НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)

“__” _____2021 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою “Інженерія

програмного забезпечення комп’ютерних систем”

спеціальності 121 “Інженерія програмного забезпечення”

на тему: Виявлення DDOS атак за допомогою машинного навчання

Виконав: студент 4 курсу, групи ІП-73

Михайліченко Ілля Андрійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник професор, д.т.н. Стіренко С. Г. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2021 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Інженерія програмного забезпечення комп’ютерних систем”

спеціальності 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2021 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Михайліченко Ілля Андрійовича

1. Тема проєкту *Виявлення DDOS атак за допомогою машинного навчання*
керівник проєкту Стіренко Сергій Григорович, доктор технічних наук, професор
Затверджені наказом по університету від 11.05.2021 року № 1139-с
2. Термін здачі студентом закінченого проєкту *2 червня 2021 р.*
3. Вихідні дані до проєкту: *технічна документація, теоретичні та статистичні дані;*
4. Зміст пояснювальної записки: *опис предметної області, дослідження способів виявлення, програма виявлення на основі машинного навчання;*
5. Перелік графічного матеріалу (з точним позначенням обов’язкових креслень): *структурна схема системи, узагальнена схема роботи системи.*

6. Консультант проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	д.т.н., проф. Сімоненко В.		

7. Дата видачі завдання 22 вересня 2020 року

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	*Затвердження теми проєкту*	*10.12.2020-30.01.2021*	
2.	*Вивчення та аналіз завдання*	*30.01.2021-1.03.2021*	
3.	*Розробка архітектури та загальної структури системи*	*1.03.2021-12.04.2021*	
4.	*Розробка структур окремих підсистем*	*12.04.2021-26.04.2021*	
5.	*Програмна реалізація системи*	*26.04.2021-16.05.2021*	
6.	*Оформлення пояснювальної записки*	*16.04.2021-23.05.2021*	
7.	*Захист програмного продукту*	*25.05.2021*	
8.	*Передзахист*	*6.06.2021*	
9.	*Захист*	*19.06.2021*	

Студент-дипломник _____
(підпис)

Керівник проєкту _____
(підпис)

Анотація

В бакалаврському дипломному проєкті розглянуто алгоритм виявлення та захисту від DDOS атак за допомогою машинного навчання. Практичною частиною роботи є реалізація програми, що моделює роботу алгоритму.

Програма дозволяє стежити за мережевим трафіком та виявляти підозрілі запити до серверу, що в свою чергу дозволяє запобігти DDOS атакам. Є можливість завантажувати власні дані щодо Інтернет-трафіку, а саме дані про запити, які аналізуються програмою та виявляються підозрілі на атаку.

Додаток розроблений у формі додатку на мові програмування Python в середовищі розробки PyCharm IDE Community Edition 2021.1.1. Для розробки графічного інтерфейсу використовувався фреймворк PySimpleGUI.

Annotation

In this project for a Bachelor's Degree, the algorithm of detection of DDOS attacks is considered. The practical part of the work is the implementation of a program that simulates the operation of the algorithm.

The program allows to monitor network traffic and detect suspicious requests to the server, which prevents DDOS attacks. It is possible to download your own data of Internet traffic requests that are analyzed by the program and detected suspicious of the attack.

The application is developed in the form of an application in the Python programming language in the PyCharm IDE Community Edition 2021.1.1 development environment. The PySimpleGUI framework is used to develop the graphical interface.

Опис альбому

до дипломного проекту

на тему: «Виявлення DDOS атак за
допомогою машинного навчання»

Київ – 2021

Справки	Формат	*Значення*	*Найменування*	Кіл. листів	№ екземпляра	*Додаток*
			Документація загальна			
			Знову розроблена			
	А4	***ІАЛЦ.467100.001 ОА***	Виявлення DDOS атак за	1		
			допомогою машинного навчання			
			Опис альбому			
	А4	***ІАЛЦ.467100.002 ТП***	Виявлення DDOS атак за	3		
			допомогою машинного навчання			
			Технічне завдання			
	А4	***ІАЛЦ.467100.003 ПЗ***	Виявлення DDOS атак за	58		
			допомогою машинного навчання			
			Пояснювальна записка			
	А1	***ІАЛЦ.467100.004 Д1***	Виявлення DDOS атак за	1		
			допомогою машинного навчання			
			Додаток А			
	А1	***ІАЛЦ.467100.005 Д2***	Виявлення DDOS атак за	1		
			допомогою машинного навчання			
			Додаток Б			
	А1	***ІАЛЦ.467100.006 Д3***	Виявлення DDOS атак за	9		
			допомогою машинного навчання			
			Додаток В			

Зм	*Лист*	*№ докум.*	*Підп.*	*Дата*	***ІАЛЦ.467100.001 ОА***		
Розроб		*Михайліченко І.А.*			*Виявлення DDOS атак за допомогою машинного навчання Опис альбому*	*Літ.*	*Аркуш* / *Аркушів*
Перев		*Стіренко С.Г.*					*1* / *1*
Реценз.						***НТУУ "КПІ" ФІОТ ІП-73***	
Н. контр.		*Сімоненко В.П.*					
Затв.		*Стіренко С.Г.*					

Технічне завдання

до дипломного проекту

на тему: «Виявлення DDOS атак за
допомогою машинного навчання»

Київ – 2021

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

Найменування: виявлення DDOS атак за допомогою машинного навчання.

Область застосування: альтернатива існуючим алгоритмам й програмам виявлення.

2. ПРИЧИНИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврського проєкту по освітньо-професійної програми “Інженерія програмного забезпечення комп’ютерних систем” за спеціальністю 121 Інженерія програмного забезпечення, затверджене кафедрою Обчислювальної техніки Національного технічного Університету України “Київський Політехнічний інститут імені Ігоря Сікорського”.

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проєкту є розгляд методів машинного навчання для виявлення DDOS та розробка програмного продукту, який реалізовуватиме дану модель.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література з теорії та практики машинного навчання та комп’ютерних мереж, технічна документація, публікації в періодичних виданнях, довідники, публікації в Інтернеті з даних питань.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

- Відносно простий і інтуїтивно зрозумілий інтерфейс системи.
- Можливість аналізу різних даних для виявлення атак.
- Незалежність - система повинна бути автономна і не залежати від встановленого програмного забезпечення.
- Технічна коректність та актуальність інформації, що становить наповнення системи.

5.2. Вимоги до програмного забезпечення

- MS Windows XP/Vista/7/8/10

5.3. Вимоги до апаратної частини

- Комп’ютер на базі процесора Intel Pentium 166 та вище
- Оперативної пам’яті не менше 64 Мбайт
- Python версії 3.0 та вище

6. Етапи розробки

	Дата
Вивчення літератури	11.01.2021
Складання і узгодження технічного завдання	15.03.2021
Створення модулів розробленої системи	29.03.2021
Тестування модулів розробленої системи	26.04.2021
Доопрацювання і виправлення помилок	17.05.2021
Оформлення документації дипломного проєкту	24.05.2021

Пояснювальна записка

до дипломного проекту

на тему: «Виявлення DDOS атак за
допомогою машинного навчання»

Київ – 2021

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

DDOS (Distributed Denial of Service)	Розподілена атака на відмову в обслуговуванні
ML(Machine learning)	Машинне навчання
SDN (Software-defined Networking)	Програмно-визначена мережа
CDN (Content Delivery Network)	Мережа доставки контенту
SVM (Support Vector Machine)	Метод опорних векторів
RST (Rough Sets Theory)	Теорія грубих множин
ROC (Receiver Operator Characteristic)	Робочі характеристики приймача
CF(Cost Function)	Функція витрат
FPR(False positive rate)	Рівень хибно позитивних
TPR(True positive rate)	Рівень істинно позитивних

ВСТУП

На сьогоднішній день в Інтернеті існує багато важливих сервісів, які мають стабільно працювати весь час, бо інакше це негативно вплине на життя багатьох людей. Наприклад, пошкодження системи банківських платежів зупиняє роботу магазинів та створює незручності великій кількості покупців. Тому розробники таких сервісів мають приділяти достатньо уваги захисту інформації та стійкості комп'ютерних систем.

Питання стійкості систем стоїть досить гостро. Регулярно відбуваються різноманітні хакерські атаки та зломи сайтів. Їх метою не завжди може бути заволодіння цінною інформацією, іноді це бажання вивести з ладу певний інтернет-сервіс, чи, як мінімум, сповільнити його роботу. Ціллю таких атак зазвичай є сайти державних установ, інтернет-магазини, чи інші бізнеси, що працюють в мережі. Такі атаки можуть знадобитися для боротьби з конкурентами або для дестабілізації ситуації(наприклад, на фондових ринках).

Для таких цілей хакери використовують різноманітні інструменти. Деякі з них шукають вразливості в коді сервісу, інші використовують комп'ютерні віруси. Такі методи є досить складними, адже вимагають спеціального підходу до кожного сервісу. Натомість існують атаки, які використовують звичайні запити до серверу, але за рахунок їх кількості чи інших особливостей(наприклад, розміру переданої інформації) можуть зупиняти роботу сервісу. Якщо запити відбуваються з різних IP-адрес, то це називають DDOS атаками.

Для захисту використовують програми, які можуть виявляти атаки, повідомляти про них або автоматично їх блокувати. Але оскільки запити можуть бути схожими на звичайні, іноді можна заблокувати користувача або пропустити атаку. Проблема полягає у тому, що не очевидно, які параметри більш важливі для класифікації запитів. До того ж, зловмисники можуть змінити тип атаки, і тоді будь-яка стратегія виявиться хибною.

Ці проблеми можна вирішити за допомогою машинного навчання. Це метод, який дозволяє класифікувати дані без явного програмування. Тобто ми чітко не знаємо, як програма вирішує, які з запитів є шкідливими, а лише надаємо дані про атаки, що вже трапилися. Для такого методу необхідно мати великий об'єм попередніх даних, обрати певний алгоритм навчання і мати достатньо часу для тренування моделі. Але після тренування модель швидко обробляє нові дані й виявляє атаки.

Завданням даної роботи є вивчення методів машинного навчання та можливості їх використання для виявлення DDOS атак; реалізація системи виявлення з використанням цих методів; оцінка результатів реалізації за допомогою спеціальних методів оцінки якості алгоритмів машинного навчання.

Для успішного виконання дипломної роботи є необхідним:

- обрати й опрацювати наукову літературу з обраної теми;
- провести огляд існуючих алгоритмів виявлення атак, інструментів машинного навчання;
- розробити алгоритм виявлення, який буде базуватися на принципах машинного навчання;
- створити модель програми для тестування і візулізації розробленого алгоритму;
- розробити програму із інтерфейсом користувача згідно прототипу.

Дана робота складається з чотирьох розділів. В процесі виконання поставленої задачі буде виконано огляд існуючих рішень, а також буде вибрано відповідну практичні практичні інструменти, за допомогою яких буде виконана програмна реалізація системи виявлення на основі машинного навчання.

Перший розділ містить огляд структури DDOS атак та їх типів. Описані існуючі методи, за допомогою яких відбувається виявлення та запобігання атакам. Також розглядаються та порівнюються запропоновані в наукових роботах системи виявлення.

Другий розділ містить постановку основної задачі та методи її вирішення. Визначений алгоритм машинного навчання, що обраний для навчання, та набір даних, що для цього використовується. Описані методи попередньої обробки даних для використання в алгоритмі машинного навчання. Розглядаються технології й засоби, використані для розробки програмного продукту.

Третій розділ містить опис розробки і реалізації програми та користувацького інтерфейсу за обраним прототипом. Описані компоненти програмного продукту та їх взаємодія. Представлені та пояснені схеми роботи програми.

Четвертий розділ містить демонстрацію результатів роботи розробленої системи виявлення DDOS атак, користувацького інтерфейсу та порівняння з іншими існуючими системами. Також представлені результати тестування програми за допомогою спеціальних інструментів оцінки якості роботи методів машинного навчання.

В кінці роботи містяться загальні висновки до всіх розділів, зокрема перелік виконаної роботи та підсумки щодо її успішності. Визначається можливість використання розробленого програмного продукту на рівні з іншими можливими рішеннями.

РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ ВИЯВЛЕННЯ DDOS АТАК

1.1. Комп'ютерні атаки

Наявність в Інтернеті цінної інформації та важливих сервісів призводить до великої кількості комп'ютерних атак. Комп'ютерні атаки направлені на злом цілі, наприклад, серверної системи, для отримання можливостей, не передбачених для звичайних користувачів.

Основні види комп'ютерних атак[1]:

1. Комп'ютерні віруси – спеціальні програми, що можуть передаватися різними шляхами. Вони можуть бути запрограмованими на погіршення якості роботи комп'ютера, видалення файлів з пам'яті, а також на розсилку своїх копій іншим користувачам для поширення.
2. Сніферінг трафіку – перехоплення пакетів даних, що передаються в мережі, для отримання з них конфіденційної інформації про користувачів. Далі така інформація можу використовуватися, наприклад, для злому акаунтів.
3. Атаки, що використовують вразливості в системі. В такому випадку в коді знаходять певні недопрацювання і стає можливим злом чи пошкодження системи. Такі атаки можуть відбуватися навіть якщо розробники виправили проблему, адже не всі користувачі одразу оновлюють програму.
4. Атаки на відмову в обслуговуванні. Один з найпоширеніших типів, бо не потребує спеціальної підготовки та майже не залежить від цілі атаки. Відбувається атака за рахунок здійснення великої кількості запитів до серверу, що призводить до відмови системи.

Зараз в Інтернеті існує багато інструментів для проведення певних комп'ютерних атак. Вони знаходяться у вільному доступі та їх використання не потребує значних навичок програмування, що значно підвищує частоту атак. Таким чином проблема їх виявлення є дуже важливою.

1.2. DDOS атаки

DDOS атака[2] це напад на комп'ютерну систему з метою зробити її недоступною для звичайних користувачів, або ускладнити доступ. Зазвичай таких атак зазнають сайти урядів, ЗМІ, інтернет-магазинів, фінансових установ. Найчастіше їх метою є вимагання грошей за припинення атаки, але іноді це політичні протести або боротьба з конкурентами.

Інструменти, що використовуються при DDOS атаках(інфраструктура, на яку вони спираються):

- Відносно стабільні ботнет-ресурси, керовані хакерськими групами. Групи злому заражають якомога більше машин спеціальними вірусами.
- Відкриті послуги. Це засоби для здійснення відбиваючих атак, тобто таких, що використовують сторонні сервіси, такі як DNS сервери.
- Публічні хмарні хости. Сервіси хмарних обчислень зростають та знижують ціни на обчислювальні та мережеві ресурси. Хакерські групи можуть використовувати ці ресурси для налаштування мереж для атак.

На рис. 1.1 можна побачити зростаючу кількість DDOS атак по роках.

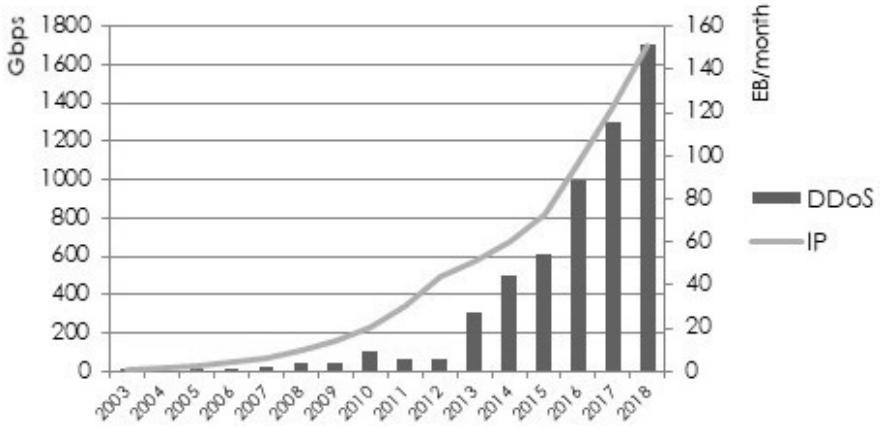


Рис. 1.1 – IP Traffic and DDOS Peak Size Development Trend[3]

Зараз DDOS атаки є одним з найпоширеніших типів комп'ютерних атак, так як можуть бути використані проти будь-якого сервісу і не залишають доказів проти зловмисників. Вони стали популярними в останні десятиліття, оскільки виникла достатня потужність для їх проведення.

1.3. Типи DDOS атак

DDOS атаки класифікують за належністю до певного рівня OSI моделі та за інструментами, які вони використовують. Наприклад, в роботі [4] розглядається класифікація на основі використаних протоколів(рис. 1.2).

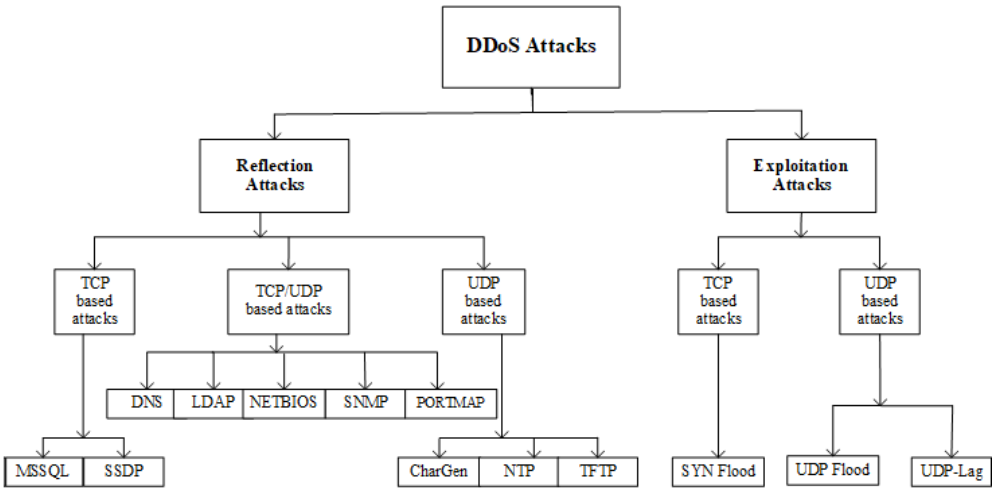


Рис 1.2 – DDoS Attack Taxonomy[4]

1.3.1. Атаки прикладного рівня

Атаки прикладного рівня націлені на веб-сторінки, що генеруються на сервері та доставляються відповідь на запити HTTP(рис. 1.3).

