

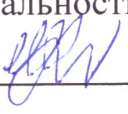
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА КІБЕРБЕЗПЕКИ

ПОЯСНЮВАЛЬНА ЗАПИСКА

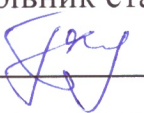
до кваліфікаційної роботи освітнього ступеня
«магістр» за спеціальністю 125 «Кібербезпека»
на тему:

«Методи та інструменти аналізу статичного мережевого
трафіку на основі штучного інтелекту»

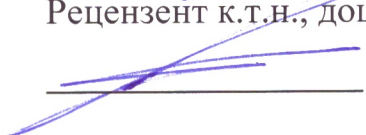
Виконав студент 2-го курсу, групи КБм-22-1
спеціальності 125 «Кібербезпека»

 П.К. Погребніченко

Керівник старший викладач кафедри КІ та КБ

 О.А. Покотило

Рецензент к.т.н., доцент, доцент кафедри КІ та КБ,

 Ю.Б. Бродський

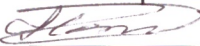
Житомир – 2023

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної
інженерії та кібербезпеки

 Андрій Єфіменко
13 жовтня 2023 р.

ЗАВДАННЯ

на випускні кваліфікаційну роботу

Студент: Погребніченко Павло Костянтинович

Тема роботи: Методи та інструменти аналізу статичного мережевого трафіку на основі штучного інтелекту

Затверджена Наказом університету від «13» жовтня 2023 р. № 572/с

Термін здачі студентом закінченої роботи 11 грудня 2023р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження): Об'єктом дослідження є методи та інструменти аналізу статичного мережевого трафіку на основі штучного інтелекту, дослідження їх можливостей, та реалізація веб-додатку на основі обраного методу, з використанням Python, KDD Dataset, та Flask. Предметом дослідження є технологія рішення задач класифікації для аналізу даних на основі набору даних з використанням Python, KDD Dataset, та Flask;

Обладнання та програмне забезпечення:

1. Мова програмування Python;
2. Набір мов програмування для розгортання веб-додатку;
3. KDD Dataset;
4. Середовище розробки Visual Studio Code.

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

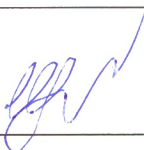
Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Пулеко І.В.	13.10.2023	13.10.2023
2	Пулеко І.В.	13.10.2023	13.10.2023
3	Пулеко І.В.	13.10.2023	13.10.2023

Студент  П.К. Погребніченко

Керівник  О.А. Покотило

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Видача завдання	13.10.2023	виконано
2	Розробка плану роботи	15.10.2023	виконано
3	Розробка розгорнутого плану	20.10.2023	виконано
4	Подача чернетки керівнику для перевірки	25.10.2023	виконано
5	Доопрацювання роботи та перевірка керівником	31.10.2023	виконано
6	Затвердження графічних матеріалів у керівника	07.11.2023	виконано
7	Проведення попереднього захисту	10.11.2023	виконано
8	Отримання відгуку керівника	17.11.2023	виконано
9	Допуск завідувача кафедри	01.12.2023	виконано
10	Подання роботи на рецензування	10.12.2023	виконано

Студент  П.К. Погребніченко

Керівник  О.А. Покотило

РЕФЕРАТ

Завдання кваліфікаційної роботи магістра полягає у вивченні та обґрунтуванні вибору моделі, мов програмування, логічних рішень та методів дослідження статичного мережевого трафіку, а також використання отриманих знань для розробки веб-додатку з імплементацією технологій машинного навчання, а саме підходу класифікації даних та прогнозування аналітичних даних та збірок на основі результатів аналізу статичного інтернет трафіку.

Кваліфікаційна робота магістра на тему «Методи та інструменти аналізу статичного мережевого трафіку на основі штучного інтелекту» складається з програмного, логічного, аналітичного, та теоретичного комплексів та пояснювальної записки.

Пояснювальна записка до випускної кваліфікаційної роботи містить в собі 84 сторінки, та 65 рисунків. Список використаних джерел містить 28 найменувань і займає 3 сторінки.

КЛЮЧОВІ СЛОВА: ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, FLASK, PYTHON, KDD DATASET, ВЕБ-ДОДАТОК, АНАЛІЗ ДАНИХ, ШТУЧНИЙ ІНТЕЛЕКТ.

ABSTRACT

The task of the master's qualification work consists in studying and justifying the choice of a model, programming languages, logical solutions and methods of static network traffic research, as well as using the acquired knowledge to develop a web application with the implementation of machine learning technologies, namely the approach of data classification and forecasting of analytical data and collections based on the results of analysis of static Internet traffic.

The qualification course on the topic "Methods and tools for static network traffic analysis based on artificial intelligence" consists of software, logical, analytical, theoretical complexes and an explanatory note.

The explanatory note to the graduation thesis contains 84 pages, and 65 figures. The list of used sources contains 28 names and occupies 3 pages.

KEYWORDS: SOFTWARE, FLASK, PYTHON, KDD DATASET, WEB APPLICATION, DATA ANALYSIS, ARTIFICIAL INTELLIGENCE.

