

Міністерство освіти і науки України  
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту  
(повна назва)

Кафедра Інформатики  
(повна назва)

**КВАЛІФІКАЦІЙНА РОБОТА**  
**Пояснювальна записка**

рівень вищої освіти другий (магістерський)

**ДОСЛІДЖЕННЯ І ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ**  
**РЕАЛІЗАЦІЇ ТРАФІКІВ ДЛЯ ДЕТЕКТУВАННЯ DDOS-АТАК**  
(тема)

Виконав:  
студент 2 курсу, групи ІНФМ-20-1

Гонтар С.О.  
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки  
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика  
(повна назва освітньої програми)

Керівник проф. Кіріченко Л.О.  
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри \_\_\_\_\_  
(підпис)

Кобилін О.А.  
(прізвище, ініціали)

2021 р.

## Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту  
(повна назва)Кафедра Інформатики  
(повна назва)Рівень вищої освіти другий (магістерський)Спеціальність 122 Комп'ютерні науки  
(код і повна назва)Тип програми освітньо-професійнаОсвітня програма Інформатика  
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:

Зав. кафедри \_\_\_\_\_  
(підпис)

«\_\_\_\_» \_\_\_\_\_ 2021 р.

**ЗАВДАННЯ**  
НА КВАЛІФІКАЦІЙНУ РОБОТУстудентові Гонтарю Сергію Олеговичу  
(прізвище, ім'я, по батькові)1. Тема роботи Дослідження і порівняння методів класифікації реалізацій трафіків для детектування DDos-атак

затверджена наказом по університету від « 22 » жовтня 2021 року № 1574Ст.

2. Термін подання студентом роботи до екзаменаційної комісії 14 грудня 2021 р.3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, теоретичні відомості про методи кластеризації, мова програмування Python.

4. Перелік питань, що потрібно опрацювати в роботі \_\_\_\_\_

1. Проаналізувати предметну область.

2. Здійснити класифікацію DDOS-атак.

3. Проаналізувати існуючі методи класифікації.

4. Здійснити вибір програмних засобів для реалізації алгоритмів.

5. Програмно реалізувати класифікатор DDOS-атаки.

6. Протестувати розроблений застосунок та провести аналіз результатів.

7. Виявити перспективи подальшої роботи.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність дослідження, об'єкт та мета дослідження, постановка задачі дослідження, вихідні дані дослідження, результати тестування, висновки, перспективи подальших досліджень

---



---



---



---

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Бєлова Н.В.		

### КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	22.10.2021	
2	Аналіз завдання, підбір літератури	07.11.21-11.11.21	
3	Аналіз літератури з досліджуваної проблеми	11.11.21-14.11.21	
4	Аналіз методів класифікації	15.11.21-20.11.21	
5	Програмна реалізація	21.11.21-30.11.21	
6	Оформлення пояснювальної записки	01.11.21-14.11.21	
7	Перевірка на плагіат	14.12.2021	
8	Рецензування	14.12.2021	
9	Підготовка презентації та доповіді	14.12.2021	
10	Занесення роботи в електронний архів	15.12.2021	
11	Попередній захист кваліфікаційної роботи	15.12.2021	

Дата видачі завдання 22 жовтня 2021 р.

Студент \_\_\_\_\_  
(підпис)

Керівник роботи \_\_\_\_\_ проф. Кіріченко Л.О.  
(підпис) (посада, прізвище, ініціали)

## РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 53 с., 26 рис., 33 джерел.

ТРАФІК, DDOS-АТАКА, КЛАСИФІКАЦІЯ, МАШИННЕ НАВЧАННЯ, НЕЙРОННА МЕРЕЖА, ПЕРЦЕПТРОН, ДЕРЕВА РІШЕНЬ, ЛОГІСТИЧНА РЕГРЕСІЯ, ПРОТОКОЛИ, ВИБІРКА, PYTHON.

Об'єктом дослідження є набір атакованого трафіку.

Метою дослідження є вивчення та реалізація методів класифікації для виявлення DDoS-атак.

У ході виконання кваліфікаційної роботи продемонстровано точність результатів експерименту щодо 4 алгоритмів для 3 різних вибірок із датасету.

Програма може використовуватись для подальшої реалізації та інтеграції в інші програмні компоненти чи модулі, що після цього дасть змогу використовувати побудовані моделі для класифікації.

TRAFFIC, DDOS-ATTACK, CLASSIFICATION, MACHINE LEARNING, NEURAL NETWORK, PERCEPTRON, DECISION TREE, LOGISTIC REGRESSION, PROTOCOLS, SAMPLE, PYTHON.

The object of research is a set of attacked traffic.

The aim of the study is to study and implement classification methods to detect DDoS attacks.

During the qualification work, the accuracy of the experimental results on 4 algorithms for 3 different samples from the dataset was demonstrated.

The program can be used for further implementation and integration into other software components or modules, which will then allow you to use the built models for classification.

## ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів	6
Вступ	7
1 Аналіз предметної області та постановка задачі дослідження	8
1.1 Аналіз предметної області	8
1.2 Класифікація DDoS-атак за OSI	11
1.2.1 SYN-флуд	14
1.2.2 UDP -флуд	15
1.2.3 HTTP-флуд	17
1.2.3 ICMP-флуд	18
1.3 Постановка задачі дослідження	20
2 Методи класифікації	22
2.1 Класифікація за допомогою kNN	22
2.2 Класифікація за допомогою дерев рішень	24
2.3 Класифікація за допомогою нейронних мереж	25
2.4 Класифікація за допомогою логістичної регресії	29
3 Реалізація методу класифікації для виявлення DDoS-атак	31
3.1 Вибір програмних засобів	31
3.1.1 Огляд мови Python та допоміжних бібліотек	32
3.1.2 Середовище програмної реалізації	34
3.2 Програмна реалізація для класифікації DDoS-атак	35
3.3 Тестування програми та аналіз результатів	40
3.4 Перспективи подальшої роботи	47
Висновки	48
Перелік джерел посилання	49

## **ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ**

DDoS – Distributed Denial-of-Service

LR – Logistic Regression

ML – Machine Learning

MLP – Multilayered Perceptron

OSI – Open Systems Interconnection

TCP – Transmission Control Protocol

UDP – User Datagram Protocol

ICMP – Internet Control Message Protocol

ІС – інформаційна система

НМ – нейронна мережа

ІІ – штучний інтелект

## ВСТУП

Зі стрімким розвитком інформаційних технологій все більше з'являються потреби у захисті будь-яких комп'ютерних систем. Джерела небезпеки для них можуть бути абсолютно різні. Деякі джерела являються непередбачуваними та час їх виникнення складно спрогнозувати, тому з ними досить складно боротися. До таких потенційних джерел небезпеки для будь-яких інформаційних систем можна віднести DDoS-атаки [1].

DDoS-атака – це напад на комп'ютерну систему з наміром зробити комп'ютерні ресурси недоступними користувачам, для яких комп'ютерна система була призначена. Під час неї відбуваються аномальні кількості запитів до системи в наслідок чого це може призвести то її збою та відмови в обслуговуванні.

Головним джерелом даних, яке несе будь-яку інформацією є трафік [2-5]. Трафік представляє собою набір даних з певним об'ємом, яких надходить до системи за певний час. Проблемою є те, що такий трафік можна штучно збільшити. Саме це є основою нападу хакерами на інформаційну систему та є суттю DDoS-атаки.

В сучасних системах захист від атак виконується на рівні програмного забезпечення або апаратного.

Актуальність дослідження полягає у аналізі існуючих методів захисту від DDoS-атак та розробці системи для їх детектування для комп'ютерних систем чи модулів.

Виконавши всі поставлені цілі, буде розроблено програмний модуль для детектування DDoS-атак. Ця система на основі вхідного трафіку відносить його до одного із двох варіантів: трафік або атакований, або – ні.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

## 1.1 Аналіз предметної області

В наші часи онлайн комунікація між різними об'єктами чи суб'єктами є невід'ємною частиною життя. Більша частина людей активно користується різними онлайн сервісами, сайтами та іншими інформаційними ресурсами в повсякденному житті. Існує велика конкуренція між інтернет ресурсами та сервісами, оскільки в інформаційній галузі все більше з'являється грошей. Така конкуренція може призвести до незаконної конкуренції. Одними із методами усунення конкурентів, або просто здійснення неправомірних дій по тим чи іншим причинам є здійснення DDoS-атак.

Популярними жертвами таких атак зазвичай є комерційні, державні та інформаційні сайти. Хакери, наприклад, останнім часом використовують такий вид атаки з метою вимагання грошей за припинення атаки. Тому DDoS-атаки є активним інструментом конкурентної боротьби, а також інформаційних воєн. На рисунку 1.1 наведено динаміку потужності DDoS-атаки для сайту GitHub [6]. Вертикаль відображає об'єм трафіку, горизонталь відображає рік фіксації.

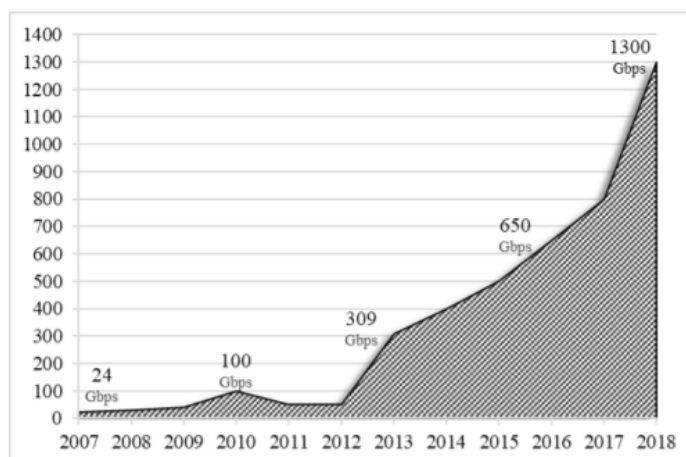


Рисунок 1.1 – Статистика потужності DDoS-атаки



Якщо брати більш «свіжі» дані, то стає все досить цікаво, особливо з початком «нового етапу» [7]. З початку 2020 року через пандемію COVID-2019 життя майже повністю перемістилося в Інтернет – люди по всьому світу зараз працюють, навчаються, роблять покупки та розважаються онлайн, як ніколи раніше. Це знайшло відображення в цілях останніх DDoS-атак, причому найбільш цільовими ресурсами в першому кварталі були веб-сайти медичних організацій, служб доставки, ігрових та освітніх платформ.

В 2020 році в США відбулися президентські вибори, а підготовка до них, як завжди, супроводжувалася DDoS-атаками. Наприклад, на початку лютого 2020 відбулася атака. Зловмисники використовували техніку PRSD (псевдовипадкова атака на субдомен), щоб відправити численні запити до неіснуючих субдоменів сайту. Однак спроба DDoS провалилася: ресурс був захищений від атак такого роду.

Не обійшли стороною і фінансові установи. У лютому 2020 криптовалютні біржі Bitfindex зазнали складних та тривалих DDoS-атак. Біржа зупинила свою роботу на деякий час, а саме декілька годин. За словами керівництва Bitfinex, це було необхідно для встановлення спеціалізованого захисту. Невідомо, чи були інциденти схожими чи пов'язаними.

З початку 2020 року домінувала пандемія коронавірусу, яка сколихнула багато речей у світі, включаючи ринок DDoS. У першому кварталі 2020 року спостерігалось значне збільшення як кількості, так і якості DDoS-атак. Кількість атак у 1 кварталі 2020 року збільшилася на 80% порівняно з 1 кварталом 2019 року. Атаки також стали довшими: спостерігалось зростання як середньої, так і максимальної тривалості. У першому кварталі кожного року спостерігається певний сплеск активності DDoS (рис. 1.2).

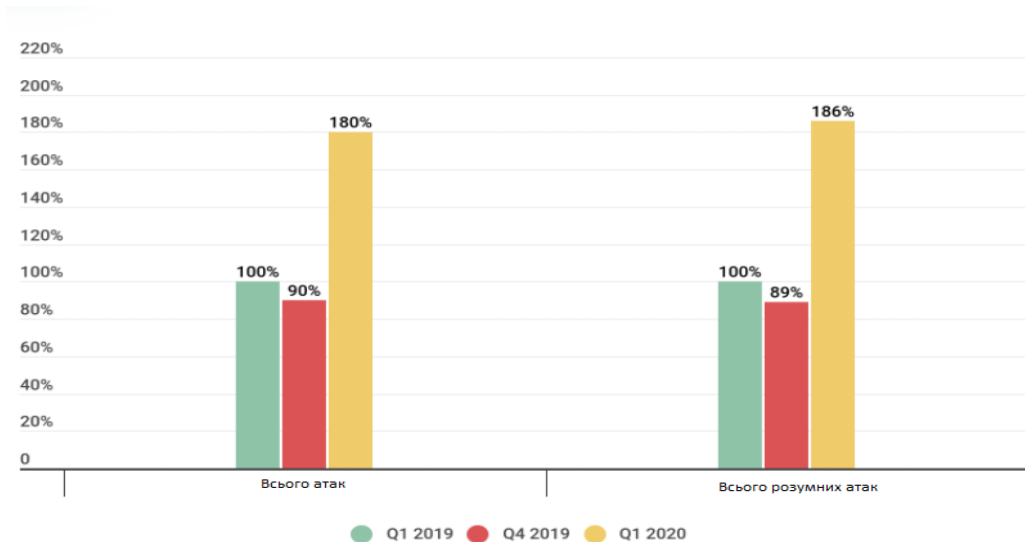


Рисунок 1.2 – Порівняльна статистика кількості атак

Також є цікавою статистика тривалості DDoS-нападів. В цілому загальний час атаки збільшився приблизно на 30%, якщо порівнювати перші квартали 2019 та 2020 років. Також неймовірним показником є збільшення максимальної тривалості атак. Це може свідчити про те, що пандемія сильно збільшила ринок онлайн-сервісів та послуг, тим самим збільшивши й потенційну кількість розумних та тривалих атак (рис. 1.3).