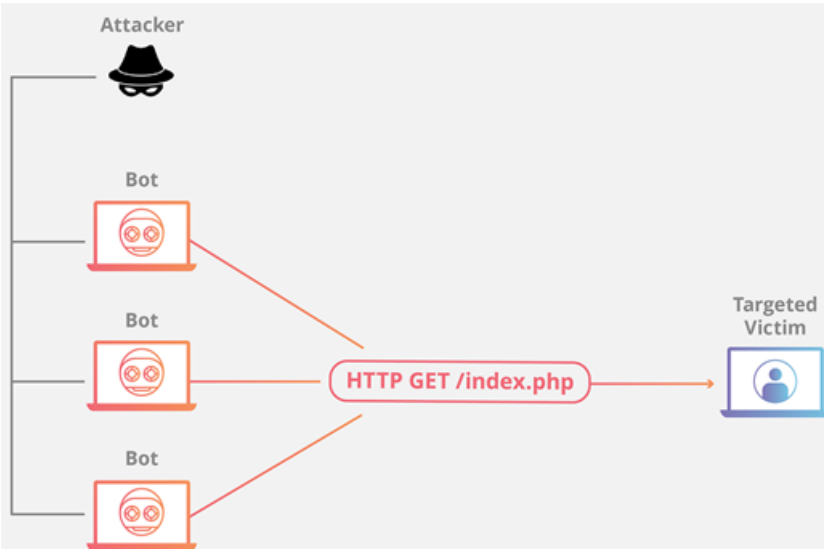


Рис. 1.3 – Application layer attack example[5]

До атак прикладного рівня належить HTTP flood. Він полягає у надсиланні

великої кількості запитів до серверу. Один HTTP-запит є обчислювально дешевим у виконанні на стороні клієнта, але для цільового сервера може бути важко реагувати, оскільки він завантажує кілька файлів і запускає запити до бази даних для створення веб-сторінки. При цьому сервер продовжує приймати нові запити.

1.3.2. Атаки протоколу

Атаки протоколу, також відомі як атаки виснаження, спричиняють злом системи через надмірне споживання ресурсів сервера або ресурсів мережевого обладнання, таких як брандмауери та балансувальники навантаження. Ці атаки використовують слабкі місця на третьому та четвертому рівні OSI моделі.

Прикладом атак протоколу є SYN flood(рис. 1.4).

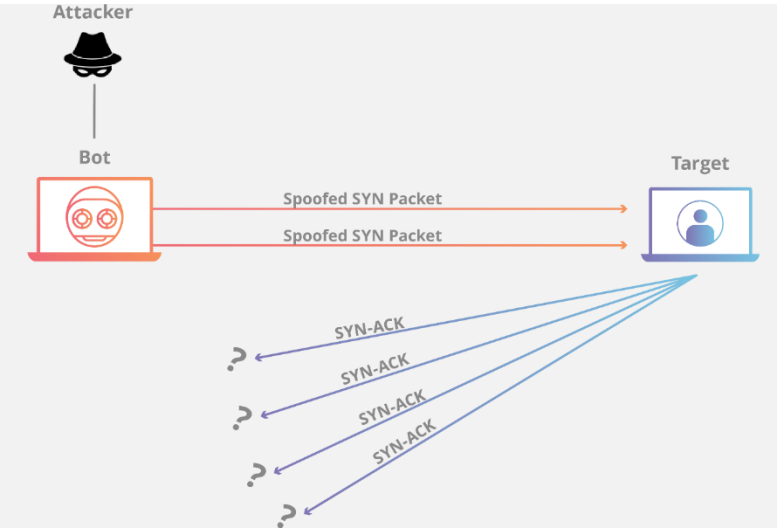


Рис. 1.4 - Protocol attack example[5]

Надсилається потік пакетів TCP, часто з підробленою адресою відправника. Кожен із цих пакетів обробляється як запит на підключення, змушуючи сервер створювати напіввідкрите з'єднання, відправляючи назад пакет TCP/SYN-ACK(пакет підтвердження), і чекаючи пакета у відповідь з адреси відправника (відповідь на пакет ACK). Оскільки IP-адреса відправника підроблена, відповідь ніколи не надходить. Ці напіввідкриті з'єднання насичують кількість доступних з'єднань, не даючи йому відповідати на звичайні запити.

1.3.3. Атаки підсилення

Атаки цієї категорії намагаються створити перевантаження, надсилаючи пакети різних протоколів та використовуючи всю доступну пропускну здатність між цільовою системою та Інтернетом. Великі обсяги даних направляються до цілі за допомогою форми посилення або іншого способу створення масового трафіку, наприклад, запитів від бот-мережі.

Прикладом атак підсилення є підсилення DNS. Надсилається запит на відкритий DNS-сервер із підробленою адресою(IP-адресою жертви), внаслідок чого цільовий хост отримує відповідь від сервера(рис. 1.5). Ця відповідь є набагато більшою запиту, що призводить до перевантаження. Такий тип атак називають “дзеркальними”, бо зловмисник не взаємодіє напряму з ціллю.

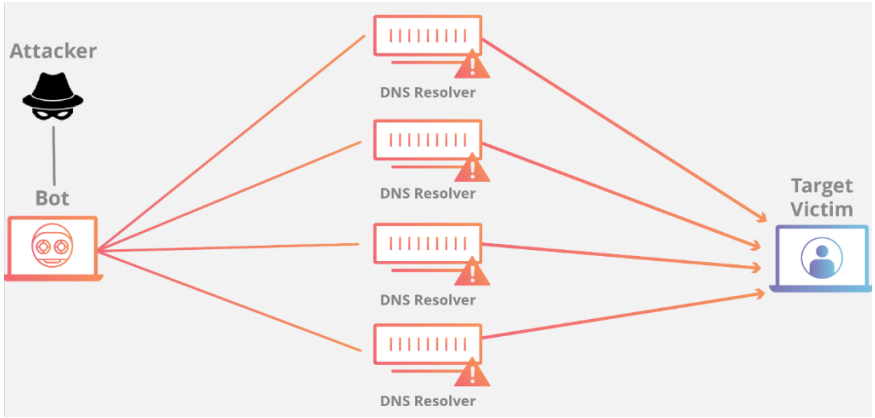


Рис. 1.5 - Amplification example[5]

Іншою атакою такого типу є UDP flood. Велика кількість пакетів UDP надсилається на хост, який має для кожного запиту відправити відповідь. Таким чином атакований хост виявляється перевантаженим. Змінивши IP-адреси в запитах, зловмисник може перенапрямити потік відповідей і зберегти працездатність атакуючих хостів, а також забезпечити їх анонімність.

Також до об’ємних атак відноситься Smurf(ICMP flood). Пакети протоколу ICMP з підробленими IP-адресами заповнює сервер жертви, що сповільнює його роботу. ICMP flood може бути реалізований з метою збору інформації про сервер (відкриті порти та адрес призначення) для організації точної атаки прикладного рівня.

1.4. Виявлення DDOS атак

Завданням виявлення DDOS атак є відмежування атак від звичайного мережевого трафіку з метою ефективного пом'якшення атаки. Для досягнення ефективності потрібно дотримуватися двох критеріїв успіху: швидкість та точність виявлення. Щоб виявити атаку, потрібно зібрати достатню інформацію про мережевий трафік, а потім провести аналіз, щоб з'ясувати, чи є трафік атакою. Цей процес може виконуватися вручну або в автоматизованому режимі. Отже, методи виявлення є ключовим фактором при формуванні надійного захисту від DDOS. Існує два основних способи виявлення DDOS атак: вбудована перевірка всіх пакетів та аналіз мережевого трафіку. Будь-який із підходів можна застосувати локально або за допомогою хмарних служб. Вбудовані інструменти виявлення DDOS мережевих пристроїв, такі як балансувальники навантаження, брандмауери або системи запобігання проникненню, могли забезпечувати виявлення коли DDOS атаки були меншими, але великі за обсягом атаки можуть перевантажити ці пристрої. Окремі системи виявлення, що працюють у мережевих каналах, є набагато ефективнішими, адже не потрібно чекати поки жертва побачить та повідомить про атаку. Виявлення здійснюється системою, яка отримує дані потоку від маршрутизаторів та комутаторів з підтримкою NetFlow, J-Flow, sFlow та IPFIX, а потім аналізує дані потоку для виявлення атак. Потім пом'якшення атак запускається вручну або автоматично за допомогою методів маршрутизації.[6]

Існує думка, що спеціальні засоби для виявлення DDOS атак не потрібні, оскільки їх неможливо не помітити. У багатьох випадках це дійсно може бути так. Проте достатньо часто спостерігаються атаки, які були помічені жертвами лише через 2-3 дні. Іноді негативні наслідки атаки виявляються в більших витратах на оплату вихідного Інтернет-трафіку.

Також для боротьби з DDOS-атаками важливим є визначення їх типу, характеру та інших характеристик. Для завчасного отримання цих даних потрібні системи виявлення.

Також для виявлення DDOS-атаки можна використовувати служби, не пов'язані з безпекою, наприклад, перенаправлення трафіку за іншими каналами зв'язку, включення резервних серверів для копіювання інформації. Таким чином, засоби для виявлення та запобігання DDOS-атаку можуть сильно відрізнятися в залежності від виду захищеної системи.

Методи виявлення DoS-атак можна розділити на кілька груп:

- сигнатурні – основані на якісному аналізі трафіка;
- статистичні – основані на кількісному аналізі трафіка;
- комбіновані – мають ознаки обох вищеназваних методів.

1.5. Захист від DDOS атак

Небезпека більшості DDOS атак в їх прозорості та схожості на запити звичайних користувачів. Адже якщо помилка в програмному забезпеченні завжди може бути виправлена, то витрата ресурсів сервісу є безповоротною. Атака, яка націлена на декілька рівнів OSI одночасно, наприклад, підсилення DNS у поєднанні з HTTP flood, є прикладом багатовекторного DDOS. Пом'якшення багатовекторної атаки вимагає різноманітних стратегій. Взагалі кажучи, чим складніша атака, тим більша ймовірність того, що її трафік буде важко відокремити від звичайного трафіку. Тож метою зловмисника є поєднання якомога більшої кількості типів, що робить зусилля щодо захисту менш ефективними.

Абсолютного рішення цієї проблеми фактично немає, проте наслідки атак і їх ефективність можна істотно понизити, правильного налаштувавши брандмауера, маршрутизатор і постійного аналізуючи аномалії в мережевому трафіку.

Основні методи захисту[6]:

- Використання фільтрувальної мережі. Така мережа приймає трафік, направлений до серверу, фільтрує його і до цільового сервера надходить тільки перевірений трафік від реальних користувачів.
- Технологія SDN. Основною ідеєю є відділення рівня управління від рівня даних у мережі. Централізована система управління мережею за допомогою SDN контролера забезпечує можливість глобального контролю мережі.
- CDN разом із розумною системою DNS допомагає пом'якшити атаки. Розумний DNS сервер розподіляє атакуючий трафік з різних місць, що робить вузол CDN центром поглинання трафіку. Після розподілу між вузлами CDN трафік може бути очищений на цих вузлах, при цьому лише нормальний трафік буде відправлений назад на вихідний веб-сайт.
- Маршрут чорної діри(Blackhole routing). Рішення полягає у створенні додаткового маршруту та спрямування трафіку на цей маршрут. У найпростішій формі, коли фільтрація реалізується без певних критеріїв обмеження, як законний, так і зловмисний мережевий трафік перенаправляється на додатковий маршрут і випадає з мережі. Це не ідеальне рішення, оскільки зловмисник досягає мети: робить мережу недоступною.
- Обмеження трафіку, тобто кількості запитів, які сервер прийме протягом певного періоду часу. Хоча обмеження швидкості є корисним для уповільнення атак, одного цього недостатньо для ефективної обробки складної атаки.

1.6. Огляд існуючих рішень

Системи виявлення DDOS атак на основі машинного навчання розглядаються в багатьох наукових роботах. Будуть розглянуті деякі з них[7].

1.6.1. Системи на основі дерев рішень

Dewan Md. Farid та ін. [8] у своїй роботі пропонують алгоритм виявлення вторгнень на основі дерев рішень. Дерево рішень використовується як прогностична модель, в якій містяться спостереження за станом об’єкту(трафіку). Набір даних моделюється у вигляді такого дерева рішень, і коли новий елемент даних надається для класифікації, він буде класифікований відповідно до попереднього набору даних.

Дерево рішень може бути побудоване з великого обсягу даних з багатьма характеристиками, оскільки розмір дерева не залежить від розміру набору даних. Дерево складається з вузлів(node) та листя(leaf). Кожен вузол містить правила, за допомогою яких дані мають бути розділені. Листи визначають рішення для прикладів, що в них потрапляють.

Структура роботи алгоритму: алгоритм ініціалізує ваги для кожного прикладу набору даних; потім оцінює попередню ймовірність для кожного класу підсумовуючи як часто кожен клас зустрічається в наборі даних. Аналогічно визначається ймовірність для кожного атрибута даних. Після цього ці ймовірності використовуються для оновлення ваг для кожного прикладу в наборі даних. Далі алгоритм будує дерево рішень, яке можу бути використане для класифікації наступних даних.

В даній роботі було використано набір даних KDD99. Дані в наборі класифікуються як один клас нормальної поведінки та чотири класи атак: probing, DOS, U2R та R2L. В наборі є 41 вхідний атрибут для кожного підключення до мережі, які мають дискретні або неперервні значення і розділені на три групи. Перша група атрибутів це основні особливості мережевого підключення, які включають тривалість, прототип та деякі прапори у з'єднаннях TCP. Друга група атрибутів складається з особливостей вмісту мережеві з'єднання, а третя група складається з статистичних ознак, які обчислюються за часовим проміжками.

1.6.2. Метод опорних векторів

SVM це метод навчання з вчителем, що використовуються для класифікації та регресії. На основі набору навчальних прикладів, кожен з яких відноситься до однієї з двох категорій, алгоритм будує модель, яка передбачає, чи потрапляє новий приклад в одну з двох категорій.

Vipin Das та ін. [9] провели експеримент з виявлення атак за допомогою RST та SVM. Запропонований алгоритм збирає пакети з мережі кожні 4 секунди. Далі RST використовується для попередньої обробки даних. Зокрема, було виділено 14 основних характеристик. Набір характеристик, вибраний RST надсилається в модель SVM для вивчення та тестування відповідно.

За допомогою алгоритму визначається оптимальний варіант класифікації. Наприклад, на рис. 1.6 є багато можливих лінійних класифікаторів, які можуть розмежувати дані різних типів, але існує лише один, який максимізує відступ (відстань між ним і найближчою точкою даних кожного класу). Цей лінійний класифікатор називається оптимальною роздільною гіперплощиною.

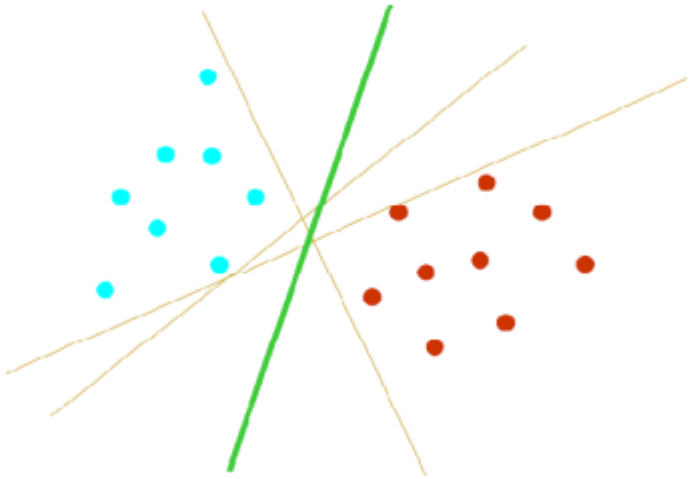


Рис 1.6. Optimal Separating Hyper Plane[8]

Використання цього методу дозволило правильно виявити 98.7% пакетів атак і попередити про це адміністратора. Відповідно до рішення адміністраторів дані трафіку зберігаються для подальшого використання.

1.6.3. Множинна лінійна регресія

Swathi Sambangi та ін. [10] пропонують підхід до виявлення DDoS атак з використанням множинної лінійної регресії. Лінійна регресія використовується коли є дві неперервні змінні – незалежна та залежна. Незалежна змінна використовується для обчислення результату. Множинна регресії полягає у передбаченні на основі декількох незалежних змінних.

Модель множинної регресії базується на таких припущеннях:

- між залежними та незалежними змінними існує лінійна залежність
- незалежні змінні не надто корелюють між собою
- дані відбираються незалежно та випадково з загального об'єму даних

В даній роботі був використаний набір даних CICIDS 2017. Початкова кількість атрибутів дорівнювала 78 з останнім атрибутом мітки класу, тобто разом із міткою класу 79 атрибутів. Міткою класу є доброякісні(звичайні) запити або DDoS(атаки).