ЗМІСТ

РЕФЕРАТ	4
ABSTRACT	5
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	7
ВСТУП.....	8
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ АКТУАЛЬНОСТІ ТЕМИ РОБОТИ.....	10
1.1 Обґрунтування актуальності теми роботи	10
1.2 Огляд існуючих методів аналізу даних, що використовують машинне навчання.....	13
1.3 Аналіз базових технологій веб-розробки	23
1.4 Аналіз мов програмування для створення штучного інтелекту	31
1.5 Постановка завдання на розробку.....	37
РОЗДІЛ 2 РОЗРОБКА ВЕБ-ДОДАТКУ АНАЛІЗУ СТАТИЧНОГО МЕРЕЖЕВОГО ТРАФІКУ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ	40
2.1 Аналіз вхідних даних.....	40
2.2 Розробка та навчання моделі штучного інтелекту	44
2.3 Проектування та розробка веб-додатку.....	46
РОЗДІЛ 3 ПЕРЕВІРКА ПРАЦЕЗДАТНОСТІ ТА ЕФЕКТИВНОСТІ ДЕТЕКТОРА АТАК НА ОСНОВІ АЛГОРИТМІВ ДЕРЕВ РІШЕНЬ	61
3.1 Тестування та оцінка якості моделі машинного навчання	61
3.2 Тестування веб-додатку	66
ВИСНОВКИ.....	71
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	73
ДОДАТКИ.....	76

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

XML – Розширювана мова розмітки (Extensible Markup Language)

HTML – Гіпертекстова мова розмітки (HyperText Markup Language)

KDD Set – Набір даних (Knowledge Discovery & Data Mining Set)

ML – Машинне навчання (Machine Learning);

JSON – Формат файлу (JavaScript Object Notation);

CSV – Числовий формат файлу з роздільником (comma-separated values);

API – Програмний інтерфейс для доадтку (application programming interface);

SQL – Мова структурованих запитів (Structured Query Language);

ROC – Крива операційної характеристики відповідей (анг. Receiver Operating Characteristics);

SL – Неглибоке Навчання (анг. Shallow Learning);

ВСТУП

Актуальність теми. Сучасне інтернет середовище на постійній основі має взаємодію з нашими даними, відкритими та персональними. Будь який сучасний сайт або портал запитує нашого дозволу на отримання «кукі» файлів, відстеження нашої локації, та безліч інших варіацій та інтеракцій з нашими даними. В останній час особливої уваги надають «аугментованим» сервісам, котрі мають за собою штучний інтелект, або навіть базуються на ньому всебічно. Ця тенденція завдає більшого ризику для будь-якого роду даних користувача, оскільки з часом буде все важче оцінити рівень дій машини, котра може отримувати чуттєву інформацію навіть з контекстних непрямих дій.

Актуальність теми машинного навчання у безпеці та аналізі даних є чималою, оскільки велика кількість комерційних та користувацьких сервісів мають за собою ту чи іншу форму штучного інтелекту, котрі набрали широкої популярності у нинішній час. Разом з тим, ще більша ділянка інтернет простору набула підвищеного рівня вразливості до маніпуляцій, зламу та підміни даних, оскільки злочинці також почали використовувати штучний інтелект для своїх інтересів. Таким чином можна сказати, що **тема роботи** направлена на розробку програмного веб додатку аналізу даних на базі Python є **актуальною**.

Мета і завдання роботи. Метою роботи є Розробка та реалізація інструменту аналізу статичного мережевого трафіку на основі штучного інтелекту.

Досягнення поставленої мети передбачає розв'язання наступних завдань:

- Проведення аналізу існуючих технологій машинного навчання у сфері аналізу даних;
- Аналізу алгоритмів та методів машинного навчання;
- Аналізу базових технологій веб-розробки;
- Аналізу вхідного набору даних та його обробки;
- Розробки та реалізації веб-додатку аналізу статичного мережевого трафіку на основі штучного інтелекту;
- Проведення підсумкових контрольних вимірювань точності та якості результатів моделі та їх відображення у веб-додатку;

Об'єктом дослідження є побудова додатку аналізу статичного мережевого трафіку на основі штучного інтелекту.

Предметом дослідження є технологія рішення задач класифікації для аналізу даних на основі набору даних з використанням Python та KDD Dataset.

Методи дослідження. В кваліфікаційній роботі було використано наступні методи: об'єктно орієнтованого програмування, двокласової класифікації, математичного моделювання та алгоритмізації коду.

Практичне значення роботи полягає у наступному:

- розроблений програмний продукт дозволяє якісно оцінювати пакети даних, аналізувати їх та прогнозувати шкідливість за допомогою мови програмування Python та KDD Dataset, що допомагає в роботі фахівцям з безпеки, дата аналітикам, та системним адміністраторам;
- розроблені пропозиції використання програмного продукту у сфері кібербезпеки, мережевого та системного адміністрування, дата аналізу.

Апробація результатів:

1. Пулеко І.В., Погребніченко П.К. Оцінка якості виявлення кібератак розробленим програмним додатком детектору атак на основі Python та KDD dataset // Тези доповідей Тези V Всеукраїнської науково-технічної конференції «Комп'ютерні технології: інновації, проблеми, рішення», м. Житомир, 01-02 грудня 2022р. — Житомир: Житомирська політехніка, 2022.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ АКТУАЛЬНОСТІ ТЕМИ РОБОТИ

1.1 Обґрунтування актуальності теми роботи

Аналітика даних має вирішальне значення в будь-якій сфері, щоб точно зрозуміти, що відбувається. Це вирівнює ігрове поле та надає цінний контекст і інформацію для прийняття рішень, орієнтованих на майбутнє.

Кібербезпека нічим не відрізняється.

Аналітика кібербезпеки побудована на основі ретельного збору даних. Необроблені дані про кібербезпеку збираються, порівнюються та аналізуються, а потім перетворюються на рекомендації, дії та звіти. Аналіз надає дані, які надходять у програмне забезпечення кібербезпеки на основі ШІ.

Ці звіти та платформи кібербезпеки потім надають поради та дії, які користувач повинен виконати далі.

Аналітика кібербезпеки — це будівельні блоки, які забезпечують проактивний підхід до заходів безпеки для вашої мережевої інфраструктури, а не просто реактивні підходи[1].

