

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-ТЕЛЕКОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**  
Навчально-науковий інститут захисту інформації  
Кафедра інформаційної та кібернетичної безпеки

**Організація проведення наукових досліджень**

**ЗВІТ**

за результатами виконання лабораторної роботи на тему:

**“Проведення наукових досліджень в кібербезпеці”**

**Варіант 31**

Виконав: Терно  
Ярослав Анатолійович,  
студент групи БСДМ-52

**Київ, 2024**

**Метою** необхідно виконати комплексну лабораторну роботу за варіантом у вигляді файлу (розширення .ipynb) та розмістити у системі Moodle. Для цього необхідно провести наступні етапи машинного навчання:

проаналізувати вхідний набір даних (Exploratory data analysis, EDA);  
підготувати дані до обробки (Data Preprocessing);  
застосувати алгоритм машинного навчання (Modelling);  
оцінити оцінювання продуктивності моделей машинного навчання (Evaluation);  
зробити висновки про доцільність застосування алгоритму машинного навчання для виявлення та реагування на шкідливі процеси в інформаційній системі.

### **Завдання 1.**

Комплексну лабораторну роботу рекомендується виконати в середовищі Jupyter Notebook (<https://jupyter.org/>) або в середовищі Google Colaboratory (<https://colab.google/>).

### **Завдання 2.**

Пропонуються наступні набори даних для застосування алгоритмів машинного навчання (див. варіант завдання нижче):

1. Wednesday-28-02-2018\_TrafficForML\_CICFlowMeter.csv

Детальна інформація про набори даних розміщено за наступним посиланням: <https://www.unb.ca/cic/datasets/ids-2018.html>

Також набори даних є доступними за наступним посиланням: <https://www.kaggle.com/datasets/solarmainframe/ids-intrusion-csv>

### **Завдання 3.**

Пропонуються наступні алгоритми машинного навчання (див. варіант завдання нижче):

для розв'язання задачі класифікації:

7. метод k найближчих сусідів (k-Nearest Neighbours, kNN);

Детальна інформація про відповідні інструменти машинного навчання розміщено в бібліотеці scikit-learn (<https://scikit-learn.org/stable/>).

### **Завдання 4.**

Пропонуються наступні методи оцінювання продуктивності моделей машинного навчання (sklearn.metrics) (див. варіант завдання нижче):

3. обчислення показника F1 (f1\_score);

Детальна інформація про відповідні інструменти машинного навчання розміщено в бібліотеці scikit-learn (<https://scikit-learn.org/stable/>).

Виконання роботи:

У ході виконання лабораторної роботи були отримані наступні результати:

1. Ознайомитися з інформацією про набори даних для дослідження шкідливих процесів в інформаційній системі організації з використання методів машинного навчання за посиланням:

CSE-CIC-IDS2018 on AWS: <https://www.unb.ca/cic/datasets/ids-2018.html>.

2. Для завантаження набору даних відповідно до вашого варіанту необхідно встановити AWS CLI. Інсталятор Ви знайдете за посиланням:

AWS Command Line Interface: <https://aws.amazon.com/cli/>.

```
C:\Users\User>msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi /qn  
C:\Users\User>msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi  
C:\Users\User>msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi /qn
```

### Процес інсталяції aws на Windows 10

3. Щоб перевірити поточну встановлену версію, скористайтеся такою командою:

```
$ aws --version  
aws-cli/2.7.24 Python/3.8.8 Linux/4.14.133-113.105.amzn2.x86_64 botocore/1.13
```

```
C:\Users\User>aws --version  
aws-cli/2.15.48 Python/3.11.8 Windows/10 exe/AMD64 prompt/off
```

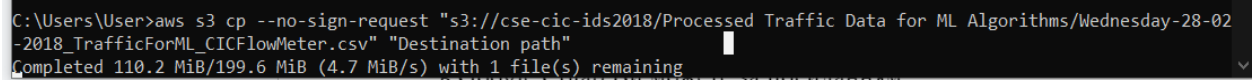
4. Для перегляду переліку наборів даних у форматі .csv скористайтеся такою командою:

```
aws s3 ls --human-readable --summarize --no-sign-request --region eu-west-3  
"s3://cse-cic-ids2018/Processed Traffic Data for ML Algorithms/"
```

```
C:\Users\User>aws s3 ls --human-readable --summarize --no-sign-request --region eu-west-3 "s3://cse-cic-ids2018/Processed Traffic Data for ML Algorithms/"  
2018-10-11 19:02:25 0 Bytes  
2018-10-11 19:02:49 336.0 MiB Friday-02-03-2018_TrafficForML_CICFlowMeter.csv  
2018-10-11 19:03:10 318.3 MiB Friday-16-02-2018_TrafficForML_CICFlowMeter.csv  
2018-10-11 19:03:33 365.1 MiB Friday-23-02-2018_TrafficForML_CICFlowMeter.csv  
2018-10-11 19:03:59 3.8 GiB Tuesday-20-02-2018_TrafficForML_CICFlowMeter.csv  
2018-10-11 19:08:38 102.8 MiB Thursday-01-03-2018_TrafficForML_CICFlowMeter.csv  
2018-10-11 19:08:48 358.5 MiB Thursday-15-02-2018_TrafficForML_CICFlowMeter.csv  
2018-10-11 19:09:20 364.9 MiB Thursday-22-02-2018_TrafficForML_CICFlowMeter.csv  
2018-10-11 19:09:44 341.6 MiB Wednesday-14-02-2018_TrafficForML_CICFlowMeter.csv  
2018-10-11 19:10:12 313.7 MiB Wednesday-21-02-2018_TrafficForML_CICFlowMeter.csv  
2018-10-11 19:10:33 199.6 MiB Wednesday-28-02-2018_TrafficForML_CICFlowMeter.csv  
  
Total Objects: 11  
Total Size: 6.4 GiB
```

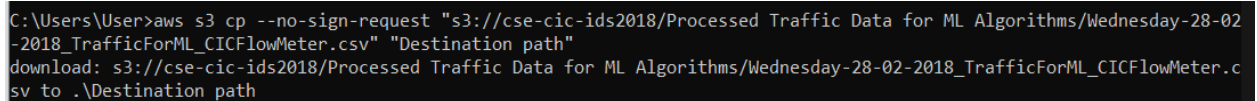
5. Для завантаження набору даних відповідно до вашого варіанту (наприклад, Wednesday-28-02-2018\_TrafficForML\_CICFlowMeter.csv) необхідно виконати таку команду:

```
aws s3 cp --no-sign-request "s3://cse-cic-ids2018/Processed Traffic Data for ML Algorithms/Wednesday-28-02-2018_TrafficForML_CICFlowMeter.csv" "Destination path"
```



```
C:\Users\User>aws s3 cp --no-sign-request "s3://cse-cic-ids2018/Processed Traffic Data for ML Algorithms/Wednesday-28-02-2018_TrafficForML_CICFlowMeter.csv" "Destination path"
Completed 110.2 MiB/199.6 MiB (4.7 MiB/s) with 1 file(s) remaining
```

Процес завантаження потрібного нам файлу

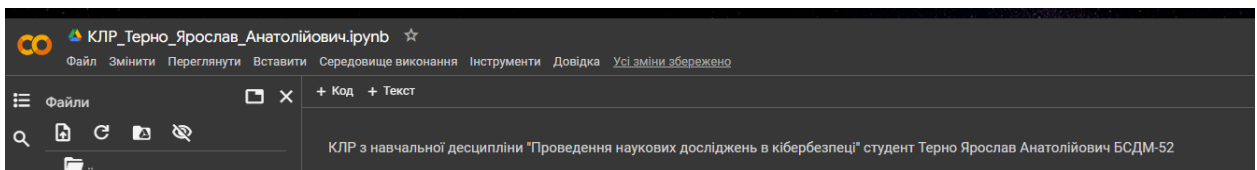


```
C:\Users\User>aws s3 cp --no-sign-request "s3://cse-cic-ids2018/Processed Traffic Data for ML Algorithms/Wednesday-28-02-2018_TrafficForML_CICFlowMeter.csv" "Destination path"
download: s3://cse-cic-ids2018/Processed Traffic Data for ML Algorithms/Wednesday-28-02-2018_TrafficForML_CICFlowMeter.csv to .\Destination path
```

6. Для проведення дослідження із застосуванням методів машинного навчання необхідно застосовувати середовище Google Colab або інші. Для використання Google Colab необхідно мати обліковий запис Google. Перейти в Google Colab Ви можете за посиланням:

[https://colab.research.google.com/#scrollTo=5fCEDCU\\_qrC0](https://colab.research.google.com/#scrollTo=5fCEDCU_qrC0)

7. У середовищі Google Colab необхідно створити новий записник (блокнот), який буде вашим звітом з комплексної лабораторної роботи.



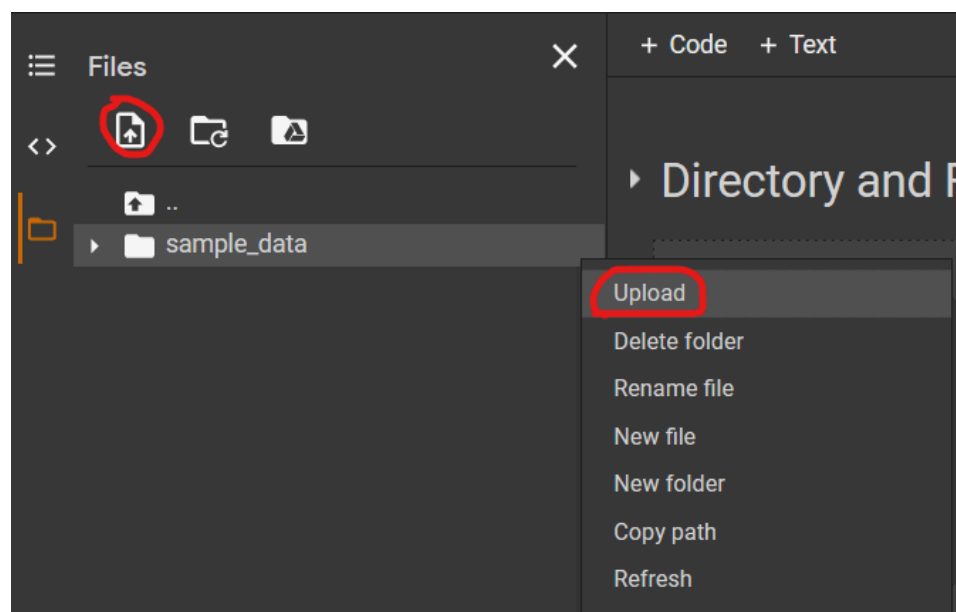
8. Здійснюється імпорт потрібних бібліотек.

```
from ast import increment_lineno
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
import tqdm.notebook as tqdm
from tqdm import tqdm_notebook as tqdm
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings('ignore')
```

9. Здійснюється імпорт набору даних (попередньо скачаний відповідно варіанту завдання).