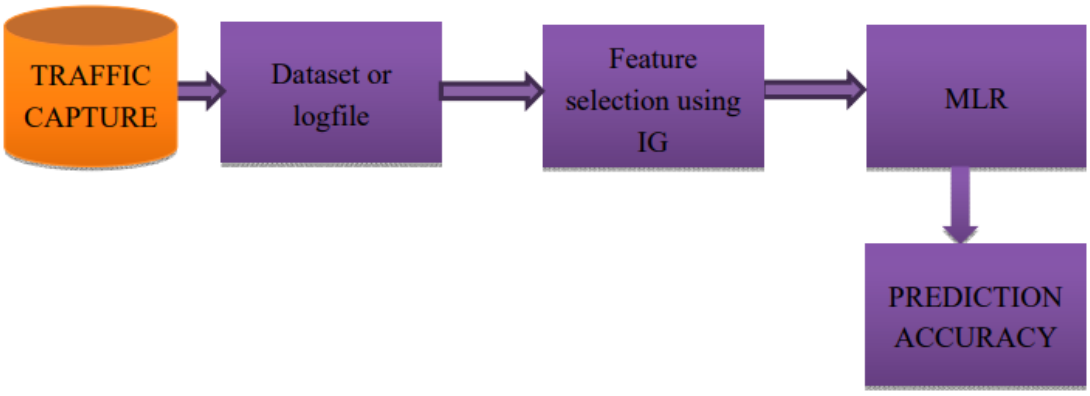


Рис. 1.7. Proposed Machine Learning approach for DDoS attack detection[9]

Далі був застосований алгоритм вибору ознак, який базується на обчисленні критерію приросту інформації (Information Gain) для кожного з атрибутів набору даних. Найкращі 16 атрибутів були розглянуті для збережені, інші були видалені. Усього кількість пакетів трафіку включала 225 746 записів. Точність моделі множинної лінійної регресії була отримана рівною 97,86%.

1.6.4. Наївний баєсів класифікатор

Sundar Raghavan та ін.[11] розглядають метод виявлення виявлення DDoS за допомогою наївного баєсівського класифікатора. Метод полягає застосуванні теореми Байєса з сильними (наївними) припущеннями про незалежність між ознаками. Теорема полягає в обчисленні ймовірності для двох(або більше) незалежних подій за формулою (1.1).

$$P(A,B) = P(A) * P(B) \quad (1.1)$$

Основним недоліком даного методу є те, що характеристики ніколи не виявляються незалежними, що протирічить теоремі. Але на практиці використання методу може давати задовільні результати.

Класифікатори NB можуть працювати з невеликими обсягами навчальних даних, а також може приймати велику кількість атрибутів, що робить їх непоганим вибором для мережевого моделювання для DDoS атак.

Для класифікації були використані заголовки пакетів TCP. Наступні параметри були досліджені:

1. Прапори TCP
2. Розмір пакету
3. Номер порту
4. IP-адреса
5. Міжпакетні проміжки часу
6. Загальний час з'єднання до поточного пакета
7. Кількість пакетів у з'єднанні

Було визначено, що параметри 6 і 7 мають сенс лише при використанні повного обсягу даних послідовності з'єднання, і вони були виключені з набору. Експериментально було встановлено, що TCP прапори найкраще підходять серед решти атрибутів.

1.6.5. Генетичний алгоритм

В роботі [12] Anurag Andhare та ін. запропонували вирішення проблеми виявлення за допомогою генетичний алгоритму. Генетичний алгоритм - це техніка програмування, яка імітує біологічну еволюцію як стратегію вирішення проблем. Він заснований на принципі еволюції найбільш придатних для оптимізації сукупності рішень-кандидатів.

GA використовує хромосомну структуру даних і еволюцію хромосом за допомогою селекції, кросоверу та операторів мутації. На рис. 1.9 показано структуру простого генетичного алгоритму.

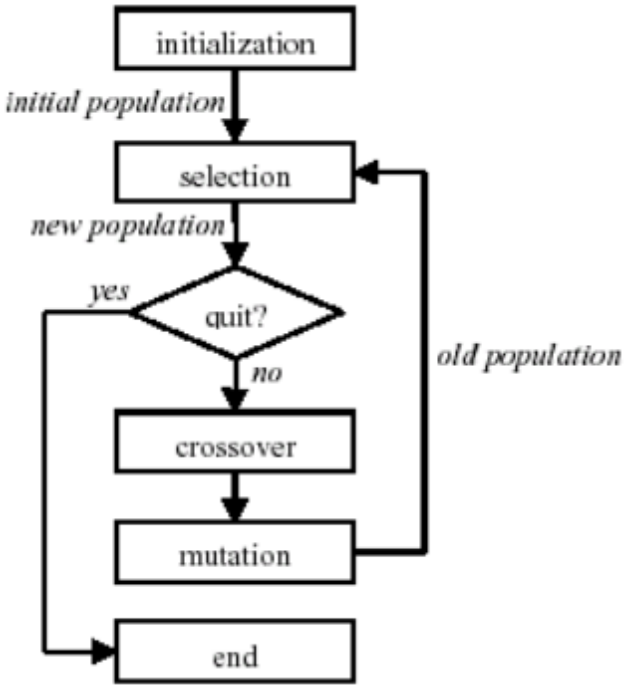


Рис 1.8. Working of Genetic Algorithm [11]

Процес починається із випадково сформованої популяції хромосом, яка представляє усі можливі рішення розглянутої задачі. Хромосоми поділяються на гени, закодовані як біти, символи або цифри. "Фітнес-функція" використовується для розрахунку придатності кожної хромосоми відповідно до бажаного рішення. Під час оцінки два основні оператори, кросовер і мутація, використовуються для імітації природного розмноження та мутація видів.

Для навчання був використаний набір даних KDDCUP 99. В ньому містяться дані про підключення TCP / IP, описані 41 дискретним і неперервними характеристиками. Підключення позначені як звичайний трафік або як один з 4 видів атак: DDOS, U2R, R2L, probing. Для тренування було обрано 65535 записів(приблизно 10% з кожного набору). Із загальної кількості вторгнень при використанні такого генетичного алгоритму у наборі тестування було виявлено більше ніж 97%. Також, якщо динамічно обробляти дані, що надходять з брендмауеру, цей підхід є досить ефективним для виявлення атак нових типів.

1.6.6. Нейронні мережі

В роботі Donghwoon Kwon та ін. [13] досліджується виявлення аномалій на основі глибинного навчання нейронної мережі. Штучні нейронні мережі це такі математичні моделі та їх програмні реалізації, побудовані за принципом організації й функціонування біологічних нейронних систем – мереж нервових клітин живих організму.

Багато різних архітектур нейронних мереж застосовується для виявлення вторгнень, найпростішою з яких є MLP.

Багатошаровий перцептрон – мережа з одиничних перцептронів, які є простими обчислювальними моделями, що нагадують біологічні нейрони. Мережа складається щонайменше з трьох повністю з'єднаних шарів перцептронів: вхідний, прихований і вихідний рівень(рис. 1.9).

При навчанні MLP зазвичай використовується метод зворотного поширення помилки, заснований на вхідних та вихідних даних мережі. Помилка між передбаченим та обчисленим результати повертаються до попередніх шарів мережі. За належної кількості нейронів і прихованих шарів, MLP дає досить точне наближення функції між вхідними та вихідними даними.

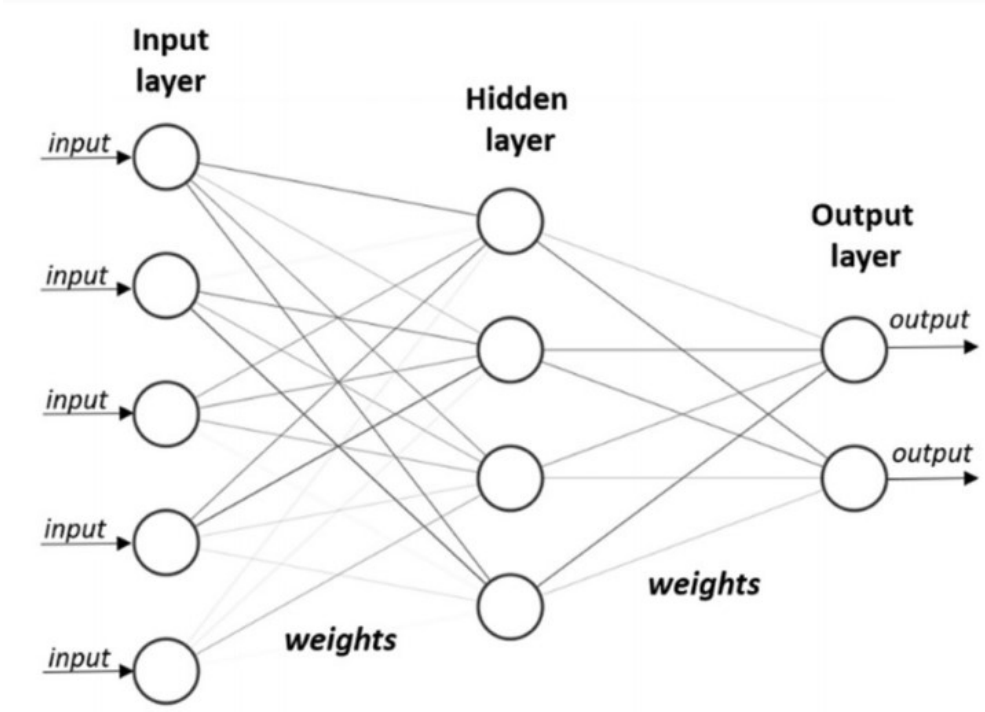


Рис. 1.9. The simple architecture of a three layer feed-forward neural network[14]

Нейронні мережі за рахунок великої кількості нейронів можуть давати вислкі показники ефективності. Але до їх недоліків відноситься те, що навчання нейронні мережі в деяких випадках призводить до тупикових ситуацій; великі часові витрати на виконання навчання не дозволяють застосовувати їх в системах реального часу; поведінка навченої мережі не завжди може бути однозначно передбачувано, що збільшує ризик застосування для управління важливими технічними об'єктами.

В даній роботі був використаний набір даних NSL-KDD. Була отримана точність виявлення 90.4%.

1.7. Порівняння рішень

Для визначення кращого методу для вирішення поставленої задачі було проведено порівняльний аналіз описаних існуючих рішень. Результати аналізу представлені в таблиці 1.1.

Таблиця 1.2

Порівняння існуючих систем виявлення DDOS атак

№	Рішення	Набір даних, кількість атрибутів	Типи атак	Отримана точність	Особливості
1	Дерево рішень[8]	KDD99, 41	DDOS, U2R, R2L, probing	98.76%	Найвища отримана точність
2	Метод опорних векторів[9]	KDD99, 14	Benign/ Attack	98.7%	Поєднання алгоритмів RST та SVM
3	Множинна лінійна регресія[10]	CICIDS 2017, 16	Benign/ attack	97,86%	Точність змінювалася від 73,79 до 97,86%
4	Наївний Байєс[11]	Потік TCP пакетів, 6	Benign/ attack	98.3%	Змінні не є незалежними
5	Генетичний алгоритм[12]	KDDCUP 99, 41	DDOS, U2R, R2L, probing	97%	Здатність алгоритму змінюватися залежно від типу атаки
6	Нейронна мережа[13]	NSL-KDD	DDOS, U2R, R2L, probing	90.4%	Довгий етап тренування

Висновок до розділу 1

В результаті проведених досліджень, направлених на аналіз проблеми виявлення DDOS атак, огляду їх типів та збір даних про поширені алгоритми машинного навчання можна прийти до наступних висновків:

1. Виявлення DDOS-атак є нетривіальною задачею. Вона не може бути задовільно вирішеною за допомогою стандартних алгоритмів, адже використання хакерами різних типів атак робить складним відрізнення атак від звичайних запитів.
2. Із дослідження методів виявлення атак було вирішено, що методи машинного навчання можуть покращити результати за рахунок тренувань на попередніх атаках.
3. Було проведено аналіз різних методів машинного навчання. Покладено за мету реалізувати деякі з них для аналізу можливості їх використання в задачах виявлення.
4. Розглянуто існуючі рішення поставленої задачі і визначено, що вони не задовольняють всіх необхідних умов, зокрема не всі є в загальному доступі, що робить неможливим їх використання.
5. Вирішено реалізувати рішення проблеми на основі алгоритмів машинного навчання.

РОЗДІЛ 2. ІНСТРУМЕНТИ РЕАЛІЗАЦІЇ СИСТЕМИ ВИЯВЛЕННЯ

2.1. Задача виявлення DDOS атак

Для ефективної розробки алгоритму виявлення DDOS атак на основі машинного навчання спершу необхідно розглянути математичну модель виявлення із якою необхідно буде мати справу.

Формальна постановка задачі виявлення DDOS атак:

Нехай дано деяку множину вхідних даних про трафік X і множина відповідей Y. Множина відповідей Y позначає приналежність даного прикладу до атаки. Є деяка навчальна вибірка $\{x_1, \ldots, x_n\} \subset X$ та вибірка відомих відповідей $\{y_1, \ldots, y_n\} \subset Y$.

Задачою виявлення атак є знаходження такого визначаючого правила(моделі або алгоритму), що дає найкраще наближення до правильних відповідей на всій множині X:

$a: X \to Y$

$x_i \subset X$ в даному випадку є набором ознак даних про трафік, $y_i \subset Y$ позначає приналежність даного прикладу до атаки. Ознаки поділяються на типи:

1. булеві, область значень яких {0, 1};
2. номінальні, область значень яких – скінченна підмножина N;
3. порядкові, що представляють собою номінальні ознаки, для яких визначено лінійний порядок;
4. кількісні, значенням яких є дійсне число.

Прикладом булевої ознаки є належність певного прикладу до вхідного(1) або вихідного(0) трафіку. Номінальною ознакою є протокол, до якого належить надісланий пакету. В якості прикладу порядкової ознаки можна привести кількість флагів у заголовку пакету. Кількісною ознакою є розмір надісланого пакету.

Навчальна вибірка будується з певного набору даних про трафік у вигляді матриці ознак, рядки якої є окремими прикладами трафіку та відповідними відомими станами системи.

Множина відповідей також може складатися з ознак різних типів:

1. булеві – Y = {0, 1}, звичайний трафік = 0 і атака = 1;
2. номінальні – Y = {0…M}, різні числа залежно від типу атаки;
3. номінальні, що перетинаються – Y = {0, 1}^M, приклад трафіку може належати одразу до декількох типів атак.

Особливістю двох останніх варіантів є те, що окремо має бути визначено, чи є даний приклад трафіку атакою, а потім вже визначено тип атаки.

Отже, був наданий формальний опис задачі виявлення, що дає змогу розробити оптимальний алгоритм вирішення цього завдання.

2.2. Машинне навчання як метод вирішення поставленої задачі

Машинне навчання можна визначити[15] як сукупність обчислювальних методів, що використовують досвід для підвищення продуктивності або для створення точних прогнозів. Під досвідом в даному випадку мається на увазі попередня інформація, тобто навчальні дані, як правило, у формі електронних даних, зібраних та наданих для аналізу. Прогнозування в даному випадку відбувається без явного програмування, а лише на основі наданих даних. Якість і розмір даних мають вирішальне значення для успіху зроблених передбачень.

Машинне навчання полягає у розробці ефективних і точних алгоритмів прогнозування. Як і в інших областях інформатики, критичною мірою якості цих алгоритмів є їх часово-просторова складність. Але в машинному навчанні додатково знадобиться поняття складності вибірки даних, які потрібно аналізувати.

Стандартними завданнями машинного навчання є:

- класифікація: присвоєння категорії кожному елементу з набору даних;
- регресія: проблема прогнозування деякого значення для кожного елементу.
- рангування: впорядкування елементів за певним критерієм.
- кластеризація: розподіл набору елементів на однорідні підмножини.
- зменшення розмірності: перетворення початкового набору даних у набір меншого розміру зі збереженням деяких властивостей.

Далі наводиться перелік визначень та термінології, що зазвичай використовується в машинному навчанні:

- приклади: це елементи або екземпляри даних, що використовуються для навчання чи оцінювання;
- параметри: набір атрибутів, часто представлених у вигляді вектора, який представляє собою приклад;
- мітки: значення або категорії, присвоєні прикладам. Саме мітки є результатом функції прогнозування;
- навчальні дані: набір прикладів, що використовуються для навчання алгоритму;
- приклад перевірки: приклади, що використовуються для налаштування параметрів алгоритму навчання;
- тестовий зразок: приклади, що використовуються для оцінки ефективності алгоритму навчання;
- функція збитків: функція, яка вимірює різницю або втрату між передбачуваною міткою та справжньою міткою.

Побудову моделі машинного навчання можна розділити на такі етапи[16]:

1. Збір та підготовка даних: основою для побудови моделі є збір та підготовка даних у форматі, який може бути наданий в якості вхідних даних для алгоритму. Дані зазвичай неструктуровані і містять багато шуму, тож дані потрібно очистити та попередньо обробити.

2. Вибір параметрів: дані, отримані з вищевказаного кроку, можуть містити численні параметри, не всі з яких мають значення для навчального процесу. Ці параметри потрібно видалити та отримати підмножину найважливіших параметрів.
3. Вибір алгоритму: не всі алгоритми машинного навчання можуть бути використані для вирішення будь-яких завдань. Певні алгоритми більше підходять для певних задач. Вибір найкращого алгоритму машинного навчання для даної задачі є обов'язковим для отримання найкращого результату.
4. Вибір моделей та параметрів: більшість алгоритмів машинного навчання вимагають певної початкової корекції параметрів для єфективнішої їх обробки(наприклад, нормалізація).
5. Навчання: після вибору відповідного алгоритму та відповідних значень параметрів модель навчається, використовуючи частину набору даних як навчальні дані.
6. Оцінка ефективності: перед впровадженням системи, модель повинна бути протестована на інших даних(не навчальних), щоб оцінити якість за допомогою таких параметрів, як accuracy(відсоток правильних відповідей), precision(точність) та recall(повнота).