І хоча ви не можете передбачити майбутнє, якщо ви розумієте навколишнє середовище та історичні атаки на кібербезпеку, аналізуєте, як вони відбувалися, хто їх здійснював, їхні цілі та засоби, а також як саме компанія стала вразливою, тоді ви можете виправити ці прогалини у своїй системі.

Аналізуючи ці дані та розуміючи, що стало можливим для кожного порушення даних, ваша система зможе виявити потенційні вразливості у власних процесах і краще виявляти потенційні атаки, перш ніж вони матимуть шанс завдати шкоди.

Технології штучного інтелекту та машинного навчання швидко розвиваються, і разом із цим ми проактивно використовуємо ці досягнення та застосовуємо їх для покращення інформаційної безпеки. Практики кібербезпеки швидко перетворюють те, що раніше було науковою фантастикою, на науковий факт.

Аналітичні платформи кібербезпеки використовують потужні алгоритми для збору та аналізу даних із ряду систем кібербезпеки. Вони збирають і вивчають історичні дані про загрози кібербезпеці та атаки, і після того, як вони зібрані, відсортовані та проаналізовані, вони перетворюються на прогнозування моделей кіберзагроз[2].

І чим більше атак, тим більше релевантних даних збирається, тим обґрунтованішою стає аналітика — тим точніше вона може прогнозувати майбутні атаки до того, як вони відбудуться.

Це також працює в режимі реального часу. Ці керовані даними моделі використовуються для збору та аналізу інформації та аналізу шаблонів, кидаючи історичний огляд на дані, щоб визначити, де поточні шаблони відповідають історичним шаблонам загроз.

Це покращує те, як ваші команди безпеки виконують виявлення аномалій у вашому мережевому трафіку, виявляють випадки зловмисного програмного забезпечення та краще визначають будь-які потенційні атаки та керування подіями безпеки. Це дає змогу повністю скоротити час відповіді та зупинити цілеспрямовані атаки ще до того, як вони відбудуться, лише на основі низки показників. Разом з цим штучний інтелект значно стійкіший до монотонної праці, що повторюється на регулярній основі, наприклад складання шаблонів чи перевірка та ідентифікація даних. Машина може значно швидше провести усю монотонну працю та передати людині готовий результат, що суттєво підвищить якість роботи та знизить ризики.

Машинне навчання широко використовується у багатьох сферах, і, в особливості, у сфері безпеки та аналізу даних. На прикладі декількох компаній ми розглянемо успішне використання засобів машинного навчання для надання послуг з кібербезпеки та аналізу даних.

1. Cybereason. Cybereason — це платформа аналітики кібербезпеки, яка забезпечує моніторинг, пошук і аналіз загроз. Це дає компаніям і організаціям кращу видимість у своєму середовищі безпеки для випередження потенційних загроз. Технологія пошуку на базі штучного інтелекту Cybereason, за допомогою автоматизації, визначає, чи зазнає атака організація, та може принести користь командам із

безпеки будь-якого розміру та рівня кваліфікації для запобігання та аналізу безпечного простору.

2. Cisco. Cisco використовує штучний інтелект для забезпечення своєї платформи кібербезпеки, яка, за їх словами, розроблена, щоб дати клієнтам змогу бачити «безпеку всього вашого бізнесу» з однієї інформаційної панелі під назвою Cisco Actionboard. Вона має 14 модулів безпеки, які можна увімкнути або вимкнути за потреби, охоплюючи все: від хмарної безпеки та безпеки мережі до керування даними кінцевої точки та безпеки електронної пошти.

3. Palo Alto Networks. Що стосується постачальників послуг кібербезпеки, то Palo Alto Networks працює з такими масивними клієнтами, як Salesforce і Accenture. Їх продукти підтримують широкий спектр потреб, від брандмауерів і хмарної безпеки до виявлення загроз і захисту кінцевих точок, з рішеннями, які використовують машинне навчання та вбудоване глибоке навчання.

4. DataDome. DataDome використовує штучний інтелект і машинне навчання для розробки рішень, які захищають мобільні програми, веб-сайти та API від атак ботів. Завдяки низькій частоті помилкових спрацьовувань і інформаційній панелі даних у реальному часі для моніторингу, компанія затверджує, що аналітика загроз і сповіщення покращують безпеку, а також оптимізують взаємодію з користувачами.

5. Anomali. Anomali надає технологію, яка допомагає компаніям і організаціям виявляти підозрілу активність ще до того, як вона потрапить у їхні мережі. Рішення компанії для аналізу загроз дозволяють групам безпеки та аналітикам виявляти загрози та зловмисників, а також співпрацювати з іншими організаціями, щоб поділитися своїми висновками.

6. VectraAI. Vectra використовує ШІ для виявлення кібератак у реальному часі. Поєднуючи людський інтелект, науку про дані та машинне навчання, Cognito автоматизує завдання, які зазвичай виконують аналітики безпеки, і значно скорочує роботу, необхідну для проведення розслідувань загроз. Vectra складається з спеціалістів із обробки даних, інженерів із мережевої безпеки, дослідників та UI дизайнерів.

7. Check Point Software Technologies, Ltd. Check Point надає рішення з безпеки комп'ютерів і мереж урядам і підприємствам у всьому світі. Його рішення для аналізу загроз дозволяє командам безпеки проактивно стримувати загрози, контролювати мережі, керувати службами безпеки та реагувати на атаки. Оскільки загрози розвиваються швидко, Check Point надає настроювану інформацію про загрози для задоволення потреб організацій у режимі реального часу[3].

1.2 Огляд існуючих методів аналізу даних, що використовують машинне навчання

Машинне навчання — це підрозділ штучного інтелекту, в якому алгоритми вивчають шаблони з даних і забезпечують прогнози на основі цих вивчених шаблонів, застосовуючи їх до нових даних. Традиційно прості інтелектуальні системи, такі як калькулятори, розробники програмують на чітко визначені кроки та процедури. Однак це неможливо масштабувати або невідповідно для більш складних проблем.