Для завантаження будь-яких файлів з локальної файлової системи до поточної робочої директорії Colab можна скористатися опцією Upload у верхній частині панелі менеджера файлів.

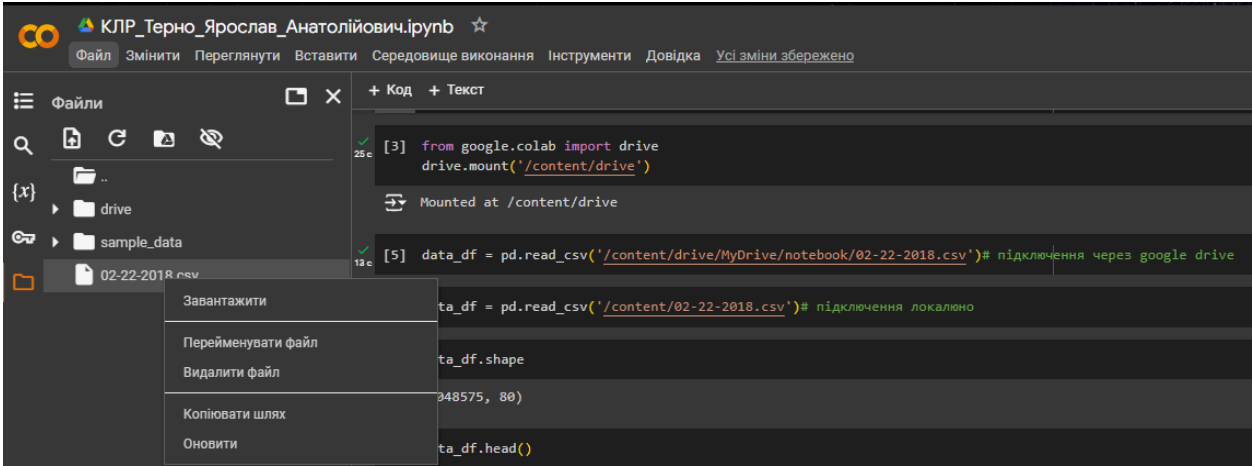
Для завантаження файлів безпосередньо в піддиректорію потрібно:  
натисніть на три точки, що з'являються при наведенні курсору на каталог;  
вибрати опцію Upload;



вибрати файли для завантаження із діалогового вікна File Upload;  
зачекати на завершення завантаження (звертаю увагу!), процес

виконання якого відображається в нижній частині панелі менеджера файлів.

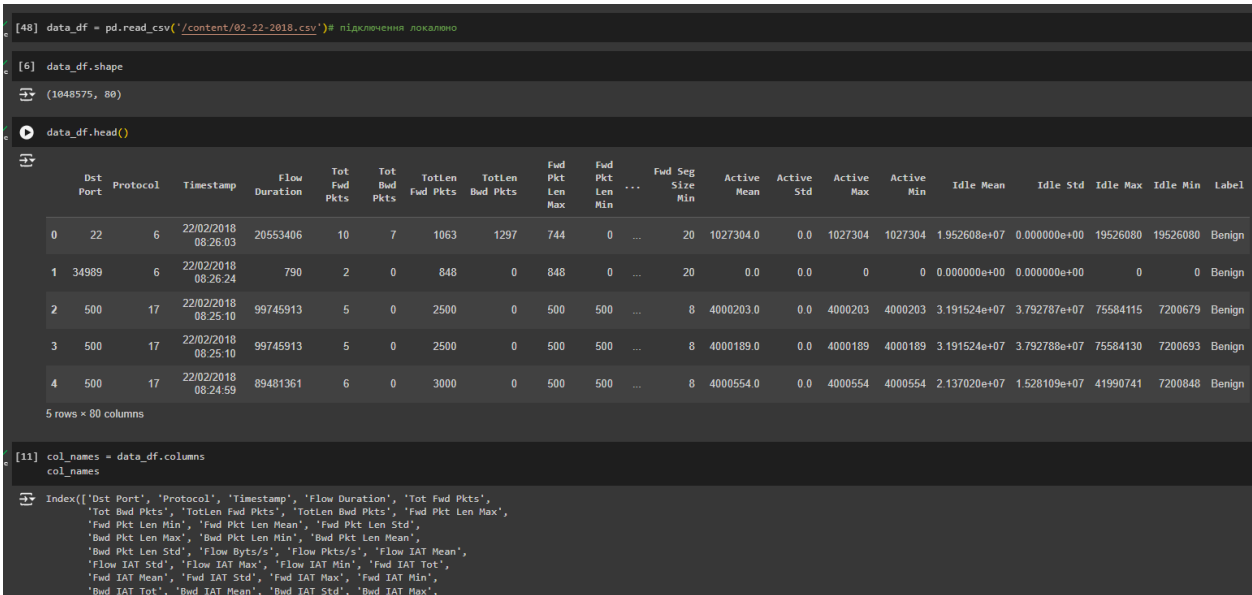
Після закінчення процесу завантаження Ви можете скопіювати шлях до датасету та відкрити в Colab. Завантажувати датасет необхідно кожного разу при входженні в середовище Colab.



The screenshot shows the Google Colab interface. On the left, the 'Files' pane shows a folder named 'sample\_data' containing a file '02-22-2018.csv'. A context menu is open over this file, showing options: 'Завантажити' (Upload), 'Перейменувати файл' (Rename file), 'Видалити файл' (Delete file), 'Копіювати шлях' (Copy path), and 'Оновити' (Refresh). The main code area shows two code cells. Cell [3] imports the 'drive' module and mounts the Google Drive at '/content/drive'. Cell [5] reads the CSV file from the mounted drive using 'pd.read\_csv' and prints its shape. The output of cell [5] shows the shape as (1048575, 80).

10. Проводиться дослідницький аналіз даних (Exploratory data analysis).

Дослідницький аналіз даних проводиться для правильного розуміння набору даних, який застосовується. Для цього здійснюється візуалізація даних, розуміння та їх аналіз для постановки прикладних задач (висунення гіпотез).



The screenshot shows the Google Colab interface with code cells [48], [6], and [11]. Cell [48] reads the CSV file from the local content directory. Cell [6] prints the shape of the data frame, which is (1048575, 80). Cell [11] prints the first few rows of the data frame using 'data\_df.head()'. The output shows a table with 80 columns and 5 rows displayed. The columns include '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 Seg Size Min', 'Active Mean', 'Active Std', 'Active Max', 'Active Min', 'Idle Mean', 'Idle Std', 'Idle Max', 'Idle Min', and 'Label'. The first five rows of data are shown, all labeled as 'Benign'.

	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 Seg Size Min	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label	
0	22	6	22/02/2018 08:26:03	20553406	10	7	1063	1297	744	0	...	20	1027304.0	0.0	1027304	1027304	1.952608e+07	0.000000e+00	19526080	19526080	Benign
1	34989	6	22/02/2018 08:26:24	790	2	0	848	0	848	0	...	20	0.0	0.0	0	0	0.000000e+00	0.000000e+00	0	0	Benign
2	500	17	22/02/2018 08:25:10	99745913	5	0	2500	0	500	500	...	8	4000203.0	0.0	4000203	4000203	3.191524e+07	3.792787e+07	75584115	7200679	Benign
3	500	17	22/02/2018 08:25:10	99745913	5	0	2500	0	500	500	...	8	4000189.0	0.0	4000189	4000189	3.191524e+07	3.792788e+07	75584130	7200693	Benign
4	500	17	22/02/2018 08:24:59	89481361	6	0	3000	0	500	500	...	8	4000554.0	0.0	4000554	4000554	2.137020e+07	1.528109e+07	41990741	7200848	Benign

```
[12] data_df['Label'].value_counts()

Label
Benign                1048213
Brute Force -Web       249
Brute Force -XSS       79
SQL Injection          34
Name: count, dtype: int64

[13] data_df['Label'].value_counts()/np.float64(len(data_df))

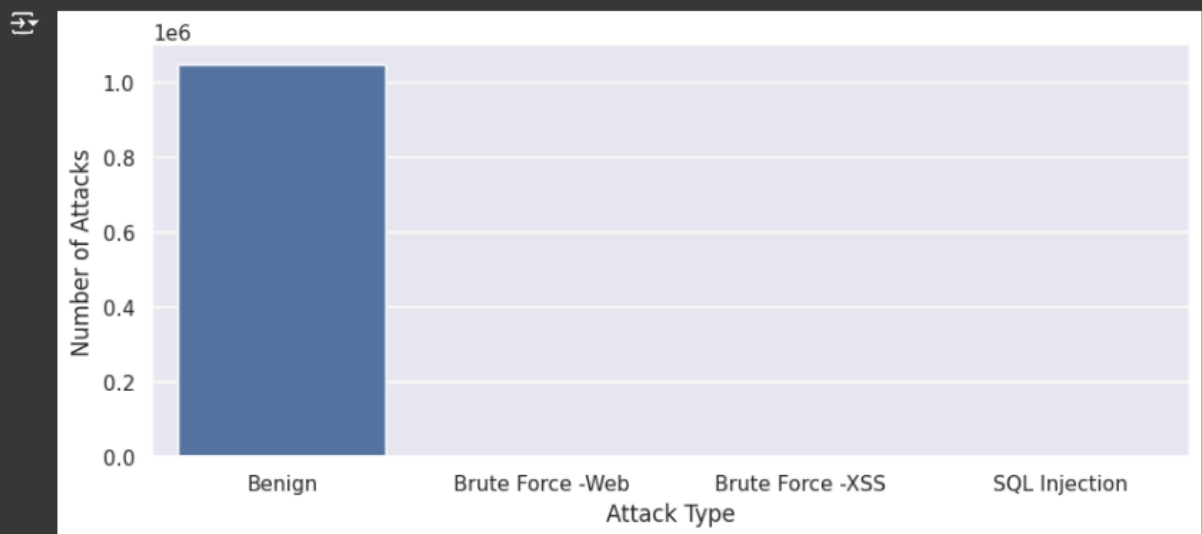
Label
Benign                0.999655
Brute Force -Web       0.000237
Brute Force -XSS       0.000075
SQL Injection          0.000032
Name: count, dtype: float64

data_df.info()

24  Fwd IAT Std          1048575 non-null float64
25  Fwd IAT Max          1048575 non-null int64
26  Fwd IAT Min          1048575 non-null int64
27  Bwd IAT Tot          1048575 non-null int64
28  Bwd IAT Mean         1048575 non-null float64
29  Bwd IAT Std          1048575 non-null float64
30  Bwd IAT Max          1048575 non-null int64
31  Bwd IAT Min          1048575 non-null int64
32  Fwd PSH Flags        1048575 non-null int64
33  Bwd PSH Flags        1048575 non-null int64
34  Fwd URG Flags        1048575 non-null int64
35  Bwd URG Flags        1048575 non-null int64
36  Fwd Header Len       1048575 non-null int64
37  Bwd Header Len       1048575 non-null int64
38  Fwd Pkts/s           1048575 non-null float64
39  Bwd Pkts/s           1048575 non-null float64
40  Pkt Len Min          1048575 non-null int64
41  Pkt Len Max          1048575 non-null int64
42  Pkt Len Mean         1048575 non-null float64
43  Pkt Len Std          1048575 non-null float64
```