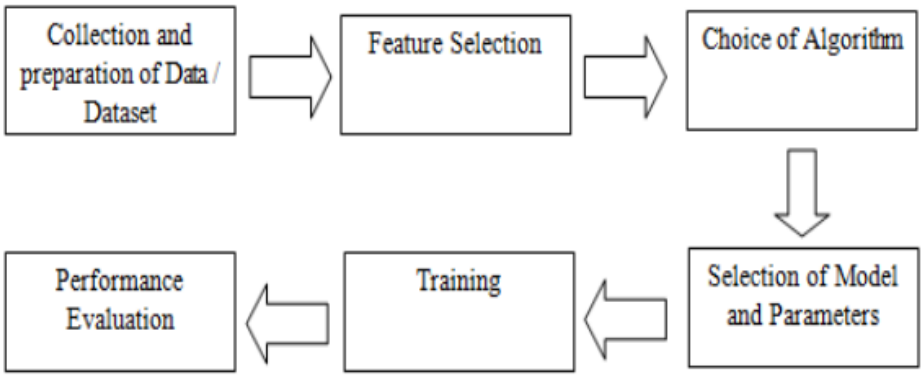


Рис. 1.10. Components of a Generic ML model[16]

Варто зауважити, що модель даних при виявленні атак схожа на модель даних в алгоритмах машинного навчання, тож цей метод є досить зручним в даному випадку.

2.3. Вибір методу машинного навчання

Для реалізації алгоритму виявлення було обрано метод логістичної регресії. Логістична регресія[17] – різновид множинної регресії, призначення якої полягає в аналізі взаємозв'язку між декількома незалежними змінними (регресорами або предикторами) і певною залежною змінною. Використання регресії полягає у побудові моделі та подальшому прогнозуванні. На вхід алгоритму надходять незалежні змінні $x = [x_1, x_2, \ldots, x_m]^T \in R^m$ а виходом є залежні змінні $y = [y_1, y_2, \ldots, y_n]^T \in R^n$. Бінарна логістична регресія застосовується в разі, коли залежна змінна є бінарною (тобто може приймати тільки два значення). За допомогою даного алгоритму можна оцінювати вірогідність того, що подія настане для конкретного випробуваного.

Регресійні моделі можуть бути описані за допомогою формули

$$y = F(x_1, x_2, \ldots, x_m)$$

В множинній лінійній регресії залежна змінна є лінійною функцією незалежних змінних:

$$y = w_1x_1 + w_2x_2 + \ldots + w_mx_m + b \quad (2.1)$$

Однак така формула не може бути використана для бінарної класифікації, адже результати передбачення не обов’язково приймаються значення 0 та 1. Логістична регресія вирішує цю проблему за допомогою передбачення безперервної змінної зі значеннями з відрізку [0,1] при будь-яких значеннях незалежних змінних. Такий результат досягається застосуванням наступного регресійного рівняння:

$$P = \frac{1}{1 + e^{-y}} \quad (2.2)$$

де P – ймовірність того, що відбудеться необхідна подія, e – основа натуральних логарифмів, y – стандартне рівняння регресії(2.1).

Залежність, що зв'язує ймовірність події і величину y, показана на рис. 2.1.

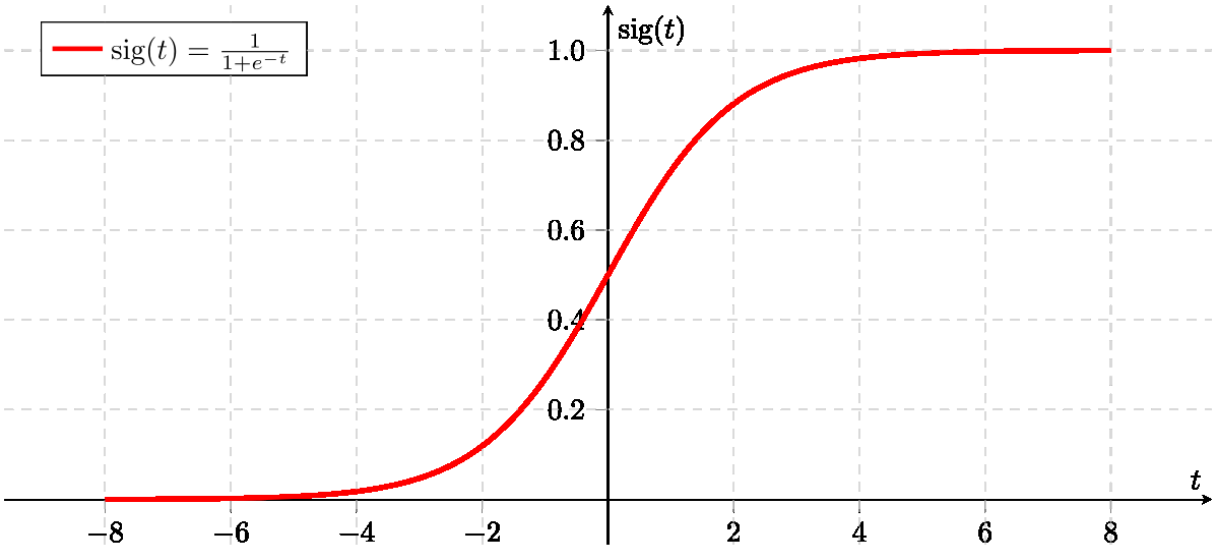


Рис. 2.1. Sigmoid Activation Function [18]

Для знаходження оптимальних вагів моделі використовується функцією витрат (CF). Задача тренування полягає у мінімізації певної функції витрат. Функція витрат в логістичній регресії базується на перехресній ентропії та має наступний вигляд:

$$\boldsymbol{Cost(h(x), y) = \begin{cases} -log\big(h(x)\big), & y = 1 \\ -log(1 - h(x)), & y = 0 \end{cases}} \tag{2.3}$$

Вона приймає *h(x)* як функцію моделі та *y* як істинні значення.

Дана функція визначає розмір похибки при передбаченні значення h(x), тоді як фактичне значення дорівнює y. У випадку, коли y = 1, похибка наближається до 0, за умови що h(x) наближається до 1. І навпаки, похибка зростає до нескінченності, коли h(x) наближається до 0 (рис. 2.2). Це є необхідною властивістю: потрібно отримувати більшу похибку коли алгоритм передбачає значення, що значно відрізняються від фактичних. Аналогічна ситуація при y = 0 (рис. 2.3).

Дві функції з формули (2.2) можуть бути зведені до однієї загальної формули:

$$\boldsymbol{Cost(h(x), y) = -y * log\big(h(x)\big) - (1 - y) * log(1 - h(x))} \tag{2.4}$$

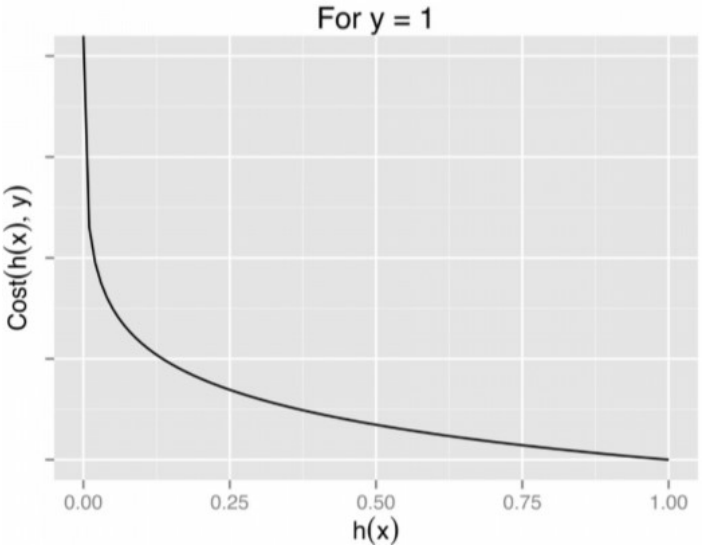


Рис. 2.2. Cost function of logistic regression if y = 1[19]

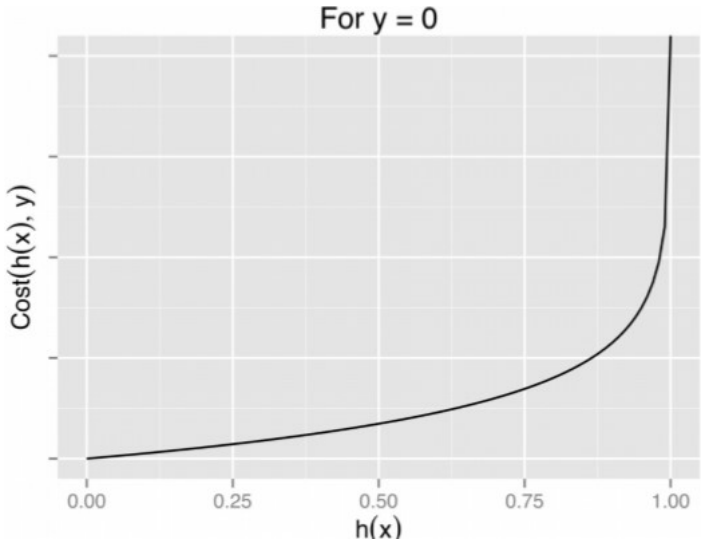


Рис. 2.3. Cost function of logistic regression if y = 0[19]

Для мінімізації такої функції використовується метод градієнтного спуску(gradient descent)[20]. Метод полягає у поступовій зміні вагів моделі залежно від якості передбачення результату. Тобто необхідно застосувати градієнтний спуск до кожного параметру:

$$\boldsymbol{\theta_j} = \boldsymbol{\theta_j} - \boldsymbol{\alpha} \frac{\boldsymbol{d}}{\boldsymbol{d\theta}} \boldsymbol{Cost}(\boldsymbol{\theta_j}) \quad (2.5)$$

Таким чином можна отримати оптимальні значення вагів, а значення функції витрат дозволяє визначати якість тренування.

2.4. ROC аналіз

ROC-крива[21] – крива, що використовується для представлення результатів бінарної класифікації в машинному навчанні. Оскільки класів два, вони називаються позитивним і негативним. Дана крива демонструє залежність кількості вірно класифікованих позитивних випадків від кількості невірно класифікованих негативних. У випадку виявлення DDOS атак позитивною вважається подія атаки.

При цьому передбачається, що класифікатор має деякий параметр, змінюючи який, отримується те чи інше розбиття на два класи. Такий параметр називають порогом, або точкою відсікання (cut-off value). Залежно від його значень отримуються різні величини помилок I і II роду.

Структуру помилок I і II роду можна визначити з таблиці спряженості (confusion matrix), яка будується на основі результатів класифікації і фактичної (об'єктивної) приналежності прикладів до класів.

Таблиця 2.1 Таблиця спряженості

Модель	Фактично позитивно	Фактично негативно
Позитивно	TP	FP
Негативно	FN	TN

TP (True Positives) – позитивні приклади, що були класифіковані вірно (істинно позитивні випадки).

TN (True Negatives) – негативні приклади, що були класифіковані вірно (істинно негативні випадки).

FN (False Negatives) – позитивні приклади, що були класифіковані як негативні (помилка I роду).

FP (False Positives) – негативні приклади, що були класифіковані як позитивні (помилка II роду).

Але при аналізі результатів замість абсолютних показників частіше оперують відносними – частками (rates), що виражені у відсотках:

Частка істинно позитивних прикладів (True Positives Rate), яка є такою властивістю класифікатора, як чутливість (Sensitivity):

$$S_e = TPR = \frac{TP}{TP+FN} * 100\% \quad (2.6)$$

Частка невірно позитивних прикладів (False Positives Rate):

$$FPR = \frac{FP}{TN+FP} * 100\% \quad (2.7)$$

Специфічність (Specificity) – частка істинно негативних випадків, що були вірно ідентифіковані моделлю:

$$S_p = 100 - FPR = \frac{TN}{TN+FN} * 100\% \quad (2.8)$$

Об'єктивна цінність будь-якого бінарного класифікатора визначається за допомогою значень чутливості і специфічності моделі

Для побудови ROC-кривої(рис. 2.4) для кожного значення порога відсікання з певним кроком розраховуються значення з таблиці спряження.

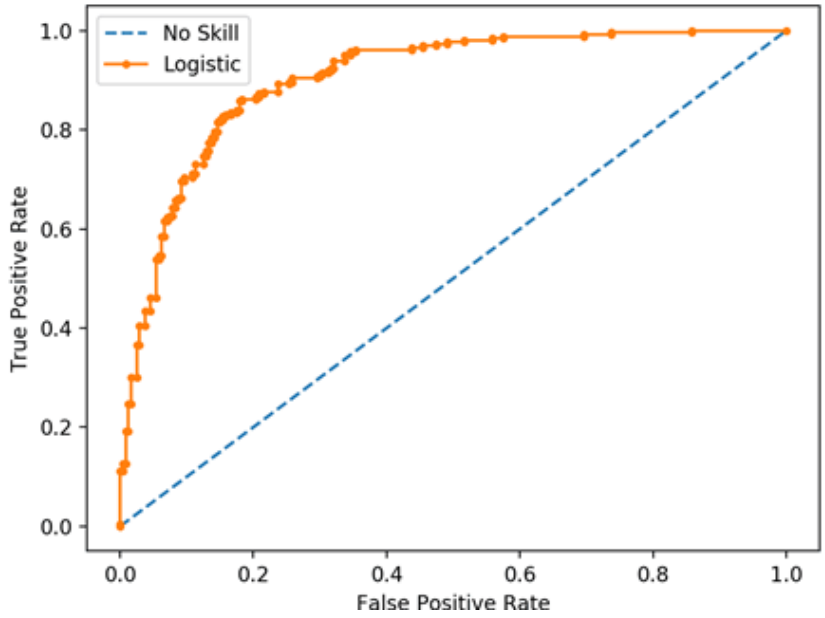


Рис. 2.4. ROC Curve Plot[22]

Для ідеального класифікатора графік ROC-кривої проходить через верхній лівий кут, де частка істинно позитивних випадків становить 100% або 1,0 (ідеальна чутливість), а частка хибно позитивних прикладів дорівнює нулю.

2.5. Характеристики для виявлення DDOS атак

Для тренування моделі було використано набір даних DDoS Evaluation Dataset (CIC-DDoS2019) від The University of New Brunswick. Даний набір даних складається с декількох фалів, кожен я яких відноситься до певного типу атаки, з яких були обрані найбільш відомі: LDAP, MSSQL, NetBIOS, Portmap, Syn, UDP.

Записи в наборах складаються з 88 характеристик, остання з яких відноситься до типу атаки чи її відсутності. Після аналізу цих характеристик було визначено, що багато з них майже завжди мають однакові значення або ж не мають важливого значення для опису трафіку. Після цього було залишено 29 основних характеристик.

Для цих характеристик було застосовано метод визначення найбільш важливих характеристик. Він полягає у тренуванні та тестуванні моделі для різних наборів характеристик, після чого визначається набір з найкращими показниками якості передбачення. Набори відрізняються як за кількістю характеристик, так і за обраними характеристиками. Найкращий набір використовується для подальшого тренування моделі. Після застосування цього методу було залишено наступні характеристики:

Табл. 2.3 Характеристики

1	Source IP	IP відправлення
2	Source Port	Порт відправлення
3	Destination IP	IP призначення
4	Destination Port	Порт призначення
5	Protocol	Протокол
6	Time	Час з останнього запиту
7	Flow Duration	Тривалість потоку
8	Total Fwd Packets	Кількість вхідних пакетів
9	Total Backward Packets	Кількість вихідних пакетів
10	Total Length of Fwd Packets	Загальна довжина вхідних пакетів
11	Total Length of Bwd Packets	Загальна довжина вихідних пакетів
12	Fwd Packet Length Max	Максимальна довжина вхідних пакетів
13	Fwd Packet Length Min	Мінімальна довжина вхідних пакетів
14	Fwd Packet Length Mean	Середня довжина вхідних пакетів

Кожен файл набору складається в середньому з 400000 записів щодо трафіку. Для репрезентативності було взято по 10000 записів з кожного набору та сформовано загальний набір для тренування.

Дані були додатково оброблені для зручнішого використання. По-перше, набір було очищено від усіх не числових даних(строки та значення nan), адже обрана модель логістичної регресії працює з числами.

Час кожного запиту, який міститься в наборі даних, було перетворено в час між двома сусідніми запитами. Адже кожен тип атаки відбувався в окремий час, тож дані про дату і час тільки заплутають алгоритм. Дані були спеціально сформовані вченими, тож час проведення атак є випадковим і не впливає на наявність атаки.

Логістична регресія найкраще працює для бінарної класифікації, тож класи атак були приведені до двох основних значень: наявність чи відсутність атаки, відповідно 1 і 0.

Також було нормалізовно дані. Нормалізація полягає у зведенні даних до певного діапазону значень, а саме [0;1].

Це необхідно для якіснішої обробки даних, адже якщо значення однієї характеристики на порядок більші за значення іншої, то вона буде враховуватися алгоритмом як більш важлива для результату. Нормалізація відбувається за наступною формулою:

$$\boldsymbol{X} = \frac{\mathbf{X} - \boldsymbol{X_{min}}}{\boldsymbol{X_{max}} - \boldsymbol{X_{min}}} \tag{2.9}$$

Для використання даних їх було розділено на дві частини: тренувальну та тестову. Вони складають відповідно 80 та 20% від набору даних. Це необхідно для правильної оцінки результатів передбачення моделі. Адже модель може бути добре навчена для прикладів з тренувального набору, але давати погані результати при інших даних.

2.6. Вибір засобів розробки

2.6.1. Мова програмування

Python[23] – високорівнева мова програмування загального призначення. Python підтримує кілька парадигм програмування, включаючи структуроване, об'єктно-орієнтоване та функціональне програмування.

В дані роботі використовується остання версія мови Python 3.0, випущена в 2008 році і не є повністю сумісною з кодом попередньої Python 2. Python стабільно входить до числа найпопулярніших мов програмування.

До основних переваг мови відносять:

- Універсальність. Python існує досить довго, тож було створено спеціальні бібліотеки для різних цілей: NumPy для багатовимірних масивів та високорівневих матриць, SciPy для інженерних розрахунків, Pandas для дослідження, аналізу та маніпуляції даними, Scikit-Learn для роботи з штучним інтелектом.
- Простота. Мова має зручний, зрозумілий сисntаксис, схожий на людську мову. Нестрога типізація дозволяє менше звертати увагу на перетворення типів даних.
- Популярність мови забезпечила велику кількість посібних матеріалів, де можна знайти всю необхідну інформацію.