Розглянемо приклад спам-фільтрів електронної пошти. Розробники можуть спробувати створити фільтри спаму, стало визначивши умови. Наприклад, вони можуть визначити програму, яка запускає фільтр спаму, якщо електронний лист має певну тему або певні посилання. Однак ця система виявиться неефективною, щойно спамери змінять тактику.

З іншого боку, рішення на основі машинного навчання прийматиме мільйони спам-листів як вхідні дані, вивчатиме найпоширеніші характеристики спам-листів за допомогою статистичних зв'язків і робитиме прогнози щодо майбутніх електронних листів на основі вивчених характеристик.

Окрім виявлення електронного спаму, деякі загальновідомі програми машинного навчання включають сегментацію клієнтів на основі демографічних даних (продажі та маркетинг), прогнозування курсу акцій (фінанси), автоматизацію затвердження претензій (страхування), рекомендації вмісту на основі історії переглядів (медіа та розваги), і багато іншого. Машинне навчання стало повсюдним і знаходить різноманітне застосування в нашому повсякденному житті.

Тепер, коли ми дали огляд машинного навчання та його відповідність іншим термінам, які ви можете зустріти в інформаційному просторі, давайте глибше розглянемо різні типи алгоритмів машинного навчання. Алгоритми машинного навчання загалом поділяються на контрольоване, неконтрольоване, підкріплення та самоконтрольоване навчання. Давайте розберемося з ними докладніше та з найбільш поширеними випадками використання.

I. Контрольоване навчання. Більшість випадків використання машинного навчання пов'язані з алгоритмами, які вивчають шаблони з історичних даних і застосовують їх до нових даних у формі прогнозів. Це часто називають навчанням під наглядом, або контрольованим навчанням.

Алгоритми контрольованого навчання показують як історичні вхідні дані, так і вихідні дані щодо конкретної проблеми, яку ми намагаємося вирішити, де вхідні дані — це, по суті, особливості або розміри спостереження, яке ми намагаємося передбачити, а вихідні дані — це результати, які ми хочемо передбачити. Давайте проілюструємо це на нашому прикладі виявлення спаму[4].

У випадку використання виявлення спаму, контрольований алгоритм навчання буде навчено на наборі даних спам-листів(рис.1.1). Вхідними даними будуть характеристики або параметри електронних листів, наприклад рядок теми електронного листа, адреса електронної пошти відправника, вміст електронного листа, наявність у електронному листі небезпечних на вигляд посилань та інша відповідна інформація, яка може дати підказки про те, чи електронний лист є спамом.

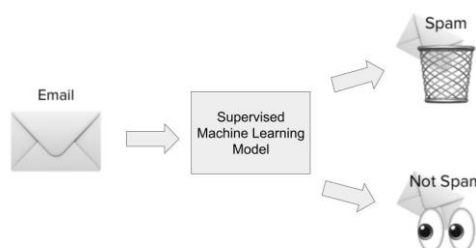


Рис.1.1 Приклад роботи алгоритму контрольованого машинного навчання для фільтрації спам листів

На виході буде виявлено, чи справді цей електронний лист був спамом чи ні. Під час фази навчання моделі алгоритм вивчає функцію для відображення статистичних зв'язків між набором вхідних змінних (різні розміри електронної пошти зі спамом) і вихідною змінною (незалежно від того, був він спамом чи ні). Потім це функціональне відображення використовується для прогнозування результату раніше невидимих даних.

Загалом існує два типи випадків використання контрольованого навчання:

- **Регресія.** Регресія використовується, коли ми намагаємося передбачити безперервний результат, який потрапляє в діапазон. Хорошим прикладом може бути прогнозування ціни на будинок на основі квадратних метрів будинку, де він розташований, кількості спалень та інших відповідних розмірів.
- **Класифікація.** Випадки використання класифікації — це коли ми намагаємося класифікувати, чи підпадає результат у дві або більше категорій. Детектори спаму є класифікаційними моделями (спам або не спам), але інші випадки використання класифікації включають прогнозування відтоку клієнтів (відпливу чи ні), ідентифікацію автомобілів на зображеннях (кілька категорій) тощо[5].

Серед методів контрольованого навчання є декілька найбільш популярних:

Лінійна регресія (Linear Regression). Простий алгоритм, який моделює лінійний зв'язок між однією або декількома пояснювальними змінними та безперервною числовою вихідною змінною(рис.1.2). Цей метод швидший для навчання порівняно з іншими алгоритмами машинного навчання. Його найбільша перевага полягає в його здатності пояснювати та інтерпретувати прогнози моделі.

Алгоритм регресії використовується для прогнозування таких результатів, як вартість життєвого циклу клієнта, ціни на житло та ціни на акції.

Метод дерева рішень (Decision Tree). Метод дерева рішень — це деревоподібна структура правил прийняття рішень, які застосовуються до вхідних функцій для прогнозування можливих результатів(рис.1.3). Його можна використовувати

для класифікації або регресії. Прогнози дерева рішень є гарною підмогою для експертів у сфері охорони здоров'я, оскільки легко інтерпретувати, як ці прогнози зроблені[6].

