors": KNeighborsClassifier(3)}

# The 7 features with the highest importance weight selected by the file "1.4.b_SelectFeature_alldata.py" are used here
features = {
        "all_data": ["Flow IAT Min", "Bwd Pkt Len Mean", "Fwd Pkt Len Std", "Fwd IAT Tot", "Bwd Pkt Len Std", "Flow Byts/s",
                                  "Fwd IAT Max", "Label"]}

# Timestamp
seconds = time.time()

# Create a CSV file to save the results
with open(result, "w", newline="", encoding="utf-8") as f:
        wrt = csv.writer(f)
        wrt.writerow(["File", "ML algorithm", "accuracy", "Precision", "Recall", "F1-score", "Time"])

# This loop runs on the list containing the filenames. Operations are repeated for all attack files
for j in csv_files:
        print('%-17s %-17s    %-15s %-15s %-15s %-15s %-15s' % (
                "File", "ML algorithm", "accuracy", "Precision", "Recall", "F1-score", "Time"))
                "File", "ML algorithm", "accuracy", "Precision", "Recall", "F1 score", "Time"])
        feature_list = list(features[j[0:-4]])
        # Read ""all_data"" file
        df = pd.read_csv(path + j, usecols=feature_list)
        # Replace NaN with 0
        df.fillna(0, inplace=True)
        # Replace Infinity with -1
        df.replace([np.inf, -np.inf], -1, inplace=True)
        attack_or_not = []

        # Change "Label" to 0/1 beased on the data is "Benign" or attack
        for i in df["Label"]:
                if i == "Benign":
                        attack_or_not.append(1)
                else:
                        attack_or_not.append(0)

        df["Label"] = attack_or_not
        # Separate label and data into 2 pieces, data=x Label=y
        y = df["Label"]
        del df["Label"]
        feature_list.remove('Label')
        X = df[feature_list]

        # This loop runs on the list containing the machine learning algorithm names. Operations are repeated for all the 7 al
        for ii in ml_list:
                precision = []
                recall = []
                f1 = []
                accuracy = []
                t_time = []

                # This loop allows cross-validation and machine learning algorithm to be repeated 10 times
                for i in range(repetition):
                        second = time.time()
                        # Cross-validation
                        # Data(X) and Labels(y) are divided into 2 parts to be sent to the machine learning algorithm (80% train
                        # In total there are 4 tracks: training data(X_train), training tag(y_train), test data(X_test) and test ta
                        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=repetition)
                        # Machine learning algorithms are applied in this section
                        # Choose algorithm from ml_list dictionary
                        clf = ml_list[ii]
                        clf.fit(X_train, y_train)
                        predict = clf.predict(X_test)
                        # Make "classification report" and assign the precision, f-measure, and recall values
                        f_1 = f1_score(y_test, predict, average='macro')
                        pr = precision_score(y_test, predict, average='macro')
                        rc = recall_score(y_test, predict, average='macro')
                        precision.append(float(pr))
                        recall.append(float(rc))
                        f1.append(float(f_1))
                        accuracy.append(clf.score(X_test, y_test))
                        t_time.append(float((time.time() - second)))

                # Print the result of ten repetitions
                print('%-17s %-17s  %-15s %-15s %-15s %-15s %-15s' % (
                        j[0:-4], ii, str(round(np.mean(accuracy), 2)), str(round(np.mean(precision), 2)),
                        str(round(np.mean(recall), 2)), str(round(np.mean(f1), 2)), str(round(np.mean(t_time), 4))))

                # Save all the values in the file
                with open(result, "a", newline="", encoding="utf-8") as f:
                        wrt = csv.writer(f)

                        for i in range(0, len(t_time)):
                                # Write file name, algorithm name, precision, recall and f-measure in CSV file
                                wrt.writerow([j[0:-4], ii, accuracy[i], precision[i], recall[i], f1[i], t_time[i]])