11. Здійснюємо візуалізацію даних. Отримавши деяку корисну інформацію про дані, необхідно створити візуальні зображення набору даних, щоб побачити, як розвивається тенденція в даних. Візуальні елементи включають стовпчасті діаграми, діаграми розподілу, точкові діаграми тощо.

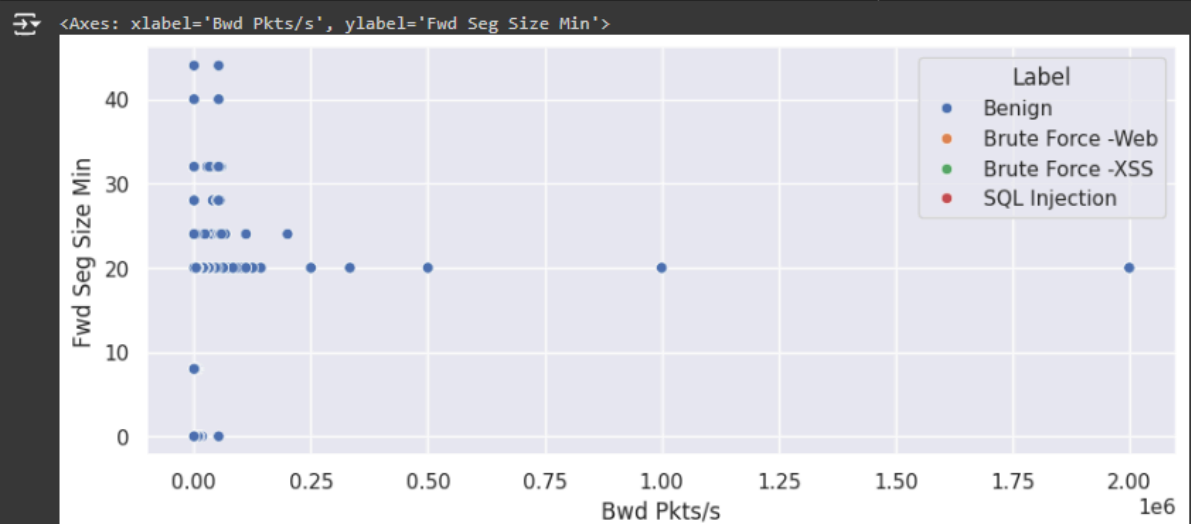
```
[15] sns.set(rc={'figure.figsize':(10,4)})
plt.xlabel('Attack Type')
sns.set_theme()
ax = sns.countplot(x='Label', data=data_df)
ax.set(xlabel='Attack Type', ylabel='Number of Attacks')
plt.show()
```



```
[16] import plotly.express as px
import plotly.offline as pyo

pyo.init_notebook_mode()
fig = px.scatter(x = data_df['Bwd Pkts/s'][:100000],
                 y = data_df['Fwd Seg Size Min'][:100000])
fig.show()
```

```
sns.set(rc={'figure.figsize':(10,4)})
sns.scatterplot(x=data_df['Bwd Pkts/s'][:50000], y=data_df['Fwd Seg Size Min'][:50000], hue='Label', data=data_df)
```



12. Здійснюємо попередню обробку даних. Попередня обробка даних відіграє важливу роль, оскільки дані можуть містити відсутні або нульові значення, а також викиди тощо.



```
[18] data_df['Label'].value_counts()

Label
Benign                1048213
Brute Force -Web       249
Brute Force -XSS        79
SQL Injection          34
Name: count, dtype: int64

[19] data_df['Label'] = data_df['Label'].map({'Benign': 1,
                                             'Brute Force -Web': 2,
                                             'Brute Force -XSS': 3,
                                             'SQL Injection': 4})
```

або так:

```
data_df = data_df.drop(['Timestamp'], axis=1)
data_df.head()
```

	Dst Port	Protocol	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 Seg Size Min	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
0	22	6	20553406	10	7	1063	1297	744	0	106.3	...	20	1027304.0	0.0	1027304	1027304	1.952608e+07	0.000000e+00	19526080	19526080	1
1	34989	6	790	2	0	848	0	848	0	424.0	...	20	0.0	0.0	0	0	0.000000e+00	0.000000e+00	0	0	1
2	500	17	99745913	5	0	2500	0	500	500	500.0	...	8	4000203.0	0.0	4000203	4000203	3.191524e+07	3.792787e+07	75584115	7200679	1
3	500	17	99745913	5	0	2500	0	500	500	500.0	...	8	4000189.0	0.0	4000189	4000189	3.191524e+07	3.792788e+07	75584130	7200693	1
4	500	17	89481361	6	0	3000	0	500	500	500.0	...	8	4000554.0	0.0	4000554	4000554	2.137020e+07	1.528109e+07	41990741	7200848	1

5 rows x 79 columns

```
[21] data_df.head()
```

	Dst Port	Protocol	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 Seg Size Min	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
0	22	6	20553406	10	7	1063	1297	744	0	106.3	...	20	1027304.0	0.0	1027304	1027304	1.952608e+07	0.000000e+00	19526080	19526080	1
1	34989	6	790	2	0	848	0	848	0	424.0	...	20	0.0	0.0	0	0	0.000000e+00	0.000000e+00	0	0	1
2	500	17	99745913	5	0	2500	0	500	500	500.0	...	8	4000203.0	0.0	4000203	4000203	3.191524e+07	3.792787e+07	75584115	7200679	1
3	500	17	99745913	5	0	2500	0	500	500	500.0	...	8	4000189.0	0.0	4000189	4000189	3.191524e+07	3.792788e+07	75584130	7200693	1
4	500	17	89481361	6	0	3000	0	500	500	500.0	...	8	4000554.0	0.0	4000554	4000554	2.137020e+07	1.528109e+07	41990741	7200848	1

5 rows x 79 columns

```
✓ [22] data_df.replace([np.inf, -np.inf], np.nan, inplace=True)
```

```
✓ [23] null_col=[]  
      check_null = data_df.isnull().sum()  
      for indx,val in check_null.items():  
          if(val!=0):  
              print('\''+indx+'\'' has ' + str(val)+ ' null values!!')  
              null_col.append(indx)
```

```
⇒ 'Flow Byts/s' has 5610 null values!!  
  'Flow Pkts/s' has 5610 null values!!
```

```
✓ [24] for i in null_col:  
      print(i)  
      print(data_df[i].describe())  
      npercent = data_df[i].isnull().sum()/len(data_df[i])*100  
      print('nan% :', npercent)  
      print('\n')
```

```
⇒ Flow Byts/s  
count    1.042965e+06  
mean      2.635930e+05  
std       4.571838e+06  
min       0.000000e+00  
25%       0.000000e+00  
50%       7.876609e+02  
75%       6.339144e+04  
max       1.806643e+09  
Name: Flow Byts/s, dtype: float64  
nan% : 0.5350118017309206
```

```
Flow Pkts/s  
count    1.042965e+06  
mean      2.730313e+04
```

[25] data\_df.head()

	Dst Port	Protocol	Flow Duration	Tot Fwd Pkts	Tot Bud Pkts	TotLen Fwd Pkts	TotLen Bud Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min	Fwd Pkt Len Mean	...	Fwd Seg Size Min	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
0	22	6	20553406	10	7	1063	1297	744	0	106.3	...	20	1027304.0	0.0	1027304	1027304	1.952608e+07	0.000000e+00	19526080	19526080	1
1	34989	6	790	2	0	848	0	848	0	424.0	...	20	0.0	0.0	0	0	0.000000e+00	0.000000e+00	0	0	1
2	500	17	99745913	5	0	2500	0	500	500	500.0	...	8	4000203.0	0.0	4000203	4000203	3.191524e+07	3.792787e+07	75584115	7200679	1
3	500	17	99745913	5	0	2500	0	500	500	500.0	...	8	4000189.0	0.0	4000189	4000189	3.191524e+07	3.792788e+07	75584130	7200693	1
4	500	17	89481361	6	0	3000	0	500	500	500.0	...	8	4000554.0	0.0	4000554	4000554	2.137020e+07	1.528109e+07	41990741	7200848	1