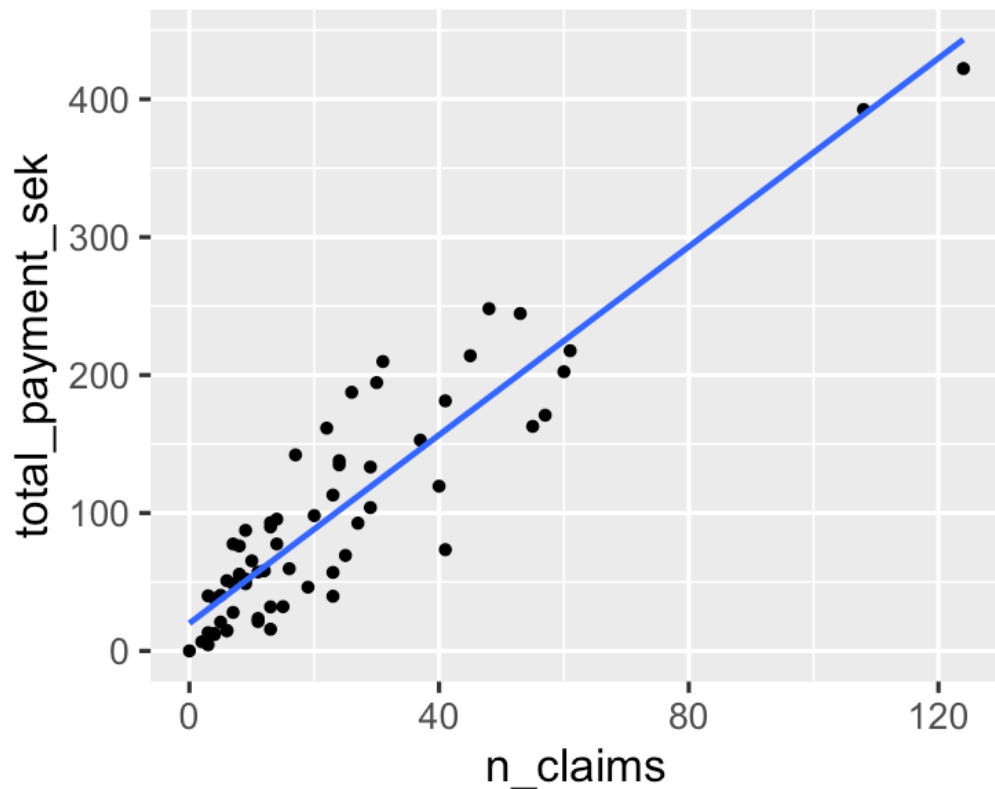


Рис. 1.2 Приклад методу лінійної регресії

Він широко використовується, коли проблема класифікації є бінарною; правда чи брехня, так чи ні і т. д. Наприклад, його можна використовувати, щоб передбачити, чи є електронний лист спамом (1) чи ні (0).

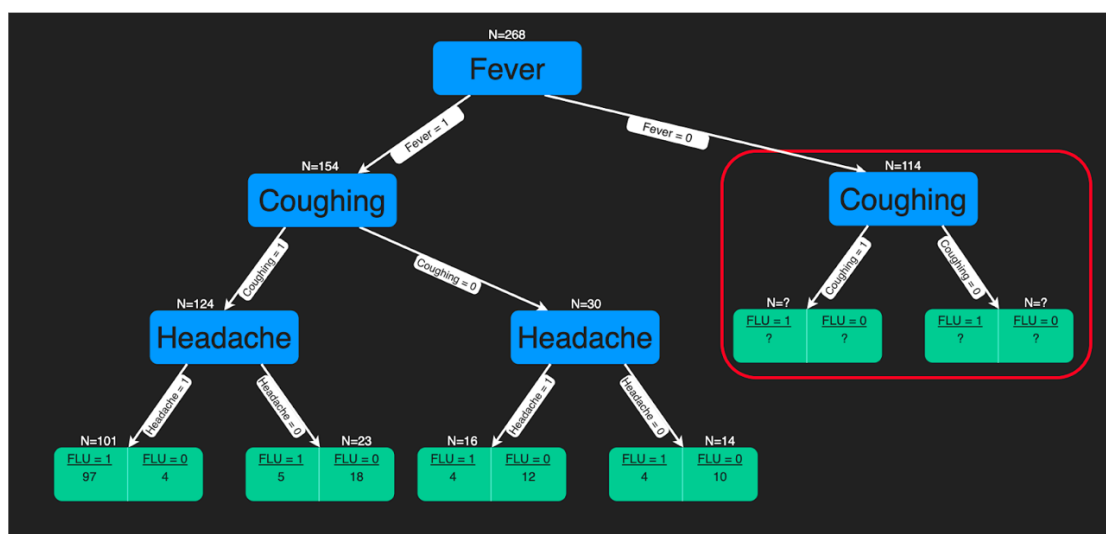


Рис.1.3 Приклад методу дерева рішень

Метод Випадкового Лісу (Random Forest). Це, мабуть, один із найпопулярніших алгоритмів, який базується на недоліках переповнення, помітних у моделях дерева рішень(рис.1.4). Переповнення виникає коли алгоритми надто добре навчені на тренувальній даних і не можуть узагальнити чи надати точні прогнози на нових невідомих даних. Випадковий ліс вирішує проблему переповнення шляхом побудови кількох дерев рішень на випадково вибраних вибірках із даних. Остаточний результат у формі найкращого передбачення виходить з голосування більшості всіх дерев у лісі.