                # Create and save the graphics results
                plt.boxplot(f1)
                plt.title("All Dataset - " + str(ii))
                plt.ylabel('F-measure')
                plt.savefig(folder_name + j[0:-4] + str(ii) + ".pdf", bbox_inches='tight', papertype='a4',
                                        orientation='portrait', format='pdf')
                plt.show()

print("Mission accomplished!")
print("Total operation time: = ", time.time() - seconds, "seconds")
```

```
from google.colab import drive
drive.mount('/content/drive')

     Mounted at /content/drive


from sklearn import metrics
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.metrics import average_precision_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
import numpy as np
import pandas as pd
import warnings
import time

%matplotlib inline
warnings.filterwarnings("ignore")
seconds = time.time()

# The 20 features selected by the file "1.4.b_SelectFeature_alldata.py" are used here
features = ["Flow IAT Min", "Bwd Pkt Len Mean", "Fwd Pkt Len Std", "Fwd IAT Tot", "Bwd Pkt Len Std", "Flow Byts/s",
                        "Flow IAT Max", "Flow Duration", "Flow IAT Mean", "Flow IAT Std", "Fwd Pkt Len Max", "Flow Pkts/s",
                        "Bwd Pkt Len Max", "Tot Bwd Pkts", "TotLen Fwd Pkts", "Tot Fwd Pkts", "TotLen Bwd Pkts", "Fwd Pkt Len M
                        "Bwd Pkt Len Min", "Fwd Pkt Len Min", "Label"]

# Read ""all_data"" file
df = pd.read_csv('/content/drive/MyDrive/ISU/all_data.csv', usecols=features)
# Replace NaN with 0
df.fillna(0, inplace=True)
# Replace Infinity with -1
df.replace([np.inf, -np.inf], -1, inplace=True)

print('%-17s %-17s ' % ("Feature Number", "Feature"))

for i in range(len(features) - 1):
        # Print features and feature numbers
        print('%-17s %-17s' % (i + 1, features[i]))

print('\n\n\n')
attack_or_not = []

# Change "Label" to 0/1 beased on the data is "Benign" or attack
for i in df.iloc[:, -1]:
        if i == "Benign":
                attack_or_not.append(1)
        else:
                attack_or_not.append(0)

df.iloc[:, -1] = attack_or_not
y = df.iloc[:, -1].values
my_list = []
least = 0

# List of machine learning algorithms:
ml_list = {"Naive Bayes": GaussianNB(),
                      "QDA": QDA(),
                      "MLP": MLPClassifier(hidden_layer_sizes=(13, 13, 13, 13, 13), max_iter=500)}

# Remove the "Label"
features.pop()
print('%-17s %-30s %-10s   %-10s %-15s ' % ("ML algorithm", "Feature Name", "F1-score", "Accuracy", "Feature List"))

# Run every machine learning algorithm
for j in ml_list:
        my_list = []

        # Run every feature
```

```
    # Run every feature
    for i in features:
        my_list.append(i)
        X = df.loc[:, my_list].values
        # Cross-validation
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)
        # Apply machine learning algorithms
        clf = ml_list[j]
        clf.fit(X_train, y_train)
        predict = clf.predict(X_test)
        f1 = clf.score(X_test, y_test)
        result = f1_score(y_test, predict, average='macro')
        accuracy = round(clf.score(X_test, y_test), 2)
        temp = "["

        # Translate property list to sequence number for less space
        for ii in my_list:
            temp += str(my_list.index(ii) + 1) + ", "
        # If the F-criterion is equal to or greater than the highest value previously accessed, keep the new feature
        if result >= least:
            least = result
            print('%-17s %-30s %-10s    %-10s %-15s %-15s ' % (
                j, i, result, accuracy, temp, "------> New feature found!!!"))
        # If not, remove it from the list
        else:
            my_list.remove(my_list[len(my_list) - 1])
            print('%-17s %-30s %-10s    %-10s %-15s ' % (j, i, result, accuracy, temp))
    # Print maximum F1 and the most efficient feature list
    print("F1=", least, j, " The most efficient feature list =", my_list, "\n\n")

print("Mission accomplished!")
print("Total operation time: = ", time.time() - seconds, "seconds")
```