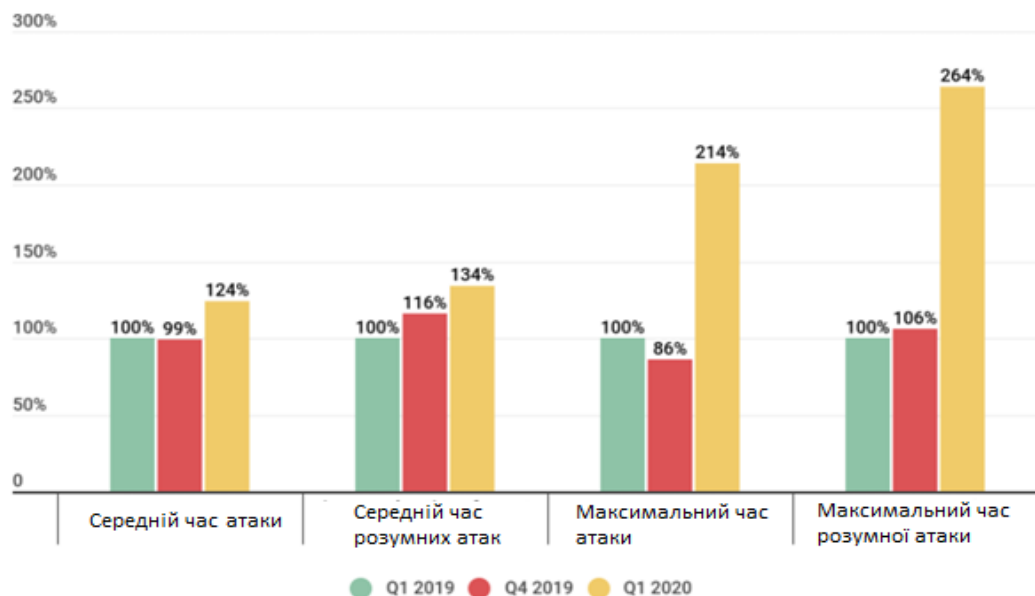


Рисунок 1.3 – Порівняльна статистика середнього часу атак

Що стосується найрозповсюджених IP-адрес по країнам, то важко сказати від чого залежить вибір країн, але вони мають такий розподіл як на рисунку 1.4.

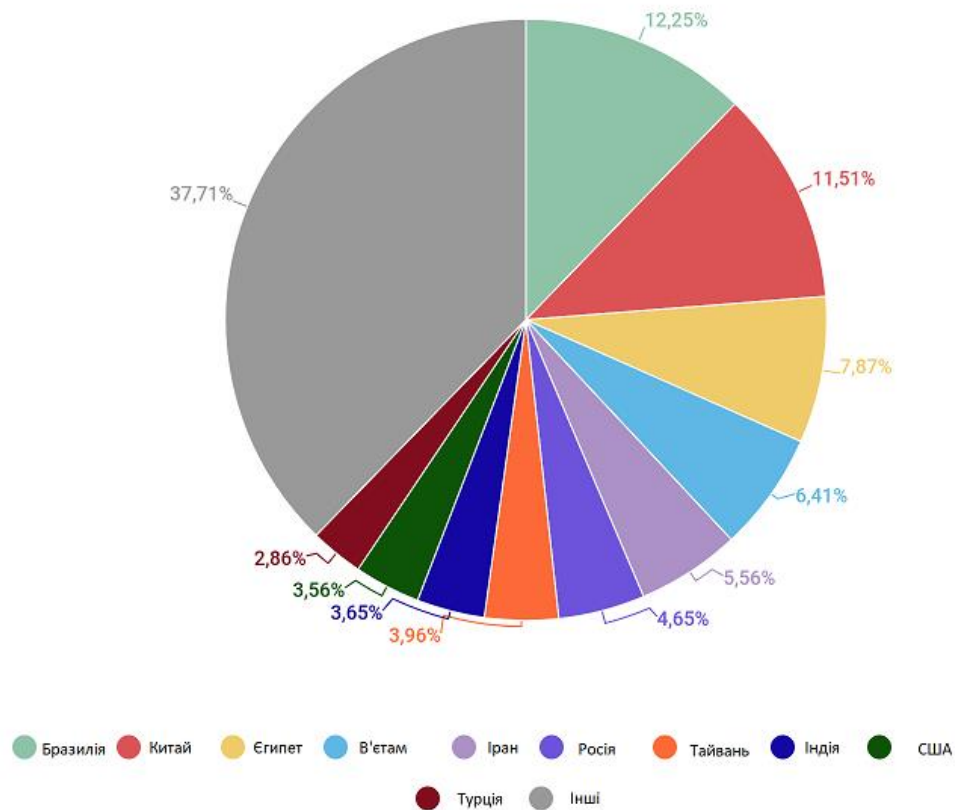


Рисунок 1.4 – Статистика унікальних IP-адрес, що використовувалися в атаках

## 1.2 Класифікація DDoS-атак за OSI

Класифікувати DDoS можна за різними критеріями [8]. Для цього використати такий опис. Наприклад, напади можна класифікувати за рівнем в моделі OSI [9]:

- фізичний;
- канальний;

- мережевий;
- транспортний;
- сеансовий;
- представлення;
- прикладний.

Також можна класифікувати DDoS-атаку за архітектурою:

- рефлектор;
- на основі Web;
- агент-обробник.

Існують варіант класифікація за типом сканування. Вона також може бути використана для опису атаки:

- випадково;
- списки;
- локальні підмережі;
- перестановочні.

Що стосується адреси, звідки відбуваються атаки, то їх класифікують на 2 глобальні класи, не дійсні джерела більш розповсюджені ніж реальні адреси:

- дійсна;
- підроблена, випадкова;
- підроблена, підмережа;
- підроблена, на маршруті.

Оскільки атака може бути направлена на різні види ресурсів, то існує й класифікація за типом жертв:

- застосунок;
- хост;
- мережа;
- інфраструктура.

Всі ці класифікації можуть більше детально описати вид атаки, що може забезпечити її подальше усунення, як мінімум це дасть змогу швидше вирішити проблему та зрозуміти її джерело.

Звернемо увагу на класифікацію DDoS-нападів за моделлю OSI. Майже на кожному рівні існують певні протоколи, що по суті і є трафіком. Тому слід звернути увагу на ті рівні, які більш за всіх використовуються в атаках.

Найбільших ризиків набули такі протоколи як:

- транспортний;
- прикладний;
- мережевий.

Стосовно цього існує статистика за 2020 рік, яка відображена на рисунку 1.5.

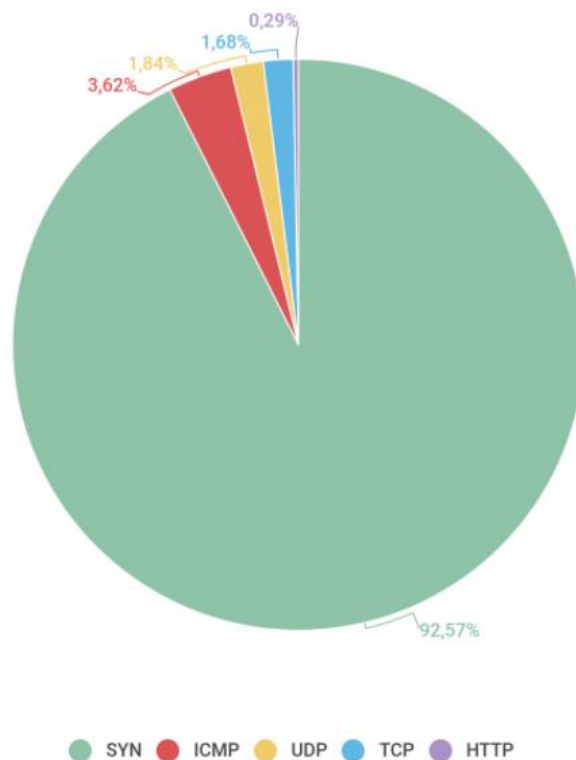


Рисунок 1.5 – Розподіл кількості атак за типом протоколу та флагами

Оскільки SYN – це один із флагів TCP-протоколу, то транспортний рівень моделі OSI найбільш уразливий. Після нього йде протокол ICMP, що використовується на мережевому рівні. Протоколи TCP та UDP відносяться до транспортного рівня. Що стосується останнього за значенням протоколу

HTTP, то він входить до самого верхнього рівня моделі OSI – прикладний рівень.

### 1.2.1 SYN-флуд

SYN-flood – це тип атаки відмови в обслуговуванні, яка спрямована на те, щоб зробити сервер недоступним для законного трафіку, споживаючи всі доступні ресурси сервера [10]. Повторно надсилаючи пакети початкового запиту на з'єднання (SYN), зловмисник може перевантажити всі доступні порти на цільовій серверній машині, змушуючи цільовий пристрій повільно працювати або взагалі не реагувати на трафік.

Така атака працює за алгоритмом:

- 1) спочатку клієнт надсилає SYN-пакет на сервер, щоб ініціювати з'єднання;
- 2) потім сервер відповідає на цей початковий пакет пакетом SYN/ACK, щоб підтвердити зв'язок;
- 3) нарешті, клієнт повертає пакет ACK, щоб підтвердити отримання пакета від сервера. Після завершення цієї послідовності надсилання та отримання пакетів TCP-з'єднання відкрито і може надсилати та отримувати дані;
- 4) далі зловмисник надсилає на цільовий сервер велику кількість пакетів SYN, часто з підробленими IP-адресам;
- 5) потім сервер відповідає на кожен із запитів на підключення і залишає відкритий порт, готовий отримати відповідь;
- 6) поки сервер чекає останній пакет ACK, який так і не надходить, зловмисник продовжує надсилати нові пакети SYN. Надходження кожного нового пакету SYN змушує сервер тимчасово підтримувати нове з'єднання з відкритим портом протягом певного періоду часу і після того, як усі доступні порти будуть використані, сервер не може нормально функціонувати.

Загальна схема зображена на рисунку 1.6. Зловмисник використовує не зачинені з'єднання та надсилає несправжні пакети до серверу.