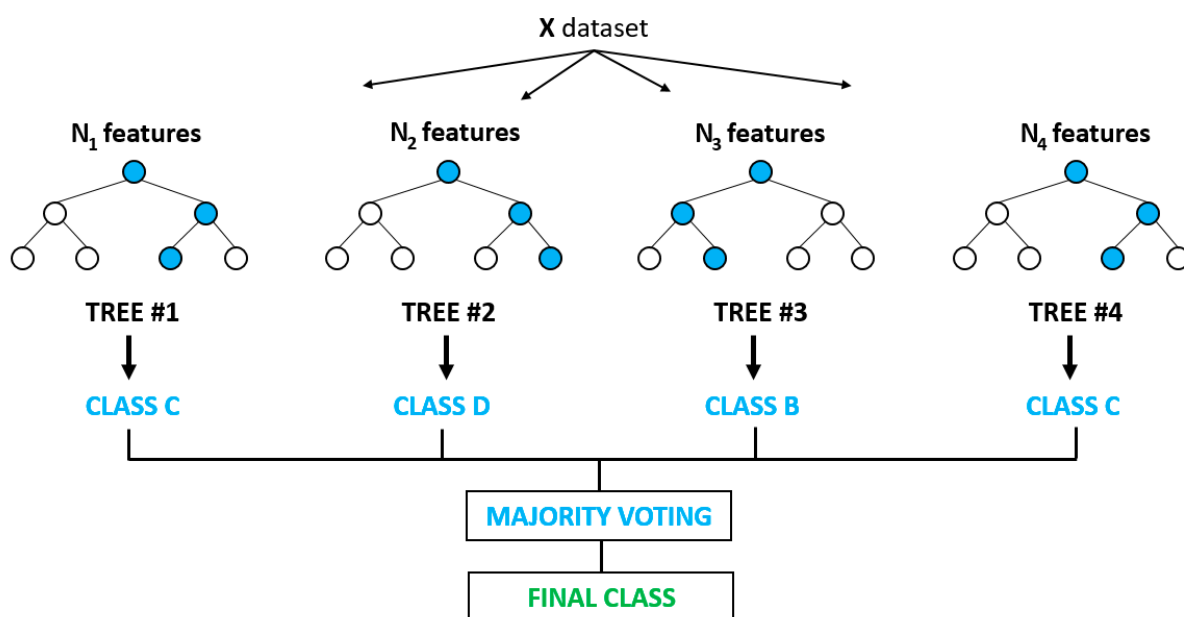


Рис.1.4 Метод випадкового лісу

Метод опорних векторів (Supporting Vector method). Метод опорних векторів, широко відомий як SVM, зазвичай використовуються для проблем класифікації. Як показано в прикладі нижче(рис.1.5), SVM знаходить гіперплощину (у цьому випадку лінію), яка розділяє два класи (червоний і зелений) і максимізує межу (відстань між пунктирними лініями) між ними. SVM зазвичай використовується для задач класифікації, але також може бути використаний у задачах регресії. Він використовується для класифікації статей новин і розпізнавання рукописного тексту[7].

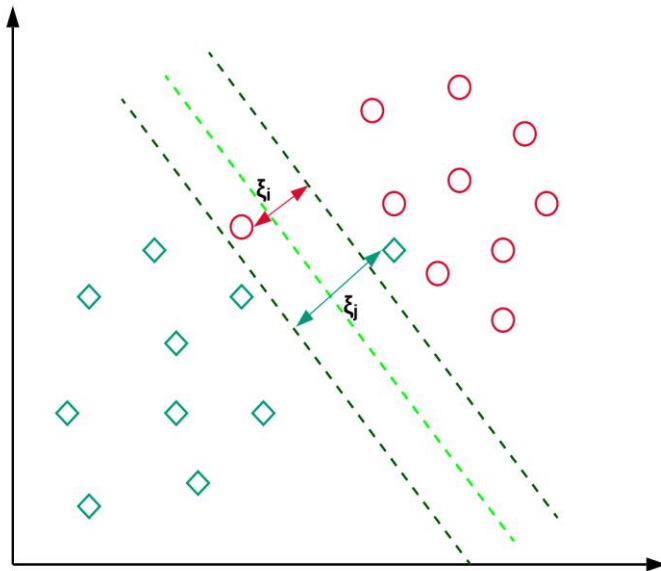


Рис. 1.5 Метод опорних векторів

II. Неконтрольоване навчання. Замість шаблонів навчання, які відображають вхідні дані та вихідні дані, алгоритми неконтрольованого навчання виявляють загальні шаблони в даних, не показуючи явно вихідні дані. Алгоритми неконтрольованого навчання зазвичай використовуються для групування та кластеризації різних об'єктів і сутностей. Чудовим прикладом неконтрольованого навчання є сегментація клієнтів. Компанії часто обслуговують різних клієнтів. Організації часто бажають використовувати підхід, заснований на фактах, для визначення сегментів клієнтів, щоб краще їх обслуговувати [8].

У цьому випадку використання алгоритм неконтрольованого навчання почне групувати клієнтів на основі різних атрибутів, таких як частота використання продукції, їхні демографічні показники, спосіб взаємодії з продуктами тощо. Потім той самий алгоритм може передбачити, до якого ймовірного сегменту належать нові клієнти, на основі подібних параметрів.

Неконтрольовані алгоритми також використовуються для зменшення розмірів у наборі даних (тобто кількості ознак) за допомогою методів зменшення розмірності. Ці алгоритми часто використовуються як проміжний крок у навчанні контрольованого алгоритму(рис.1.6).

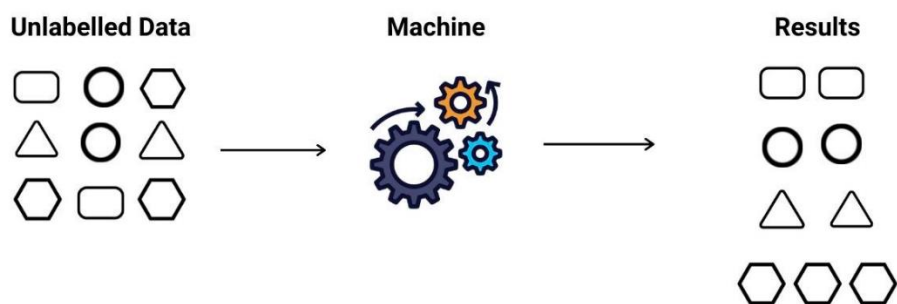


Рис. 1.6. Прикладна схема роботи алгоритму неконтрольованого навчання

Під час навчання алгоритмів машинного навчання науковці часто стикаються з великим компромісом між продуктивністю та точністю прогнозування. Як правило, чим більше інформації вони мають про певну проблему, тим краще. Однак це також може призвести до сповільнення часу навчання та продуктивності. Методи зменшення розмірності допомагають зменшити кількість ознак, присутніх у наборі даних, не жертвуючи прогностичною цінністю.