```
Feature Number   Feature
1                Flow IAT Min
2                Bwd Pkt Len Mean
3                Fwd Pkt Len Std
4                Fwd IAT Tot
5                Bwd Pkt Len Std
6                Flow Byts/s
7                Flow IAT Max
8                Flow Duration
9                Flow IAT Mean
10               Flow IAT Std
11               Fwd Pkt Len Max
12               Flow Pkts/s
13               Bwd Pkt Len Max
14               Tot Bwd Pkts
15               TotLen Fwd Pkts
16               Tot Fwd Pkts
17               TotLen Bwd Pkts
18               Fwd Pkt Len Mean
19               Bwd Pkt Len Min
20               Fwd Pkt Len Min




ML algorithm     Feature Name                   F1-score    Accuracy   Feature List
Naive Bayes      Flow IAT Min                   0.2730517012608505  0.31       [1,             ------> New feature found!!!
Naive Bayes      Bwd Pkt Len Mean               0.2730517012608505  0.31       [1, 2,          ------> New feature found!!!
Naive Bayes      Fwd Pkt Len Std                0.2730517012608505  0.31       [1, 2, 3,       ------> New feature found!!!
Naive Bayes      Fwd IAT Tot                    0.3462846045466874  0.36       [1, 2, 3, 4,    ------> New feature found!!!
Naive Bayes      Bwd Pkt Len Std                0.3462846045466874  0.36       [1, 2, 3, 4, 5,  ------> New feature found!!!
Naive Bayes      Flow Byts/s                    0.3479107406216101  0.36       [1, 2, 3, 4, 5, 6,  ------> New feature found!!!
Naive Bayes      Flow IAT Max                   0.3439284111725323  0.36       [1, 2, 3, 4, 5, 6, 7,
Naive Bayes      Flow Duration                  0.3520771211107586  0.37       [1, 2, 3, 4, 5, 6, 7,  ------> New feature found!!!
Naive Bayes      Flow IAT Mean                  0.3516934780118647  0.37       [1, 2, 3, 4, 5, 6, 7, 8,
Naive Bayes      Flow IAT Std                   0.35195114761868307  0.37       [1, 2, 3, 4, 5, 6, 7, 8,
Naive Bayes      Fwd Pkt Len Max                0.3520771211107586  0.37       [1, 2, 3, 4, 5, 6, 7, 8,  ------> New feature found!!!
Naive Bayes      Flow Pkts/s                    0.3525791222629955  0.37       [1, 2, 3, 4, 5, 6, 7, 8, 9,  ------> New feature found!!!
Naive Bayes      Bwd Pkt Len Max                0.3525791222629955  0.37       [1, 2, 3, 4, 5, 6, 7, 8, 9, 10,  ------> New feature foun
Naive Bayes      Tot Bwd Pkts                   0.3525791222629955  0.37       [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,  ------> New feature
Naive Bayes      TotLen Fwd Pkts                0.35306742303338456  0.37       [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,  ------> New fea
Naive Bayes      Tot Fwd Pkts                   0.35307258123265894  0.37       [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,  ------> New
Naive Bayes      TotLen Bwd Pkts                0.3526949733082554  0.37       [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
Naive Bayes      Fwd Pkt Len Mean               0.35307258123265894  0.37       [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,  ------>
Naive Bayes      Bwd Pkt Len Min                0.35307258123265894  0.37       [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,  ---
Naive Bayes      Fwd Pkt Len Min                0.35307258123265894  0.37       [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
F1= 0.35307258123265894 Naive Bayes  The most efficient feature list = ['Flow IAT Min', 'Bwd Pkt Len Mean', 'Fwd Pkt Len Std', 'Fwd IAT To


QDA              Flow IAT Min                   0.2730517012608505  0.31       [1,
from google.colab import drive
drive.mount('/content/drive')

    Mounted at /content/drive


from sklearn import metrics
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.metrics import average_precision_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
import matplotlib.pyplot as plt
from sklearn import preprocessing
import numpy as np
import os
import pandas as pd
import csv
import time
import warnings
import math

%matplotlib inline
warnings.filterwarnings("ignore")

# Create "feature_graph" folder
def folder(f_name):
        try:
                if not os.path.exists(f_name):
                        os.makedirs(f_name)
        except OSError:
                print("The folder could not be created!")

