Activity 1.1 : Introduction to Machine Learning

Objective(s):

This activity aims to introduce how to use the different toolsets in machine learning.

Intended Learning Outcomes (ILOs):

- Demonstrate how to use different toolsets in machine learning.
- Demonstrate how to import, manipulate and analyze data using pandas and numpy.
- Demonstrate how to visualize data in graphs using matplotlib and seaborn

Resources:

- Jupyter Notebook
- Iris_Data.csv

Procedure:

The iris data set will be used for this activity. It is a well-known data set containing iris species and sepal and petal measurements.

Import the libraries and the dataset

In [1]:
```
#import the libraries
import pandas as pd
import numpy as np

# import the dataset
data = pd.read_csv('Dataset/Iris_Data.csv')
#check the content of the dataframe
data.head()
```

Out[1]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Determine the following:

- The number of data points (rows).
- The column names.
- The data types for each column.

In [2]:
```
#the number of datapoints
print('number of rows: ',  data.shape[0],  '\n')

#the column names
print('Column names:')
for x in data.columns.tolist():
    print(x)
print('\n')
#the data types for each column
print('column data types:')
print(data.dtypes)
```

```
number of rows:  150

Column names:
sepal_length
sepal_width
petal_length
petal_width
species


column data types:
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```

Examine the species names and note that they all begin with 'Iris-'. Remove this portion of the name so the species name is shorter.

In [3]:
```
#remove the 'Iris-' portion of the name
data['species'] = data.species.str.replace('Iris-', '')
data.head()
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Determine the following:

- The number of each species present.
- The mean, median, and quantiles and ranges (max-min) for each petal and sepal measurement.

In [4]:
```
#the number of each species present
data.species.value_counts()

# the mean, median and quartiles and ranges
stats_df = data.describe()
stats_df.loc['range'] = stats_df.loc['max'] - stats_df.loc['min']

out_fields = ['mean','25%','50%','75%', 'range']
stats_df = stats_df.loc[out_fields]
stats_df.rename({'50%': 'median'}, inplace=True)
stats_df
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width
mean	5.843333	3.054	3.758667	1.198667
25%	5.100000	2.800	1.600000	0.300000
median	5.800000	3.000	4.350000	1.300000
75%	6.400000	3.300	5.100000	1.800000
range	3.600000	2.400	5.900000	2.400000

Calculate the following for each species in a separate dataframe:

- The mean of each measurement (sepal_length, sepal_width, petal_length, and petal_width).
- The median of each of these measurements.

In [5]:
```
# The mean calculation
data.groupby('species').mean()
```

Out[5]:

species	sepal_length	sepal_width	petal_length	petal_width
setosa	5.006	3.418	1.464	0.244
versicolor	5.936	2.770	4.260	1.326
virginica	6.588	2.974	5.552	2.026

In [6]:
```
# The median calculation
data.groupby('species').median()
```

Out[6]:

species	sepal_length	sepal_width	petal_length	petal_width
setosa	5.0	3.4	1.50	0.2
versicolor	5.9	2.8	4.35	1.3
virginica	6.5	3.0	5.55	2.0

In [7]:
```
from pprint import pprint

agg_dict = {field: ['mean', 'median'] for field in data.columns if field != 'specie
agg_dict['petal_length'] = 'max'
pprint(agg_dict)
data.groupby('species').agg(agg_dict)
```

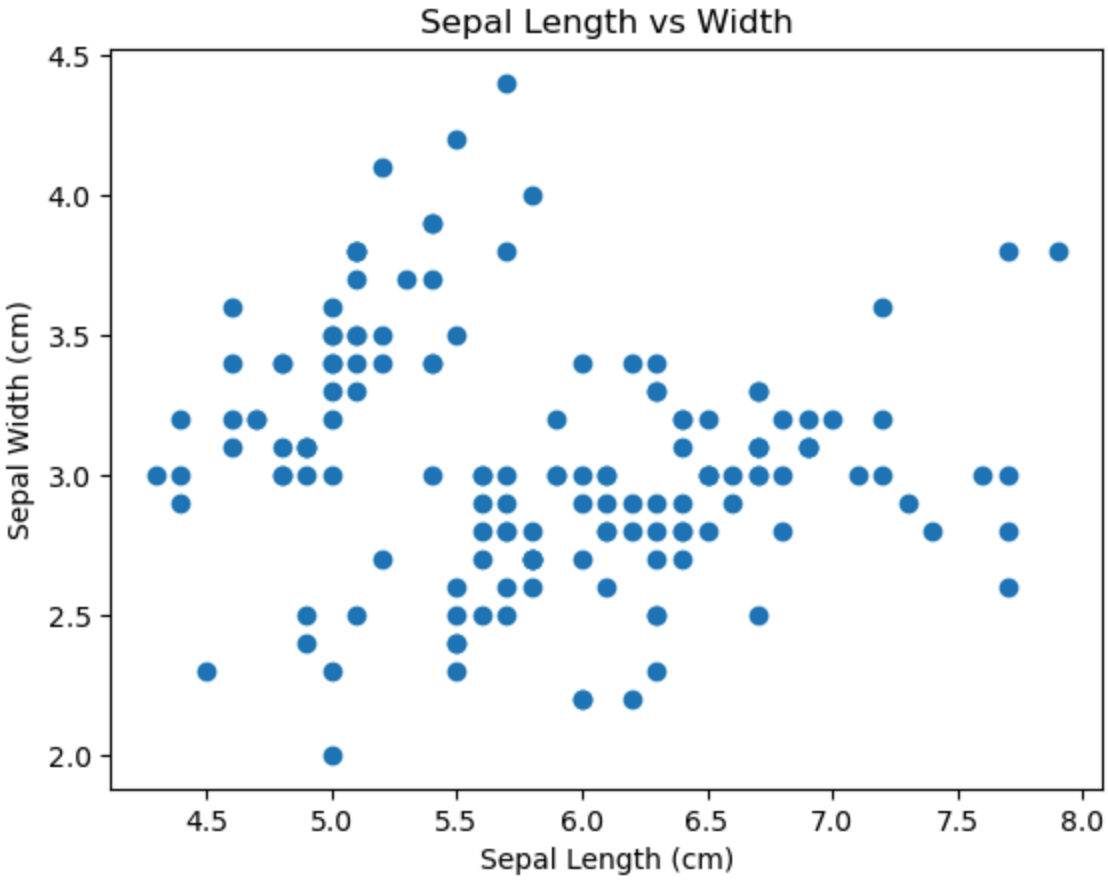
```
{'petal_length': 'max',
 'petal_width': ['mean', 'median'],
 'sepal_length': ['mean', 'median'],
 'sepal_width': ['mean', 'median']}
```

Out[7]:

	sepal_length		sepal_width		petal_length	petal_width	
	mean	median	mean	median	max	mean	median
species							
setosa	5.006	5.0	3.418	3.4	1.9	0.244	0.2
versicolor	5.936	5.9	2.770	2.8	5.1	1.326	1.3
virginica	6.588	6.5	2.974	3.0	6.9	2.026	2.0

Make a scatter plot of `sepal_length` vs `sepal_width` using Matplotlib. Label the axes and give the plot a title.

In [8]:
```
#Scatter plot of sepal_length versus sepal_width using Matplotlib

import matplotlib.pyplot as plt
%matplotlib inline
# A simple scatter plot with Matplotlib
ax = plt.axes()

ax.scatter(data.sepal_length, data.sepal_width)