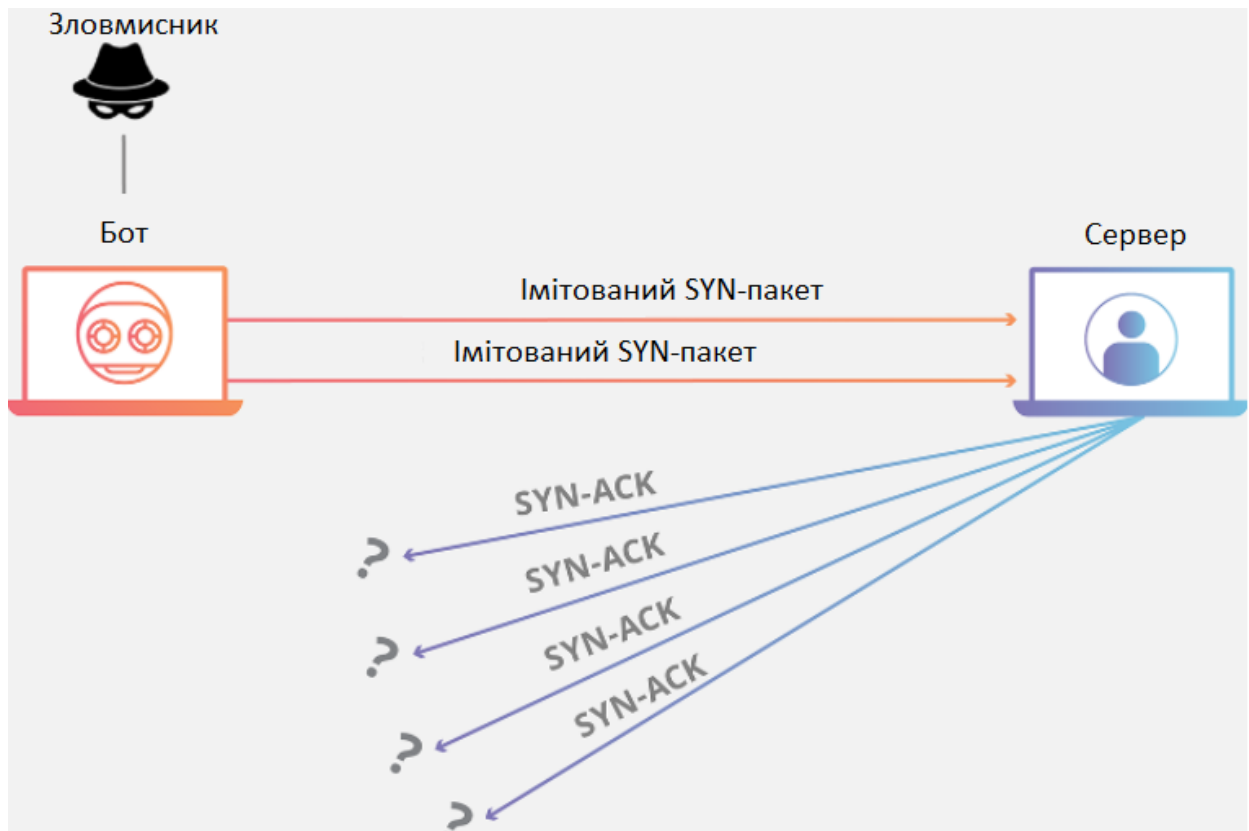


Рисунок 1.6 – Схема SYN-флуд атаки

### 1.2.2 UDP-флуд

UDP-флуд – це тип атаки відмови в обслуговуванні, при якій зловмисник переповнює випадкові порти цільового хосту IP-пакетами, що містять датаграми UDP [11]. Приймаючий хост перевіряє програми, пов'язані з цими датаграмами, і, не знайшовши жодної, надсилає назад пакет «Destination Unreachable». Оскільки все більше і більше пакетів UDP отримується і відправляється, система стає перевантаженою і не реагує на інших клієнтів. У рамках атаки UDP-флуд зловмисник також може підробити IP-адресу пакетів, щоб переконатися, що ICMP-пакети не надходять до свого хоста, і щоб бути анонімним під час атаки. Існує ряд комерційно доступних програмних пакетів,

які можна використовувати для виконання атаки UDP-флуд (наприклад, UDP Unicorn).

Потік UDP працює насамперед, використовуючи кроки, які виконує сервер, коли відповідає на пакет UDP, надісланий на один із його портів. У звичайних умовах, коли сервер отримує пакет UDP на певному порту, він проходить два кроки у відповідь:

- 1) сервер спочатку перевіряє, чи запущені програми, які в даний момент прослуховують запити на вказаному порту;
- 2) якщо жодна програма не отримує пакети на цьому порту, сервер відповідає пакетом ICMP, щоб повідомити відправника про те, що адресат недоступний.

У результаті того, що цільовий сервер використовує ресурси для перевірки та відповіді на кожен отриманий UDP-пакет, ресурси серверу можуть швидко вичерпатися при отриманні великого потоку UDP-пакетів, що призведе до відмови в обслуговуванні звичайного трафіку. Схематично відображено на рисунку 1.7.

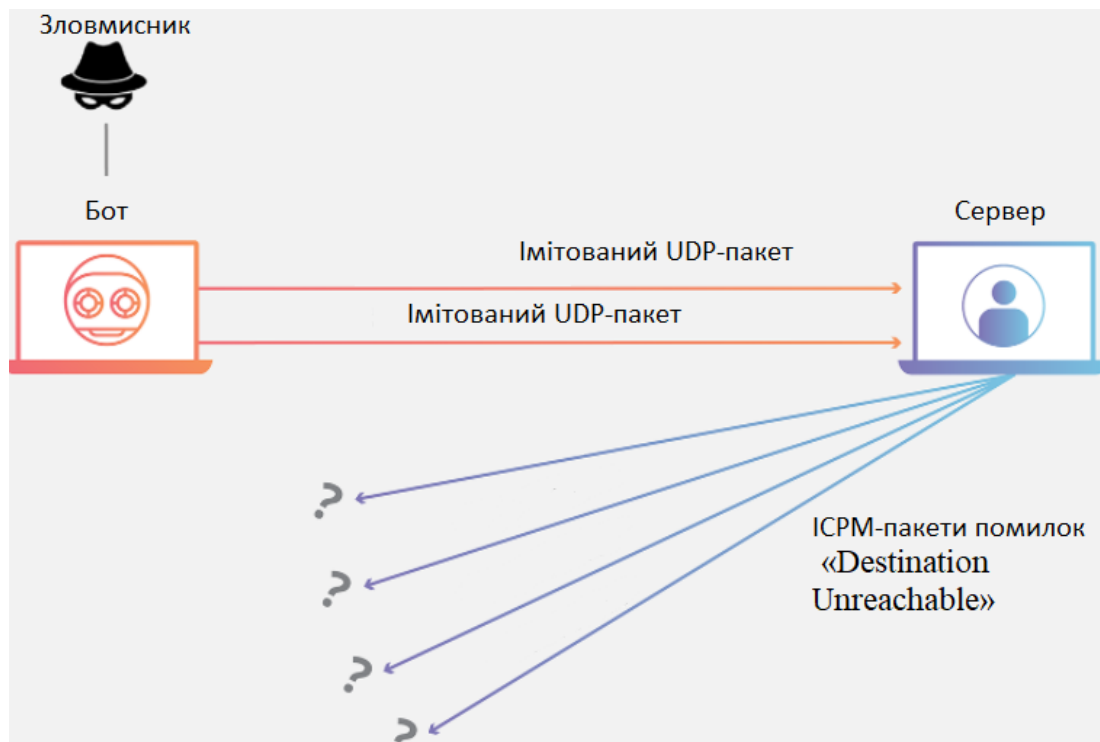


Рисунок 1.7 – Схема UDP-флуд атаки



### 1.2.3 HTTP-флуд

HTTP-флуд – це тип атаки відмови в обслуговуванні, призначеної для перевантаження цільового сервера HTTP-запитами [12]. Після того, як ціль буде насичена запитами і не зможе відповідати на звичайний трафік, відбудеться відмова в обслуговуванні для додаткових запитів від реальних користувачів.

HTTP flood атаки є різновидом DDoS-атаки верхнього рівня моделі OSI. Сьомий рівень є прикладним рівнем і відноситься до Інтернет-протоколів, таких як HTTP. HTTP є основою інтернет-запитів на основі браузера і зазвичай використовується для завантаження веб-сторінок або надсилання вмісту форм через Інтернет. Пом'якшення атак прикладного рівня є особливо складним, оскільки шкідливий трафік важко відрізнити від звичайного трафіку.

Існує два різновиди HTTP-флуд-атак:

1) атака HTTP-get – у цій формі атаки кілька комп'ютерів або інших пристроїв координуються для надсилання кількох запитів на зображення, файли чи інший актив із цільового сервера. Коли ціль переповнена вхідними запитами та відповідями, відмова в обслуговуванні буде відбуватися на додаткові запити від реальних джерел трафіку, тому користувачі не зможуть отримати ресурси сайту;

2) атака HTTP-post – атака, коли дані надсилаються на веб-сайті, сервер повинен обробляти вхідний запит і передавати дані на рівень збереження, найчастіше базу даних. Процес обробки даних та виконання необхідних команд бази даних є відносно інтенсивним порівняно з обсягом обчислювальної потужності та пропускної здатності, необхідної для відправки запиту POST. Ця атака використовує нерівність у відносному споживанні ресурсів, надсилаючи багато запитів на адреси безпосередньо на цільовий сервер, поки його потужність не буде насичена, то відмова в обслуговуванні не буде відбуватися.

Загальна схема такої атаки може виглядати так (рис. 1.8):

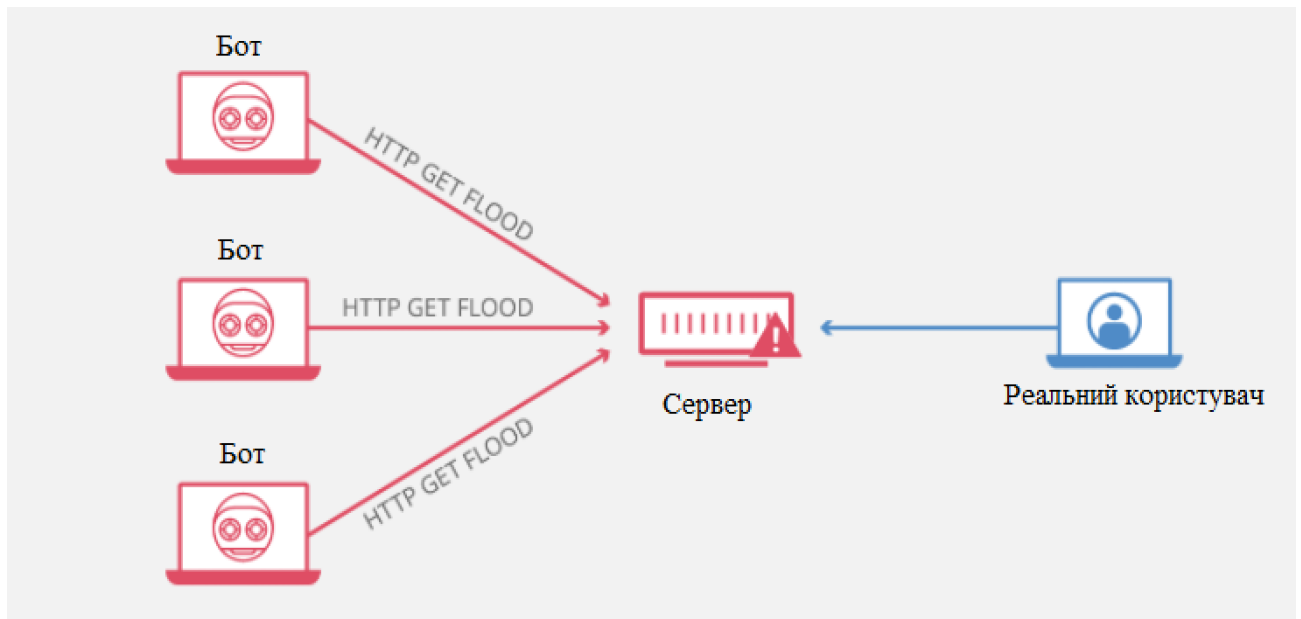


Рисунок 1.8 – Схема HTTP-флуд атаки

Щоб досягти максимальної ефективності, зловмисники зазвичай використовують або створюють ботнети, щоб максимізувати вплив своєї атаки. Використовуючи багато пристроїв, заражених вірусами комп'ютерами, зловмисник може використовувати свої зусилля, запускаючи більший обсяг атаки.

#### 1.2.4 ICMP-флуд

Протокол ICMP, який використовується в атаці Ping Flood, є протоколом мережевого рівня в моделі OSI, який використовується мережевими пристроями для зв'язку. Інструменти діагностики мережі Traceroute і Ping працюють за допомогою ICMP [13].

Зазвичай повідомлення ICMP з ехо-запитом та ехо-відповіддю використовуються для Ping мережевого пристрою з метою діагностики

працездатності та підключення пристрою та з'єднання між відправником і пристроєм.

Запит ICMP вимагає певних ресурсів сервера для обробки кожного запиту та надсилання відповіді. Запит також вимагає пропускну здатності як для вхідного повідомлення (ехо-запит), так і для вихідної відповіді (ехо-відповідь).

Атака Ping Flood має на меті придушити здатність цільового пристрою відповідати на велику кількість запитів та перевантажити мережеве з'єднання фіктивним трафіком.

Завдяки тому, що багато пристроїв у ботнеті націлені на ту саму інтернет-власність або компонент інфраструктури із запитами ICMP, трафік атаки значно збільшується, що потенційно може призвести до порушення нормальної діяльності мережі.

Історично зловмисники часто підробляли підроблену IP-адресу, щоб замаскувати пристрій-відправник. При сучасних атаках ботнетів зловмисники нечасто бачать потребу маскувати IP-адресу бота, а замість цього покладаються на велику мережу непідроблених ботів для насичення можливостей цілі.

Цю атаку можна описати в 2 кроки:

- 1) зловмисник надсилає багато пакетів ICMP-запитів на цільовий сервер за допомогою кількох пристроїв;
- 2) цільовий сервер потім надсилає пакет ICMP-відповіді на IP-адресу кожного пристрою, що запитує, як відповідь.