# The name and path of the results
result = "/content/drive/MyDrive/ISU/results/results_Final.csv"
csv_files = ["all_data.csv"]
path = "/content/drive/MyDrive/ISU/"
repetition = 10

# Create "results" and "result_graph_Final" folder
def folder(f_name):
        try:
                if not os.path.exists(f_name):
                        os.makedirs(f_name)
        except OSError:
                print("The folder could not be created!")

# Path:
folder_name = "/content/drive/MyDrive/ISU/results/"
folder(folder_name)
folder_name = "/content/drive/MyDrive/ISU/results/result_graph_Final/"
folder(folder_name)

# The 20 features selected by the file "1.4.b_SelectFeature_alldata.py" are used here
usecols = ["Flow IAT Min", "Bwd Pkt Len Mean", "Fwd Pkt Len Std", "Fwd IAT Tot", "Bwd Pkt Len Std", "Flow Byts/s",
                    "Flow IAT Max", "Flow Duration", "Flow IAT Mean", "Flow IAT Std", "Fwd Pkt Len Max", "Flow Pkts/s",
                    "Bwd Pkt Len Max", "Tot Bwd Pkts", "TotLen Fwd Pkts", "Tot Fwd Pkts", "TotLen Bwd Pkts", "Fwd Pkt Len Mea
                    "Bwd Pkt Len Min", "Fwd Pkt Len Min", "Label"]

# List of machine learning algorithms:
ml_list = {"Naive Bayes": GaussianNB(), "QDA": QDA(),
                    "MLP": MLPClassifier(hidden_layer_sizes=(13, 13, 13, 13, 13), max_iter=500),
                    "Random Forest": RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
                    "ID3": DecisionTreeClassifier(max_depth=5, criterion="entropy"), "AdaBoost": AdaBoostClassifier(),
                    "Nearest Neighbors": KNeighborsClassifier(3)}

# The features to be used for Random Forest, ID3, AdaBoost, Nearest Neighbors are defined in a list
# The first 7 of the features created by the file "1.4.b_SelectFeature_alldata.py" are used here
others = ["Flow IAT Min", "Bwd Pkt Len Mean", "Fwd Pkt Len Std", "Fwd IAT Tot", "Bwd Pkt Len Std", "Flow Byts/s",
                  "Flow IAT Max"]
```

```
# In this part different sets of properties for machine learning methods are defined as follows:
# For "Naive Bayes", "QDA", and "MLP", each method has a different feature list
# Other algorithms (Random Forest, ID3, AdaBoost, and Nearest Neighbors) use the first 7 of the features created by the file
algorithms_features = {
        "Naive Bayes": ['Flow IAT Min', 'Bwd Pkt Len Mean', 'Fwd Pkt Len Std', 'Fwd IAT Tot', 'Bwd Pkt Len Std',
                                        'Flow Byts/s', 'Flow Duration', 'Fwd Pkt Len Max', 'Flow Pkts/s', 'Bwd Pkt Len Max', 'Tot B
                                        'TotLen Fwd Pkts', 'Tot Fwd Pkts', 'Fwd Pkt Len Mean', 'Bwd Pkt Len Min', 'Fwd Pkt Len Mi
        "QDA": ['Bwd Pkt Len Mean', 'Fwd Pkt Len Std', 'Bwd Pkt Len Std', 'Fwd Pkt Len Max', 'Flow Pkts/s',
                        'TotLen Fwd Pkts', 'Fwd Pkt Len Mean'],
        "MLP": ['Flow Byts/s', 'Flow IAT Max', 'Flow Duration', 'Tot Bwd Pkts'],
        "Random Forest": others,
        "ID3": others,
        "AdaBoost": others,
        "Nearest Neighbors": others}