5 rows × 79 columns

[26] # Fill empty values with mean  
print('Filling \'Flow Pkts/s\' nan values with mean :', data\_df['Flow Pkts/s'].mean())  
print('Filling \'Flow Byts/s\' nan values with mean :', data\_df['Flow Byts/s'].mean())  
  
data\_df['Flow Byts/s'].fillna(data\_df['Flow Byts/s'].mean(),inplace=True)  
data\_df['Flow Pkts/s'].fillna(data\_df['Flow Pkts/s'].mean(),inplace=True)  
  
data\_df.fillna(data\_df.mean(), inplace=True)

Filling 'Flow Pkts/s' nan values with mean : 27393.127841855376  
Filling 'Flow Byts/s' nan values with mean : 263592.9800085811

data\_df.isnull().sum()

Dst Port 0  
Protocol 0  
Flow Duration 0  
Tot Fwd Pkts 0  
Tot Bud Pkts 0  
TotLen Fwd Pkts 0  
TotLen Bud Pkts 0  
Fwd Pkt Len Max 0  
Fwd Pkt Len Min 0  
Fwd Pkt Len Mean 0  
...  
Fwd Seg Size Min 0  
Active Mean 0  
Active Std 0  
Active Max 0  
Active Min 0  
Idle Mean 0  
Idle Std 0  
Idle Max 0

[28] round(data\_df.describe(),2)

	Dst Port	Protocol	Flow Duration	Tot Fwd Pkts	Tot Bud Pkts	TotLen Fwd Pkts	TotLen Bud Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min	Fwd Pkt Len Mean	...	Fwd Seg Size Min	Active Mean	Active Std	Active Max	Active Min	Idle Mean
count	1048575.00	1048575.00	1.048575e+06	1048575.00	1048575.00	1048575.00	1048575.00	1048575.00	1048575.00	1048575.00	...	1048575.00	1.048575e+06	1048575.00	1.048575e+06	1.048575e+06	1.048575e+06
mean	8972.82	9.12	1.493482e+07	6.37	8.23	398.87	7344.81	177.17	12.87	45.18	...	16.23	6.129767e+04	29607.11	1.117695e+05	4.508249e+04	1.169886e+07
std	18109.54	5.11	1.475475e+09	71.34	203.59	4970.65	295808.49	267.48	28.51	54.03	...	5.87	6.180167e+05	251855.08	8.115030e+05	5.890562e+05	8.814511e+08
min	0.00	0.00	-8.282200e+11	1.00	0.00	0.00	0.00	0.00	0.00	0.00	...	0.00	0.000000e+00	0.00	0.000000e+00	0.000000e+00	0.000000e+00
25%	53.00	6.00	5.090000e+02	1.00	0.00	0.00	0.00	0.00	0.00	0.00	...	8.00	0.000000e+00	0.00	0.000000e+00	0.000000e+00	0.000000e+00
50%	443.00	6.00	8.159100e+04	2.00	1.00	45.00	103.00	41.00	0.00	36.00	...	20.00	0.000000e+00	0.00	0.000000e+00	0.000000e+00	0.000000e+00
75%	3389.00	17.00	5.235859e+06	8.00	7.00	455.00	582.00	189.00	32.00	56.89	...	20.00	0.000000e+00	0.00	0.000000e+00	0.000000e+00	0.000000e+00
max	65533.00	17.00	1.200000e+08	11478.00	38418.00	3982514.00	56069495.00	3804.00	1460.00	1808.42	...	44.00	1.084622e+08	63575307.43	1.084622e+08	1.084622e+08	3.955714e+11

8 rows × 79 columns

```

plt.figure(figsize=(24,20))

plt.subplot(4,2,3)
fig = data_df.boxplot(column='Flow Duration')
fig.set_title('')
fig.set_ylabel('Flow Duration')

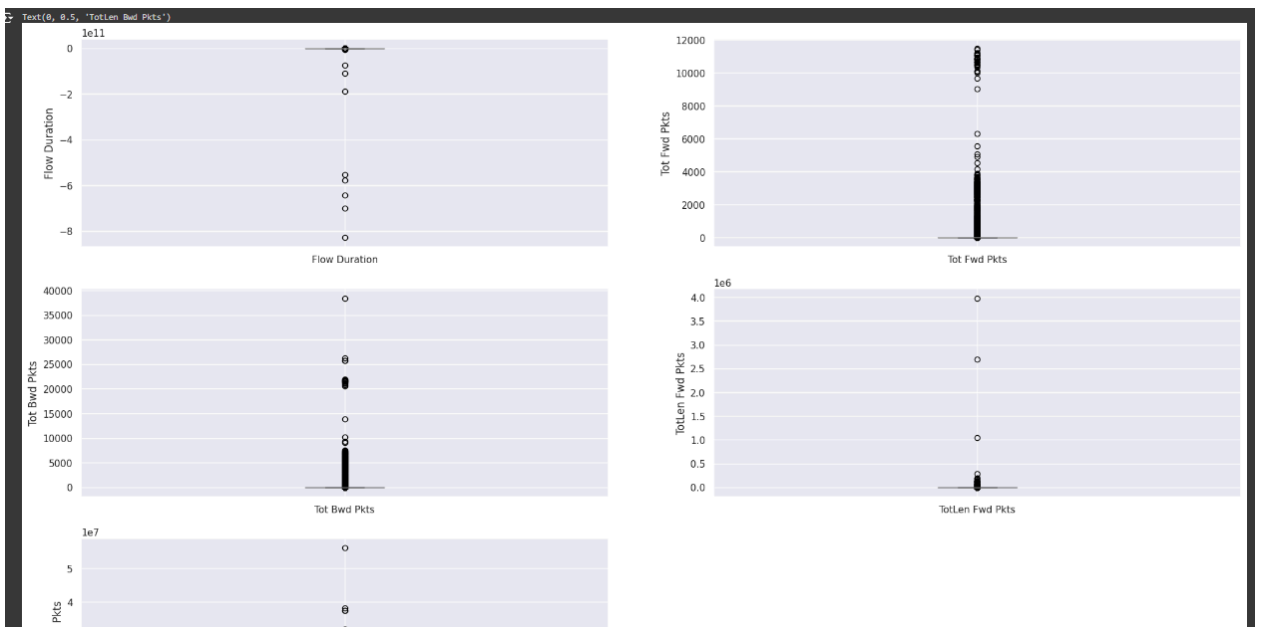
plt.subplot(4,2,4)
fig = data_df.boxplot(column='Tot Fwd Pkts')
fig.set_title('')
fig.set_ylabel('Tot Fwd Pkts')

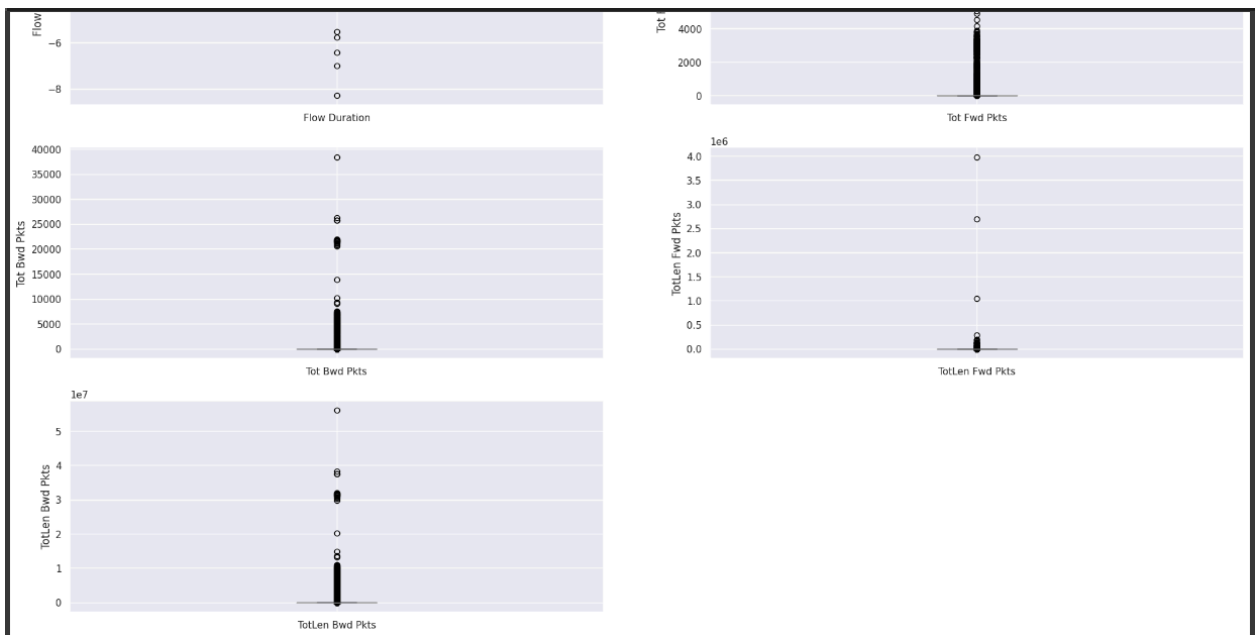
plt.subplot(4,2,5)
fig = data_df.boxplot(column='Tot Bwd Pkts')
fig.set_title('')
fig.set_ylabel('Tot Bwd Pkts')

plt.subplot(4,2,6)
fig = data_df.boxplot(column='TotLen Fwd Pkts')
fig.set_title('')
fig.set_ylabel('TotLen Fwd Pkts')

plt.subplot(4,2,7)
fig = data_df.boxplot(column='TotLen Bwd Pkts')
fig.set_title('')
fig.set_ylabel('TotLen Bwd Pkts')