Загальна схема такої атаки може виглядати так (рис. 1.9):

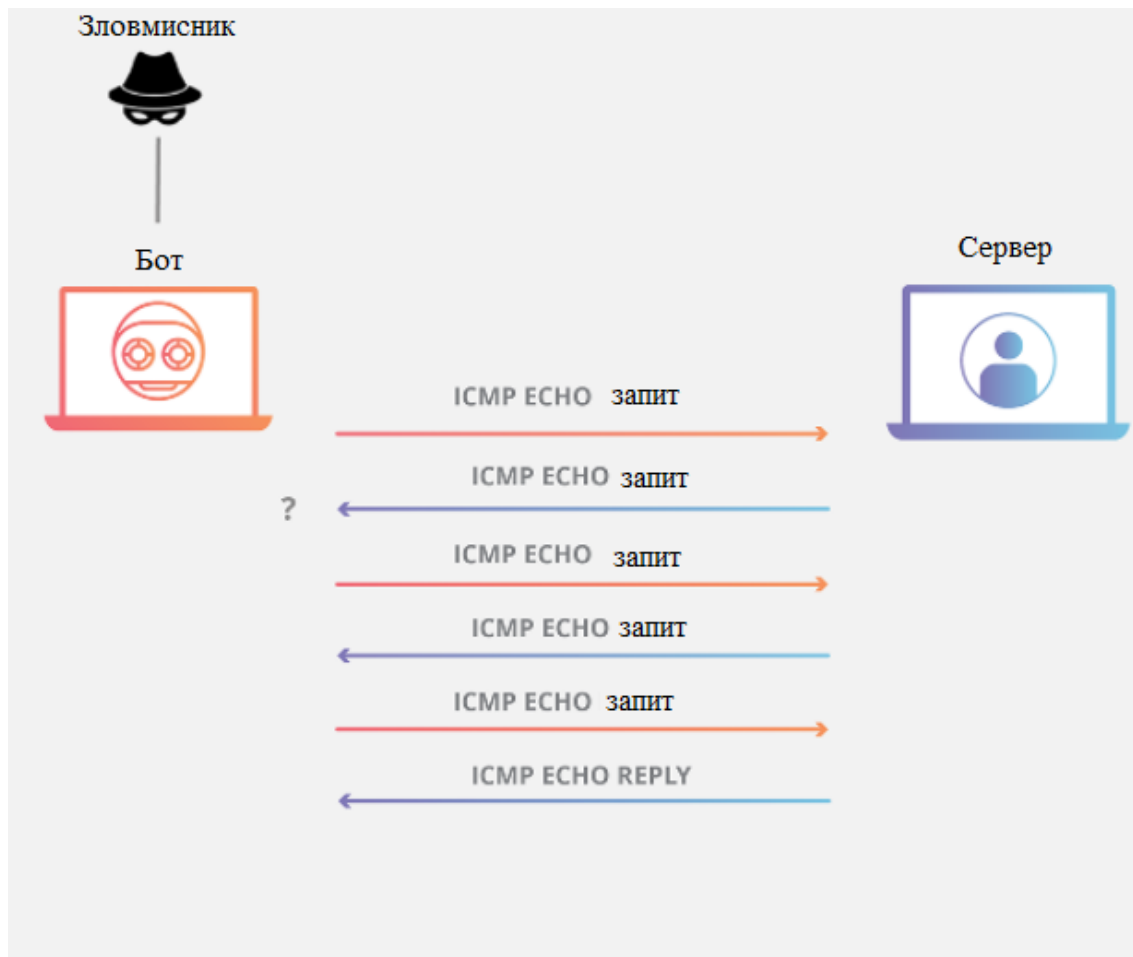


Рисунок 1.9 – Схема ICMP-флуд атаки

### 1.3 Постановка задачі дослідження

Дивлячись на статистику DDoS-атак та на в цілому розвиток цієї індустрії, можна сказати, що дослідження та створення систем, що будуть запобігати цьому та будуть розпізнавати атаки на ранніх стадіях за допомогою алгоритмів машинного навчання, чи нейронних мереж є актуальним рішенням для розв'язку задачі.

Об'єктом дослідження є вибірка даних, яка являє собою трафік.

Метою дослідження є розробка системи детектування, що базуються на використанні алгоритмів машинного навчання та нейронної мережі для класифікації трафіку на атакований або не атакований.

Для досягнення мети необхідно вирішити такі завдання:

- аналіз предметної області;
- аналіз класифікацій DDoS-атак;
- аналіз існуючих методів класифікації;
- аналіз існуючих мов програмування та додаткових бібліотек для використання в реалізації;
- створення системи детектування DDoS-атак на основі обраних інструментів;
- аналіз створеної системи детектування DDoS-атак.

## 2 МЕТОДИ КЛАСИФІКАЦІЇ

Аналіз даних має кількість задач. Однієї з таких задач є класифікація. Класифікація – це процес віднесення об’єктів предметної області до заздалегідь відомих груп [14-18]. Ці групи називаються класами, тому ця задача має таку назву. В випадку реалізації алгоритму об’єктами будуть виступати строки з наборів даних, а значення цих об’єктів – це атрибути в наборі даних.

Для класифікації будується спеціальна модель, яка має назву класифікатор. Класифікатором може виступати як статистичні моделі, так і складні методи машинного навчання [15-21]. Серед таких класифікаторів можуть бути використані:

- нейронні мережі;
- дерева рішень;
- логістична регресія та інші статистичні методи.

### 2.1 Класифікація за допомогою kNN

Одним із найпопулярнішим алгоритмом класифікації є алгоритм k-найближчих сусідів [22]. Його суть полягає в тому, щоб розділити n спостережень на k кластерів, так щоб кожне спостереження належало до кластера з найближчим до нього середнім значенням. Метод базується на мінімізації суми квадратів відстаней між кожним спостереженням та центром його кластера, тобто функції:

$$\sum_{i=1}^N d(x_i, m_j(x_i))^2, \quad (2.1)$$

де  $d$  – метрика;

$x_i$  – це  $i$ -тий об'єкт даних;

$m_j(x_i)$  – центр кластера, якому на  $j$ -тій ітерації приписаний елемент  $x_i$ .

Загальний опис алгоритму такий, що маємо деякий масив об'єктів, кожен з яких має певні значення по атрибутам. Відповідно до цих значень об'єкт розташовується у багатовимірному просторі. Алгоритм дії такий:

- 1) дослідник визначає кількість кластерів, що необхідно утворити. В випадку вирішення задачі детектування будемо мати тільки 2 класи об'єктів – це трафік, що належить до атакованого, та трафік, що не належить до атакованого, тобто він є нормальним;
- 2) випадковим чином обирається  $k$  спостережень, які на цьому кроці вважаються центрами кластерів;
- 3) кожне спостереження «приписується» до одного з  $n$  кластерів – того, відстань до якого найкоротша;
- 4) розраховується новий центр кожного кластера як елемент, ознаки якого розраховуються як середнє арифметичне ознак об'єктів, що входять у цей кластер;
- 5) відбувається така кількість ітерацій (повторюються кроки 3-4), поки кластерні центри стануть стійкими (тобто при кожній ітерації в кожному кластері опинятимуться одні й ті самі об'єкти), дисперсія всередині кластера буде мінімізована, а між кластерами – максимізована.

Принцип алгоритму полягає в пошуку таких центрів кластерів та наборів елементів кожного кластера при наявності деякої функції  $\Phi(^{\circ})$ , що виражає якість поточного розбиття множини на  $k$  кластерів, коли сумарне квадратичне відхилення елементів кластерів від центрів цих кластерів буде найменшим.

$$V = \sum_{i=1}^k \sum_{x_j \in S_i} (x_j - \mu_i)^2, \quad (2.2)$$

де  $k$  – число кластерів;

$S_i$  – отримані кластери;

$i_j(x_i) - i = 1, 2, \dots k, u_i$  – центри мас векторів  $x_j$ .

## 2.2 Класифікація за допомогою дерев рішень

Метод дерев рішень є одним із найбільш популярних методів розв’язання задач класифікації й прогнозування [23]. Іноді цей метод також називають деревами класифікації чи регресії. Як видно з назви, за допомогою дерев може бути й вирішена задача класифікації та прогнозування.

Якщо залежна змінна приймає безперервні значення, то дерево рішень установлює залежність цієї змінної від незалежних змінних та означає розв’язок задачі чисельного прогнозування. Якщо ж залежна, тобто цільова змінна приймає дискретні значення, за допомогою методу дерева рішень розв’язується задача класифікації. У найбільш простому вигляді дерево рішень – це спосіб показу правил в послідовній структурі, що має певну ієрархію. Основа такої структури – відповіді «Так» або «Ні» на низку питань, тому й дерево можна називати бінарним, адже існують тільки 2 стани.

Процес створення дерева відбувається зверху вниз, тобто є низхідним. У ході процесу алгоритм повинен знайти такий критерій розщеплення, щоб розбити множину на підмножини, які б асоціювалися з даним вузлом перевірки. Кожний вузол перевірки повинен бути позначений певним атрибутом. Існує правило вибору атрибута: він повинен розбивати вихідну множину даних таким чином, щоб об’єкти підмножин, що одержуються в результаті цієї розбивки, були представниками одного класу або ж були максимально наближені до такої розбивки.

Існують різні критерії розщеплення. Серед них найбільш відомі – міра ентропії й індекс Gini [24]. У деяких методах для вибору атрибута розщеплення використовується так звана міра інформативності підпросторів атрибутів, яка ґрунтується на підході ентропії.



Інший критерій розщеплення, запропонований Брейманом та ін., реалізований в алгоритмі CART і називається індексом Gini. За допомогою цього індексу атрибут вибирається на підставі відстаней між розподілами класів.

$$gini(T) = 1 - \sum_{j=1}^n p_j^2, \quad (2.3)$$

де  $T$  – поточний узол;

$p_j$  – ймовірність класу  $j$  у вузлі  $T$ ;

$n$  – кількість класів.

Також слід пам'ятати про таке поняття як зупинка. Зупинка – це такий момент, при якому подальша побудова дерева повинна припинитися. Зупинку можна задати по різному. Це можуть бути певні правила, або в якості зупинки може виступати максимальна глибина дерева.

Взагалі існують такі види алгоритмів побудови дерев рішень:

- CART;
- C4.5;
- CHAID;
- C4.5;
- NEWID та інші.

### 2.3 Класифікація за допомогою нейронних мереж

Нейронні мережі – це обчислювальні системи, що будується на принципі функціонування біологічних нейронних мереж [25]. Вони можуть навчатися на наборі даних та таким чином вдосконалюються. Найпоширенішою реалізацією нейронної мережі є перцептрон або персептрон.

Для того, щоб нейронна мережа була здатна виконати поставлене завдання класифікації, її необхідно навчити. Розрізняють алгоритми навчання з учителем та без учителя. Процес навчання з учителем являє собою пред'явленням мережі вибірки навчальних прикладів. Кожен зразок подається на входи мережі, потім проходить обробку всередині структури НМ, обчислюється вихідний сигнал мережі, який порівнюється з відповідним значенням цільового вектору, який являє собою необхідний вихід мережі.

Потім за певним правилом обчислюється помилка і відбувається зміна вагових коефіцієнтів зв'язків у мережі залежно від обраного алгоритму. Вектори навчальної множини пред'являються послідовно, обчислюються помилки і ваги підлаштовуються для кожного вектору, поки помилка по всьому навчального масиву не досягне потрібного низького рівня.

Також нейронні мережі можна класифікувати за кількістю шарів в ній:

- одношарова нейронна мережа;
- багатошарова нейронна мережа.

Розглянемо послідовність кроків під час навчання одношарової нейронної мережі за правилом навчання Уїдроу-Хоффа:

- 1) задаються крок навчання  $\alpha$  ( $0 < \alpha < 1$ ) та бажана середньоквадратична помилка мережі  $E_m$ ;
- 2) ініціалізуються випадковим чином вагові коефіцієнти  $w_{i,j}$  та порогові  $b_j$  значення нейронів;
- 3) подаються послідовно об'єкти з навчальної вибірки на вхід нейронної мережі. Обчислюється вихідні значення нейронів;
- 4) проводиться зміна вагових коефіцієнтів та порогів нейронних

елементів;

- 5) Обчислюється сумарна помилка нейронної мережі  $E$ ;
- 6) Якщо  $E > E_m$ , то відбувається перехід до кроку 3, інакше виконання алгоритму завершується.

На рисунку 2.1 представлена схема реалізації одношарової НМ. При такому варіанті вона має тільки один прихований шар.