# Timestamp
seconds = time.time()

# Create a CSV file to save the results
with open(result, "w", newline="", encoding="utf-8") as f:
        wrt = csv.writer(f)
        wrt.writerow(["File", "ML algorithm", "accuracy", "Precision", "Recall", "F1-score", "Time"])

# This loop runs on the list containing the filenames. Operations are repeated for all attack files
for j in csv_files:
        print('%-17s %-17s    %-15s %-15s %-15s %-15s %-15s' % (
                "File", "ML algorithm", "accuracy", "Precision", "Recall", "F1-score", "Time"))
        feature_list = usecols
        # Read ""all_data"" file
        df = pd.read_csv(path + j, usecols=feature_list)
        # Replace NaN with 0
        df.fillna(0, inplace=True)
        # Replace Infinity with -1
        df.replace([np.inf, -np.inf], -1, inplace=True)
        attack_or_not = []

        # Change "Label" to 0/1 beased on the data is "Benign" or attack
        for i in df["Label"]:
                if i == "Benign":
                        attack_or_not.append(1)
                else:
                        attack_or_not.append(0)

        df["Label"] = attack_or_not
        # Separate label and data into 2 pieces, data=x Label=y
        y = df["Label"]
        del df["Label"]
        feature_list.remove('Label')

        # This loop runs on the list containing the machine learning algorithm names. Operations are repeated for all the 7 al
        for ii in ml_list:
                X = df[algorithms_features[ii]]
                precision = []
                recall = []
                f1 = []
                accuracy = []
                t_time = []

                # This loop allows cross-validation and machine learning algorithm to be repeated 10 times
                for i in range(repetition):
                        second = time.time()
                        # Cross-validation
                        # Data(X) and Labels(y) are divided into 2 parts to be sent to the machine learning algorithm (80% train
                        # In total there are 4 tracks: training data(X_train), training tag(y_train), test data(X_test) and test ta
                        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=repetition)
                        # Machine learning algorithms are applied in this section
                        # Choose algorithm from ml_list dictionary
                        clf = ml_list[ii]
                        clf.fit(X_train, y_train)
                        predict = clf.predict(X_test)
                        # Make "classification report" and assign the precision, f-measure, and recall values
                        f_1 = f1_score(y_test, predict, average='macro')
                        pr = precision_score(y_test, predict, average='macro')
                        rc = recall_score(y_test, predict, average='macro')
                        precision.append(float(pr))
                        recall.append(float(rc))
                        f1.append(float(f_1))
                        accuracy.append(clf.score(X_test, y_test))
                        t_time.append(float((time.time() - second)))
        # Print the result of ten repetitions
        print('%-17s %-17s   %-15s %-15s %-15s %-15s %-15s' % (
            j[0:-4], ii, str(round(np.mean(accuracy), 2)), str(round(np.mean(precision), 2)),
            str(round(np.mean(recall), 2)), str(round(np.mean(f1), 2)), str(round(np.mean(t_time), 4))))

        # Save all the values in the file
        with open(result, "a", newline="", encoding="utf-8") as f:
            wrt = csv.writer(f)

            for i in range(0, len(t_time)):
                # Write file name, algorithm name, precision, recall and f-measure in CSV file
                wrt.writerow([j[0:-4], ii, accuracy[i], precision[i], recall[i], f1[i], t_time[i]])

        # Create and save the graphics results
        plt.boxplot(f1)
        plt.title("All Dataset - " + str(ii))
        plt.ylabel('F-measure')
        plt.savefig(folder_name + j[0:-4] + str(ii) + ".pdf", bbox_inches='tight', papertype='a4',
                    orientation='portrait', format='pdf')
        plt.show()

print("Mission accomplished!")
print("Total operation time: = ", time.time() - seconds, "seconds")
```