2.6.2. Бібліотеки

Pandas[24] – найважливіший інструмент для обробки та аналізу даних в Python. Дана бібліотека працює з такими типами даних, як Series(вектори) DataFrame(матриці, або ж таблиці даних). Pandas дозволяє легко працювати з різними джерелами даних: файли у форматі csv або xlsx, бази даних. Дані з джерел автоматично форматуються в DataFrame та одразу можуть бути оброблюваними. Аналогічно вони можуть бути записані назад.

Для обробки даних є інструменти для їх очищення та фільтрації, наприклад знаходження середніх, максимальних, мінімальних значень, медіани та ін.

NumPy[25] – це бібліотека Python, яка забезпечує використання багатовимірних масивів, різних похідних об'єктів (наприклад, маскованих масивів та матриць) та асортименту підпрограм для швидких операцій над масивами, включаючи математичні, логічні, маніпуляції з фігурами, сортування, вибір, введення/виведення, дискретні перетворення Фур'є, основна лінійна алгебра, основні статистичні операції, випадкове моделювання та багато іншого.

В основі пакета NumPy лежить об'єкт ndarray. Він інкапсулює n-вимірні масиви однорідних типів даних, причому багато операцій виконуються в скомпільованому коді для продуктивності. Існує кілька важливих відмінностей між масивами NumPy та стандартними масивами Python:

- Масиви NumPy мають фіксований розмір при створенні, на відміну від списків Python (які можуть динамічно зростати). Зміна розміру ndarray створить новий масив і видалить оригінал.
- Всі елементи масиву NumPy повинні мати однаковий тип даних, і, отже, матимуть однаковий розмір у пам'яті.
- Масиви NumPy пристосовані до вдосконалених математичних та інших типів операцій над великою кількістю даних. Зазвичай такі операції виконуються ефективніше і з меншим кодом, ніж це можливо за допомогою вбудованих списків Python.
- Велика кількість науково-математичних пакетів на основі Python використовують масиви NumPy; хоча вони, як правило, підтримують введення списків Python, вони перетворюють такий вхід у масиви NumPy перед обробкою.
- Швидкість обробки даних, що є особливо важливим для наукових обчислень.

Отже, обрані бібліотеки надають всі необхідні інструменти для реалізації програмних компонентів.

PySimpleGUI[26] – це пакет Python, який дозволяє створювати графічні інтерфейси. Інетрфейси створюються за допомогою макетів та віджетів, що значно спрощує процес розробки. Інструменти пакету реалізують більшу частину типового коду для створення інтерфейсів та є досить простими в використанні.

2.6.3. Середовище розробки

PyCharm[27] – це інтегроване середовище розробки (IDE), що використовується в комп'ютерному програмуванні, зокрема для мови Python. Він забезпечує аналіз коду, графічний графічне відображення, інтегроване тестування модулів, інтеграцію із системами контролю версій, а також підтримує веб-розробку з Django та аналіз даних з Anaconda. PyCharm є крос-платформним, з версіями для Windows, macOS та Linux.

Корисними функціями даного IDE є:

- Допомога та аналіз кодування з доповненням коду, виділенням синтаксису та помилок, інтеграцією лінтерів та швидкими виправленнями.
- Навігація проектами та кодами: спеціалізовані системи роботи з проектами, та файлів та швидке переключення між файлами, класами, та методами.
- Рефакторинг: включає перейменування, метод вилучення, введення змінної, введення константи, ымпорт даних.
- Інтегроване модульне тестування з покриттям коду за рядками.
- Інтеграція контролю версій: уніфікований користувальницький інтерфейс для Mercurial, Git, Subversion, Perforce та CVS зі списками змін та злиттям.
- Підтримка наукових інструментів, таких як matplotlib, numpy та scipy.

Тож дане середовище розробки дозволяє ефективно розробляти програмні компоненти, зокрема реалізацію алгоритмів машинного навчання.

Висновок до розділу 2

В результаті розглянутої в другому розділі інформації, були сформовані такі висновки:

- Було формально визначено задачу виявлення DDOS атак, зокрема формати вхідних та вихідних даних.
- Розглянуто структуру та вміст набору даних, що використовується для тренування моделі.
- Обрано інструменти для очищення, нормалізації та підготовки даних до навчання.
- Оглянуто алгоритм логістичної лінійної регресії для вирішення поставленої задачі, зокрема необхідні формули для обчислення оптимальних вагів моделі.
- Визначено інструменти для оцінки якості роботи моделі машинного навчання, зокрема показники таблиці мпряженості та побудову ROC-кривої.
- Оглянуто засоби та технології для розробки програмного забезпечення, зокрема обрано мову програмування, бібліотеки для роботи з даними та середовище розробки.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНИХ КОМПОНЕНТІВ

Базуючись на досліджених матеріалах й обраних засобах реалізації можна перейти до етапу розробки програмних компонентів. В даному розділі розглядається реалізація схеми, зображеної на рисунку 3.1.

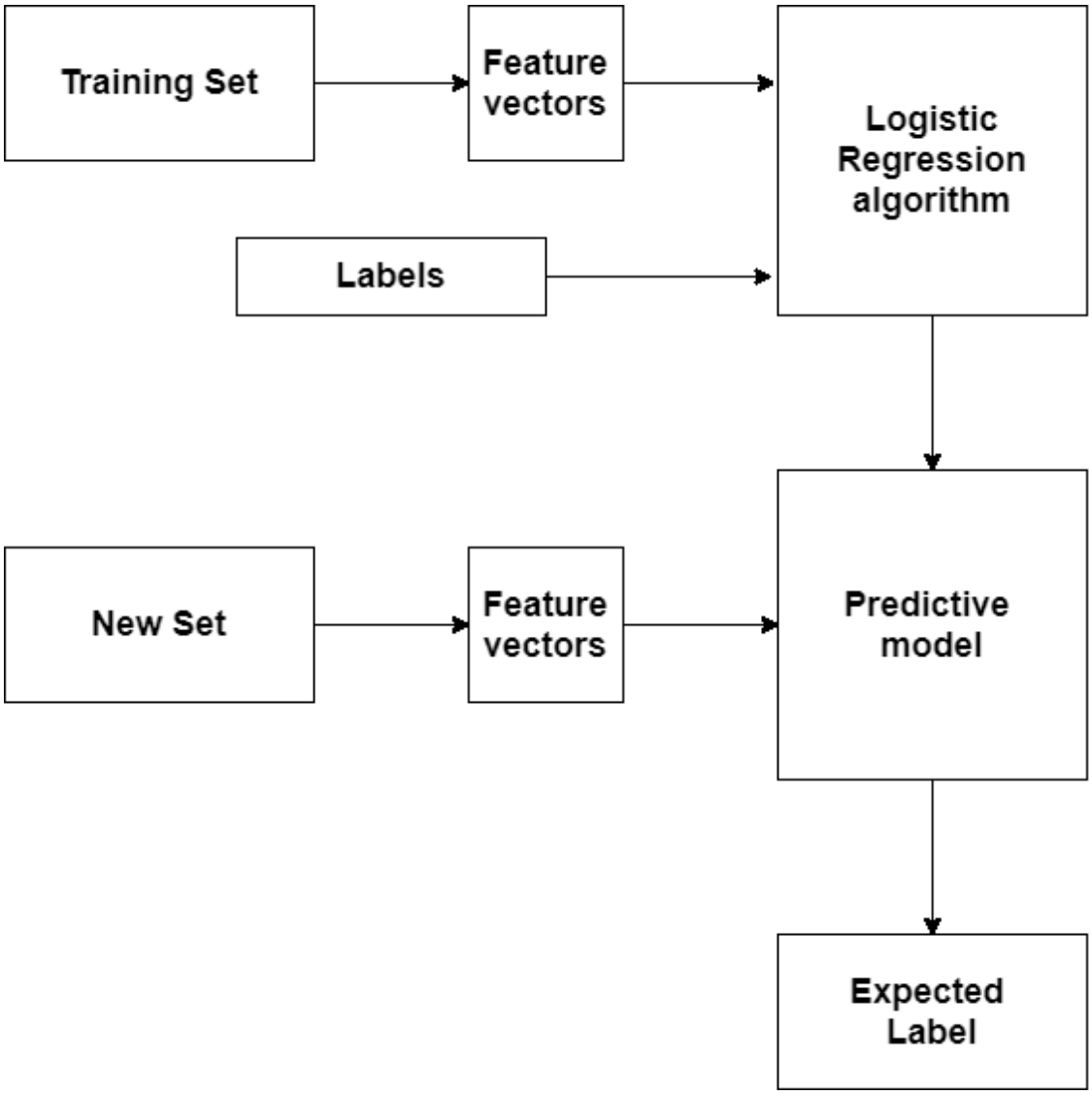


Рис. 3.1. Схема роботи алгоритму логістичної регресії

Для цього описується структура розробленої програми, її компонентів та їх взаємодія. Програмний продукт розділений на модулі відповідно до виконуваних функції, зокрема, модуль підготовки даних для навчання, навчання моделі, тестування отриманих результатів та його візуалізація, інтерфейс користувача.

3.1. Клас Model

Клас Model це модель машинного навчання, що може приймати дані, тренуватися, та прогнозувати результати отримуючи тестові дані. Цей клас має три атрибути: weights, bias та costs. На рисунку 3.2 можна побачити структуру класу.

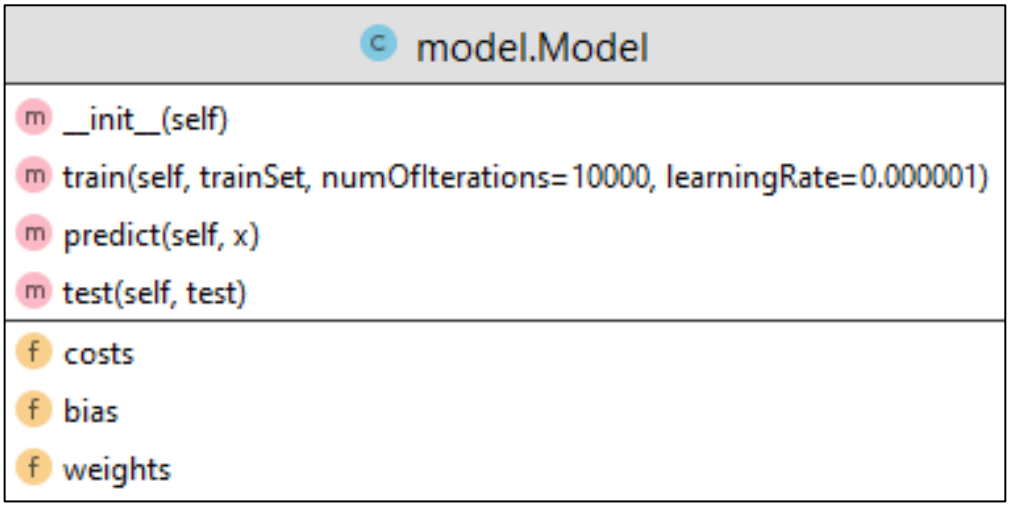


Рис. 3.2. UML діаграма класу Model

weights – масив типу np.array, що містить числові значення типу float64. Масив містить значення вагів моделі логістичної регресії. Довжина масиву дорівнює кількості параметрів.

bias – числове значення типу float64. Значення похибки моделі логістичної регресії.

cost - np.array, масив чисел float64. Зберігає дані про зміну функції витрат на кожній ітерації тренування моделі. Використовується для оцінки якості тренування.

Конструктор класу ініціалізує дані атрибути як пусті масиви (*weights, cost*) та 0 (*bias*).

Основні функції класу реалізовані в наступних методах(з використанням допоміжних функцій, що знаходяться з межами класу):

1. *train(self, trainSet, numOfIterations=10000, learningRate=0.000001)* – основний метод класу, що відповідає за тренування моделі.

Він приймає наступні атрибути:

- *trainSet* – np.array, тестовий набір даних;
- *numOfIterations* – float64, кількість ітерацій навчання моделі(за замовчуванням визначена як 10000);
- *learningRate* – float64 швидкість начання, тобто коефіцієнт α з формули (2.4) (за замовчуванням визначено як 0,000001).

На кожній операції викликається допоміжна функція *propagate(weights, bias, x, y)*. Ця функція приймає наступні параметри:

- *weights* – np.array, ваги;
- *bias* – float64, похибка;
- *x* – np.array, тренувальні приклади;
- *y* – np.array, значеня класу атаки для тренувальних прикладів.

В цій функції за формулою (2.5) визначаються параметри *dw*, *db*, що позначають зміну ваг моделі, а також *cost* – функцію витрат за формулою (2.4). Для обчислення цих формул використовується допоміжна функція *sigmoid(z),* яка приймає масив чисел float64 та обчислює для них функцію з формули (2.2) і повертає масив відповідних значень.

Функція *propagate* повертає значення *dw*, *db* та *cost*, які далі фикористовуються в методі *train*.

Метод *train* після ітерацій змін вагів зберігає їх в поля даного об’єкту класу, що робить можливим їх використання в інших методах без повторного тренування.

2. *predict(self, x)* – метод, що відповідає за передбачення результатів. Приймає *x* – np.array, приклади даних без атрибуту атаки. Обраховує для кожного прикладу значення функції з формули (2.2) та повертає вектор результатів np.array.

3. *test(self, test)* – метод тестування моделі. Приймає *test* – np.array, приклади даних з атрибутами атаки. Дані розділяються на характеристики та результати, відповідно x та y.

Для кожного прикладу викликається метод predict та результати записуються в вектор. Далі результати порівнюються з дійсними та обраховуються основні показники оцінки моделі машинного навчання:

- truePositive – істинно позитивні;
- trueNegative – істинно негативні;
- falsePositive – хибно позитивні;
- falseNegative – хибно негативні.

Ці показники записуються в np.array масив *spec.* Метод повертає даний масив, який потім використовується для аналізу якості роботи моделі.

3.2. Модуль підготовки даних

Модуль prepData містить основні функції для отримання, підготовки та нормалізації даних(рис. 3.3). Модуль імпортує бібліотеки pandas та numpy, які містять інструменти для обробки даних.

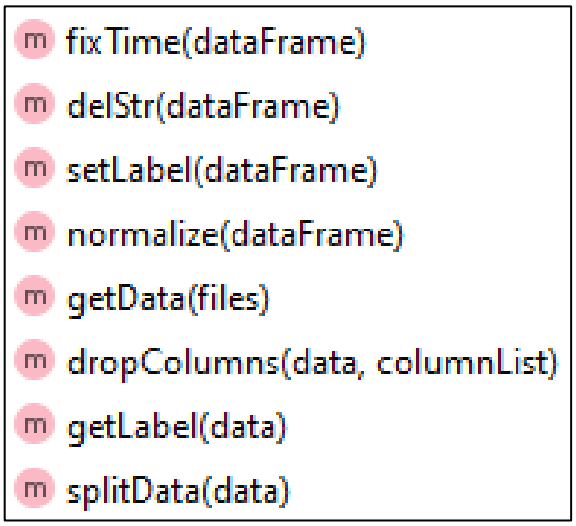


Рис. 3.3. Діаграма модулю prepData

1. *getData(files)* – основна функція, яка викликає інші функції модуля для підготовки даних. Приймає на вхід *files* - масив строк, які містять шляхи до файлів з даними.

Для отримання даних з файлів використовується бібліотека pandas, зокрема функція *pd.read_csv(file, dtype='str', nrows=1000).* Дані по черзі отримуються з кожного файлу та записуються в набір даних pd.DataFrame.

Цей формат є зручним для роботи з наборами даних. Він містить стовпці та рядки даних,при чому стовпці мають назви, отримані з першого рядку файлу даних.

За допомогою функції *DataFrame.drop([], axis=1, inplace=True),* яка приймає масив назв стовпців, видаляються зайві стовпці даних, які не містять важливої інформації.

Дані об'єднуються в один загальний набір pd.DataFrame за допомогою функції *pd.concat([dataFrame, newData], axis=0),* яка приймає основний та новий набір даних. Потім дані обробляються за допомогою функцій *fixTime, dataFrame.replace, delStr, setLabel, normalize, dataFrame.sample,* які будуть розглянуті далі.

2. *delStr(dataFrame)*(рис 3.4) – функція для очистки даних від строкових значень. Приймає pd.DataFrame, з якого видаляються дані. Останній стовпчик даних містить назву атаки, тож він складається з строк і їх не потрібно видаляти. Далі використовується метод *dataFrame.drop.apply(pd.to_numeric, errors='coerce')*, який перетворює всі значення на числові, і у випадку, якщо це не можливо, записує значення nan.

```
def delStr(dataFrame):
    label = dataFrame[" Label"]
    dataFrame = dataFrame.drop(" Label", axis=1).apply(pd.to_numeric, errors='coerce')
    dataFrame = pd.concat([dataFrame, label], axis=1)
    dataFrame.dropna(inplace=True)
    dataFrame.reset_index(drop=True, inplace=True)
    return dataFrame
```

Рис. 3.4. Функція *delStr*

Рядки з цими значеннями видаляються за допомогою функції *dataFrame.dropna(inplace=True).* Вона виділяє не тільки значення, які раніше були строковими, але й будь-які інші дані, які не можливо було б використати для тренування моделі. Після цього необхідно оновити індекси(номери рядків) за допомогою *dataFrame.reset_index(drop=True, inplace=True),* адже деякі з них були видалені.

3. *fixTime(dataFrame)*(рис. 3.5) – функція, що замінює дані часу запиту. Початковий набір містить дані щодо дати та часу запиту, але їх досить важко перетворити в числовий формат, до того ж дати та час майже однакові. Ці дані не мають впливу на наявність атаки, адже дані були згенеровані вченими, а не взяті з реальних процесів. Але важливою може виявитися інформація щодо часу між сусідніми запитами.

```
def fixTime(dataFrame):
    prev = float(dataFrame[" Timestamp"][0].split(":")[-1])
    for index, row in dataFrame.iterrows():
        curr = float(row[" Timestamp"].split(":")[-1])
        dataFrame.loc[index, " Timestamp"] = curr - prev
        prev = curr
```

Рис. 3.5. Функція *fixTime*

Дана функція розділяє дату на складові й визначає різницю між сусідніми значеннями. Ці значення записуються назад у стовпчик " Timestamp". Повертає pd.DataFrame з виправленими значеннями.