Серед алгоритмів неконтрольованого навчання також існує декілька найбільш популярних методів:

Кластеризація методом k-середніх (k-means clustering). Метод К-середніх є найпоширенішим підходом кластеризації — він визначає К кластери на основі евклідової відстані(рис.1.7). Це дуже популярний алгоритм для сегментації клієнтів і систем рекомендацій[9].

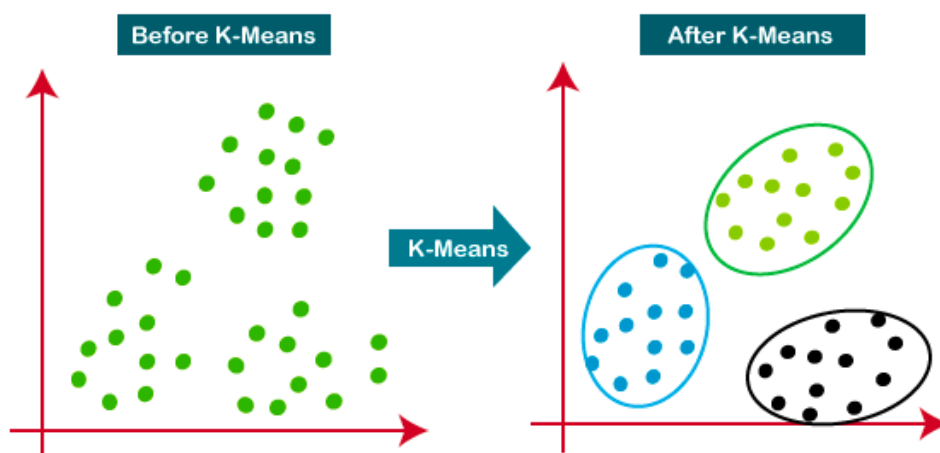


Рис. 1.7. Приклад методу К-середніх

Метод головних компонент. Аналіз головних компонент (Principal Component Analysis) — це статистична процедура, яка використовується для узагальнення інформації з великого набору даних шляхом проектування її на підпростір нижчої розмірності(1.8). Його також називають технікою зменшення розмірності, яка забезпечує збереження суттєвих частин даних із більш високою інформацією.

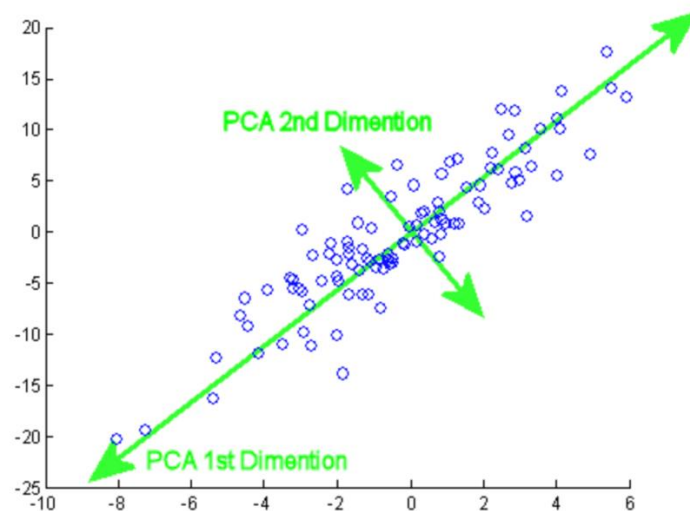


Рис. 1.8. Метод Аналізу головних компонент

Ієрархічна кластеризація (Hierarchical Clusterization). Це зворотній (bottom-up) підхід, коли кожна точка даних розглядається як окремий кластер, а потім два найближчі кластери об'єднуються разом(рис.1.9). Її найбільша перевага перед кластеризацією K-means полягає в тому, що вона не вимагає від користувача вказувати очікувану кількість кластерів на початку. Він знаходить застосування в кластеризації документів на основі подібності[10].

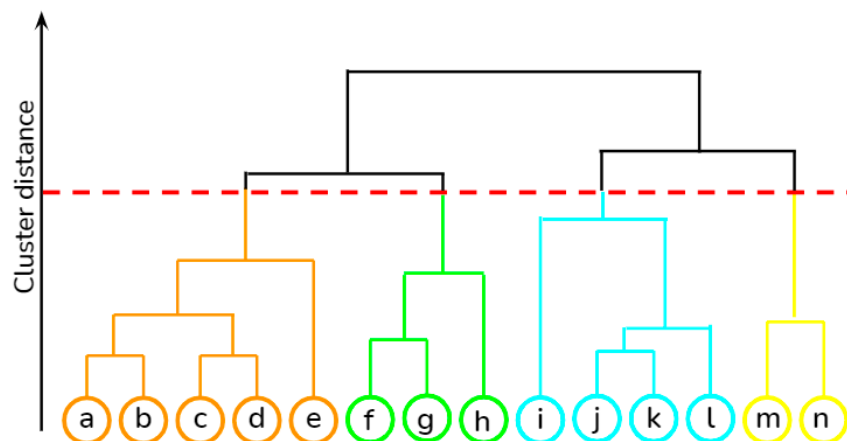


Рис. 1.9 Метод Ієрархічної кластеризації

Змішана модель Гауса. Це ймовірнісна модель для моделювання нормально розподілених кластерів у наборі даних. Він відрізняється від стандартних алгоритмів кластеризації в тому сенсі, що оцінює ймовірність того, що спостереження належить конкретному кластеру, а потім занурюється в висновки щодо його субпопуляції(рис.1.10).