```





```
plt.figure(figsize=(24,20))

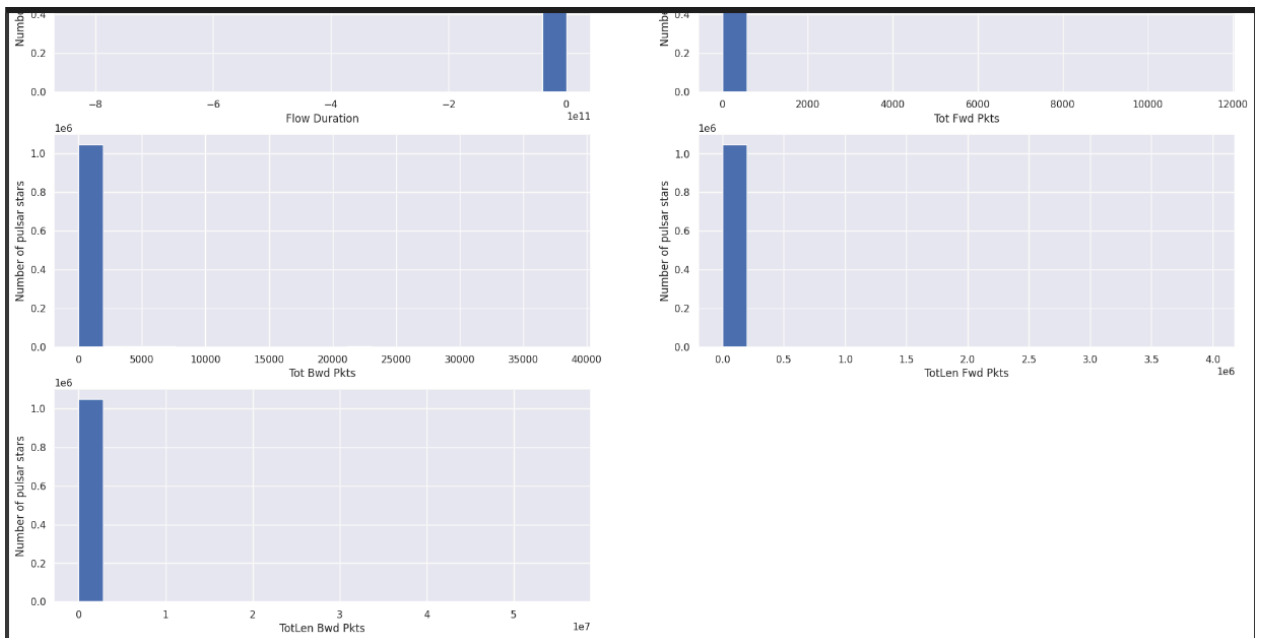
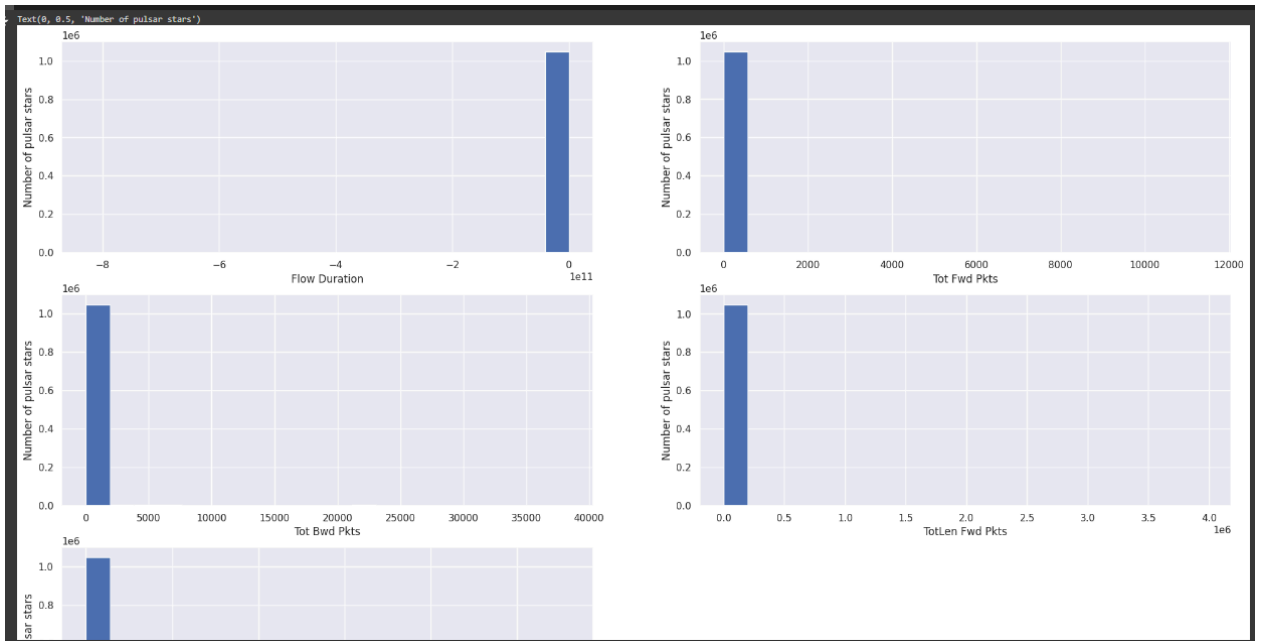
plt.subplot(4,2,3)
fig = data_df['Flow Duration'].hist(bins=20)
fig.set_xlabel('Flow Duration')
fig.set_ylabel('Number of pulsar stars')

plt.subplot(4,2,4)
fig = data_df['Tot Fwd Pkts'].hist(bins=20)
fig.set_xlabel('Tot Fwd Pkts')
fig.set_ylabel('Number of pulsar stars')

plt.subplot(4,2,5)
fig = data_df['Tot Bwd Pkts'].hist(bins=20)
fig.set_xlabel('Tot Bwd Pkts')
fig.set_ylabel('Number of pulsar stars')

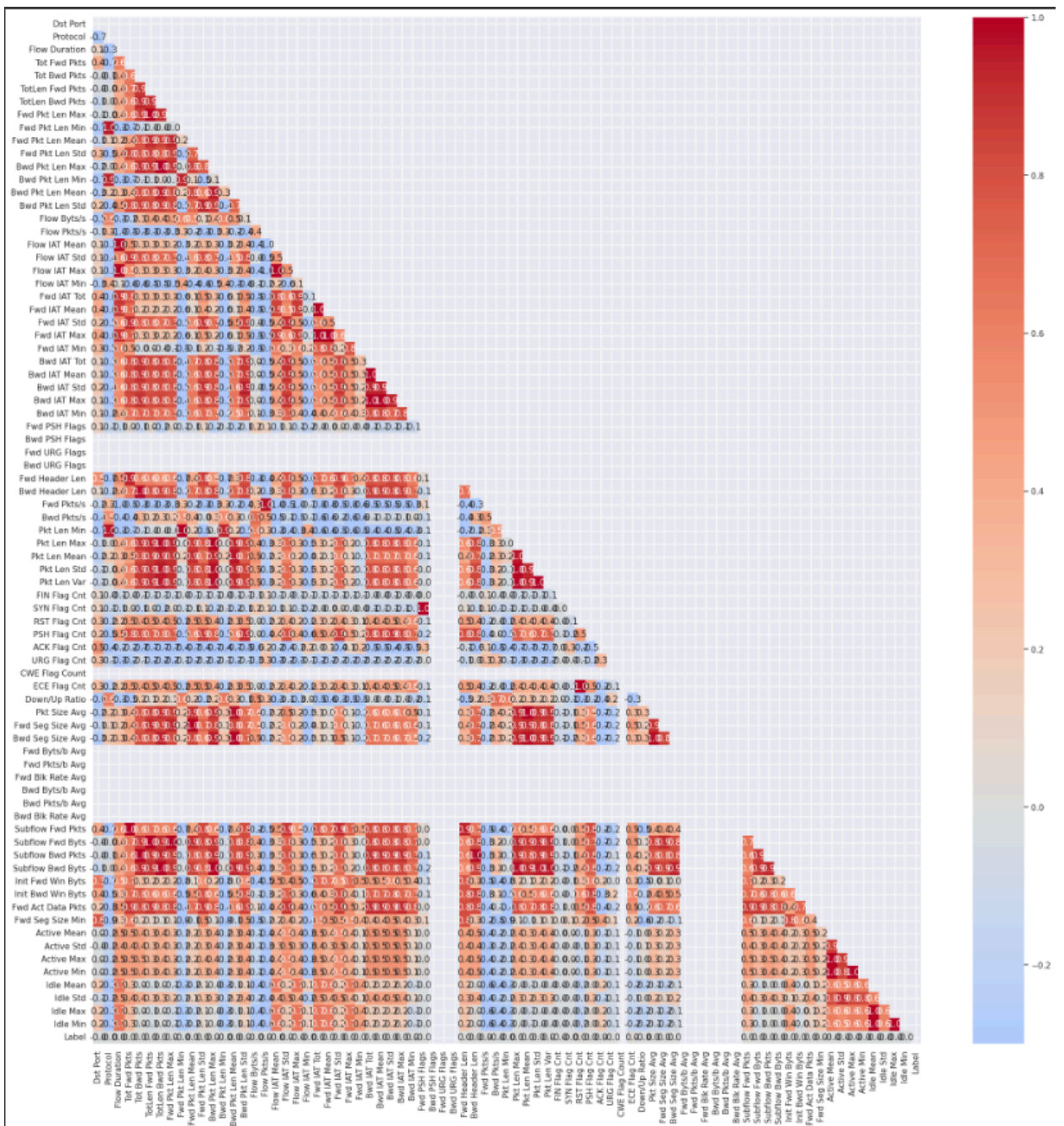
plt.subplot(4,2,6)
fig = data_df['TotLen Fwd Pkts'].hist(bins=20)
fig.set_xlabel('TotLen Fwd Pkts')
fig.set_ylabel('Number of pulsar stars')

plt.subplot(4,2,7)
fig = data_df['TotLen Bwd Pkts'].hist(bins=20)
fig.set_xlabel('TotLen Bwd Pkts')
fig.set_ylabel('Number of pulsar stars')
```



13. Здійснюємо кореляційний аналіз даних.

```
matrix = np.triu(data_df.corr(method='spearman'))
plt.figure(figsize=(25,25))
sns.heatmap(data_df.corr(method='spearman'), annot=True,
            fmt='.1f', vmin=-0.3, center=0, cmap='coolwarm', mask=matrix)
```



```
[32] corr_matrix=data_df.corr().round(2)
```

```
corr_matrix
```