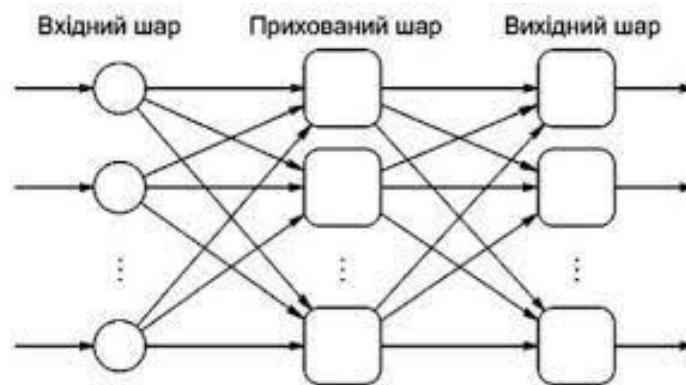


Рисунок 2.1 – Архітектура одношарової нейронної мережі на базі перцептрону

Щоб побудувати багатошаровий перцептрон необхідно вибрати його параметри за наступним алгоритмом [26]:

- 1) обрати вихідний вектор  $Y$  таким чином, щоб його компоненти містили повну відповідь для поставленого завдання;
- 2) обрати тип функції активації нейронів. При цьому бажано врахувати специфіку завдання, оскільки успішний вибір збільшить швидкість навчання.
- 3) обрати кількість шарів та нейронів у шарі;
- 4) задати діапазон зміни входів, виходів, ваг та порогових рівнів на основі вибраної функції активації;
- 5) призначити початкові значення ваги та пороговим рівням. Початкові значення не повинні бути більшими, щоб нейрони не опинилися в насиченні (на горизонтальній ділянці функції активації), інакше навчання буде дуже повільним. Початкові значення не повинні бути занадто малими, щоб

виходи більшої частини нейронів були рівні нулю, інакше навчання теж сповільниться;

б) провести навчання, тобто підібрати параметри мережі так, щоб завдання вирішувалася якнайкраще. Після закінчення навчання мережа зможе вирішувати завдання того типу, яким вона навчена;

7) подати на вхід мережі умови завдання у вигляді вектору  $X$ . Розрахувати вихідний вектор  $Y$ , який дасть формалізоване розв'язання задачі.

На рисунку 2.2 представлена схема реалізації багатошарової нейронної мережі. При такому варіанті можна конфігурувати кількість слоїв між вхідними та вихідними шарами та задавати кількість нейронів в цих шарах.

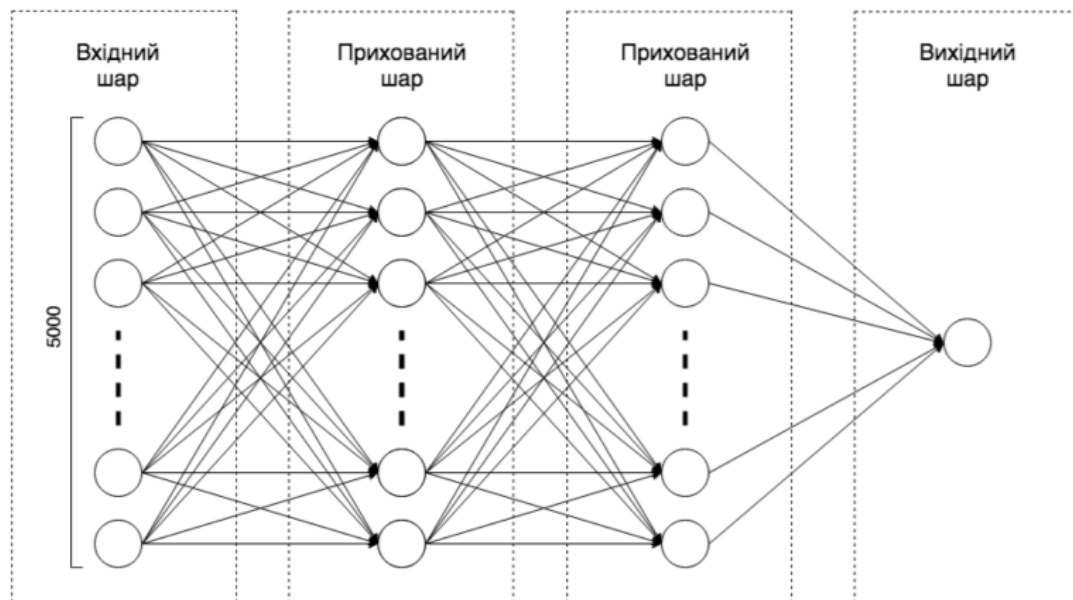


Рисунок 2.2 – Архітектура багатошарової нейронної мережі на базі перцептрону

Вхідний шар являє собою певний набір вхідних даних, який потрібен для навчання. Прихований шар є сукупністю логічних операцій над даними, а вихідний шар – це результат роботи нейронної мережі. Вихідний шар може мати як один так і декілька результатів. Також нейронна мережа може мати декілька прихованих шарів, тоді вона називається багатошаровою.

## 2.4 Класифікація за допомогою логістичної регресії

Логістична регресія або логіт-регресія – це статистичний регресійний метод, що застосовують у випадку, коли залежна змінна є бінарною, тобто може набувати тільки двох значень (0 або 1). При запровадженні порогового значення може знаходити застосування у задачах класифікації [27].

Логістичну модель можна описати так: нехай є деяка випадкова величина  $Y$ , що може набувати лише двох значень, які, як правило, позначаються цифрами 0 і 1. Нехай ця величина залежить від деякої множини пояснювальних змінних  $x = (1, x_1, \dots, x_n)^T$ . При цьому залежність  $Y$ , від  $x_1, \dots, x_n$  можна визначити ввівши додаткову змінну  $y^*$ . При цьому:

$$y^* = \theta^T x = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n + \varepsilon. \quad (2.4)$$

Тоді ми маємо:

$$Y = \begin{cases} 0, & y^* \leq 0 \\ 1, & y^* > 0 \end{cases}. \quad (2.5)$$

При визначенні логістичної моделі стохастичний доданок  $\varepsilon$  вважається випадковою величиною з логістичним розподілом ймовірностей. Відповідно для певних конкретних значень змінних  $x^* = x_1^*, \dots, x_n^*$  одержується відповідне значення  $y^*$  ймовірність того, що  $Y = 1$  така:

$$p(Y = 1) = p(y^* > 0) = p(\theta^T x^* + \varepsilon > 0) = p(\varepsilon > -\theta^T x^*) = p(\varepsilon \leq \theta^T x^*) = \Lambda(\theta^T x^*). \quad (2.6)$$

Передостання рівність впливає з симетричності логістичного розподілу,  $\Lambda$  позначає логістичну функцію – функцію розподілу логістичного розподілу:

$$\Lambda(x) = \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}}. \quad (2.7)$$

При цьому логіт-модель задовольняє наступній умові:

$$\ln \frac{p(1|X)}{1 - p(1|X)} = \ln \frac{p(1|X)}{p(0|X)} = b_0 + b_1 x_1 + \dots + b_J x_J. \quad (2.8)$$

Модель регресійного аналізу може виглядати так для випадкових точок даних (рис. 2.3).

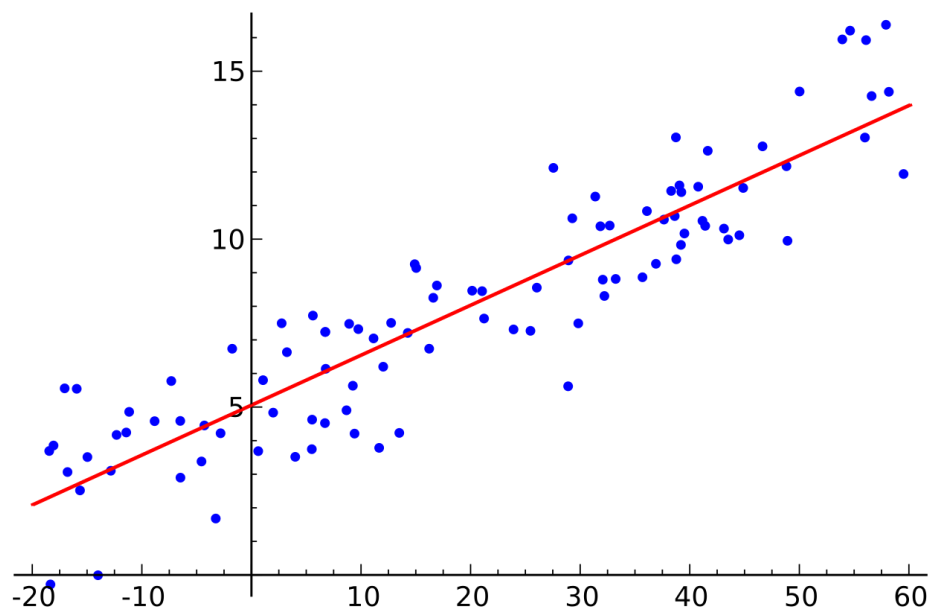


Рисунок 2.3 – Випадкові точки даних та їх лінійна регресія

### **3 РЕАЛІЗАЦІЯ МЕТОДУ КЛАСИФІКАЦІЇ ДЛЯ ВИЯВЛЕННЯ DDOS-АТАК**

#### **3.1 Вибір програмних засобів**

Для правильного вибору мови програмування було створені певні критерії відбору цих мов, а саме:

- напрямки використання;
- власна компетенція;
- існуючі знання в рамках мови програмування;
- наявність необхідних бібліотек для реалізації алгоритмів.

Проаналізувавши існуючі мови програмування за цими критеріями, було вирішено, що для успішної реалізації поставленої задачі можуть бути використані такі мови програмування як C# або Python.

Великою перевагою мови Python над мовою C# є те, що перша з них має велику базу корисних та потужних бібліотек пов'язаних за машинним навчанням, нейронними мережами та іншими сферами ШІ. В цих бібліотеках реалізуються різні алгоритми, які спрощують розробку та потребують переписування цих алгоритмів «з нуля». Також ці бібліотеки мають досить гарну та зрозумілу документацію, що дасть змогу налаштувати алгоритм так, як це потрібно розробнику.

Ще для демонстрації результатів роботи на своєму «борті» Python має декілька зручних бібліотек відрисовки даних. Результати можна відобразити в різному виді, будь то діаграми, гістограми чи різноманітні графіки. А відображення цих графіків можна закодувати в декілька строк, що є досить зручним інструментом для розробника.

Враховуючи всі переваги, саме мову програмування Python було обрано в якості мови для розробки програми.

### 3.1.1 Огляд мови Python та допоміжних бібліотек

Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою [28]. Структури даних високого рівня та використання динамічної семантики і зв'язуванням забезпечую дуже швидко розробку програм, порівняно із іншими мовами програмування високого рівня.

До основних аспектів мови можна віднести такі якості мови:

- використання чистого синтаксису, що означає важливість відступів в логічних блоках програмного коду;
- наявність корисних модулів в стандартному дистрибутиві;
- потужна база для роботи з різними числами, наприклад, комплексними числами;
- можливість використання в діалоговому режимі, що прискорює отримання прототипів програмного продукту.

Також, слід звернути увагу на бібліотеки, що можуть бути використані в розробці. Це теж є основоположною частиною під час розробки, адже наявністю таких бібліотек є великою перевагою над іншими мовами, в тому числі й С#. До таких бібліотек можна віднести:

- NumPy;
- Pandas;
- Sys;
- Pickle;
- Sklearn.

NumPy – це відкритий модуль для Python, який надає загальні математичні та числові операції у вигляді пре-компільованих, швидких функцій. Вони поєднуються у пакети високого рівня та можуть бути підключені до програми. Вони забезпечують функціонал, який можна порівняти із функціоналом MatLab. NumPy надає базові методи для маніпуляції з великими масивами та матрицями. Цю бібліотеку можна



вважати еталоном для роботи з даними та структурами, що були вказані вище. Також існує бібліотека розширення під назвою SciPy. Вона розширює функціонал NumPy величезною колекцією корисних алгоритмів, таких як мінімізація та максимізація, регресія, перетворення Фур'є та інші математичні перетворення та алгоритми. В роботі необхідність використання SciPy є мінімальною, адже таких операцій не буде використовуватися.

Pandas – це відкритий модуль для Python, яких слугує для аналізу даних. У порівнянні з NumPy, бібліотека знаходиться на більш високому рівні, адже вона побудована поверх цього ж NumPy. В глобальному використанні, Pandas є найбільш просунутою бібліотекою, що швидко розвивається, для обробки та аналізу даних. Бібліотека містить спеціальні структури даних DataFrame та Series, які є основою у використанні. Цю бібліотеку в програмному коді буде використано для зчитування даних з файлів розширення .csv або інших.

Sys – додатковий модуль, що забезпечує доступ до деяких змінних та функцій, що взаємодіють з інтерпретатором Python. Цей модуль зручно використовувати для задання параметрів під час роботи та тестування алгоритмів. Параметри передаються в консолі через послідовний набір конфігурацій, після чого, використовуючи методи бібліотеки, ці параметри можна зчитати та діяти згідно їх значення. Досить зручна бібліотека на етапах тестування, що прискорює розробку.

Модуль Pickle реалізує потужний алгоритм серіалізації та десеріалізації об'єктів Python. В рамках бібліотеки є поняття «pickling», що означає серіалізацію. Це такий собі процес перетворення об'єкта Python в потік байтів, а зворотній процес в результаті якої потік байтів перетворюється назад на Python-об'єкт називається «unpickling». Ці процеси використовуються для запису та зчитування складних об'єктів до файлів. Це досить корисна річ. Наприклад, можна її використовувати для збереження створених моделей під час роботи програми.