# Label the axes
ax.set(xlabel='Sepal Length (cm)',
       ylabel='Sepal Width (cm)',
       title='Sepal Length vs Width');
```



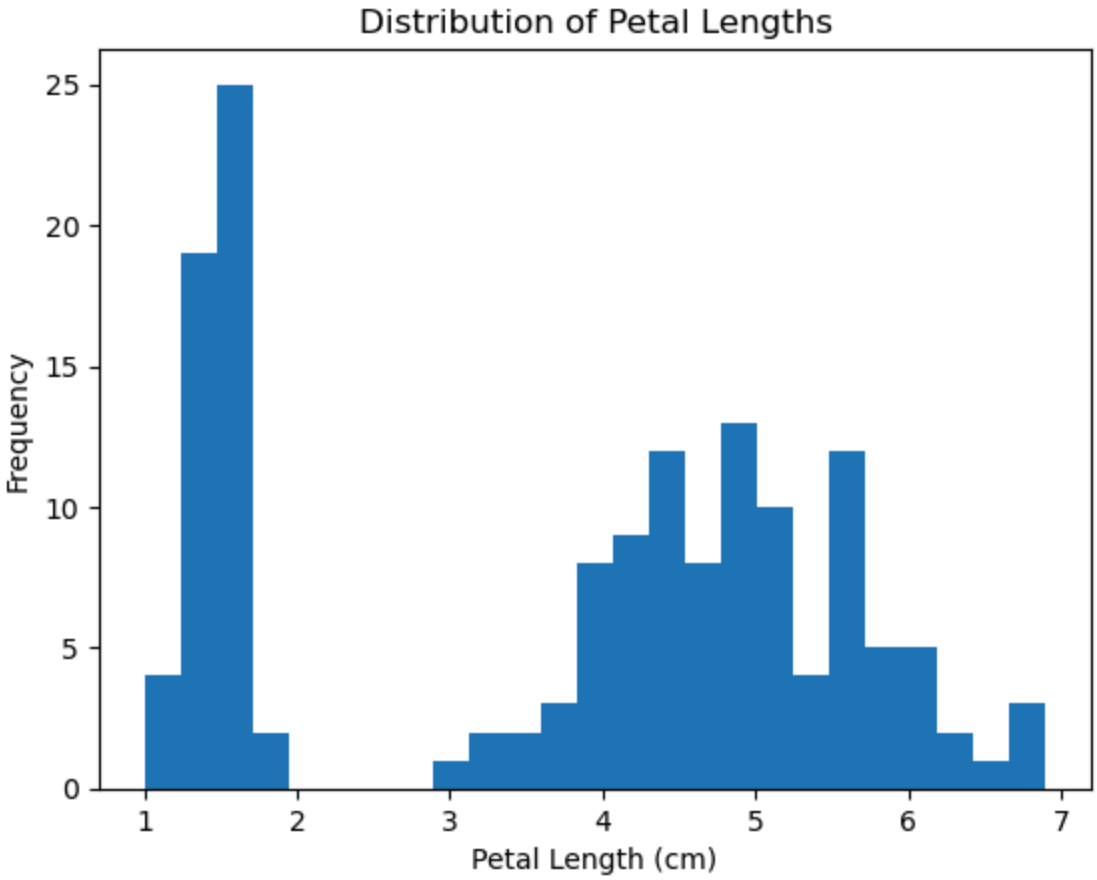
Interpret the result of the scatter plot.

Type your answer here

Make a histogram of any one of the four features. Label axes and title it as appropriate. What is the function of the histogram ?

Type your answer here

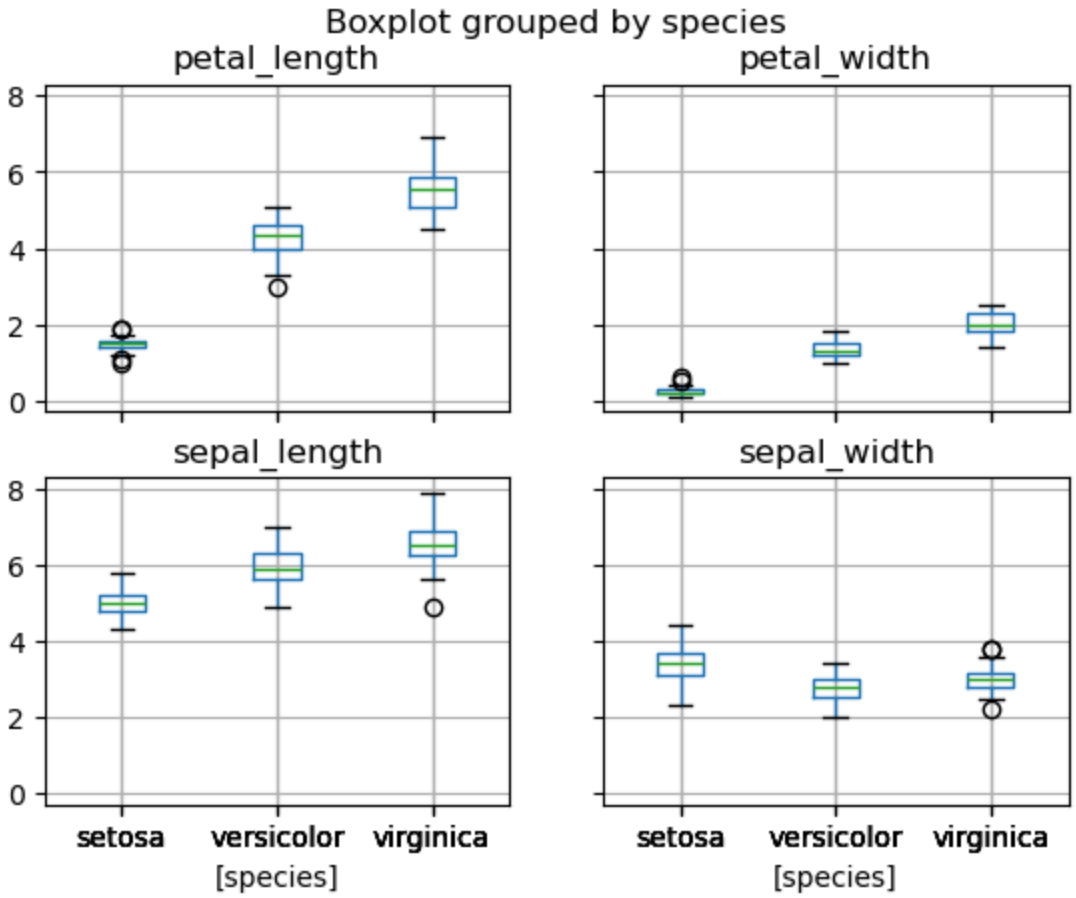
In [9]:
```
#histogram
ax = plt.axes()
ax.hist(data.petal_length, bins=25);