	Dst Port	Protocol	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 Seg Size Min	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
Dst Port	1.00	-0.30	0.00	-0.03	-0.02	-0.02	-0.01	-0.23	-0.21	-0.29	...	0.31	-0.04	-0.06	-0.06	-0.02	0.01	-0.00	0.00	0.02	-0.01
Protocol	-0.30	1.00	-0.00	-0.04	-0.02	-0.04	-0.01	-0.30	0.68	-0.00	...	-0.83	-0.05	-0.08	-0.08	-0.03	-0.01	-0.00	-0.01	-0.03	-0.01
Flow Duration	0.00	-0.00	1.00	0.00	0.00	0.00	0.00	0.00	-0.00	0.00	...	0.01	0.00	0.01	0.01	0.00	-0.64	-0.64	-0.66	-0.18	0.00
Tot Fwd Pkts	-0.03	-0.04	0.00	1.00	0.79	0.09	0.78	0.08	-0.03	0.03	...	0.04	0.18	0.10	0.19	0.16	0.00	0.00	0.00	0.00	0.01
Tot Bwd Pkts	-0.02	-0.02	0.00	0.79	1.00	0.04	1.00	0.04	-0.02	0.00	...	0.02	0.20	0.09	0.20	0.19	0.00	0.00	0.00	0.00	0.00
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Idle Mean	0.01	-0.01	-0.64	0.00	0.00	-0.00	0.00	-0.01	-0.01	-0.01	...	-0.00	0.00	0.00	0.00	0.00	1.00	0.98	0.98	0.49	-0.00
Idle Std	-0.00	-0.00	-0.64	0.00	0.00	-0.00	0.00	-0.00	-0.00	-0.00	...	-0.01	0.00	0.00	0.00	0.00	0.98	1.00	0.99	0.34	-0.00
Idle Max	0.00	-0.01	-0.66	0.00	0.00	-0.00	0.00	-0.00	-0.00	-0.00	...	-0.01	0.00	0.00	0.00	0.00	0.98	0.99	1.00	0.38	-0.00
Idle Min	0.02	-0.03	-0.18	0.00	0.00	-0.00	0.00	-0.01	-0.02	-0.02	...	0.01	0.01	0.01	0.01	0.00	0.49	0.34	0.38	1.00	-0.00
Label	-0.01	-0.01	0.00	0.01	0.00	0.05	0.00	0.01	-0.01	0.03	...	0.01	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	1.00

79 rows x 79 columns

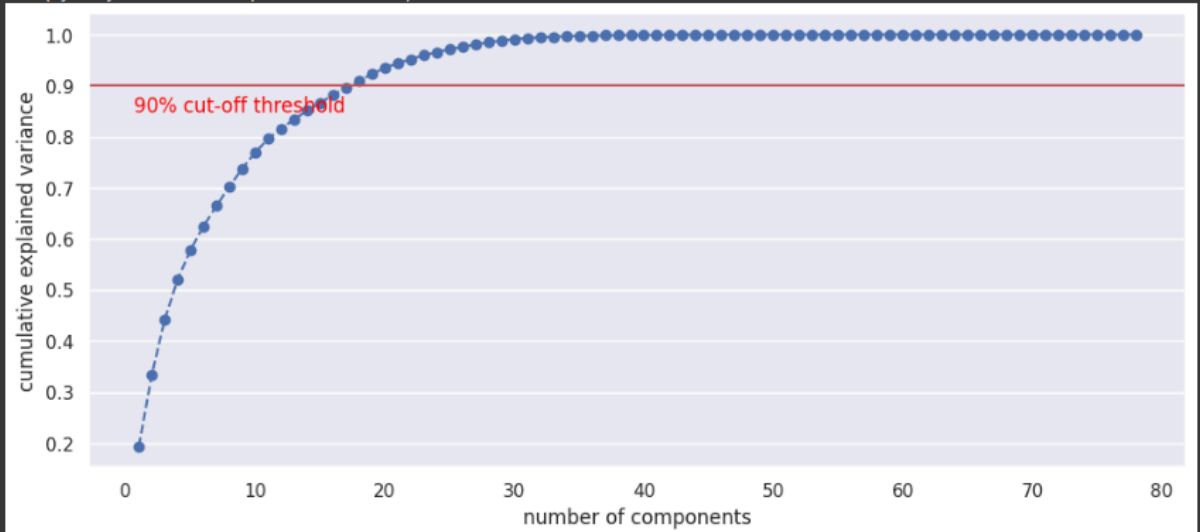
## 14. Застосуємо метод головних компонент для вирішення проблеми

великої розмірності набору даних.

```
[33] from sklearn.preprocessing import StandardScaler  
      from sklearn.decomposition import PCA
```

```
X = data_df.drop('Label', axis = 1)  
X_scaled = StandardScaler().fit_transform(X)  
pca = PCA().fit(X_scaled)  
plt.figure(figsize=(12,5))  
x = np.arange(1, len(pca.explained_variance_ratio_)+1, 1)  
plt.plot(x, np.cumsum(pca.explained_variance_ratio_), marker='o', linestyle='--', color='b')  
plt.axhline(y=0.9, color='r', linestyle='--')  
plt.text(0.6, 0.85, '90% cut-off threshold', color='red', fontsize=12)  
plt.grid(axis='x')  
  
plt.xlabel('number of components')  
plt.ylabel('cumulative explained variance')
```

```
Text(0, 0.5, 'cumulative explained variance')
```





```

pca = PCA(n_components = 6).fit(X_scaled)

components = pca.fit_transform(X_scaled)
total_var = pca.explained_variance_ratio_.sum() * 100
n_components = len(pca.explained_variance_ratio_)

labels = {str(i): f"PC {i+1}" for i in range(n_components)}
labels['color'] = 'Median Price'
labels = {
    str(i): f"PC {i+1} ({var:.1f}%)"
    for i, var in enumerate(pca.explained_variance_ratio_ * 100)
}

fig = px.scatter_matrix(
    components,
    dimensions=range(n_components),
    labels=labels,
    title=f'Total Explained Variance: {total_var:.2f}%',
)
fig.update_traces(marker_size=1, diagonal_visible=False)

fig.update_layout(
    font=dict(size=10, color='black'),
    autosize=False,
    width=1000,
    height=800,
)
fig.show()

```

```

[36] map = data_df.dtypes
indx_set = set(map.values.astype(str))
datatype_dict = {i: [] for i in indx_set}
for index, val in data_df.dtypes.items():
    datatype_dict[str(val)].append(index)

if 'object' in datatype_dict:
    print('Column with object datatype: \n', datatype_dict['object'], '\n')
else:
    print('No columns with object datatype found.\n')

if 'int64' in datatype_dict:
    print('Column with int64 datatype: \n', np.array(datatype_dict['int64']), '\n')
else:
    print('No columns with int64 datatype found.\n')

if 'float64' in datatype_dict:
    print('Column with float64 datatype: \n', np.array(datatype_dict['float64']))
else:
    print('No columns with float64 datatype found.')

```

15. Продовжуємо підготовку даних для навчання.

```

map = data_df.dtypes
indx_set = set(map.values.astype(str))
datatype_dict = {i: [] for i in indx_set}
for index, val in data_df.dtypes.items():
    datatype_dict[str(val)].append(index)

if 'object' in datatype_dict:
    print('Column with object datatype: \n', datatype_dict['object'], '\n')
else:
    print('No columns with object datatype found.\n')

if 'int64' in datatype_dict:
    print('Column with int64 datatype: \n', np.array(datatype_dict['int64']), '\n')
else:
    print('No columns with int64 datatype found.\n')

if 'float64' in datatype_dict:
    print('Column with float64 datatype: \n', np.array(datatype_dict['float64']))
else:
    print('No columns with float64 datatype found.')

```

Column with object datatype:  
['Timestamp', 'Label']

Column with int64 datatype:  
['Dst Port' 'Protocol' 'Flow Duration' 'Tot Fwd Pkts' 'Tot Bwd Pkts'  
'TotLen Fwd Pkts' 'TotLen Bwd Pkts' 'Fwd Pkt Len Max' 'Fwd Pkt Len Min'  
'Bwd Pkt Len Max' 'Bwd Pkt Len Min' 'Flow IAT Max' 'Flow IAT Min'  
'Fwd IAT Tot' 'Fwd IAT Max' 'Fwd IAT Min' 'Bwd IAT Tot' 'Bwd IAT Max'  
'Bwd IAT Min' 'Fwd PSH Flags' 'Bwd PSH Flags' 'Fwd URG Flags'  
'Bwd URG Flags' 'Fwd Header Len' 'Bwd Header Len' 'Pkt Len Min'  
'Pkt Len Max']

Column with float64 datatype:  
['Fwd Pkt Len Mean' 'Fwd Pkt Len Std' 'Bwd Pkt Len Mean' 'Bwd Pkt Len Std'  
'Flow Byts/s' 'Flow Pkts/s' 'Flow IAT Mean' 'Flow IAT Std' 'Fwd IAT Mean'  
'Fwd IAT Std' 'Bwd IAT Mean' 'Bwd IAT Std' 'Fwd Pkts/s' 'Bwd Pkts/s'  
'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 Max' 'Active Min' 'Idle Mean' 'Idle Std' 'Idle Max' 'Idle Min']

```

from tqdm import tqdm
import numpy as np

drop_lst = []
catfeat = []
for col in tqdm(data_df.columns):
    if len(data_df[col].unique()) == 1:
        drop_lst.append(col)
    elif len(data_df[col].unique()) <= 2:
        catfeat.append(col)
print('Columns with single value: \n', np.array(drop_lst), '\n')
print('Categorical columns: \n', np.array(catfeat), '\n')

```

100%|██████████| 80/80 [00:00<00:00, 109.26it/s]Columns with single value:  
['Bwd PSH Flags' 'Fwd URG Flags' 'Bwd URG Flags']