Для використання алгоритмів було обрано бібліотеку Sklearn. Sklearn – один з найбільш широко використовуваних пакетів Python для Data Science та Machine Learning. Бібліотека досить стара, але вона є актуальною та активно підтримується розробниками, оскільки на момент її використання останній реліз приписується на жовтень місяць 2021 року. В своєму арсеналі модуль має таку підтримку:

- попередньої обробки даних;
- зміна розмірності;
- класифікації;
- регресії;
- кластеризації.

Також перевагою є те, що ця бібліотека побудована на основі NumPy, а це означає, що будь-які маніпуляції та перетворення будуть більш зручними, адже існують одні й ті ж правила та специфікації.

### 3.1.2 Середовище програмної реалізації

Для реалізації обрано локальне середовище Jupyter Notebook [29]. Також зручний засіб для створення та редагування коду Visual Studio Code.

Jupyter Notebook – це спеціальне середовище, в основу якого лежить консольний підхід до інтерактивних обчислень. Ці обчислення представлені у вигляді окремих логічних блоків коду. А сама програма складається за таких блоків.

Зручність такого підходу є в тому, що на кожному етапі ми можемо бачити якість проміжні результати роботи певного блоку, тому це досить зручно під час тестування та розробки.

Така середа розробки може бути як на веб сервісі, тобто онлайн платформи, так і може бути встановлена локально. Було обрано другий

варіант, так як це не вимагає підключення до мережі та швидкість роботи програми та роботи з програмними компонентами вище.

Для встановлення локального середовища Jupyter Notebook було використано розширення Jupyter (рис. 3.1).

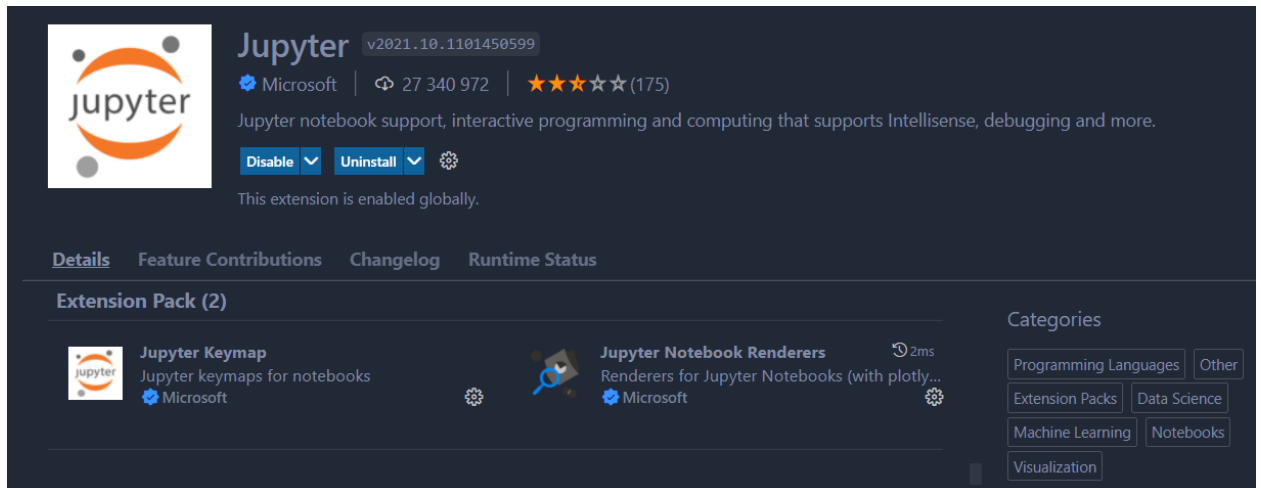


Рисунок 3.1 – Розширення Jupyter для локальної розробки

### 3.2 Програмна реалізація для класифікації DDoS-атак

В якості основного було обрано набір даних під назвою «KDD Cup 1999» [30-31]. Це набір даних, який використовується в деяких міжнародних конкурсах. Завданням конкурсу було створення детектора вторгнення в мережу та класифікації трафіку на плохий та хороший. Ця база даних містить стандартний набір даних, що підлягають аудиту, що включає широкий спектр вторгнень, змодельованих у військовому мережевому середовищі.

В цілому набір має 52 поля, опишемо деякі з основних:

- duration (тривалість підключення);
- protocol\_type (тип протоколу);
- src\_bytes (кількість байтів від джерела до міста призначення);

- `dst_bytes` (кількість байтів від міста призначення до джерела);
- `service` (мережевий сервіс в пункті призначення);
- `wrong_fragment`;
- `count`;
- `num_compromised`;
- `srv_count`;
- `srv_serror_rate`;
- `serror_rate`.

Не всі поля є інформативними та потрібними, тому їх використання можна обмежитися. Але це може впливати на точність. Слід звернути увагу на поле типу протоколу, воно має три різних значення:

- `UDP`;
- `TCP`;
- `ICMP`.

Саме за цими протоколами фільтрувалася вибірка, тому отримуються результати за трьома цими протоколами.

Також для задачі класифікації за допомогою бібліотеки `Sklearn` та її реалізації алгоритмів, використовуються такі реалізації алгоритмів:

- найближчих сусідів;
- дерев рішень;
- багат шарового перцептрону;
- логістичної регресії.

Як вже було виявлено, використання `Jupyter Notebook` розуміє під собою консольний підхід, де існують певні логічні блоки, які можуть вивести результат роботи цього блоку. Тому доречно розглядати такі логічні блоки окремо.

Блок 1. Підключення модулів.

Лістинг 3.1 Підключення модулів:

```
import numpy as np
```

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import sys
import pickle
```

Блок 2. Попередня обробка даних та фільтрація для ТСП.

Лістинг 3.2 Попередня обробка даних та фільтрація для ТСП:

```
df = pd.read_csv(r"revised_kddcup_dataset.csv", index_col=0)
df.head()

tcp_syn_df = df[df.loc[:, "protocol_type"] == "tcp"]
tcp_syn_df = tcp_syn_df[tcp_syn_df.loc[:, "srv_error_rate"] > 0.7]
tcp_syn_df.head()

service_values = np.unique(tcp_syn_df.loc[:, "service"])
mid = (len(service_values)+1)/2
for i in range(len(service_values)):
    tcp_syn_df = tcp_syn_df.replace(service_values[i], (i-mid)/10)
```

Блок 3. Витяг даних за основними полями.

Лістинг 3.3 Витяг даних за основними полями:

```
features =
["service", "src_bytes", "wrong_fragment", "count", "num_compromised", "srv_count",
"srv_error_rate", "error_rate"]
target = "result"
```

Блок 4. Попередня обробка класів.

Лістинг 3.4 Попередня обробка класів:

```

x = tcp_syn_df.loc[:,features]
y =tcp_syn_df.loc[:,target]
classes = np.unique(y)
for i in range(len(classes)):
    if i == 2:
        tcp_syn_df = tcp_syn_df.replace(classes[i], 0)
    else:
        tcp_syn_df = tcp_syn_df.replace(classes[i], 1)
tcp_syn_df=tcp_syn_df.replace("eco_i",-0.1)
tcp_syn_df=tcp_syn_df.replace("ecr_i",0.0)
tcp_syn_df=tcp_syn_df.replace("tim_i",0.1)
tcp_syn_df=tcp_syn_df.replace("urp_i",0.2)

```

Блок 5. Підключення модулів для використання алгоритмів.

Лістинг 3.5 Підключення модулів для використання алгоритмів:

```

from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

```

Блок 6. Підключення модулів для використання алгоритмів.

Лістинг 3.6 Підключення модулів для використання алгоритмів:

```

rs = RandomForestClassifier()

```

```

rs.fit(X,y)
print(pd.Series(rs.feature_importances_,index=features).sort_values(ascending=False))
X
=
X.loc[:,["service","count","srv_count","src_bytes","serror_rate"]]
for i in X.index:
    if y[i] == 0:
        print(i,":",X.loc[i,:])
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42,
test_size=0.3)
models = [LogisticRegression(), KNeighborsClassifier(n_neighbors=3),
MLPClassifier(alpha=0.005),DecisionTreeClassifier()]
classifiers = ["LR", "KNN", "MLP", "DecisionTree"]
scores = []
for model in models:
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    score = accuracy_score(y_test, y_pred)*100
    scores.append(score)
    print("Accuracy: ", score)
    conf_matrix = confusion_matrix(y_test,y_pred)
    class_report = classification_report(y_test,y_pred)
    print("CMatrix:\n",conf_matrix)
    print("Report:\n",class_report)

```

Блок 7. Відображення діаграми точності алгоритмів.

Лістинг 3.7 Відображення діаграми точності алгоритмів:

```

plt.bar(classifiers,scores)

```

```
plt.title("TCP Sync Attack")
plt.ylim(99.5,100)
plt.show()
```

Реалізувавши кожний із логічних блоків в локальному розширенні Jupyter, отримаємо робоче середовище, яке має такий вигляд (рис 3.2). На ньому видно, що кожний логічний блок виводить результат роботи, якщо це потрібно, якщо один із етапів не виконується, то буде помилка та наступні етапи не будуть виконуватися. Це досить зручно при розробці.

```
classes = np.unique(y)
print(classes)
```

```
[87] ✓ 0.3s Python
... ['apache2.', 'back.', 'guess_passwd.', 'land.', 'mscan.', 'neptune.', 'nmap.',
'normal.', 'processtable.', 'saint.', 'satan.', 'warezmaster.', 'xsnoop.']
```

```
#replacing all classes of attack with 1 and normal result with 0 in our icmp_df
for i in range(len(classes)):
    if i == 2:
        tcp_syn_df = tcp_syn_df.replace(classes[i], 0)
    else:
        tcp_syn_df = tcp_syn_df.replace(classes[i], 1)

#turning the service attribute to categorical values
tcp_syn_df=tcp_syn_df.replace("eco_i",-0.1)
tcp_syn_df=tcp_syn_df.replace("ecr_i",0.0)
tcp_syn_df=tcp_syn_df.replace("tim_i",0.1)
tcp_syn_df=tcp_syn_df.replace("urp_i",0.2)
```