ax.set(xlabel='Petal Length (cm)',
       ylabel='Frequency',
       title='Distribution of Petal Lengths');
```



Make a boxplot of each petal and sepal measurement. What is the function of the boxplot?

type your answer here

In [10]:
```
#boxplot
data.boxplot(by='species');
```



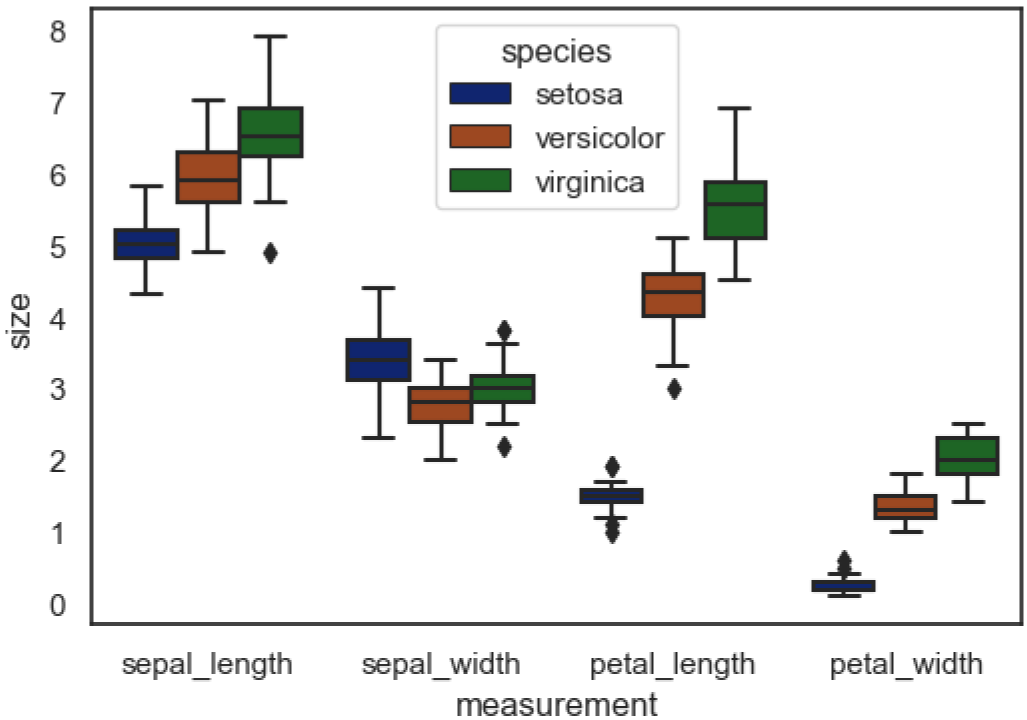
Make a single boxplot where the features are separated in the x-axis and species are colored with different hues.

In [11]:
```
#single boxplot
plot_data = (data
             .set_index('species')
             .stack()
             .to_frame()
             .reset_index()
             .rename(columns={0:'size', 'level_1':'measurement'})
            )

plot_data.head()
```

Out[11]:

	species	measurement	size
0	setosa	sepal_length	5.1
1	setosa	sepal_width	3.5
2	setosa	petal_length	1.4
3	setosa	petal_width	0.2
4	setosa	sepal_length	4.9

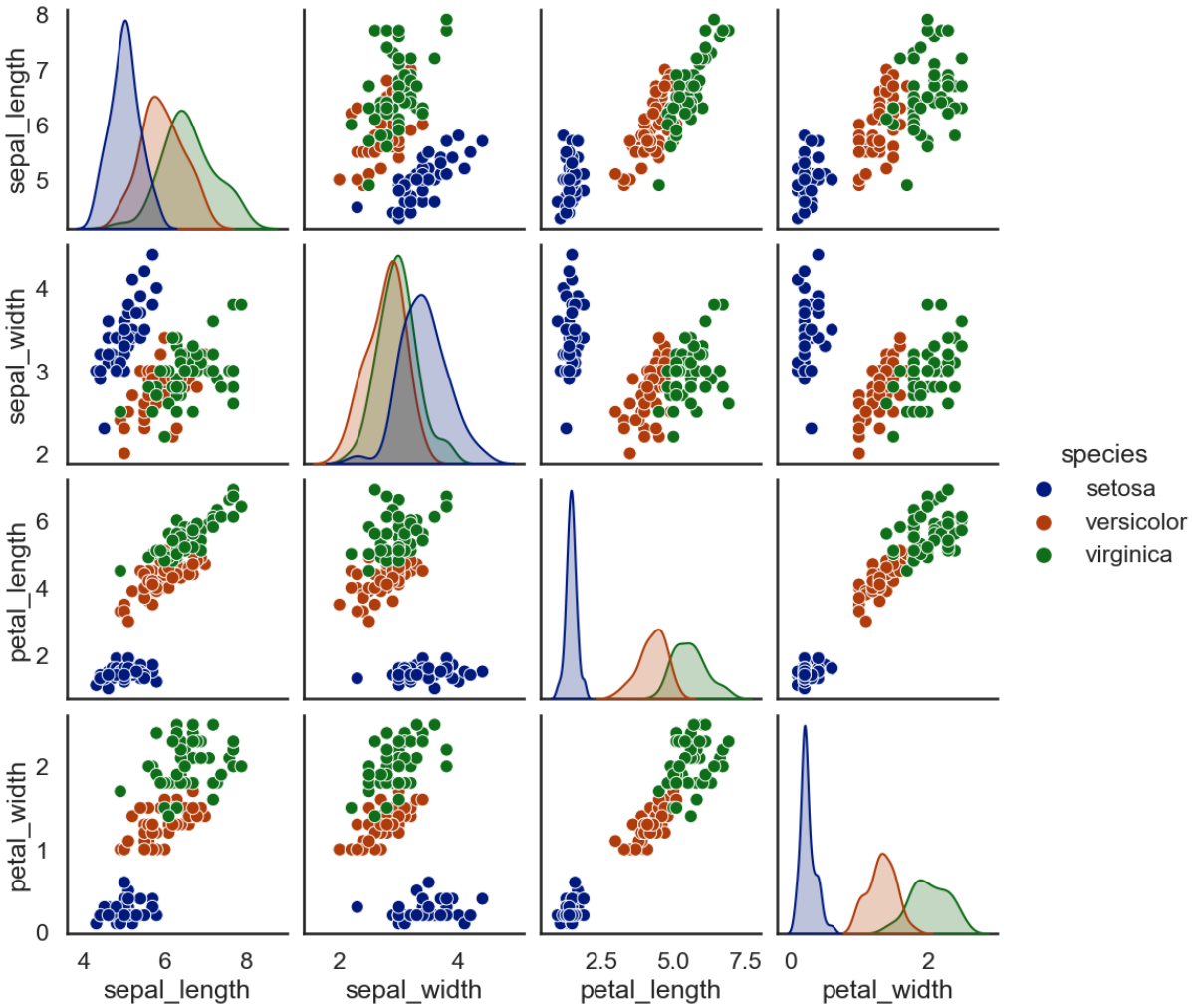
In [12]:
```
import seaborn as sns
sns.set_style('white')
sns.set_context('notebook')
sns.set_palette('dark')

f = plt.figure(figsize=(6,4))
sns.boxplot(x='measurement', y='size',
            hue='species', data=plot_data);
```



Make a pairplot with Seaborn to examine the correlation between each of the measurements.

In [13]:
```
#pair plot
sns.set_context('talk')
sns.pairplot(data, hue='species');
```

```
E:\AnacondaNavigator\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
E:\AnacondaNavigator\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
E:\AnacondaNavigator\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
E:\AnacondaNavigator\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



Interpret the result of correlation using the pairplot

Supplementary Activity:

- Choose your own dataset
- Import the dataset
- Determine the number of datapoints, columns and data types
- Remove unneccesary columns
- Do data cleaning such as removing empty values.
- Perform descriptive statistics such as mean, median and mode
- Compare and analyze your data using data visualization

Supplementary Activity

Dataset: RT-IoT Dataset

Import dataset

In [14]:
```python
# type your code here
import pandas as pd
import numpy as np
from ucimlrepo import fetch_ucirepo

# fetch dataset
rt_iot2022 = fetch_ucirepo(id=942)

# data (as pandas dataframes)
X = rt_iot2022.data.features
y = rt_iot2022.data.targets

# metadata
print(rt_iot2022.metadata)

# variable information
print(rt_iot2022.variables)
```

{'uci_id': 942, 'name': 'RT-IoT2022 ', 'repository_url': 'https://archive.ics.uci.edu/dataset/942/rt-iot2022', 'data_url': 'https://archive.ics.uci.edu/static/public/942/data.csv', 'abstract': 'The RT-IoT2022, a proprietary dataset derived from a real-time IoT infrastructure, is introduced as a comprehensive resource integrating a diverse range of IoT devices and sophisticated network attack methodologies. This dataset encompasses both normal and adversarial network behaviours, providing a general representation of real-world scenarios.\nIncorporating data from IoT devices such as ThingSpeak-LED, Wipro-Bulb, and MQTT-Temp, as well as simulated attack scenarios involving Brute-Force SSH attacks, DDoS attacks using Hping and Slowloris, and Nmap patterns, RT-IoT2022 offers a detailed perspective on the complex nature of network traffic. The bidirectional attributes of network traffic are meticulously captured using the Zeek network monitoring tool and the Flowmeter plugin. Researchers can leverage the RT-IoT2022 dataset to advance the capabilities of Intrusion Detection Systems (IDS), fostering the development of robust and adaptive security solutions for real-time IoT networks. ', 'area': 'Engineering', 'tasks': ['Classification', 'Regression', 'Clustering'], 'characteristics': ['Tabular', 'Sequential', 'Multivariate'], 'num_instances': 123117, 'num_features': 83, 'feature_types': ['Real', 'Categorical'], 'demographics': [], 'target_col': ['Attack_type'], 'index_col': ['id'], 'has_missing_values': 'no', 'missing_values_symbol': None, 'year_of_dataset_creation': 2023, 'last_updated': 'Fri Mar 08 2024', 'dataset_doi': '10.24432/C5P338', 'creators': ['B. S.', 'Rohini Nagapadma'], 'intro_paper': {'title': 'Quantized autoencoder (QAE) intrusion detection system for anomaly detection in resource-constrained IoT devices using RT-IoT2022 dataset', 'authors': 'B. S. Sharmila, Rohini Nagapadma', 'published_in': 'Cybersecurity', 'year': 2023, 'url': 'https://www.semanticscholar.org/paper/753f6ede01b4acaa325e302c38f1e0c1ade74f5b', 'doi': None}, 'additional_info': {'summary': None, 'purpose': None, 'funded_by': None, 'instances_represent': None, 'recommended_data_splits': None, 'sensitive_data': None, 'preprocessing_description': None, 'variable_info': 'Column Details:\nid.orig_p\nid.resp_p\nproto\nservice\nflow_duration\nfwd_pkts_tot\nbwd_pkts_tot\nfwd_data_pkts_tot\nbwd_data_pkts_tot\nfwd_pkts_per_sec\nbwd_pkts_per_sec\nflow_pkts_per_sec\ndown_up_ratio\nfwd_header_size_tot\nfwd_header_size_min\nfwd_header_size_max\nbwd_header_size_tot\nbwd_header_size_min\nbwd_header_size_max\nflow_FIN_flag_count\nflow_SYN_flag_count\nflow_RST_flag_count\nfwd_PSH_flag_count\nbwd_PSH_flag_count\nflow_ACK_flag_count\nfwd_URG_flag_count\nbwd_URG_flag_count\nflow_CWR_flag_count\nflow_ECE_flag_count\nfwd_pkts_payload.min\nfwd_pkts_payload.max\nfwd_pkts_payload.tot\nfwd_pkts_payload.avg\nfwd_pkts_payload.std\nbwd_pkts_payload.min\nbwd_pkts_payload.max\nbwd_pkts_payload.tot\nbwd_pkts_payload.avg\nbwd_pkts_payload.std\nflow_pkts_payload.min\nflow_pkts_payload.max\nflow_pkts_payload.tot\nflow_pkts_payload.avg\nflow_pkts_payload.std\nfwd_iat.min\nfwd_iat.max\nfwd_iat.tot\nfwd_iat.avg\nfwd_iat.std\nbwd_iat.min\nbwd_iat.max\nbwd_iat.tot\nbwd_iat.avg\nbwd_iat.std\nflow_iat.min\nflow_iat.max\nflow_iat.tot\nflow_iat.avg\nflow_iat.std\npayload_bytes_per_second\nfwd_subflow_pkts\nbwd_subflow_pkts\nfwd_subflow_bytes\nbwd_subflow_bytes\nfwd_bulk_bytes\nbwd_bulk_bytes\nfwd_bulk_packets\nbwd_bulk_packets\nfwd_bulk_rate\nbwd_bulk_rate\nactive.min\nactive.max\nactive.tot\nactive.avg\nactive.std\nidle.min\nidle.max\nidle.tot\nidle.avg\nidle.std\nfwd_init_window_size\nbwd_init_window_size\nfwd_last_window_size\nAttack_type', 'citation': None}}

	name	role	type	demographic	description	units	\
0	id.orig_p	Feature	Integer	None	None	None	
1	id.resp_p	Feature	Integer	None	None	None	
2	proto	Feature	Categorical	None	None	None	
3	service	Feature	Continuous	None	None	None	
4	flow_duration	Feature	Continuous	None	None	None	
..	...	...	...	...	...	...	
80	fwd_init_window_size	Feature	Integer	None	None	None	
81	bwd_init_window_size	Feature	Integer	None	None	None	
82	fwd_last_window_size	Feature	Integer	None	None	None	
83	Attack_type	Target	Categorical	None	None	None	

```
84                  id       ID      Integer        None        None  None

   missing_values
0              no
1              no
2              no
3              no
4              no
..            ...
80             no
81             no
82             no
83             no
84             no

[85 rows x 7 columns]
```

In [15]:
```
# joining the feature and target dataset into one
rtdata = pd.concat([X,y],axis=1)
```

In [16]:
```
rtdata.head()
```

Out[16]:

	id.orig_p	id.resp_p	proto	service	flow_duration	fwd_pkts_tot	bwd_pkts_tot	fwd_dat
0	38667	1883	tcp	mqtt	32.011598	9	5	
1	51143	1883	tcp	mqtt	31.883584	9	5	
2	44761	1883	tcp	mqtt	32.124053	9	5	
3	60893	1883	tcp	mqtt	31.961063	9	5	
4	51087	1883	tcp	mqtt	31.902362	9	5	

5 rows × 84 columns

In [17]:
```
# checking if there are null values
rtdata.isna().sum()
```

Out[17]:
```
id.orig_p               0
id.resp_p               0
proto                   0
service                 0
flow_duration           0
                       ..
idle.std                0
fwd_init_window_size    0
bwd_init_window_size    0
fwd_last_window_size    0
Attack_type             0
Length: 84, dtype: int64
```