Categorical columns:  
['Fwd PSH Flags' 'CWE Flag Count' 'Fwd Byts/b Avg' 'Fwd Pkts/b Avg'  
'Fwd Blk Rate Avg' 'Bwd Byts/b Avg' 'Bwd Pkts/b Avg' 'Bwd Blk Rate Avg']

```

# Видаляємо відповідні стовпці
print('Dataframe shape before trimming:', data_df.shape)

# drop_lst.extend(['Timestamp']) # Timestamp has been deleted before
print('\nDropping following columns:\n', np.array(drop_lst))
data_df = data_df.drop(drop_lst,axis=1)
print('\nDataframe shape after trimming:', data_df.shape)

```

Dataframe shape before trimming: (252612, 80)

Dropping following columns:  
['Bwd PSH Flags' 'Fwd URG Flags' 'Bwd URG Flags']

Dataframe shape after trimming: (252612, 77)

```

[53] X = data_df.drop('Label', axis = 1)
     y = data_df['Label']

```

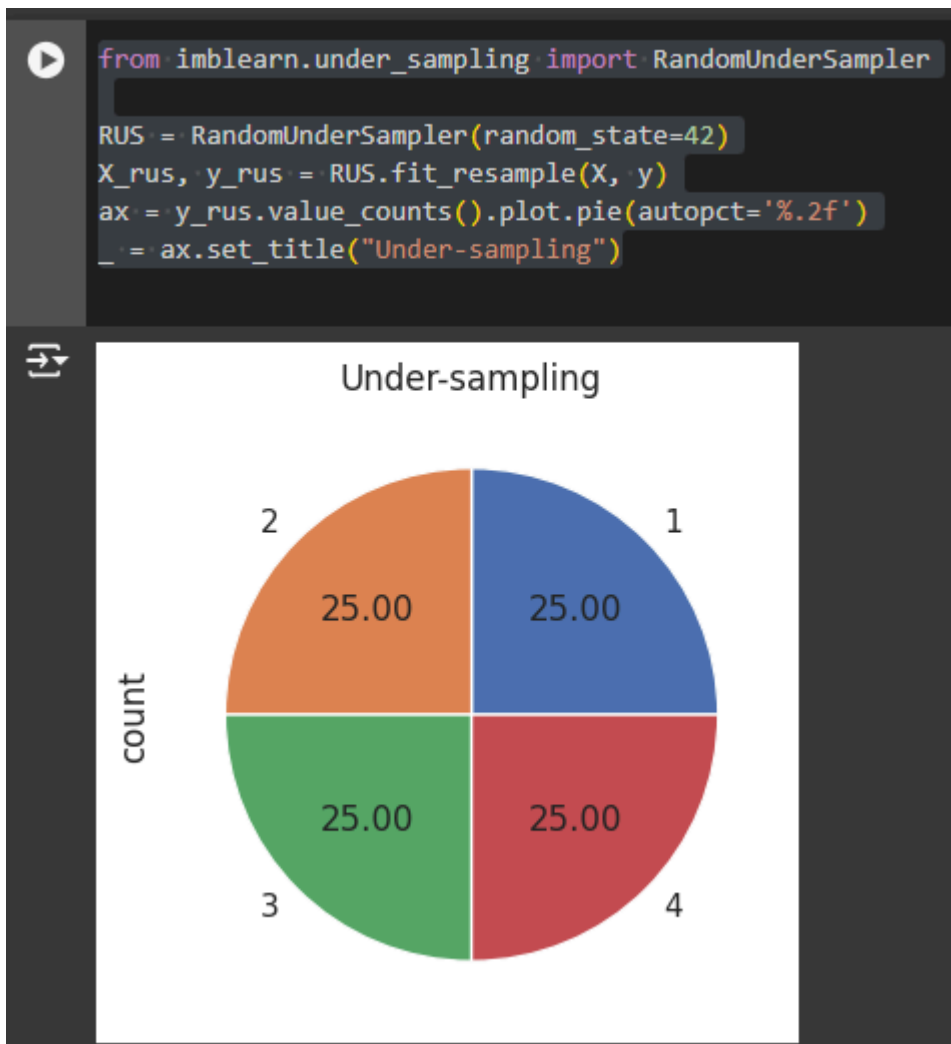
```

!pip install -U imbalanced-learn

```

Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.10/dist-packages (0.10.1)  
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.24.3)  
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.10.1)  
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.3.2)  
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.3.2)  
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (3.2.0)

+ Кол + Текст



```
[56] from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_rus, y_rus, test_size=0.3, random_state=42)

[57] X_train.shape, X_test.shape
((95, 68), (41, 68))

[58] cols = X_train.columns

[59] from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

[60] X_train = pd.DataFrame(X_train, columns=cols)
X_test = pd.DataFrame(X_test, columns=cols)

X_train.describe()
```

	Dst Port	Protocol	Flow Duration	Tot Fwd Pkts	Tot Bud Pkts	TotLen Fwd Pkts	TotLen Bud Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min	Fwd Pkt Len Mean	...	Fwd Act Data Pkts	Fwd Seg Size Min	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Id
count	9.500000e+01	9.500000e+01	9.500000e+01	9.500000e+01	9.500000e+01	9.500000e+01	9.500000e+01	9.500000e+01	9.500000e+01	9.500000e+01	...	9.500000e+01	9.500000e+01	9.500000e+01	9.500000e+01	9.500000e+01	9.500000e+01	9.500000e+01	9.500000e+01
mean	2.103500e-17	-2.045148e-18	-7.011935e-18	1.051790e-17	4.966787e-17	6.661338e-17	4.148728e-17	4.908354e-17	3.564400e-17	3.973430e-17	...	-4.090295e-18	4.324027e-17	2.045148e-17	-2.045148e-18	-2.980072e-17	-4.674623e-18	-2.863207e-17	
std	1.005305e+00	1.005305e+00	1.005305e+00	1.005305e+00	1.005305e+00	1.005305e+00	1.005305e+00	1.005305e+00	1.005305e+00	1.005305e+00	...	1.005305e+00	1.005305e+00	1.005305e+00	1.005305e+00	1.005305e+00	1.005305e+00	1.005305e+00	1.005305e+00
min	-2.543062e-01	-2.596454e-01	-6.068396e-01	-5.238899e-01	-5.435601e-01	-5.052821e-01	-4.586948e-01	-9.953791e-01	-2.488722e-01	-8.966779e-01	...	-5.144016e-01	-3.557962e+00	-1.463737e-01	-1.031421e-01	-1.447943e-01	-1.342744e-01	-2.044438e-01	
25%	-2.480499e-01	-2.596454e-01	-6.068323e-01	-5.097468e-01	-5.435601e-01	-5.052821e-01	-4.586948e-01	-9.953791e-01	-2.488722e-01	-8.966779e-01	...	-5.144016e-01	1.976424e-01	-1.463737e-01	-1.031421e-01	-1.447943e-01	-1.342744e-01	-2.044438e-01	
50%	-2.480499e-01	-2.596454e-01	-4.609943e-01	-4.814606e-01	-4.679920e-01	-4.882799e-01	-4.527792e-01	-4.975756e-01	-2.488722e-01	-3.852010e-01	...	-4.886544e-01	1.976424e-01	-1.463737e-01	-1.031421e-01	-1.447943e-01	-1.342744e-01	-2.044438e-01	
75%	-2.480499e-01	-2.596454e-01	-2.935256e-01	-3.683159e-01	-3.294505e-01	-4.415819e-01	-4.237690e-01	1.002019e+00	-2.488722e-01	5.867450e-01	...	-3.599185e-01	1.976424e-01	-1.463737e-01	-1.031421e-01	-1.447943e-01	-1.342744e-01	-2.044438e-01	
max	5.125773e+00	3.851407e+00	3.785591e+00	2.333015e+00	2.202081e+00	2.118655e+00	2.715778e+00	2.650026e+00	5.686167e+00	2.117216e+00	...	2.086064e+00	3.952847e+00	7.227009e+00	9.695360e+00	7.825035e+00	9.014514e+00	5.798017e+00	

8 rows x 68 columns

16. Проводимо навчання моделі відповідно до варіанту завдання.

Як приклад, застосовуємо метод k найближчих сусідів (k-Nearest Neighbours, kNN);

```
[189] from sklearn.neighbors import KNeighborsClassifier
      from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

[196] # Створюємо та навчаємо модель k-найближчих сусідів

      k = 3 # Задаємо кількість сусідів
      knn = KNeighborsClassifier(n_neighbors=k)
      knn.fit(X_train, y_train)

      KNeighborsClassifier
      KNeighborsClassifier(n_neighbors=3)

[197] # Робимо передбачення
      y_pred = knn.predict(X_test)

# Оцінка моделі
      print('Model accuracy score with knn kernel and C=100.0 : {0:0.4f}'.format(accuracy_score(y_test, y_pred)))

Model accuracy score with knn kernel and C=100.0 : 0.6098
```

```
[215] # Докладний звіт про класифікацію
      print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
1	1.00	0.58	0.74	12
2	0.55	0.50	0.52	12
3	0.39	1.00	0.56	7
4	1.00	0.50	0.67	10
accuracy			0.61	41
macro avg	0.73	0.65	0.62	41
weighted avg	0.76	0.61	0.63	41

```
# Матриця неточностей (confusion matrix)
print(confusion_matrix(y_test, y_pred))

[[7 0 5 0]
 [0 6 6 0]
 [0 0 7 0]
 [0 5 0 5]]
```

17. Здійснюємо оцінювання отриманої моделі.