4. *setLabel(dataFrame)* – функція заміни значення класу атаки. Оскільки логістична регресія працює з числовими даними та з бінарним значенням результату, необхідно замінити значення класу атаки на 0(відсутність атаки) або 1(наявність атаки). Приймає pd.DataFrame та повертає його копію з заміненими даними.

5. *normalize(dataFrame)*(рис. 3.6) – функція нормалізації даних.

```
def normalize(dataFrame):
    for column in dataFrame.columns:
        dataFrame[column] = (dataFrame[column] - dataFrame[column].min()) / \
                            (dataFrame[column].max() - dataFrame[column].min())
    return dataFrame
```

Рис. 3.6. Функція normalize

Приймає pd.DataFrame та повертає нормалізовані дані в тому ж форматі. Нормалізація даних необхідна для коректної роботи алгоритму та відбувається за формулою (2.9).

Оскільки нормалізувати дані необхідно у ствопчиках, то це зручно зробити з методом pd.DataFrame[column], який повертає окремо стовпці набору. Для цих векторів значень обраховується мінімальне та максимальне значення за допомогою dataFrame[column].max() та dataFrame[column].min() та підставляється у формулу нормалізації.

6. *dropColumns(data, columnList)* – функція видалення ствопців. Оскільки алгоритм логістичної регресії працює з масивами типу np.array, потрібно реалізувати таку функцію. Приймає два масиви np.array: data – набір даних та columnList – список стовпців для видалення. Повертає масив np.array.

7. *getLabel(data)* - функція отримання атрибуту класу. Для тренування моделі необхідно мати список атрибутів класу окремо від прикладів. Функція приймає масив np.array data – набір даних та повертає список *labelList*.

8. *splitData(data)*(рис. 3.7) – функція розділяє набір даних на тренувальний(80%) та тестувальний(20%) набори. Функція приймає масив np.array data – набір даних та повертає два масиви np.array: *train* та *test.*

```
def splitData(data):
    train = dropColumns(data[:int(0.8 * len(data))], [data.shape[1] - 1])
    test = dropColumns(data[int(0.8 * len(data)):int(len(data))], [data.shape[1] - 1])

    return train, test
```

Рис. 3.6. Функція *splitData*

3.3. Модуль вибору параметрів

Даний модуль відповідає за вибір оптимальних параметрів для тренування моделі. Оскільки він використовує модель для тренування та тестування, він є окремим від модулю підготовки даних. Хоча результати роботи цього модуля потім використовуються при підготовці даних. Модуль містить основну функцію *chooseFeatures(data)*. Вона приймає параметр *data* – np.array, дані для тренування та тестування типу. Повертає список атрибутів, при використанні яких було досягнуто найкращі показники ефективності роботи моделі.

Дана функція перебирає різні кількості та комбінації вхідних параметрів, розділяє на тренувальний та тестувальний набори. Потім тренує на них модель та тестує якість передбачень.

```
def chooseFeatures(data):
    mainFeatures = []
    bestF1 = 2

    for setLen in range(5, len(data[0])):
        featuresSet = generateSet(setLen)

        for col in range(len(data[0])):
            if col not in featuresSet:
                np.delete(data, col, axis=1)

        trainSet, testSet = prepData.splitData(data)
        myModel = model.Model()
        myModel.train(trainSet)

        spec = myModel.test(testSet)
        precision = round(spec[0] / (spec[0] + spec[2]), 3)
        recall = round(spec[0] / (spec[0] + spec[2]), 3)
        f1 = (2 * precision * recall) / (precision + recall)
        if f1 > bestF1:
            bestF1 = f1
            mainFeatures = set

    return mainFeatures
```

Рис. 3.6. Функція *chooseFeatures*

Отримавши значення TP, FP, FN, TN, обчислюються такі наступні показники оцінки якості передбачення:

Точність(precision):

$$\boldsymbol{precision} = \frac{\mathbf{TP}}{\mathbf{TP+FP}}$$

Повнота(recall):

$$\boldsymbol{recall} = \frac{\mathbf{TP}}{\mathbf{TP+FN}}$$

Основним показником оцінки є F-міра, тобто середнє гармонічне з двох попередніх показників:

$$\boldsymbol{F1} = \frac{\mathbf{2*precision*recall}}{\mathbf{precision+recall}}$$

Далі обирається набір параметрів з найвищими показниками та найменшою кількість параметрів.

3.4. Модуль аналізу результатів

Цей модуль призначений для аналізу результатів тестування алгоритму. Він містить основну функцію *analyze(model, testData)*(рис. 3.9), яка відповідає за аналіз результатів роботи алгоритму.

```
def analyze(model, testData):
    tpr = []
    fpr = []

    for i in range(100):
        cutOff = i / 100
        spec = model.test(testData, cutOff)

        tpr.append(calculate(spec)[0])
        fpr.append(calculate(spec)[1])

    printROC(fpr, tpr)
```

Рис. 3.9 – Функція *analyze*

Вона приймає два параметри: *model* – об’єкт класу *Model* та *testData* – дані для тестування. Модель має бути вже натренованою на навчальних даних, а тестові дані отримані не з навчальних. Функція для кожного значення порога відсікання, яке змінюється від 0 до 1 з кроком 0,01 розраховує значення FPR та TPR за допомогою функції *calculate(spec)*. Ця функція приймає масив з чотирьох значень: TP, FP, FN, TN та повертає FPR і TPR. У випадку, якщо сума TN + FP чи TP + FN дорівнюють 0, відповідні показники повертаються рівними 0.

З них формується список значень, який передається в функцію *printROC(fpr, tpr),* яка відповідає за побудову кривої. Вона приймає два масиви з значеннями параметрів, які будуть точками на графіку. По осі X відкладаються значення FPR, по осі Y – TPR. Графік будується за допомогою набору функцій pyplot з бібліотеки matplotlib та виводиться у вікно з бібліотеки PySimpleGUI.

Висновок до розділу 3

В цьому розділі було розглянуто компоненти програмного продукту. В результаті роботи сформовано такі висновки:

- Розроблено програмні компоненти відповідно до поставленої задачі. Вони виконують основні функції, необхідні для тренування та тестування моделі машинного навчання.
- Було програмно реалізовано алгоритм логістичної регресії з можливістю зміни основних параметрів навчання.
- Реалізовано можливість збереження тренованої моделі для її подальшого використання.
- Програма має модульну структуру, тож було описано основні модулі програми, їх структуру та взаємодію.
- Програма використовує такі бібліотеки, як Pandas, NumPy, Matplotlib, PySimpleGUI, що значно спростило написання програми та зменшило кількість коду.
- Обрані інструменти розробки виявилися зручними для роботи над програмними компонентами, пришвидшили та полегшили процес написання коду.

РОЗДІЛ 4. ТЕСТУВАННЯ ТА АНАЛІЗ РОЗРОБЛЕНОЇ ПРОГРАМИ

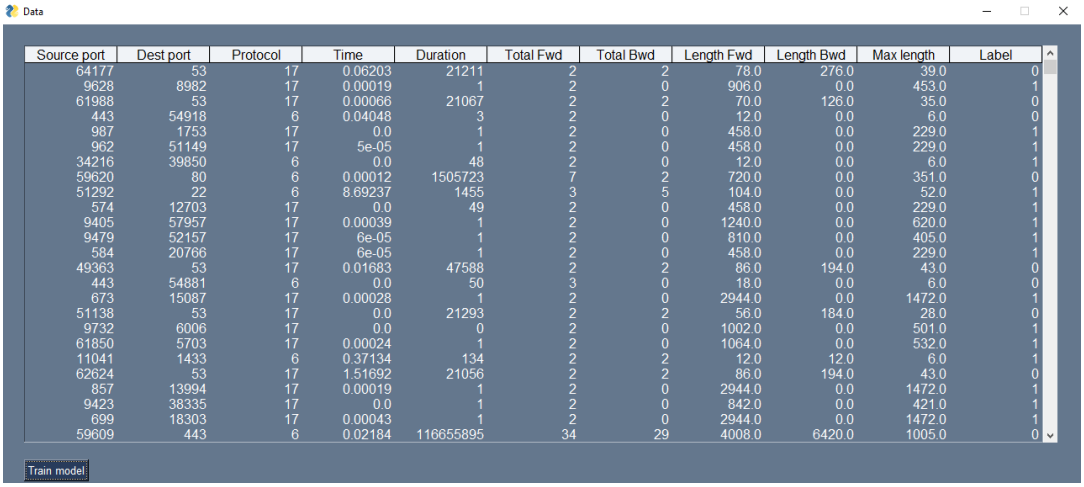
В цьому розділі описуються результати розробки програмного продукту. Проводиться тестування та порівняльна характеристика розробленого алгоритму.

4.1. Інтерфейс користувача

Стартовий інтерфейс користувача(рис. 4.1) дозволяє завантажувати дані для тренування або використати треновану модель машинного навчання. Є можливість вписати шлях до файлу з даними або обрати з файлової системи.

Рис. 4.1 – Стартовий інтерфейс користувача

Після цього дані завантажуються та виводяться в таблиці(рис. 4.2), де можна переглянути наявні характеристики, тобто назви стовпців.



Source port	Dest port	Protocol	Time	Duration	Total Fwd	Total Bwd	Length Fwd	Length Bwd	Max length	Label
64177	53	17	0.06203	21211	2	2	78.0	276.0	39.0	0
9628	8982	17	0.00019	1	2	0	906.0	0.0	453.0	1
61988	53	17	0.00066	21067	2	2	70.0	126.0	35.0	0
443	54918	6	0.04048	3	2	0	12.0	0.0	6.0	0
987	1753	17	0.0	1	2	0	458.0	0.0	229.0	1
962	51149	17	5e-05	1	2	0	458.0	0.0	229.0	1
34216	39850	6	0.0	48	2	0	12.0	0.0	6.0	1
59620	80	6	0.00012	1505723	7	2	720.0	0.0	351.0	0
51292	22	6	8.69237	1455	3	5	104.0	0.0	52.0	1
574	12703	17	0.0	49	2	0	458.0	0.0	229.0	1
9405	57957	17	0.00039	1	2	0	1240.0	0.0	620.0	1
9479	52157	17	6e-05	1	2	0	810.0	0.0	405.0	1
584	20766	17	6e-05	1	2	0	458.0	0.0	229.0	1
49363	53	17	0.01683	47588	2	2	86.0	194.0	43.0	0
443	54881	6	0.0	50	3	0	18.0	0.0	6.0	0
673	15087	17	0.00028	1	2	0	2944.0	0.0	1472.0	1
51138	53	17	0.0	21293	2	2	56.0	184.0	28.0	0
9732	6006	17	0.0	0	2	0	1002.0	0.0	501.0	1
61850	5703	17	0.00024	1	2	0	1064.0	0.0	532.0	1
11041	1433	6	0.37134	134	2	2	12.0	12.0	6.0	1
62624	53	17	1.51692	21056	2	2	86.0	194.0	43.0	0
857	13994	17	0.00019	1	2	0	2944.0	0.0	1472.0	1
9423	38335	17	0.0	1	2	0	842.0	0.0	421.0	1
699	18303	17	0.00043	1	2	0	2944.0	0.0	1472.0	1
59609	443	6	0.02184	116655895	34	29	4008.0	6420.0	1005.0	0

Рис. 4.2 – Вивід даних

З цього вікна за допомогою кнопки “Train model” можна перейти до тренування моделі.

Вікно тренування моделі(рис. 4.3) містить вибір різних характеристик для тренування моделі. Можна вписати атрибути для навчання, або скористатися алгоритмом отримання оптимального набору атрибутів. Є можливість задати кількість ітерації, розмір набору навчальних даних у відсотках від завантаженого набору, опція нормалізації даних.

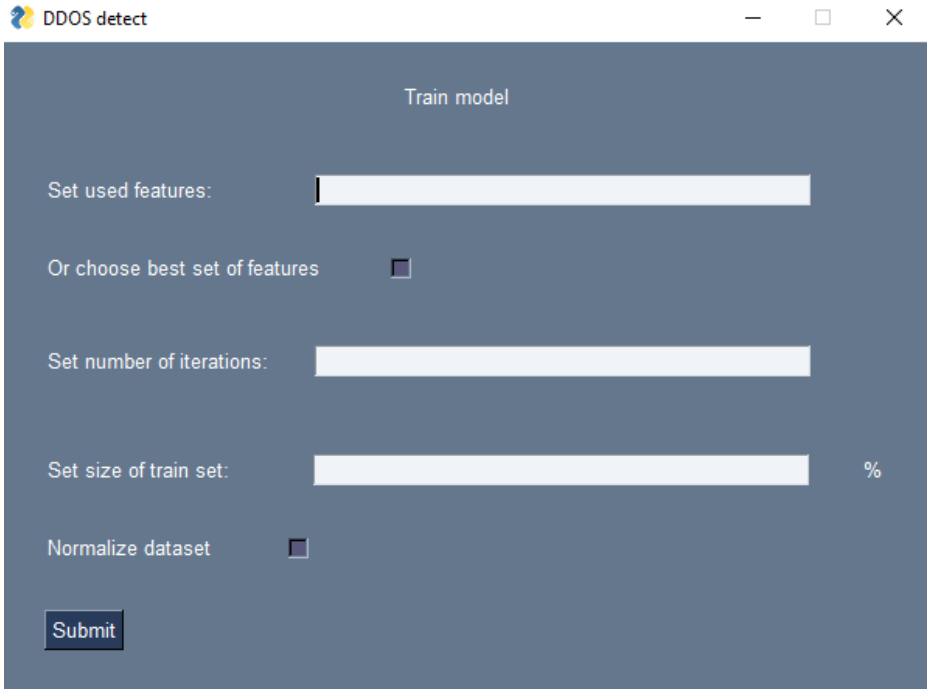


Рис. 4.3 – Вікно тренування моделі

Після визначення параметрів переходимо у вікно тестування моделі(рис. 4.4).

Рис. 4.4 – Вікно завантаження тестових даних

Можна завантажити окремі дані для тестування або використати дані, які не були використані для навчання на попередньому етапі. Це можливо якщо було обрано менше 100% даних для тренування.

4.2. Тестування алгоритму

Після завантаження тестових даних відбувається тестування моделі та визначаються основні показники ефективності. Результати відображаються у вигляді ROC-кривої(рис. 4.5) та основних показників ефективності(рис. 4.6).

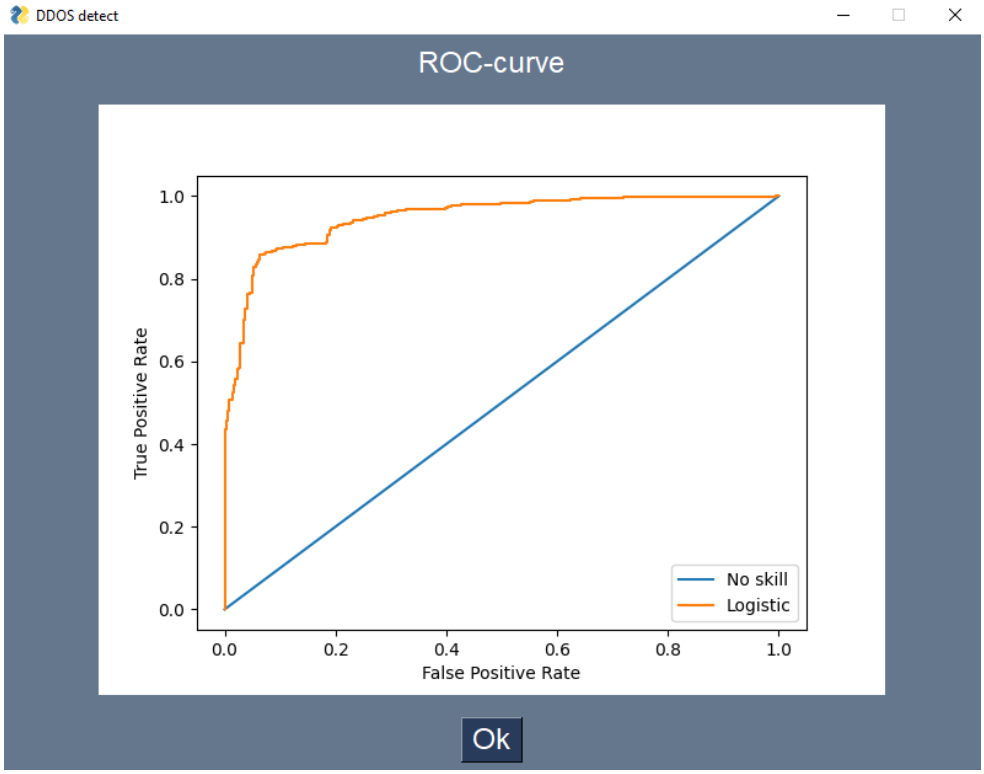


Рис. 4.5 – Графік отриманої ROC-кривої

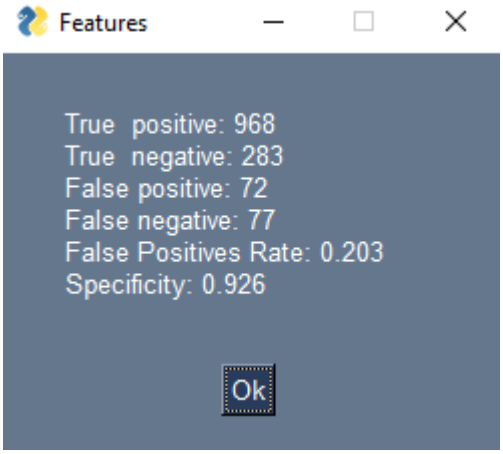


Рис. 4.6 – Вікно відображення показників моделі

Використовуючи дані результати можна порівнювати якість різних моделей, або однієї моделі при різних параметрах. Це дає можливість обрати оптимальні умови для навчання.