In [18]:
```
# checking the all of the attack types that we can classify
rtdata['Attack_type'].unique()
```

Out[18]:
```
array(['MQTT_Publish', 'Thing_Speak', 'Wipro_bulb', 'ARP_poisioning',
       'DDOS_Slowloris', 'DOS_SYN_Hping', 'Metasploit_Brute_Force_SSH',
       'NMAP_FIN_SCAN', 'NMAP_OS_DETECTION', 'NMAP_TCP_scan',
       'NMAP_UDP_SCAN', 'NMAP_XMAS_TREE_SCAN'], dtype=object)
```

Determine the number of datapoints, columns and data types

In [19]:
```
# determine the number of data points
print('number of datapoints:', len(rtdata),'\n')

#determine the number of columns
print('columns in RT-IoT Dataset:')
for x in rtdata.columns.tolist():
    if x == 'Attack_type':
        print(x, end='.\n')
    else:
        print(x, end=', ')


#determine the number of datatypes
print('\ndatatype of rt-IoT columns:\n', rtdata.dtypes)
```

```
number of datapoints: 123117

columns in RT-IoT Dataset:
id.orig_p, id.resp_p, proto, service, flow_duration, fwd_pkts_tot, bwd_pkts_tot, fwd
_data_pkts_tot, bwd_data_pkts_tot, fwd_pkts_per_sec, bwd_pkts_per_sec, flow_pkts_per
_sec, down_up_ratio, fwd_header_size_tot, fwd_header_size_min, fwd_header_size_max,
bwd_header_size_tot, bwd_header_size_min, bwd_header_size_max, flow_FIN_flag_count,
flow_SYN_flag_count, flow_RST_flag_count, fwd_PSH_flag_count, bwd_PSH_flag_count, fl
ow_ACK_flag_count, fwd_URG_flag_count, bwd_URG_flag_count, flow_CWR_flag_count, flow
_ECE_flag_count, fwd_pkts_payload.min, fwd_pkts_payload.max, fwd_pkts_payload.tot, f
wd_pkts_payload.avg, fwd_pkts_payload.std, bwd_pkts_payload.min, bwd_pkts_payload.ma
x, bwd_pkts_payload.tot, bwd_pkts_payload.avg, bwd_pkts_payload.std, flow_pkts_paylo
ad.min, flow_pkts_payload.max, flow_pkts_payload.tot, flow_pkts_payload.avg, flow_pk
ts_payload.std, fwd_iat.min, fwd_iat.max, fwd_iat.tot, fwd_iat.avg, fwd_iat.std, bwd
_iat.min, bwd_iat.max, bwd_iat.tot, bwd_iat.avg, bwd_iat.std, flow_iat.min, flow_ia
t.max, flow_iat.tot, flow_iat.avg, flow_iat.std, payload_bytes_per_second, fwd_subfl
ow_pkts, bwd_subflow_pkts, fwd_subflow_bytes, bwd_subflow_bytes, fwd_bulk_bytes, bwd
_bulk_bytes, fwd_bulk_packets, bwd_bulk_packets, fwd_bulk_rate, bwd_bulk_rate, activ
e.min, active.max, active.tot, active.avg, active.std, idle.min, idle.max, idle.tot,
idle.avg, idle.std, fwd_init_window_size, bwd_init_window_size, fwd_last_window_siz
e, Attack_type.

datatype of rt-IoT columns:
 id.orig_p                int64
id.resp_p                 int64
proto                    object
service                  object
flow_duration           float64
                         ...
idle.std                float64
fwd_init_window_size      int64
bwd_init_window_size      int64
fwd_last_window_size      int64
Attack_type              object
Length: 84, dtype: object
```

Remove Unnecessary Column

In [20]:
```
# no unnecessary column
rtdata.head()
```

Out[20]:

	id.orig_p	id.resp_p	proto	service	flow_duration	fwd_pkts_tot	bwd_pkts_tot	fwd_dat
0	38667	1883	tcp	mqtt	32.011598	9	5	
1	51143	1883	tcp	mqtt	31.883584	9	5	
2	44761	1883	tcp	mqtt	32.124053	9	5	
3	60893	1883	tcp	mqtt	31.961063	9	5	
4	51087	1883	tcp	mqtt	31.902362	9	5	

5 rows × 84 columns

In [21]:
```
rtdata['proto'].value_counts()
```

Out[21]:
```
proto
tcp     110427
udp      12633
icmp        57
Name: count, dtype: int64
```

In [22]:
```
rtdata['service'].value_counts()
```

Out[22]:
```
service
-         102861
dns         9753
mqtt        4132
http        3464
ssl         2663
ntp          121
dhcp          50
irc           43
ssh           28
radius         2
Name: count, dtype: int64
```

In [23]:
```
ssh = rtdata[rtdata['service'] == 'dns']
```

In [24]:
```
ssh
```

Out[24]:

	id.orig_p	id.resp_p	proto	service	flow_duration	fwd_pkts_tot	bwd_pkts_tot	fw
4147	37818	53	udp	dns	0.022455	2	2	
4149	56221	53	udp	dns	0.032374	2	2	
4151	57780	53	udp	dns	0.049626	2	2	
4153	50716	53	udp	dns	0.030501	2	2	
4156	50389	53	udp	dns	0.032210	2	2	
...	...	...	...	...	...	...	...	
120943	64181	5353	udp	dns	0.000000	1	0	
120955	64185	5353	udp	dns	0.000000	1	0	
121108	52980	53	udp	dns	0.037393	2	2	
121109	41729	53	udp	dns	0.001469	1	1	
121110	42501	53	udp	dns	0.036275	1	1	

9753 rows × 84 columns

In [25]:
```
# removing the non integer datatype columns
rtdata.drop(columns=['proto','service'],inplace = True)
```

In [26]:
```
# checking if we removed the unnecessary columns
rtdata.head()
```

Out[26]:

	id.orig_p	id.resp_p	flow_duration	fwd_pkts_tot	bwd_pkts_tot	fwd_data_pkts_tot	bwd_
0	38667	1883	32.011598	9	5	3	
1	51143	1883	31.883584	9	5	3	
2	44761	1883	32.124053	9	5	3	
3	60893	1883	31.961063	9	5	3	
4	51087	1883	31.902362	9	5	3	

5 rows × 82 columns

Do data cleaning such as removing empty values.