```

# Матрица неточностей (confusion matrix)
cm = confusion_matrix(y_test, y_pred)

print('Confusion matrix\n\n', cm)

print('\nTrue Positives(TP) = ', cm[0,0])

print('\nTrue Negatives(TN) = ', cm[1,1])

print('\nFalse Positives(FP) = ', cm[0,1])

print('\nFalse Negatives(FN) = ', cm[1,0])

```

Confusion matrix

```

[[7 0 5 0]
 [0 6 6 0]
 [0 0 7 0]
 [0 5 0 5]]

```

True Positives(TP) = 7

True Negatives(TN) = 6

False Positives(FP) = 0

False Negatives(FN) = 0

```

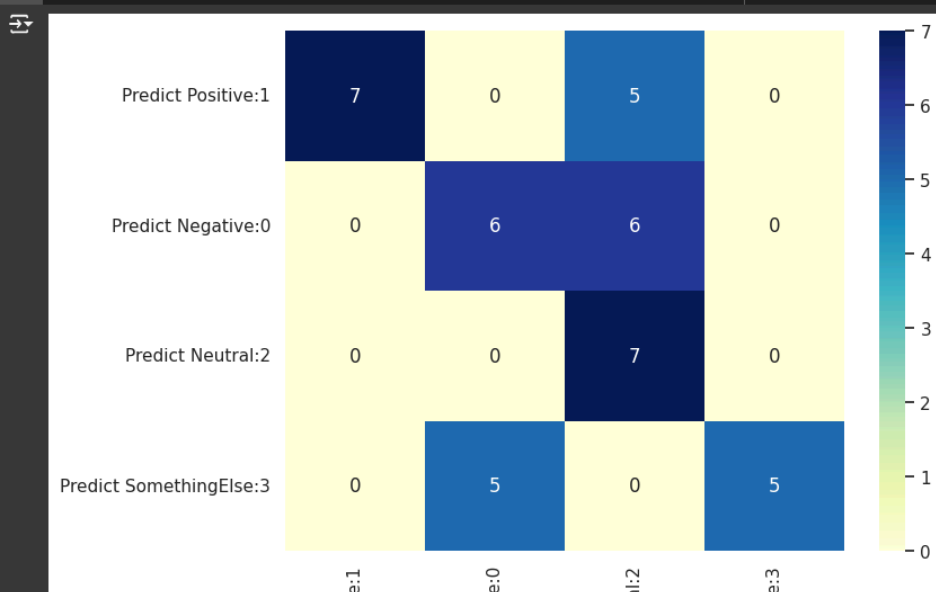
import matplotlib.pyplot as plt
import seaborn as sns

# Візуалізуємо матрицю плутанини
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive:1', 'Actual Negative:0', 'Actual Neutral:2', 'Actual SomethingElse:3'],
                        index=['Predict Positive:1', 'Predict Negative:0', 'Predict Neutral:2', 'Predict SomethingElse:3'])

plt.figure(figsize=(8, 6))
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')

plt.show()

```



```
[219] # Обчислюємо F1-показник
from sklearn.metrics import f1_score, classification_report
f1 = f1_score(y_test, y_pred, average='macro') # Ви можете використовувати 'micro', 'weighted' або None для інших варіантів обчислення
print(f'F1 Score (macro): {f1:.2f}')
```

F1 Score (macro): 0.62

```
# Докладний звіт про класифікацію
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
1	1.00	0.58	0.74	12
2	0.55	0.50	0.52	12
3	0.39	1.00	0.56	7
4	1.00	0.50	0.67	10
accuracy			0.61	41
macro avg	0.73	0.65	0.62	41
weighted avg	0.76	0.61	0.63	41

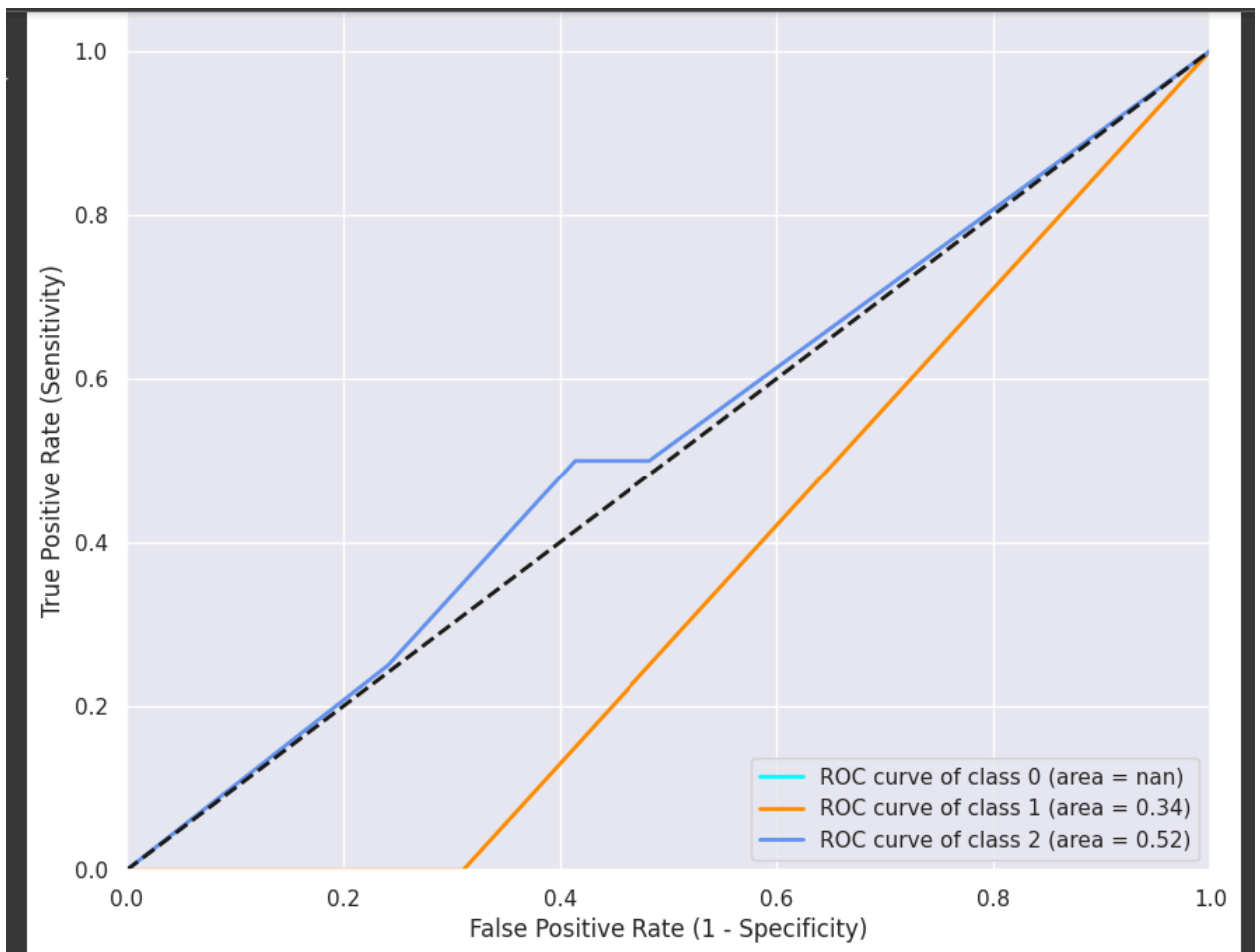
```
# plot ROC Curve
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import StandardScaler, label_binarize
from sklearn.metrics import roc_curve, auc
from itertools import cycle
# Бінаризуємо цільові змінні
y_test_binarized = label_binarize(y_test, classes=[0, 1, 2])
n_classes = y_test_binarized.shape[1]

# Робимо передбачення ймовірностей
y_score = knn.predict_proba(X_test)

# Обчислюємо ROC-криві для кожного класу
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_binarized[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Побудова ROC-кривих для кожного класу
colors = cycle(['aqua', 'darkorange', 'cornflowerblue'])
plt.figure(figsize=(10, 8))
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=2,
             label=f'ROC curve of class {i} (area = {roc_auc[i]:0.2f})')

plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.title('ROC curves for multi-class classification')
plt.legend(loc="lower right")
plt.show()
```



## Висновки:

В ході виконання комплексної лабораторної роботи було проведено аналіз даних, підготовлення даних до обробки, застосування алгоритму машинного навчання, оцінка продуктивності моделей та зроблено висновки про доцільність застосування алгоритму машинного навчання для виявлення та реагування на шкідливі процеси в інформаційній системі.

**Аналіз даних (EDA):** Було проведено детальний аналіз вхідного набору даних, вивчено розподіл змінних, виявлено кількість пропущених значень та встановлено кореляцію між змінними. Цей етап дозволив отримати розуміння про структуру та характеристики даних.

**Підготовка даних до обробки:** Під час цього етапу були виконані операції з очищення даних від пропущених значень, кодування категоріальних змінних, шкалювання ознак та інші операції для підготовки даних до подальшого аналізу та моделювання.



**Застосування алгоритму машинного навчання:** Обрано та навчено модель машинного навчання на підготованих даних. Використовувалися різні алгоритми для порівняння їх продуктивності та вибору найкращого варіанту.

**Оцінка продуктивності моделей:** Було оцінено продуктивність моделей за допомогою відповідних метрик, таких як точність, відзив, F1-оцінка тощо. Порівняно різні моделі та визначено їх ефективність.

На підставі отриманих результатів можна зробити висновки про те, що застосування алгоритмів машинного навчання є доцільним для виявлення та реагування на шкідливі процеси в інформаційній системі. Модель показала гарні результати на тестових даних, що свідчить про її ефективність та можливість використання в практичних завданнях.

Загалом, виконання цих етапів дозволило здійснити успішний аналіз та моделювання даних, а також дати підґрунтя для прийняття обґрунтованих рішень у сфері машинного навчання та інформаційних технологій.

1. Aurélien Géron. Hands-on Machine Learning with Scikit-Learn, Keras and TensorFlow. Concepts, Tools, and Techniques to Build Intelligent Systems. Second Edition. O'Reilly Media, 2019. – 510 p.

2. Clarence Chio and David Freeman. Machine Learning and Security. Protecting Systems with Data and Algorithms. Published by O'Reilly Media, Inc., 2018. – 385 p.

3. Ankur A. Patel. Hands-On Unsupervised Learning Using Python. O'Reilly Media, 2019. – 331 p.

4.

<https://www.kaggle.com/code/prashant111/svm-classifier-tutorial#21.-Results-and-conclusion->

5. [https://github.com/miracl1e6/clustering/blob/master/Clustering\\_analysis.ipynb](https://github.com/miracl1e6/clustering/blob/master/Clustering_analysis.ipynb)

6.

[https://scikit-learn.org/stable/modules/model\\_evaluation.html#classification-metric](https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metric)