4.3. Порівняння з існуючими системами

Розроблена система є альтернативою для розглянутих в першому розділі систем виявлення атак, що базуються на методах машинного навчання. Обрана модель відрізняється простотою та відносною швидкістю навчання від запропонованих моделей, адже логістична регресія є одним за найпростіших алгоритмів машинного навчання.

Модель в середньому показала точність 92,6%, що є високим показником відносно розглянутих систем. Додаткові покращення алгоритму, такі як відбір навчальних даних того типу атаки, який є більш можливим у випадку застосування можуть підвищити даний показник до рівня найкращих альтернатив. Також можна вдосконалити модель, реалізувавши модель множинної логістичної регресії, що дасть можливість не тільки виявляти атаки, але й визначати їх тип.

Важливою відмінністю розробленої моделі є те, що більшість розглянутих систем базуються на одному наборі даних, який містить невелику кількість типів атак. В даній роботі було використано великий набір з великою кількістю атак, що робить його пристосованим до більшої кількості умов.

Висновок до розділу 4

В цьому розділі було проаналізовано результати розробки. В результаті роботи сформовано такі висновки:

- Розроблено зручний інтерфейс користувача з підтримкою основних функцій, що необхідні для тренування та тестування моделі логістичної регресії.
- Тестування алгоритму свідчить про високу точність моделі, що дає можливість використовувати її в реальних застосунках для виявлення та блокування DDOS атак.
- Основною перевагою розробленого алгоритму є його простота та універсальність, адже модель можу бути натренованою на будь-яких даних й використовуватися для різних типів атак.
- Недоліком програми є тривалість навчання моделі та велика кількість необхідних даних. Також використання алгоритму в реальних застосунках поки що може не бути можливим через вірогідність похибок та не пристосованість до певних умов.
- При порівнянні з існуючими системами було виявлено, що модель має досить високу точність порівняно з іншими системами й може складати конкуренцію розглянутим розробкам.

ВИСНОВОК

В ході даної роботи було розроблено алгоритм виявлення DDOS атак та реалізовано модель алгоритму. Алгоритм базується на методах машинного навчання, що дозволяє уникнути явного програмування класифікації запитів.

В першому розділі були визначені теоретичні аспекти даної роботи. Розглянуто основні типи DDOS атак, їх структуру та характеристики. Було визначено основні методи виявлення та захисту від атак, оцінено їх ефективність. Було проведено огляд й порівняльну характеристику існуючих систем.

В другому розділі було сформовано математичну задача й шляхи її вирішення. Визначено та описано оптимальний алгоритм машинного навчання для поставленої задачі. Розглянуто обрані інструменти для реалізації алгоритму.

В третьому розділі міститься опис розробленої програми, її структура та схема роботи. Окремо розглянуті основні модулі програми та їх взаємодія.

В четвертому розділі продемонстровано результати розробки, зокрема інтерфейс користувача. Було проведено тестування розробленого алгоритму та порівняно з існуючими аналогами.

Отже, розробка виявилася успішною й поставлена задача була виконана. Розроблений додаток має простий інтерфейс з різними інструментами роботи з даними. Алгоритм показав високу якість передбачення, що робить можливим його використання в реальних застосунках.

ЛІТЕРАТУРА

1. Hansman, S. & Hunt, R. (2005). *A taxonomy of network and computer attacks.* Computers & Security 24 (pp. 31-43).
2. Lau, F., Rubin, S.H., Smith, M.H., Trajkovi´c, L. (2000). *Distributed denial of service attacks.* Proceedings of IEEE International Conference on Systems, Man and Cybernetics vol. 3 (pp. 2275–2280).
3. DDOS in the Past Decade [Електронний ресурс] – Режим доступу: https://nsfocusglobal.com/ddos-in-the-past-decade/
4. Sharafaldin, I., Lashkari, A. H., Hakak, S. & Ghorbani, A. A. (2019). *Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy.* In Proceedings of the 2019 International Carnahan Conference on Security Technology (ICCST) (pp. 1-8).
5. What is a DDoS Attack? [Електронний ресурс] – Режим доступу: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/
6. Nisha, H. Bhandari. (2013) *Survey on DDoS Attacks and its Detection & Defence Approaches.* International Journal of Science and Modern Engineering (IJISME), ISSN: 2319-6386, Volume-1, Issue-3 (pp. 67-71).
7. Niharika Sharma, Amit Mahajan & Vibhakar Mansotra. (2016). *Machine Learning Techniques Used in Detection of DOS Attacks: A Literature Review*. International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 6.
8. Dewan Md. Farid, Nouria Harbi, Emna Bahri, Mohammad Zahidur Rahman & Chowdhury Mofizur Rahman. (2010). *Attacks Classification in Adaptive Intrusion Detection using Decision Tree.* In International Conference on Computer Science (ICCS 2010) (pp. 29-31).
9. Vipin Das, Vijaya Pathak, Sattvik Sharma, Sreevathsan, MVVNS.Srikanth & Gireesh Kumar T. (2010). *Network intrusion detection system based on machine learning algorithms.* In International Journal of Computer Science & Information Technology (IJCSIT), Vol 2, No 6, (pp. 138-150).

10. Swathi Sambangi & Lakshmeeswari Gondi. (2020). *A Machine Learning Approach for DDoS (Distributed Denial of Service) Attack Detection Using Multiple Linear Regression.* In Proceedings of the 14th International Conference INTER-ENG 2020 Interdisciplinarity in Engineering, 63(1).
11. Raghavan, S.V., Vijayasarathy, R., & Ravindran, B. (2011). *A system approach to network modeling for DDoS detection using a Naive Bayesian classifier.* In Communication Systems and Networks (COMSNETS) (pp. 1-10).
12. Mr. Anurag Andhare & Prof. Arvind Bhagat Patil. (2012). *Denial-of-Service Attack Detection Using Genetic-Based Algorithm.* In International Journal of Engineering Research and Applications (IJERA), Vol. 2, Issue 2, (pp. 094-098).
13. Donghwoon Kwon, Hyunjoo Kim, Jinoh Kim, Sang C. Suh, Ikkyun Kim & Kuinam J. Kim. (2019). *A survey of deep learning-based network anomaly detection.* Cluster Comput 22, (pp. 949-961).
14. Anna Drewek-Ossowicka, Mariusz Pietrołaj & Jacek Rumiński. (2021). *A survey of neural networks usage for intrusion detection systems*. Journal of Ambient Intelligence and Humanized Computing 12 (pp. 497-514).
15. Mehryar Mohri, Afshin Rostamizadeh, & Ameet Talwalkar. (2018). *Foundations of Machine Learning (2nd. ed.).* The MIT Press.
16. Jafar Alzubi, Anand Nayyar, Akshi Kumar. (2018). *Machine Learning from Theory to Algorithms: An Overview*. Second National Conference on Computational Intelligence (NCCI 2018).
17. Dayton, C.M. (1992). *Logistic Regression Analysis.* ResearchGate Publications. (pp. 1- 9).
18. Logistic Regression — Detailed Overview [Електронний ресурс] – Режим доступу: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc
19. Introduction to Logistic Regression [Електронний ресурс] – Режим доступу: https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148
20. Andrew Ng. *Stanford CS229 Lecture Notes*.

21. Hoo ZH, Candlish J, Teare D. (2017). *What is an ROC curve?* Emergency Medicine Journal. (pp.357-359).
22. How to Use ROC Curves and Precision-Recall Curves for Classification in Python [Електронний ресурс] – Режим доступу: https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/
23. Why Python for Machine Learning? [Електронний ресурс] – Режим доступу: https://pythonbasics.org/why-python-for-machine-learning/
24. Intro to pandas [Електронний ресурс] – Режим доступу: https://pandas.pydata.org/docs/getting_started/index.html
25. Numpy tutorials [Електронний ресурс] – Режим доступу: https://numpy.org/learn/
26. Python GUIs for Humans [Електронний ресурс] – Режим доступу: https://pysimplegui.readthedocs.io/en/latest/
27. PyCharm Features [Електронний ресурс] – Режим доступу: https://www.jetbrains.com/pycharm/features/

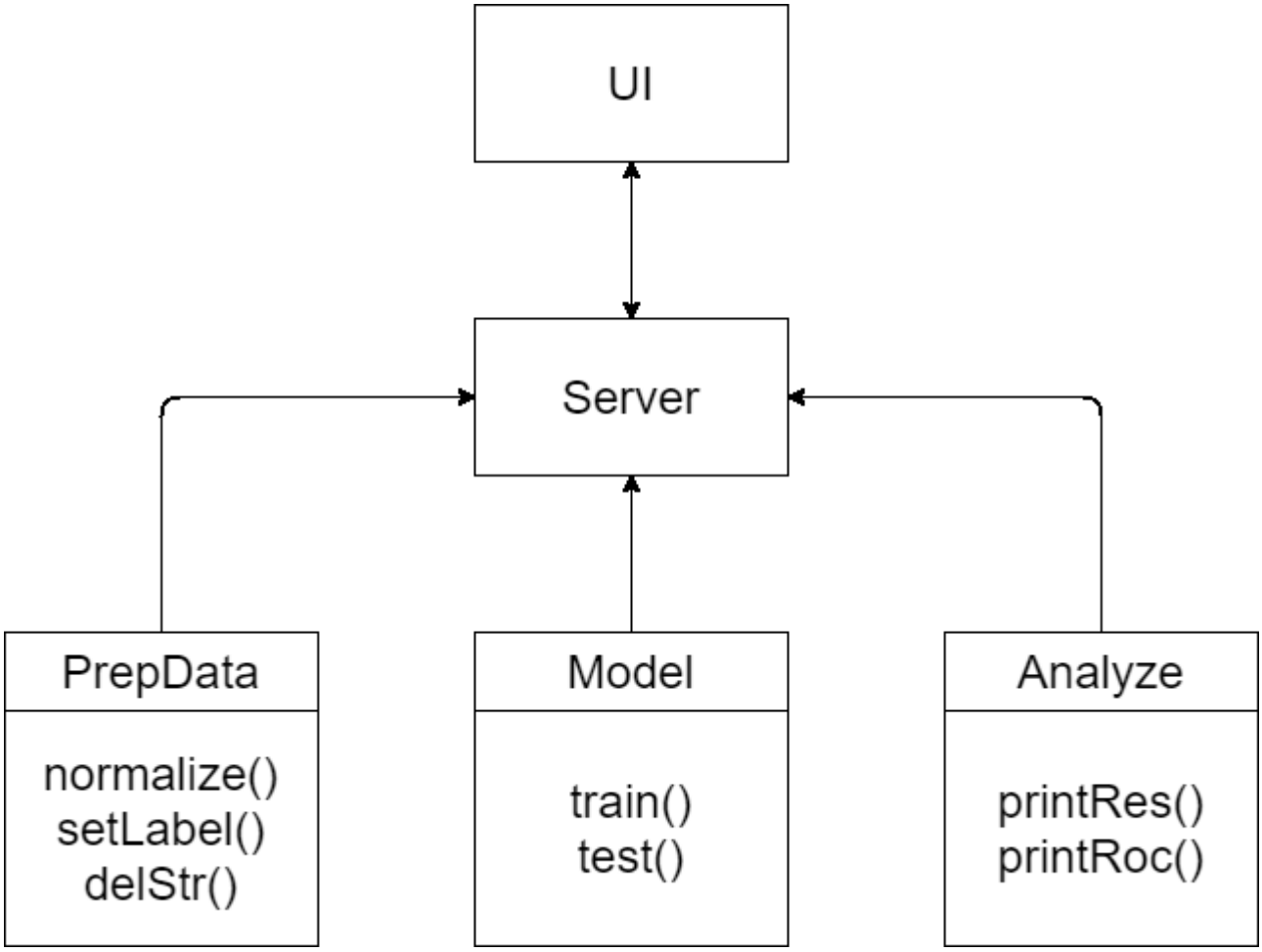
Додаток А

СТРУКТУРНА СХЕМА ПРОГРАМИ

до дипломного проекту

на тему: «Виявлення DDOS атак за
допомогою машинного навчання»

Київ – 2021



					ІАЛЦ.467100.001 ОА		
					Виявлення DDOS атак за допомогою машинного навчання	***Літ.***	***Маса*** / ***Масштаб***
Розроб.	***Михайліченко І.А.***						
Перев.	***Стіренко С.Г.***						
Реценз.					*Структурна схема*	***Аркуш 1***	***Аркушів 1***
Н. контр.	***Сімоненко В.П.***				*Дипломний проєкт*	***НТУУ "КПІ" ФІОТ ІП-73***	
Затв.	***Стіренко С.Г.***						

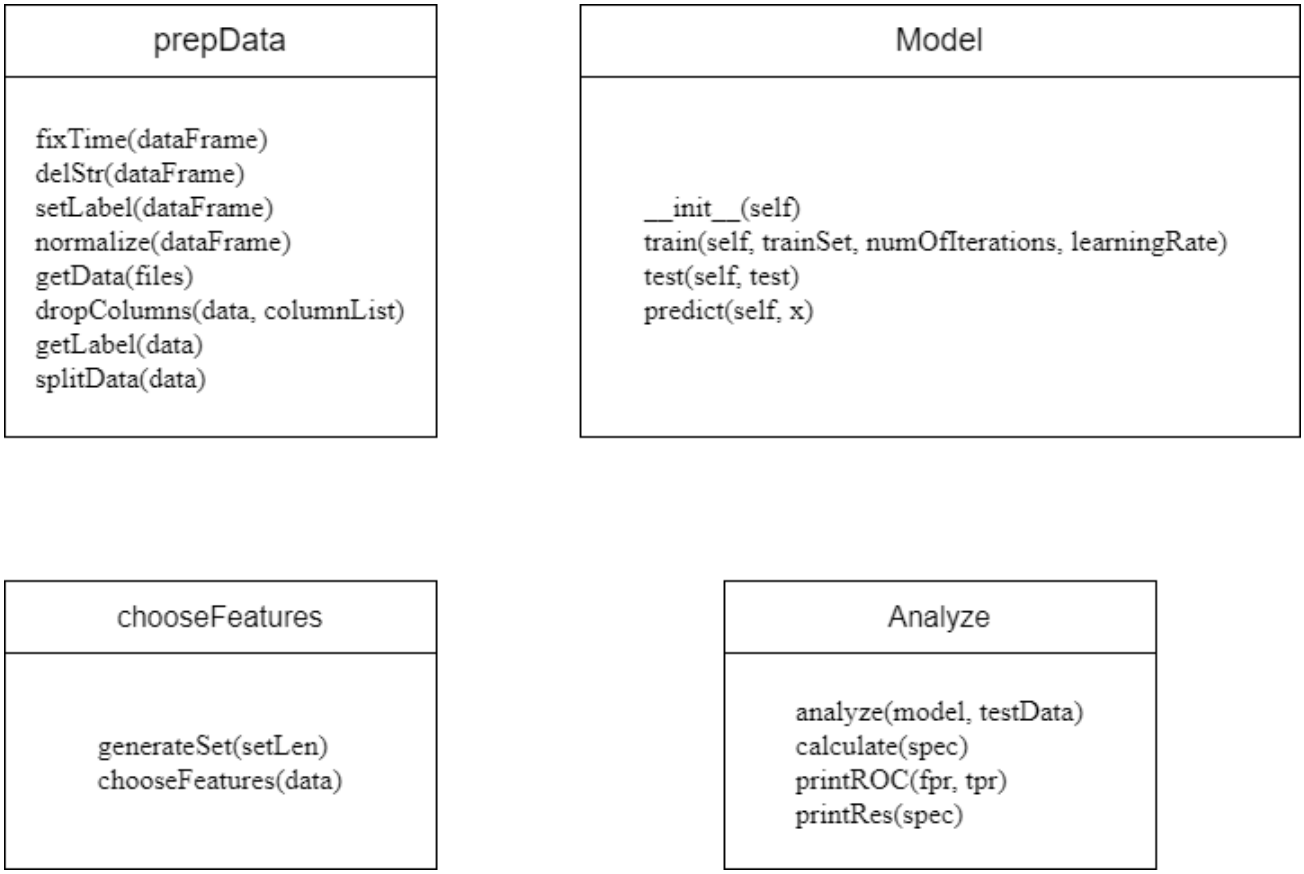
Додаток Б

ФУНКЦІОНАЛЬНА СХЕМА ПРОГРАМИ

до дипломного проекту

на тему: «Виявлення DDOS атак за
допомогою машинного навчання»

Київ – 2021



					ІАЛЦ.467100.001 ОА		
					Виявлення DDOS атак за допомогою машинного навчання Функціональна схема	***Літ.***	***Маса*** / ***Масштаб***
Розроб.	***Михайліченко І.А.***						
Перев.	***Стіренко С.Г.***					***Аркуш 1***	***Аркушів 1***
Реценз.							
Н. контр.	***Сімоненко В.П.***				*Дипломний проект*	***НТУУ "КПІ" ФІОТ ІП-73***	
Затв.	***Стіренко С.Г.***						