In [27]:
```
# checking for null values or - in the data
for x in rtdata.columns.tolist():
    print(x,': ', len(rtdata[rtdata[x] == '-']))
```

```
id.orig_p :  0
id.resp_p :  0
flow_duration :  0
fwd_pkts_tot :  0
bwd_pkts_tot :  0
fwd_data_pkts_tot :  0
bwd_data_pkts_tot :  0
fwd_pkts_per_sec :  0
bwd_pkts_per_sec :  0
flow_pkts_per_sec :  0
down_up_ratio :  0
fwd_header_size_tot :  0
fwd_header_size_min :  0
fwd_header_size_max :  0
bwd_header_size_tot :  0
bwd_header_size_min :  0
bwd_header_size_max :  0
flow_FIN_flag_count :  0
flow_SYN_flag_count :  0
flow_RST_flag_count :  0
fwd_PSH_flag_count :  0
bwd_PSH_flag_count :  0
flow_ACK_flag_count :  0
fwd_URG_flag_count :  0
bwd_URG_flag_count :  0
flow_CWR_flag_count :  0
flow_ECE_flag_count :  0
fwd_pkts_payload.min :  0
fwd_pkts_payload.max :  0
fwd_pkts_payload.tot :  0
fwd_pkts_payload.avg :  0
fwd_pkts_payload.std :  0
bwd_pkts_payload.min :  0
bwd_pkts_payload.max :  0
bwd_pkts_payload.tot :  0
bwd_pkts_payload.avg :  0
bwd_pkts_payload.std :  0
flow_pkts_payload.min :  0
flow_pkts_payload.max :  0
flow_pkts_payload.tot :  0
flow_pkts_payload.avg :  0
flow_pkts_payload.std :  0
fwd_iat.min :  0
fwd_iat.max :  0
fwd_iat.tot :  0
fwd_iat.avg :  0
fwd_iat.std :  0
bwd_iat.min :  0
bwd_iat.max :  0
bwd_iat.tot :  0
bwd_iat.avg :  0
bwd_iat.std :  0
flow_iat.min :  0
flow_iat.max :  0
flow_iat.tot :  0
flow_iat.avg :  0
```

```
flow_iat.std :  0
payload_bytes_per_second :  0
fwd_subflow_pkts :  0
bwd_subflow_pkts :  0
fwd_subflow_bytes :  0
bwd_subflow_bytes :  0
fwd_bulk_bytes :  0
bwd_bulk_bytes :  0
fwd_bulk_packets :  0
bwd_bulk_packets :  0
fwd_bulk_rate :  0
bwd_bulk_rate :  0
active.min :  0
active.max :  0
active.tot :  0
active.avg :  0
active.std :  0
idle.min :  0
idle.max :  0
idle.tot :  0
idle.avg :  0
idle.std :  0
fwd_init_window_size :  0
bwd_init_window_size :  0
fwd_last_window_size :  0
Attack_type :  0
```

In [28]:
```
# checking for null value count
rtdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 123117 entries, 0 to 123116
Data columns (total 82 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   id.orig_p                 123117 non-null  int64  
 1   id.resp_p                 123117 non-null  int64  
 2   flow_duration             123117 non-null  float64
 3   fwd_pkts_tot              123117 non-null  int64  
 4   bwd_pkts_tot              123117 non-null  int64  
 5   fwd_data_pkts_tot         123117 non-null  int64  
 6   bwd_data_pkts_tot         123117 non-null  int64  
 7   fwd_pkts_per_sec          123117 non-null  float64
 8   bwd_pkts_per_sec          123117 non-null  float64
 9   flow_pkts_per_sec         123117 non-null  float64
 10  down_up_ratio             123117 non-null  float64
 11  fwd_header_size_tot       123117 non-null  int64  
 12  fwd_header_size_min       123117 non-null  int64  
 13  fwd_header_size_max       123117 non-null  int64  
 14  bwd_header_size_tot       123117 non-null  int64  
 15  bwd_header_size_min       123117 non-null  int64  
 16  bwd_header_size_max       123117 non-null  int64  
 17  flow_FIN_flag_count       123117 non-null  int64  
 18  flow_SYN_flag_count       123117 non-null  int64  
 19  flow_RST_flag_count       123117 non-null  int64  
 20  fwd_PSH_flag_count        123117 non-null  int64  
 21  bwd_PSH_flag_count        123117 non-null  int64  
 22  flow_ACK_flag_count       123117 non-null  int64  
 23  fwd_URG_flag_count        123117 non-null  int64  
 24  bwd_URG_flag_count        123117 non-null  int64  
 25  flow_CWR_flag_count       123117 non-null  int64  
 26  flow_ECE_flag_count       123117 non-null  int64  
 27  fwd_pkts_payload.min      123117 non-null  int64  
 28  fwd_pkts_payload.max      123117 non-null  int64  
 29  fwd_pkts_payload.tot      123117 non-null  int64  
 30  fwd_pkts_payload.avg      123117 non-null  float64
 31  fwd_pkts_payload.std      123117 non-null  float64
 32  bwd_pkts_payload.min      123117 non-null  int64  
 33  bwd_pkts_payload.max      123117 non-null  int64  
 34  bwd_pkts_payload.tot      123117 non-null  int64  
 35  bwd_pkts_payload.avg      123117 non-null  float64
 36  bwd_pkts_payload.std      123117 non-null  float64
 37  flow_pkts_payload.min     123117 non-null  int64  
 38  flow_pkts_payload.max     123117 non-null  int64  
 39  flow_pkts_payload.tot     123117 non-null  int64  
 40  flow_pkts_payload.avg     123117 non-null  float64
 41  flow_pkts_payload.std     123117 non-null  float64
 42  fwd_iat.min               123117 non-null  float64
 43  fwd_iat.max               123117 non-null  float64
 44  fwd_iat.tot               123117 non-null  float64
 45  fwd_iat.avg               123117 non-null  float64
 46  fwd_iat.std               123117 non-null  float64
 47  bwd_iat.min               123117 non-null  float64
 48  bwd_iat.max               123117 non-null  float64
 49  bwd_iat.tot               123117 non-null  float64
 50  bwd_iat.avg               123117 non-null  float64
```

```
 51  bwd_iat.std               123117 non-null  float64
 52  flow_iat.min              123117 non-null  float64
 53  flow_iat.max              123117 non-null  float64
 54  flow_iat.tot              123117 non-null  float64
 55  flow_iat.avg              123117 non-null  float64
 56  flow_iat.std              123117 non-null  float64
 57  payload_bytes_per_second  123117 non-null  float64
 58  fwd_subflow_pkts          123117 non-null  float64
 59  bwd_subflow_pkts          123117 non-null  float64
 60  fwd_subflow_bytes         123117 non-null  float64
 61  bwd_subflow_bytes         123117 non-null  float64
 62  fwd_bulk_bytes            123117 non-null  float64
 63  bwd_bulk_bytes            123117 non-null  float64
 64  fwd_bulk_packets          123117 non-null  float64
 65  bwd_bulk_packets          123117 non-null  float64
 66  fwd_bulk_rate             123117 non-null  float64
 67  bwd_bulk_rate             123117 non-null  float64
 68  active.min                123117 non-null  float64
 69  active.max                123117 non-null  float64
 70  active.tot                123117 non-null  float64
 71  active.avg                123117 non-null  float64
 72  active.std                123117 non-null  float64
 73  idle.min                  123117 non-null  float64
 74  idle.max                  123117 non-null  float64
 75  idle.tot                  123117 non-null  float64
 76  idle.avg                  123117 non-null  float64
 77  idle.std                  123117 non-null  float64
 78  fwd_init_window_size      123117 non-null  int64
 79  bwd_init_window_size      123117 non-null  int64
 80  fwd_last_window_size      123117 non-null  int64
 81  Attack_type               123117 non-null  object
dtypes: float64(47), int64(34), object(1)
memory usage: 77.0+ MB
```