```
[88] ✓ 0.1s Python
```

```
tcp_syn_df.head()
```

```
[89] ✓ 0.4s Python
```

	duration	protocol_type	service	flag	src bytes	dst bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_san
...	170	0	tcp	-2.8	S1	286040	383476	0	0	0	0	...	1
	8033	0	tcp	-1.0	S3	305	421	0	0	0	0	...	253
	8105	0	tcp	-2.8	S1	1256	11240	0	0	0	0	...	1

Рисунок 3.2 – Вигляд розширення Jupyter в Visual Studio Code

### 3.3 Тестування програми та аналіз результатів

Для тестування та роботи програми та дослідження результатів вхідну вибірку було відфільтровано за кожним типом протоколу. Для кожної з цієї



відфільтрованої вибірки побудовано класифікатори по 4 із представлених в другому розділі алгоритмам.

Результати роботи класифікаторів представлені в виді гістограм, на яких відображена точність роботи по кожному із класифікаторів. Розглянемо спочатку вибірку, об'єкти яких відносяться до протоколу TCP та є цифровим відображенням SYN-флуду.

Запустимо програму та отримаємо графік точності роботи алгоритму для TCP SYNC флуду на рисунку 3.3.

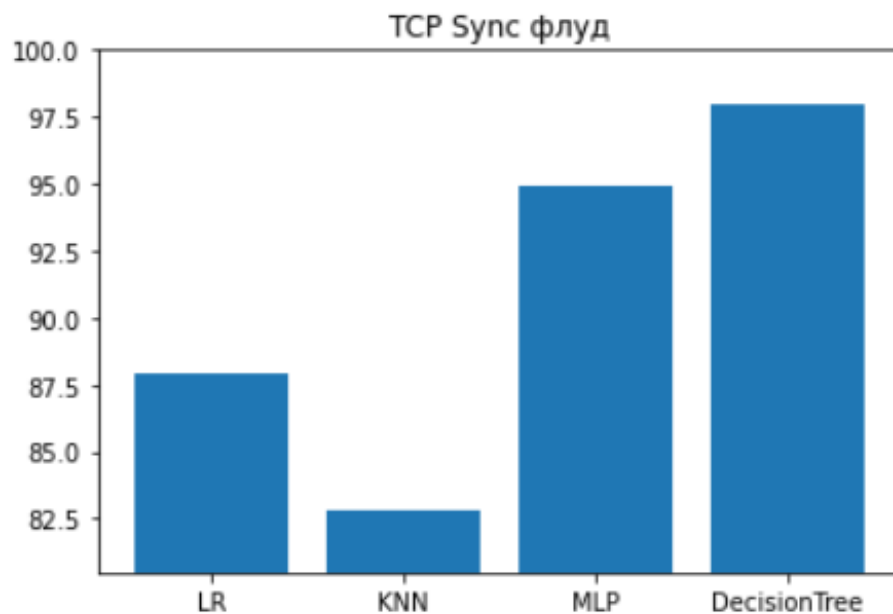


Рисунок 3.3— Значення точності TCP для базових параметрів класифікаторів для TCP

Як бачимо найгірше показав себе в плані точності алгоритм kNN. Це можна пояснити тим, що при заданні параметрів для алгоритму було вказано кількість сусідів, яке дорівнювало 1. Спробуємо збільшити це значення та задаємо кількість сусідів 5. В результаті на рисунку 3.4 отримуємо більшу точність, яка приблизно збільшилася на 5%.

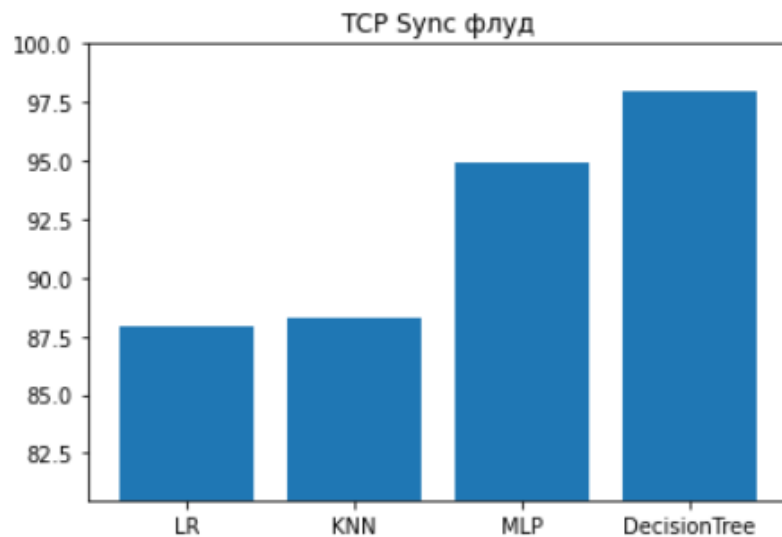


Рисунок 3.4 – Збільшення точності kNN при  $k=5$

Можна спробувати ще збільшити кількість сусідів та встановити 10 (рис. 3.5).

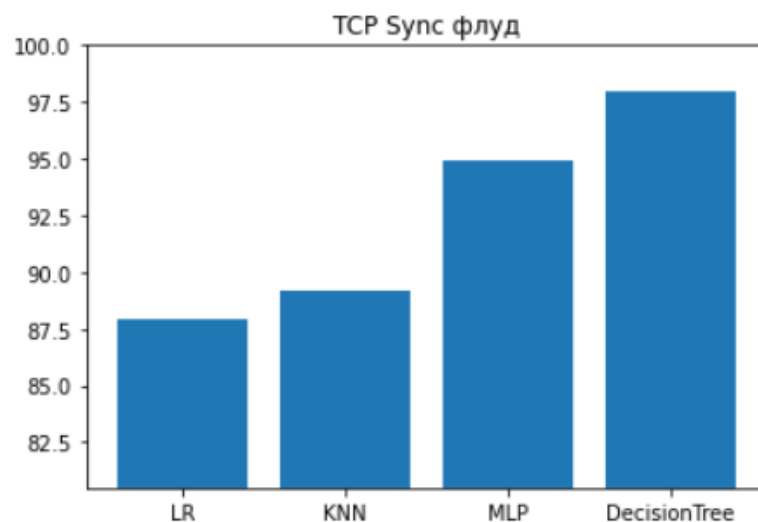


Рисунок 3.5 – Мізерне збільшення точності kNN при  $k=10$

Як бачимо на рисунку 3.5 результат покращився трохи більше ніж на 1%. Це свідчить про те, що подальше збільшення кількості  $k$  не впливатиме на точність роботи алгоритму, але час його роботи росте, тому оптимальним значенням кількості сусідів є значення не більше 5.

Спробуємо змінити коефіцієнт тестової вибірки та зменшити його в 30 разів. Результати на рисунку 3.6 не змінилися, що свідчить про великий об'єм вибірки та її однорідність.

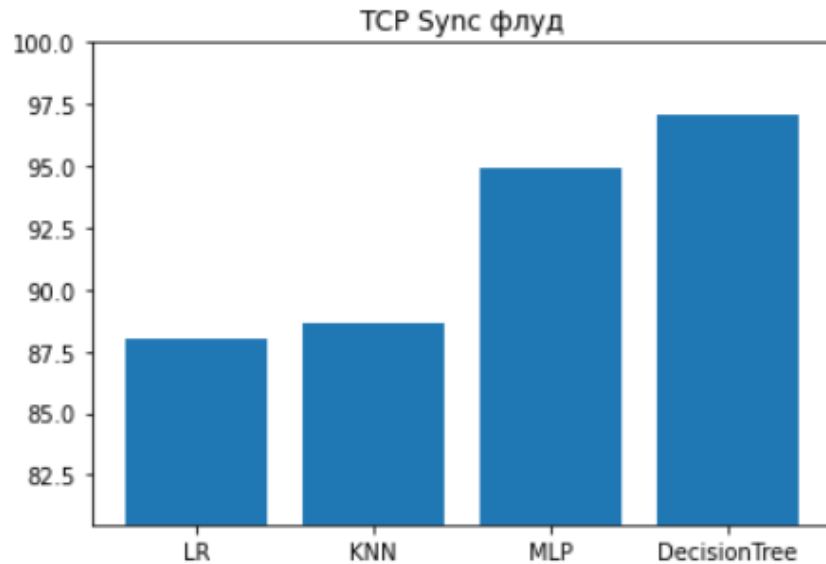


Рисунок 3.6 – Результат при зменшенні тестової вибірки

На більшості гістограм найкращу точність показав класифікатор дерева рішень, спробуємо задати ліміт його глибини за допомогою параметру `max_depth`, який буде дорівнювати 1 (рис. 3.7).

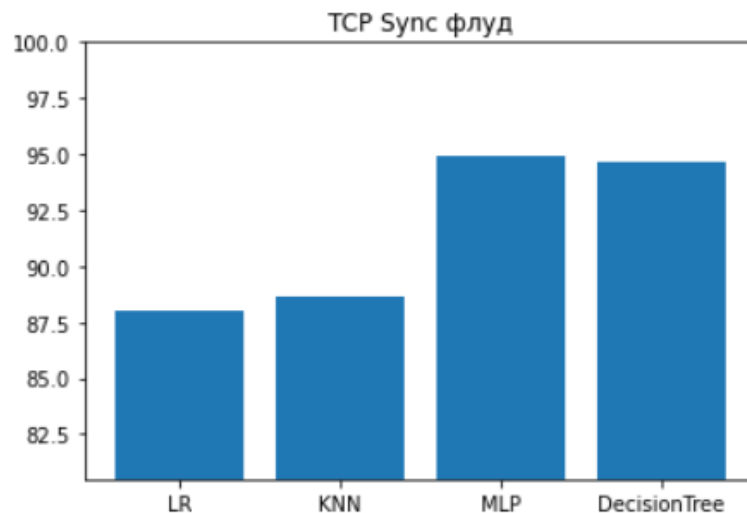


Рисунок 3.7 – Зменшення точності DecisionTree при заданні ліміту глибини

Як бачимо на рисунку 3.5 зміна цього параметру не сильно, але зменшила точність роботи. Тепер він знаходиться приблизно на рівні з класифікатором на основі нейронної мережі.

Подивимося на результати для UDP-флуду (рис. 3.8).

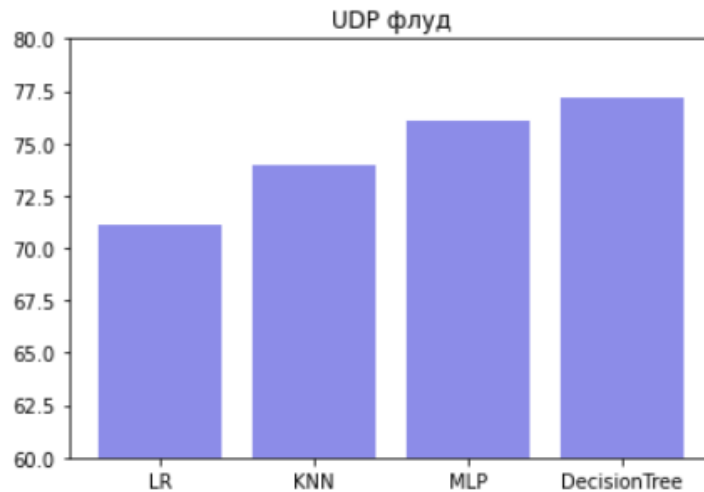


Рисунок 3.8 – Точність класифікаторів для UDP

Спробуємо змінити значення альфа-параметру для MLP-алгоритму. Це значення було 0,05. Спробуємо змінити його на 0,5 (рис. 3.9). А потім змінити на 0,01 (рис. 3.10).

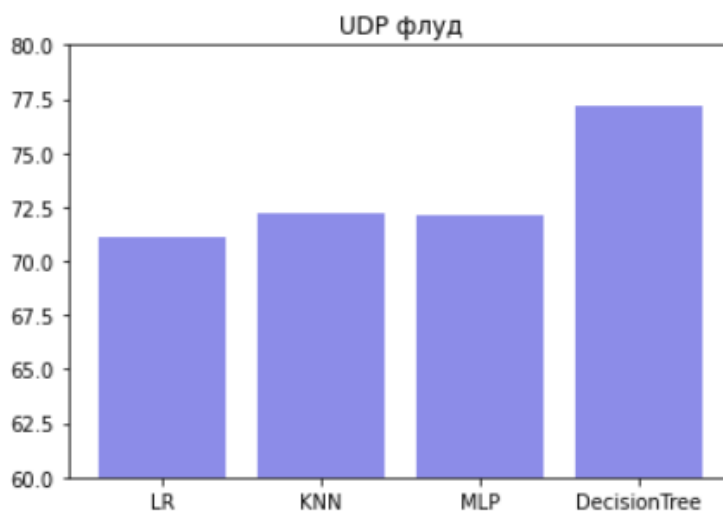


Рисунок 3.9 – Зменшення точності MLP при  $\alpha = 0,5$  для UDP

Як бачимо значення точності на рисунку 3.10 для MLP чуть зменшилося, тому слід зазначити, що первинне значення альфа-коефіцієнту є оптимальним, тобто дорівнює 0,05.

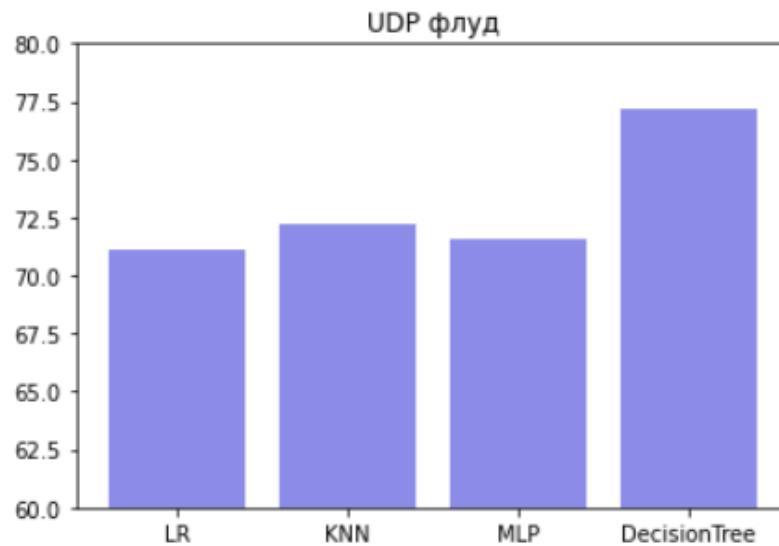


Рисунок 3.10 – Зменшення точності MLP при  $\alpha = 0,01$  для UDP

Також цікаве спостерігати те, що при збільшенні числа сусідів з 3 до 5 чи 10, значення точності для kNN алгоритму майже не збільшуються. Збільшення точності становить приблизно до 3% (рис. 3.11).

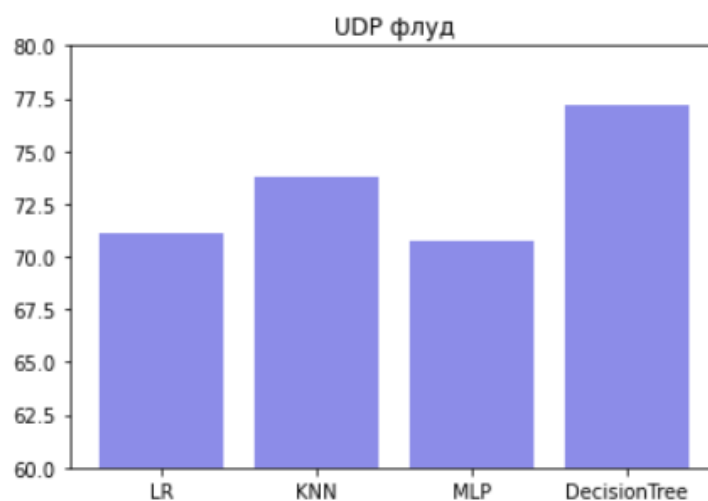
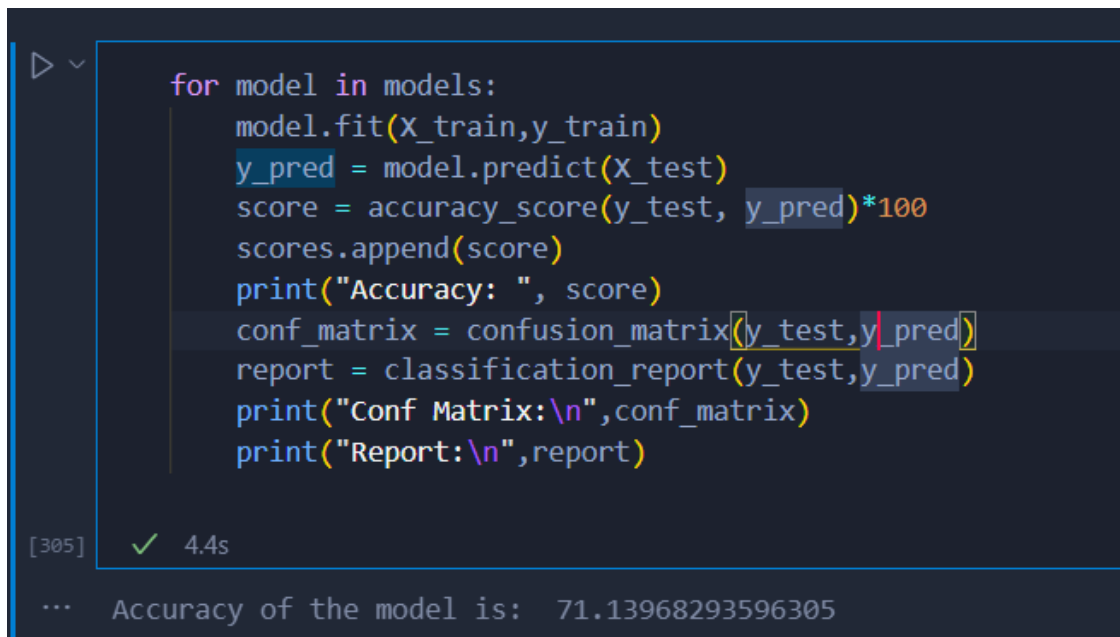


Рисунок 3.11 – Незначне збільшення точності для kNN при збільшенні сусідів

Перейдемо до протоколу ICMP. При тестуванні цього протоколу швидкість роботи побудови класифікаторів становила приблизно 40 секунд, в той час як побудова, наприклад, класифікаторів для протоколу UDP, становить від 4 до 10 секунд (рис. 3.12). Час для ICMP час 45 секунд (рис. 3.13).