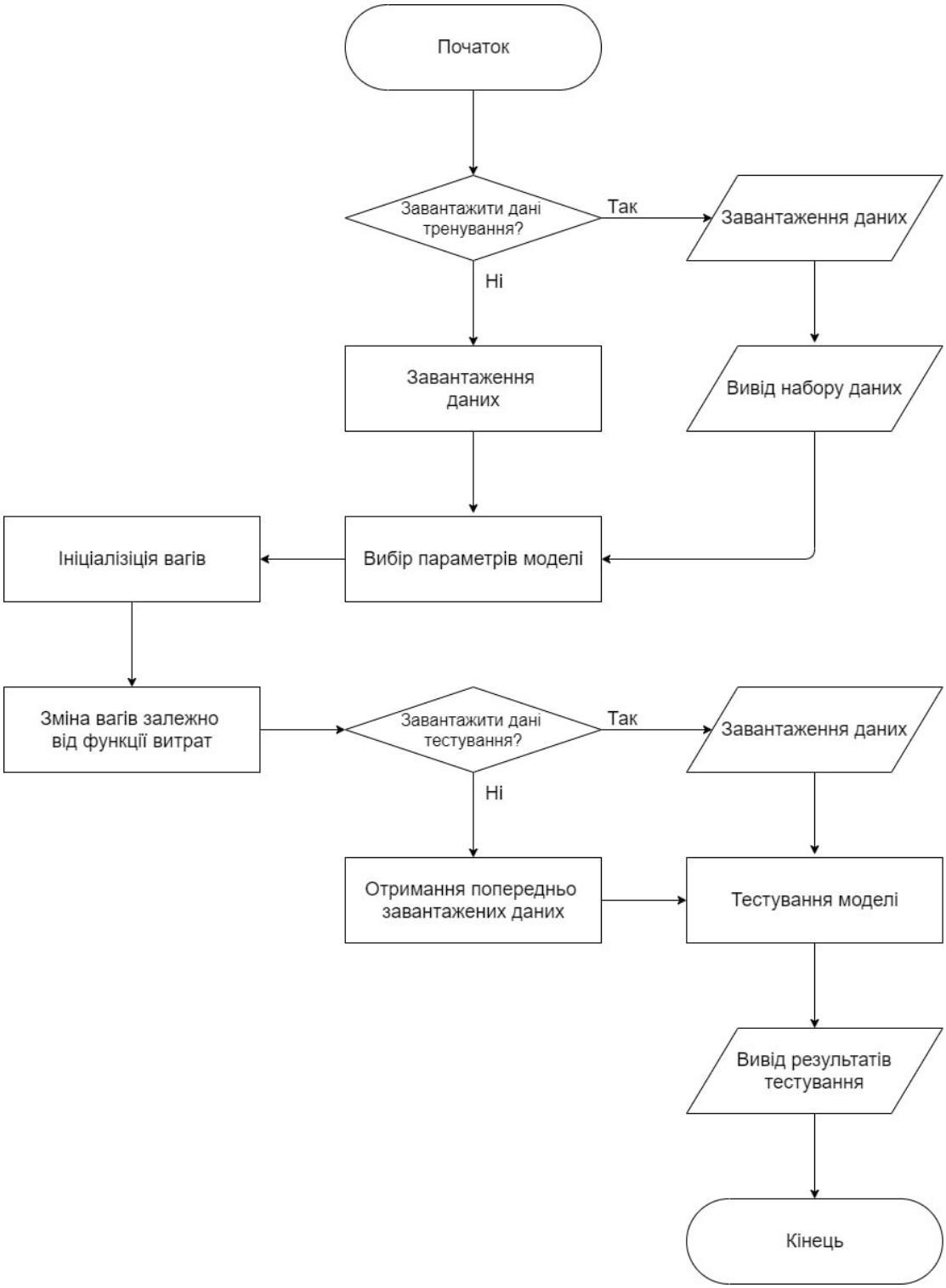
Додаток В

ПРИНЦИПОВА СХЕМА ПРОГРАМИ

до дипломного проекту

на тему: «Виявлення DDOS атак за
допомогою машинного навчання»

Київ – 2021



					ІАЛЦ.467100.001 ОА		
					Виявлення DDOS атак за допомогою машинного навчання Принципова схема	***Літ.***	***Маса*** / ***Масштаб***
Розроб.	***Михайліченко І.А.***						
Перев.	***Стіренко С.Г.***					***Аркуш 1***	***Аркушів 1***
Реценз.							
Н. контр.	***Сімоненко В.П.***				*Дипломний проект*	***НТУУ "КПІ" ФІОТ ІП-73***	
Затв.	***Стіренко С.Г.***						

Додаток В

ЛІСТИНГ КОДУ ПРОГРАМИ

до дипломного проекту

на тему: «Виявлення DDOS атак за
допомогою машинного навчання»

Київ – 2021

ЛІСТИНГ ПРОГРАМИ

```
###############
# Interface
###############

import PySimpleGUI as sg
import analyze
import model
import prepData


def popup(title, message):
    layout = [[sg.Text(message, pad=(20, 20))],
              [sg.Button("Ok", pad=(75, 10))]]

    window = sg.Window(title, layout, size=(200, 130))

    while True:
        event, values = window.read()
        if event == sg.WIN_CLOSED or event == "Exit" or event == "Ok":
            break

    window.close()


def result(message):
    layout = [[sg.Text(message, pad=(20, 20))],
              [sg.Button("Ok", pad=(100, 10))]]

    window = sg.Window("Features", layout, size=(250, 200))

    while True:
        event, values = window.read()
        if event == sg.WIN_CLOSED or event == "Exit" or event == "Ok":
            break

    window.close()


def test(testModel, data):
    layout = [[sg.Text("Choose test data file: ", pad=(20, 20)), sg.Input(key="-IN-",
change_submits=True),
               sg.FileBrowse(key="-IN-")],
```

```
            [sg.Button("Submit", pad=(20, 0))],
            [sg.Text("Or ", pad=(20, 30)), sg.Button("Use uploaded data")]]

    window = sg.Window('DDOS detect', layout, size=(600, 200))

    while True:
        event, values = window.read()
        if event == sg.WIN_CLOSED or event == "Exit":
            break
        elif event == "Submit":
            filePath = values["-IN-"]
            if filePath == '':
                popup('Error', 'Enter pass')
            else:
                data = prepData.getData([filePath])
                break
        elif event == "Use uploaded data":
            break

    analyze.analyze(testModel, data)
    result(analyze.printRes(testModel.test(data)))

    window.close()


def train(data):
    layout = [[sg.Text("Train model", pad=(250, 20))],
              [sg.Text("Set used features: ", pad=(20, 20)), sg.Input(key="-IN1-", pad=(40, 0))],
        [sg.Text("Or choose best set of features", pad=(20, 10)), sg.Checkbox('', size=(10, 1),
pad=(20, 0))],
        [sg.Text("Set number of iterations: ", pad=(20, 30)), sg.Input(key="-IN2-", pad=(4, 0))],
              [sg.Text("Set size of train set: ", pad=(20, 20)),
               sg.Input(key="-IN3-", pad=(28, 0)), sg.Text("%")],
              [sg.Text("Normalize dataset ", pad=(20, 10)), sg.Checkbox('', size=(10, 1), pad=(20,
0))],
              [sg.Button("Submit", pad=(20, 20))]]

    window = sg.Window('DDOS detect', layout, size=(600, 420))

    while True:
        event, values = window.read()
        if event == sg.WIN_CLOSED or event == "Exit":
            break
        elif event == "Submit":
            trainSet, testSet = prepData.splitData(data.to_numpy())

            testModel = model.Model()
            testModel.train(trainSet)

            test(testModel, testSet)
            break

    window.close()
```

```
def show_table(data):
    table = data[1:].values.tolist()
    table = [[round(i, 5) for i in num] for num in table]

    layout = [
        [sg.Table(values=table,
                  headings=data.columns,
                  font='Helvetica',
                  pad=(20, 20),
                  display_row_numbers=False,
                  auto_size_columns=False,
                  max_col_width=1,
                  num_rows=min(25, len(data)))],
        [sg.Button("Train model", pad=(20, 0))],
    ]

    window = sg.Window("Data", layout, grab_anywhere=False)
    event, values = window.read()
    while True:
        if event == sg.WIN_CLOSED or event == "Exit":
            break
        elif event == "Train model":
            train(data)
            break

    window.close()

def main():
    layout = [[sg.Text("Choose train data file: ", pad=(20, 20)), sg.Input(key="-IN-",
change_submits=True),
               sg.FileBrowse(key="-IN-")],
              [sg.Button("Submit", pad=(20, 0))],
              [sg.Text("Or ", pad=(20, 30)), sg.Button("Use trained model")]]

    window = sg.Window('DDOS detect', layout, size=(600, 200))

    while True:
        event, values = window.read()
        if event == sg.WIN_CLOSED or event == "Exit":
            break
        elif event == "Submit":
            filePath = values["-IN-"]
            if filePath == '':
                popup('Error', 'Enter pass')
            else:
                data = prepData.getData([filePath])
                show_table(data)
                break

    window.close()

main()
```

```
###############
# prepData
###############

import pandas as pd
import numpy as np


# Change absolute time to relative
def fixTime(dataFrame):
    prev = float(dataFrame[" Timestamp"][0].split(":")[-1])
    for index, row in dataFrame.iterrows():
        curr = float(row[" Timestamp"].split(":")[-1])
        dataFrame.loc[index, " Timestamp"] = curr - prev
        prev = curr


# Clear data from strings and nan values
def delStr(dataFrame):
    label = dataFrame[" Label"]
    dataFrame = dataFrame.drop(" Label", axis=1).apply(pd.to_numeric, errors='coerce')
    dataFrame = pd.concat([dataFrame, label], axis=1)
    dataFrame.dropna(inplace=True)
    dataFrame.reset_index(drop=True, inplace=True)
    return dataFrame


# Change string label to binary
def setLabel(dataFrame):
    dataFrame.loc[dataFrame[" Label"] != "BENIGN", " Label"] = 1
    dataFrame.loc[dataFrame[" Label"] == "BENIGN", " Label"] = 0
    return dataFrame


# Data normalization
def normalize(dataFrame):
    for column in dataFrame.columns:
        if column != " Label":
            if dataFrame[column].max() - dataFrame[column].min() == 0:
                dataFrame.drop(column, axis=1, inplace=True)
            else:
                dataFrame[column] = (dataFrame[column] - dataFrame[column].min()) / \
                                    (dataFrame[column].max() - dataFrame[column].min())
    return dataFrame


# Data preparing for training
def getData(files):
    dataFrame = pd.DataFrame()

    for file in files:
        newData = pd.read_csv(file, dtype='str', nrows=1000)
```

```
newData.drop(["Flow ID", " Source IP", " Destination IP", " Fwd Packet Length Min",
            " Fwd Packet Length Mean", " Fwd Packet Length Std", "Bwd Packet Length Max",
            " Bwd Packet Length Min", " Bwd Packet Length Mean", " Bwd Packet Length Std",
            "Flow Bytes/s", " Flow Packets/s", " Flow IAT Mean", " Flow IAT Std",
            " Flow IAT Max", " Flow IAT Min", "Fwd IAT Total", " Fwd IAT Mean",
            " Fwd IAT Std", " Fwd IAT Max", " Fwd IAT Min", "Bwd IAT Total",
            " Bwd IAT Mean", " Bwd IAT Std", " Bwd IAT Max", " Bwd IAT Min",
            "Fwd PSH Flags", " Bwd PSH Flags", " Fwd URG Flags", " Bwd URG Flags",
            " Fwd Header Length", " Bwd Header Length", "Fwd Packets/s"," Bwd Packets/s",
            " Min Packet Length", " Max Packet Length", " Packet Length Mean",
            " Packet Length Std", " Packet Length Variance", "FIN Flag Count",
            " SYN Flag Count", " RST Flag Count", " PSH Flag Count", " ACK Flag Count",
            " URG Flag Count", " CWE Flag Count", " ECE Flag Count", " Down/Up Ratio",
            " Average Packet Size", " Avg Fwd Segment Size", " Avg Bwd Segment Size",
            " Fwd Header Length.1", "Fwd Avg Bytes/Bulk", " Fwd Avg Packets/Bulk",
            " Fwd Avg Bulk Rate", " Bwd Avg Bytes/Bulk", " Bwd Avg Packets/Bulk",
            "Bwd Avg Bulk Rate", "Subflow Fwd Packets", " Subflow Fwd Bytes",
            " Subflow Bwd Packets", " Subflow Bwd Bytes", "Init_Win_bytes_forward",
            " Init_Win_bytes_backward", " act_data_pkt_fwd", " min_seg_size_forward",
            "Active Mean", " Active Std", " Active Max", " Active Min", "Idle Mean",
            " Idle Std", " Idle Max", " Idle Min", "SimillarHTTP", " Inbound"],
            axis=1, inplace=True)

        fixTime(newData)

        dataFrame = pd.concat([dataFrame, newData], axis=0)
    dataFrame.replace(np.nan, '', inplace=True)
    dataFrame = delStr(dataFrame)
    dataFrame = setLabel(dataFrame)
    dataFrame = normalize(dataFrame)
    dataFrame = dataFrame.sample(frac=1)
    return dataFrame


# Drop some columns from np.array
def dropColumns(data, columnList):
    return np.delete(data, columnList, axis=1)

# Get list of labels from dataset
def getLabel(data):
    labelList = []
    for row in data:
        labelList.append(row[-1])

    return labelList

# Split data on training and test sets
def splitData(data):
    train = dropColumns(data[:int(0.8 * len(data))], [data.shape[1] - 1])
    test = dropColumns(data[int(0.8 * len(data)):int(len(data))], [data.shape[1] - 1])

    return train, test
```

```
###############
# Model
###############

import numpy as np
import prepData


# Sigmoid function of logistic regression
def sigmoid(z):

    z = 1.0 / (1.0 + np.exp(-z))

    return z


# Propagation function calculates grades dw and db
def propagate(weights, bias, x, y):
    m = x.shape[0]

    Z = np.array(np.dot(x, weights) + bias, dtype=np.float64)

    A = sigmoid(Z)

    cost = -1.0 / m
    # * np.sum(y * np.log(A) + (1.0 - y) * np.log(1.0 - A))
    dw = 1.0 / m * np.dot(A - y, x)
    db = 1.0 / m * np.sum(A - y)

    return dw, db, cost


# Model class with hold weights and bias for training
class Model:
    def __init__(self):
        self.weights = np.array([], dtype=np.float64)
        self.bias = 0
        self.costs = []

    # Method of training of the model
    def train(self, trainSet, numOfIterations=10000, learningRate=0.000001):
        x = prepData.dropColumns(trainSet, [len(trainSet[0]) - 1])
        y = prepData.getLabel(trainSet)

        self.weights = np.zeros(x.shape[1])

        for i in range(numOfIterations):
            dw, db, cost = propagate(self.weights, self.bias, x, y)

            # self.costs.append(cost)
            self.weights = self.weights - learningRate * dw
            self.bias = self.bias - learningRate * db
```

```
    # Method of predicting of result
    def predict(self, x):
        return sigmoid((np.dot(x, self.weights.T) + self.bias) - 1) < 0.5

    # Method of testing model
    def test(self, test):
        x = prepData.dropColumns(test, [len(test[0])-1])
        y = prepData.getLabel(test)

        prediction = self.predict(x)

        # truePositive, trueNegative, falsePositive, falseNegative
        spec = [0, 0, 0, 0]

        for index in range(len(test)):
            if y[index] == prediction[index]:
                if y[index] == 1:
                    spec[0] += 1
                else:
                    spec[1] += 1
            else:
                if y[index] == 1:
                    spec[2] += 1
                else:
                    spec[3] += 1

        return spec


################
# Choose features
################


import numpy as np
import prepData
import model


# Generate set of used features
def generateSet(setLen):
    featuresSet = []
    for num in range(setLen):
        featuresSet.append(num)
    return featuresSet


# Choose best set of features for detecting
def chooseFeatures(data):
    mainFeatures = []
    bestF1 = 2
```

```
    for setLen in range(5, len(data[0])):
        featuresSet = generateSet(setLen)

        for col in range(len(data[0])):
            if col not in featuresSet:
                np.delete(data, col, axis=1)

        trainSet, testSet = prepData.splitData(data)
        myModel = model.Model()
        myModel.train(trainSet)

        spec = myModel.test(testSet)
        precision = round(spec[0] / (spec[0] + spec[2]), 3)
        recall = round(spec[0] / (spec[0] + spec[2]), 3)
        f1 = (2 * precision * recall) / (precision + recall)
        if f1 > bestF1:
            bestF1 = f1
            mainFeatures = set

    return mainFeatures

################
# Analyze
################

import PySimpleGUI as sg
from matplotlib import pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg


# Calculate main features of model analyze
# spec = truePositive, trueNegative, falsePositive, falseNegative
def calculate(spec):
    fpr = round(spec[2] / (spec[1] + spec[2]), 3) if (spec[1] + spec[2]) else 0
    tpr = round(spec[0] / (spec[0] + spec[3]), 3) if (spec[0] + spec[3]) else 0

    return fpr, tpr


# Print ROC curve
def printROC(fpr, tpr):

    fig = plt.figure(dpi=100)
    plt.xlabel(fpr)
    plt.ylabel(tpr)
    fig.add_subplot(111).scatter(fpr, tpr, color='blue', edgecolor='k')

    layout = [[sg.Text('Plot')],
              [sg.Canvas(key='-CANVAS-',
                         size=(700, 500),
                         pad=(15, 15))],
              [sg.Button('Ok')]]
```

```
    window = sg.Window('Plot',
                       layout,
                       size=(800, 600),
                       finalize=True,
                       element_justification='center',
                       font='Helvetica 18')

    def draw_figure(canvas, figure):
        figure_canvas_agg = FigureCanvasTkAgg(figure, canvas)
        figure_canvas_agg.draw()
        figure_canvas_agg.get_tk_widget().pack(side='top', fill='both', expand=1)
        return figure_canvas_agg

    fig_canvas_agg = draw_figure(window['-CANVAS-'].TKCanvas, fig)

    event, values = window.read()

    window.close()


# Get features for ROC analyze
def analyze(model, testData):
    tpr = []
    fpr = []

    for i in range(100):
        spec = model.test(testData)

        tpr.append(calculate(spec)[0])
        fpr.append(calculate(spec)[1])

    printROC(fpr, tpr)


# Print main features of model testing
def printRes(spec):
    return "True  positive: " + str(spec[0]) + "\nTrue  negative: " + str(spec[1]) + \
           "\nFalse positive: " + str(spec[2]) + "\nFalse negative: " + str(spec[3])
```