Perform descriptive statistics such as mean, median and mode

In [29]:
```
# getting the stat summary of my dataset
rtdata.describe()
```

Out[29]:

	id.orig_p	id.resp_p	flow_duration	fwd_pkts_tot	bwd_pkts_tot	fwd_da
count	123117.000000	123117.000000	123117.000000	123117.000000	123117.000000	123
mean	34639.258738	1014.305092	3.809566	2.268826	1.909509	
std	19070.620354	5256.371994	130.005408	22.336565	33.018311	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	17702.000000	21.000000	0.000001	1.000000	1.000000	
50%	37221.000000	21.000000	0.000004	1.000000	1.000000	
75%	50971.000000	21.000000	0.000005	1.000000	1.000000	
max	65535.000000	65389.000000	21728.335580	4345.000000	10112.000000	4

8 rows × 81 columns

In [30]:
```
# gets the mode
mode = rtdata.mode().drop(index=1,columns='Attack_type')
# renames the index as mode
mode.rename(index={0:'mode'},inplace=True)
```

In [31]:
```
# putting the mode in the stat describe of rtdata
rtstats = pd.concat([rtdata.describe(), mode])
# the median is originally named 50% so we will rename it median
rtstats.rename(index={'50%':'median'},inplace = True)
#filtering the describe dedicated to mean, median, mode
rtstats.loc[['mean','median','mode']]
```

Out[31]:

	id.orig_p	id.resp_p	flow_duration	fwd_pkts_tot	bwd_pkts_tot	fwd_data_p
mean	34639.258738	1014.305092	3.809566	2.268826	1.909509	1.
median	37221.000000	21.000000	0.000004	1.000000	1.000000	1.
mode	36242.000000	21.000000	0.000004	1.000000	1.000000	1.

3 rows × 81 columns

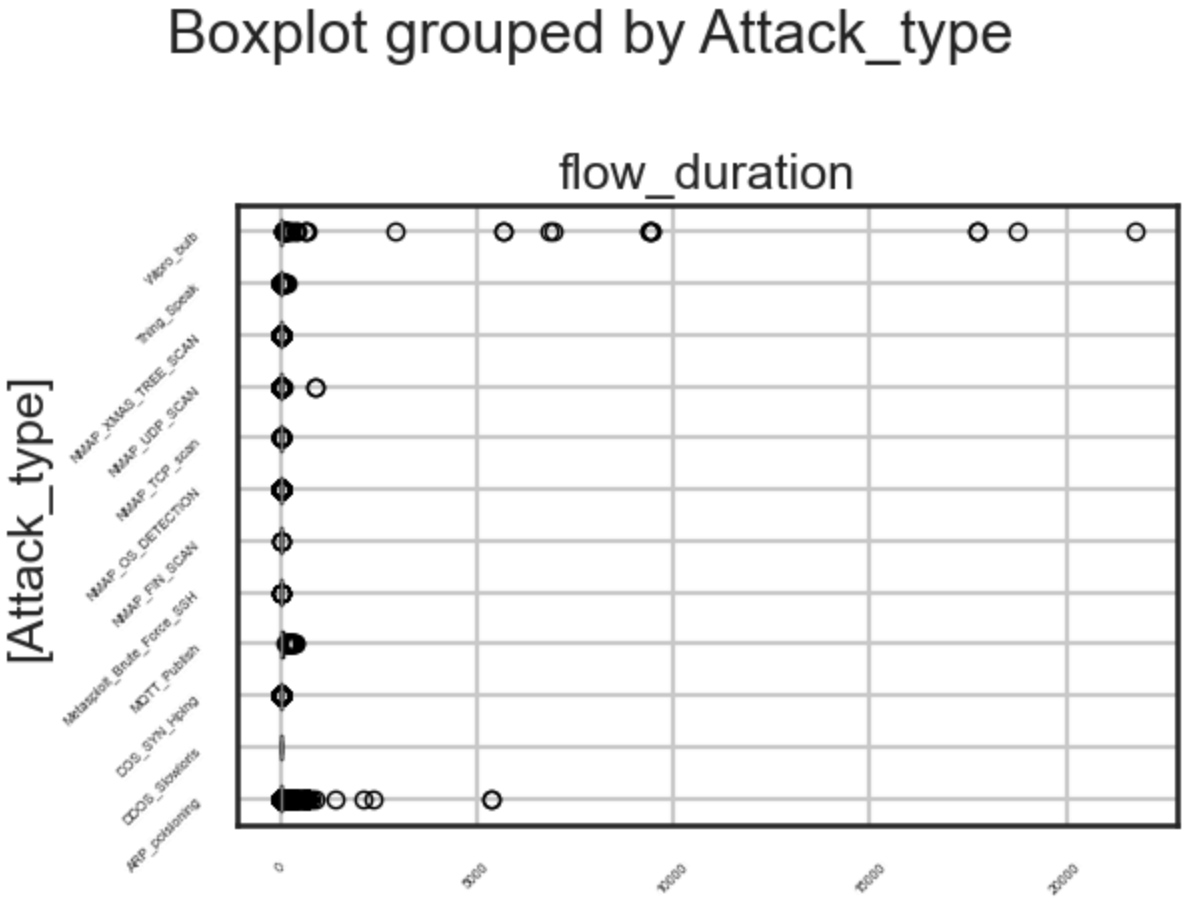
Compare and analyze your data using data visualization

In [65]:
```
print('columns in RT-IoT Dataset:')
for x in rtdata.columns.tolist():
    if x == 'Attack_type':
        print(x, end='.\n')
    else:
        print(x, end=', ')
```

```
columns in RT-IoT Dataset:
id.orig_p, id.resp_p, flow_duration, fwd_pkts_tot, bwd_pkts_tot, fwd_data_pkts_tot,
bwd_data_pkts_tot, fwd_pkts_per_sec, bwd_pkts_per_sec, flow_pkts_per_sec, down_up_ra
tio, fwd_header_size_tot, fwd_header_size_min, fwd_header_size_max, bwd_header_size_
tot, bwd_header_size_min, bwd_header_size_max, flow_FIN_flag_count, flow_SYN_flag_co
unt, flow_RST_flag_count, fwd_PSH_flag_count, bwd_PSH_flag_count, flow_ACK_flag_coun
t, fwd_URG_flag_count, bwd_URG_flag_count, flow_CWR_flag_count, flow_ECE_flag_count,
fwd_pkts_payload.min, fwd_pkts_payload.max, fwd_pkts_payload.tot, fwd_pkts_payload.a
vg, fwd_pkts_payload.std, bwd_pkts_payload.min, bwd_pkts_payload.max, bwd_pkts_paylo
ad.tot, bwd_pkts_payload.avg, bwd_pkts_payload.std, flow_pkts_payload.min, flow_pkts
_payload.max, flow_pkts_payload.tot, flow_pkts_payload.avg, flow_pkts_payload.std, f
wd_iat.min, fwd_iat.max, fwd_iat.tot, fwd_iat.avg, fwd_iat.std, bwd_iat.min, bwd_ia
t.max, bwd_iat.tot, bwd_iat.avg, bwd_iat.std, flow_iat.min, flow_iat.max, flow_iat.t
ot, flow_iat.avg, flow_iat.std, payload_bytes_per_second, fwd_subflow_pkts, bwd_subf
low_pkts, fwd_subflow_bytes, bwd_subflow_bytes, fwd_bulk_bytes, bwd_bulk_bytes, fwd_
bulk_packets, bwd_bulk_packets, fwd_bulk_rate, bwd_bulk_rate, active.min, active.ma
x, active.tot, active.avg, active.std, idle.min, idle.max, idle.tot, idle.avg, idle.
std, fwd_init_window_size, bwd_init_window_size, fwd_last_window_size, Attack_type.
```