```
for model in models:
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    score = accuracy_score(y_test, y_pred)*100
    scores.append(score)
    print("Accuracy: ", score)
    conf_matrix = confusion_matrix(y_test,y_pred)
    report = classification_report(y_test,y_pred)
    print("Conf Matrix:\n",conf_matrix)
    print("Report:\n",report)
```

[305] ✓ 4.4s

... Accuracy of the model is: 71.13968293596305

Рисунок 3.12 – Час роботи 4.4 секунди для UDP



```
for model in models:
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    score = accuracy_score(y_test, y_pred)*100
    scores.append(score)
    print("Accuracy: ", score)
    conf_matrix = confusion_matrix(y_test,y_pred)
    report = classification_report(y_test,y_pred)
    print("Conf Matrix:\n",conf_matrix)
    print("Report:\n",report)
```

[206] ✓ 45.9s

Рисунок 3.13 – Час роботи 45,9 секунди для ICMP

Самі результати для цього типу флуду становлять майже 100% для кожного із класифікаторів. Результати відображені на рисунку 3.14.

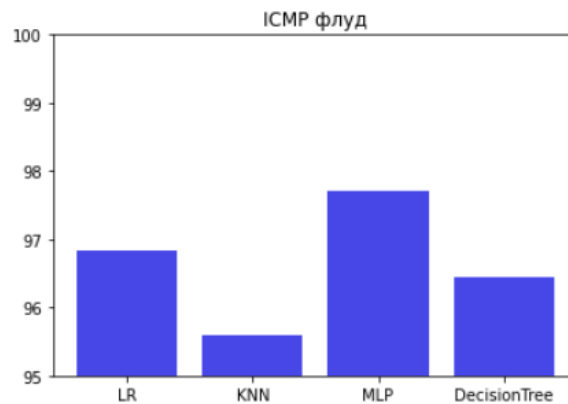


Рисунок 3.14 – Результати точності для ICMP

### 3.4 Перспективи подальшої роботи

Програмний засіб, що був розроблений, можна буде використовувати як основу для подальшого його вдосконалення. Також його можна інтегрувати в інші компоненти. Але для цього потрібно реалізувати додаткові інтерфейси взаємодії, де на вхід буде подаватися не тестовий набір даних, а набір даних за певний період із реального стеку запитів до серверу.

## ВИСНОВКИ

В ході кваліфікаційної роботи було проаналізовано предметну область. Було розглянуто статистику та типи DDoS-атак, які поширені на даний момент. Були детально розглянута класифікація DDoS-атак за моделлю OSI.

Також були детально розглянуті типи та способи класифікації в задачі класифікації. Всі методи мають свої переваги та недоліки. Це можуть бути як і простота алгоритму, його точність, час роботи, чи ефективність.

На основі розглянутих класифікаторів було створено програму, що їх реалізувала. В якості набору даних було обрано вибірку «KDD Cup 1999». До цієї вибірки входять 3 основні типи протоколів, які були розглянуті. Для кожного з цих протоколів окремо побудовано класифікатори за розглянутими алгоритмами та методами.

На основі критеріїв вибору програмних інструментів та інструментів розробки, було обрано мову Python та відповідні бібліотеки. Для локальної розробки було використано розширення Jupyter для Visual Studio Code.

В ході аналізу результатів кваліфікаційної роботи та тестування було отримано досить непогані результати. Середня точність класифікаторів, побудованих на основі нейронної мережі та дерева рішень, становила приблизно 93%. Класифікатори на основі kNN та логістичної регресії чуть гірші, за попередні та мають середню точність від 80% до 85%. Деякі результати є досить цікавими та непередбачуваними. Це можна пояснити специфікою вибірки та налаштуваннями вхідних параметрів для кожного із класифікаторів.

Результати роботи апробовано у вигляді 2 тез доповідей під час Міжнародних науково-практичних конференцій «MODERN TRENDS IN DEVELOPMENT SCIENCE AND PRACTICE» [32] та «SCIENCE AND PRACTICE, ACTUAL PROBLEMS, INNOVATIONS» [33].



## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Douligieris, C., & Mitrokotsa, A. (2004). DDoS attacks and defense mechanisms: classification and state-of-the-art. *Computer networks*, 44(5), 643-666.
2. Radivilova, T., Kirichenko, L., Tawalbeh, M., Zinchenko, P., & Bulakh, V. (2020). THE LOAD BALANCING OF SELF-SIMILAR TRAFFIC IN NETWORK INTRUSION DETECTION SYSTEMS. *Cybersecurity: Education, Science, Technique*, 3(7), 17–30.
3. Kirichenko, L.O., Tkachenko, A.E., Radivilova, T.A. (2019). Clustering of noisy time series. *System technologies. Regional mizhvuzivskiy zbirnik naukovikh prats*, 3 (122), 133-139.
4. Kirichenko, L. O., Ostroverkh, N. V., & Timko, A. V. (2012). Analysis of multifractal properties of chaotic maps. *Восточно-Европейский журнал передовых технологий*, 3(9 (57)).
5. Kirichenko, L. O., & Radivilova, T. A. (2019). Fractal analysis of self-similar and multifractal hour series. *Kharkiv, Kh-NURE [in Ukrainian]*.
6. How GitHub Survived the Biggest DDoS Attack Ever Recorded? URL: <https://medium.com/technology-hits/how-github-survived-the-biggest-ddos-attack-ever-recorded-6907ba6f5c98> (дата звернення 12.10.2021).
7. DDoS attacks in Q1 2020. URL: <https://securelist.com/ddos-attacks-in-q1-2020/96837> (дата звернення 22.10.2021).
8. Kesavamoorthy, R., Alaguvathana, P., Suganya, R., & Vigneshwaran, P. (2020). Classification of DDoS attacks—A survey. *Test Eng. Manag.*, 83, 12926-12932.
9. Li, Y., Li, D., Cui, W., & Zhang, R. (2011, May). Research based on OSI model. In *2011 IEEE 3rd International Conference on Communication Software and Networks* (pp. 554-557). IEEE.

10. Bogdanoski, M., Suminoski, T., & Risteski, A. (2013). Analysis of the SYN flood DoS attack. *International Journal of Computer Network and Information Security (IJCNIS)*, 5(8), 1-11.
11. Xiaoming, L., Sejdini, V., & Chowdhury, H. (2010). Denial of service (dos) attack with udp flood. *School of Computer Science, University of Windsor, Canada*.
12. Prasad, K. M., Reddy, A. R. M., & Rao, K. V. (2017). BIFAD: Bio-inspired anomaly based HTTP-flood attack detection. *Wireless Personal Communications*, 97(1), 281-308.
13. Gupta, N., Jain, A., Saini, P., & Gupta, V. (2016, March). DDoS attack algorithm using ICMP flood. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 4082-4084). IEEE.
14. Kirichenko, L., Radivilova, T., & Bulakh, V. (2019). Machine Learning in Classification Time Series with Fractal Properties. *Data*, 4(5), 1-13.
15. Kirichenko, L., Zinchenko, P., Radivilova, T., & Tawalbeh, M. (2019, November). Machine Learning Detection of DDoS Attacks Based on Visualization of Recurrence Plots. In *CMiGIN* (pp. 23-34).
16. Daradkeh, Y. I., Kirichenko, L., & Radivilova, T. (2018). Development of QoS methods in the information networks with fractal traffic. *International Journal of Electronics and Telecommunications*, 64.
17. Kirichenko, L. (2014). The method of distinction monofractal and multifractal process from time series.
18. Kirichenko, L. O., Demerchan, K. A., Kayali, E., & Habachyova, A. Y. (2012). МОДЕЛИРОВАНИЕ ТЕЛЕКОММУНИКАЦИОННОГО ТРАФИКА С ИСПОЛЬЗОВАНИЕМ СТОХАСТИЧЕСКИХ МУЛЬТИФРАКТАЛЬНЫХ КАСКАДНЫХ ПРОЦЕССОВ. *Radio Electronics, Computer Science, Control*, (1).
19. Kirichenko, L., & Zinchenko, P. (2019, November). Time Series Classification Based on Visualization of Recurrence Plots. In *International Conference on Tools and Methods for Program Analysis* (pp. 101-108). Springer, Cham.

20. Kirichenko, L. O., & Habacheva, A. Y. (2014). Comparative analysis of the complexity of chaotic and stochastic time series. *Радіоелектроніка, інформатика, управління*, (2 (31)).
21. Kirichenko, L., Radivilova, T., & Bulakh, V. (2019, May). Binary classification of fractal time series by machine learning methods. In *International Scientific Conference "Intellectual Systems of Decision Making and Problem of Computational Intelligence"* (pp. 701-711). Springer, Cham
22. Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003, November). KNN model-based approach in classification. In *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"* (pp. 986-996). Springer, Berlin, Heidelberg.
23. Myles, A. J., Feudale, R. N., Liu, Y., Woody, N. A., & Brown, S. D. (2004). An introduction to decision tree modeling. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 18(6), 275-285.
24. Raileanu, L. E., & Stoffel, K. (2004). Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, 41(1), 77-93.
25. Féraud, R., & Clérot, F. (2002). A methodology to explain neural network classification. *Neural networks*, 15(2), 237-246.
26. Orhan, U., Hekim, M., & Ozer, M. (2011). EEG signals classification using the K-means clustering and a multilayer perceptron neural network model. *Expert Systems with Applications*, 38(10), 13475-13481.
27. Wright, R. E. (1995). Logistic regression.
28. Vallat, R. (2018). Pingouin: statistics in Python. *Journal of Open Source Software*, 3(31), 1026.
29. Randles, B. M., Pasquetto, I. V., Golshan, M. S., & Borgman, C. L. (2017, June). Using the Jupyter notebook as a tool for open science: An empirical study. In *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)* (pp. 1-2). IEEE.

30. KDD Cup 1999 Data. URL:  
<http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html> (дата звернення 01.12.2021).

31. KDD-CUP-99 Task Description. URL:  
<http://kdd.ics.uci.edu/databases/kddcup99/task.html> (дата звернення 01.12.2021).

32. Кіріченко, Л., Гонтар, С. (2021). Дослідження і порівняння методів класифікації реалізацій трафіків для детектування DDos-атак.

33. Гонтар, С. (2021). Використання нейронної мережі для класифікації реалізацій трафіків для детектування DDos-атак.