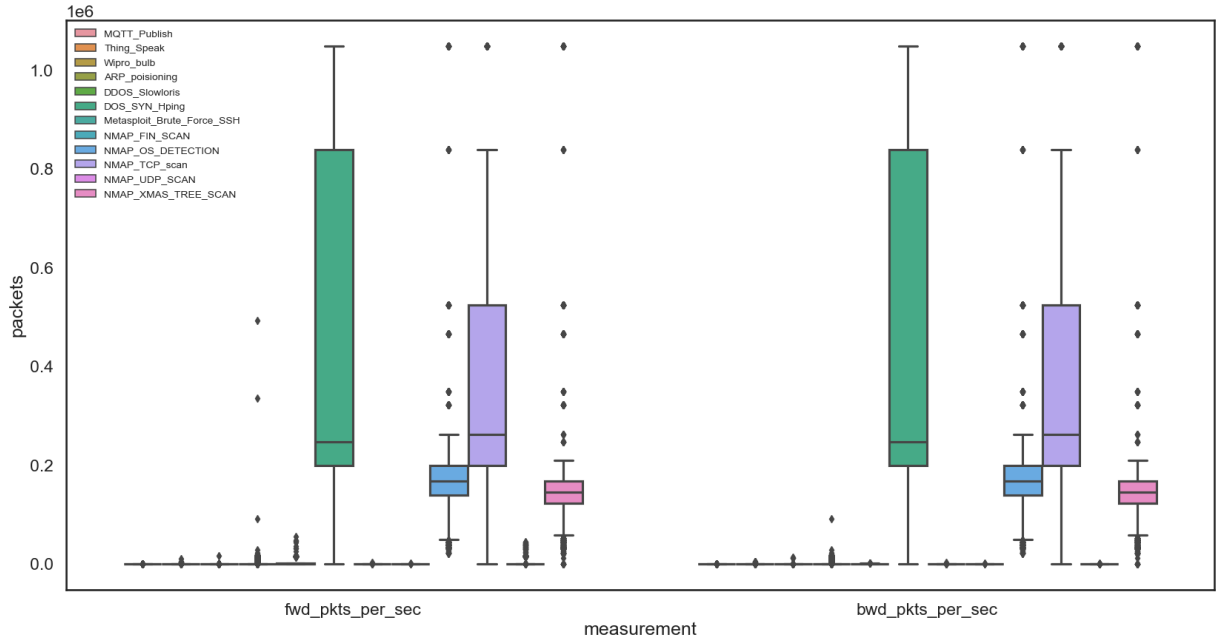
In [64]:
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.figure().set_figwidth(50)
flow_dur = rtdata[['flow_duration','Attack_type']]
flow_dur.boxplot(by='Attack_type',vert=False)
plt.xticks(fontsize=5,rotation=45)
plt.yticks(fontsize=5,rotation=45)
plt.tight_layout()
```

```
<Figure size 5000x480 with 0 Axes>
```



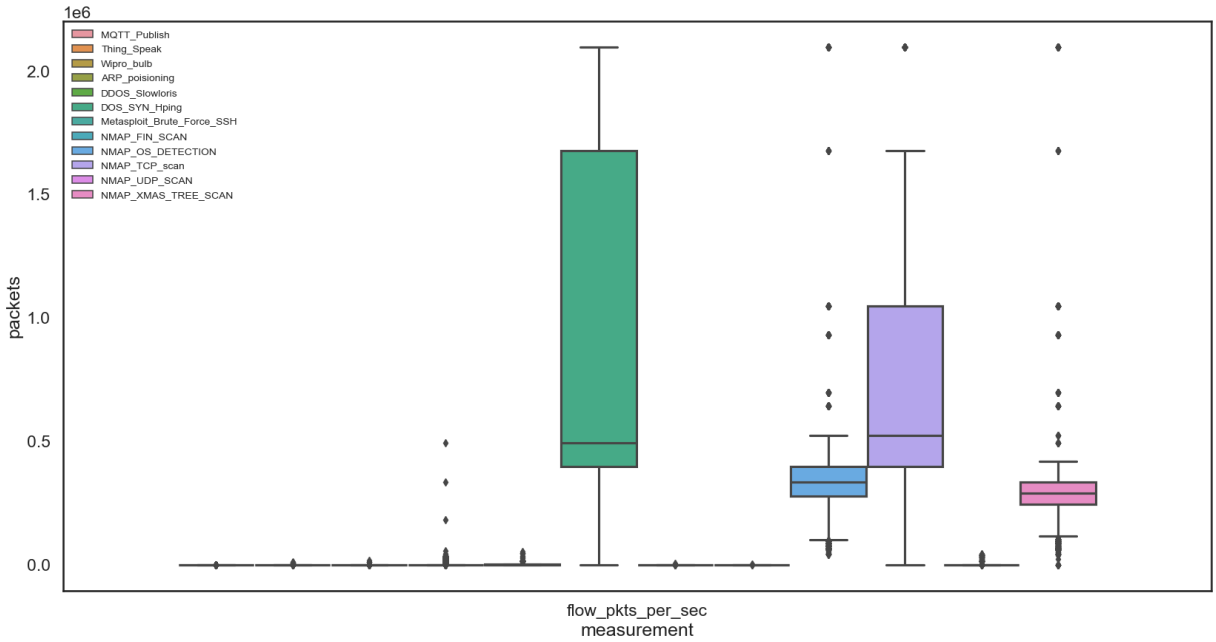
In [89]:
```
import seaborn as sns
pkts_persec = rtdata[['fwd_pkts_per_sec','bwd_pkts_per_sec','Attack_type']]
pkts_persecp = (pkts_persec
                .set_index('Attack_type')
                .stack()
                .to_frame()
                .reset_index()
                .rename(columns={0:'packets','level_1':'measurement'})
                 )
plt.figure(figsize=(20, 10))
pkts_plot = sns.boxplot(x='measurement', y='packets',
                        hue='Attack_type', data=pkts_persecp)
pkts_plot.legend(frameon=False, fontsize=10)
```

Out[89]: <matplotlib.legend.Legend at 0x1e3f0c91e90>



In [112…
```
import seaborn as sns
pkts_persec = rtdata[['flow_pkts_per_sec','Attack_type']]
pkts_persecp = (pkts_persec
                .set_index('Attack_type')
                .stack()
                .to_frame()
                .reset_index()
                .rename(columns={0:'packets','level_1':'measurement'})
                 )
plt.figure(figsize=(20, 10))
pkts_plot = sns.boxplot(x='measurement', y='packets',
                        hue='Attack_type', data=pkts_persecp)
pkts_plot.legend(frameon=False, fontsize=10)
```

Out[112… <matplotlib.legend.Legend at 0x1e389912450>



Conclusion:

In this hands-on activity we reviewed some of our modules in from our data scie 1 and did it in this activity, those modules are importing data from apis, data cleaning, manipulation, and aggregation in pandas, and data visualization using matplotlib and seaborn. I concluded that using data visualizations we can draw insights by just merely looking at the graph(for example, in the flow packets of every attack type in the network, we can draw an insight that when an attack is a DDOS the flow of packets increases compared to the normal attacks),and also using data manipulation can help us adjust the information we need to give out using the data visualizations that we will use.