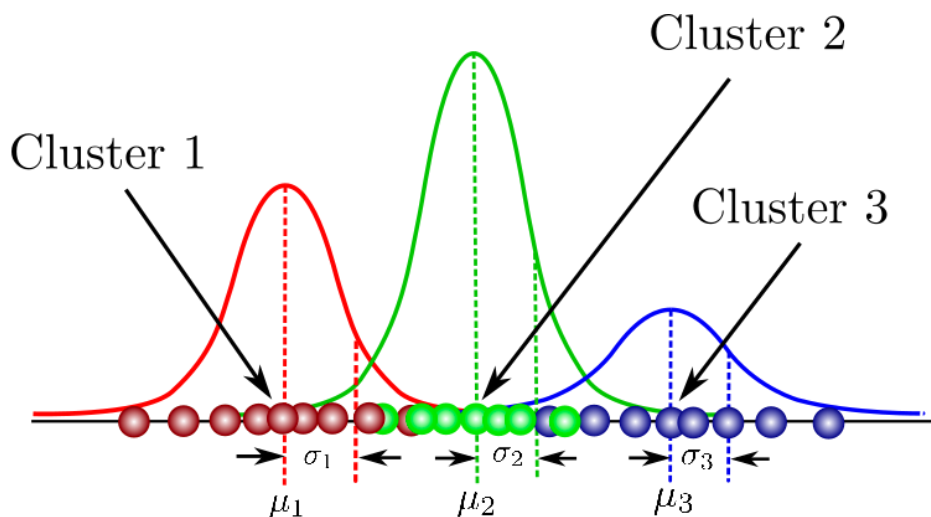


Рис. 1.10 Метод змішаної моделі Гауса

III. Навчання з підкріпленням. Навчання з підкріпленням – це підмножина алгоритмів машинного навчання, які використовують винагороди для сприяння бажаній поведінці чи прогнозу та покарання в іншому випадку. Незважаючи на те, що навчання з підкріпленням все ще залишається дослідницькою областю машинного навчання, навчання з підкріпленням відповідає за алгоритми, які перевищують інтелект людського рівня в таких іграх, як шахи, го тощо.

Це техніка поведінкового моделювання, коли модель навчається за допомогою механізму проб і помилок, постійно взаємодіючи з навколишнім середовищем(рис.1.11). Проілюструємо це на шаховому прикладі. На високому рівні алгоритму навчання з підкріпленням (часто його називають агентом) надають середовище (шахова дошка), де він може приймати різноманітні рішення (відтворювати ходи).

Кожен хід має набір пов'язаних балів, винагороду за дії, які приводять агента до перемоги, і штраф за ходи, які призводять до програшу.

Агент продовжує взаємодіяти з навколишнім середовищем, щоб навчитися діям, які приносять найбільшу винагороду, і продовжує повторювати ці дії. Таке повторення рекомендованої поведінки називається фазою експлуатації. Коли агент шукає нові шляхи отримання винагороди, це називається фазою дослідження. Більш загально це називається парадигмою розвідки та експлуатації[11].

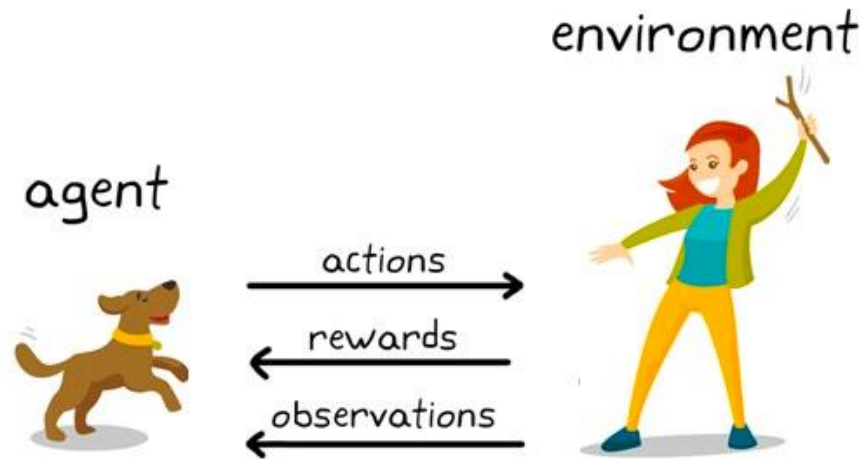


Рис. 1.11. Приклад навчання з підкріпленням

IV. Машинне навчання з самоконтролем. Самоконтрольоване навчання – це метод машинного навчання з ефективним використанням даних, у якому модель навчається з нерозміченого зразка набору даних. Як показано в наведеному нижче прикладі(рис.1.12), перша модель отримує кілька нерозмічених вхідних зображень, які вона кластеризує за допомогою функцій, створених із цих зображень.

Деякі з цих прикладів мали б високу впевненість машини щодо належності до кластерів, тоді як інші ні. На другому етапі використовуються дані з високою достовірністю з позначками першого кроку для навчання класифікатора, який має тенденцію бути потужнішим, ніж підхід однокрокової кластеризації.

Різниця між самоконтрольованим і контрольованим алгоритмами полягає в тому, що результат класифікації у першому алгоритмі все одно не матиме класів, зіставлених з реальними об'єктами. Також самоконтрольоване навчання не залежить від набору мануально створених міток, а генерує мітки самостійно, звідси й назва «самонавчання» або «самоконтроль»[11].

Для побудови моделі було обрано контрольований метод машинного навчання, а саме алгоритм Дерева прийняття рішень, адже він простий для розуміння

та візуалізації, вимагає невеликої підготовки даних та гарно підходить для поставленої задачі.

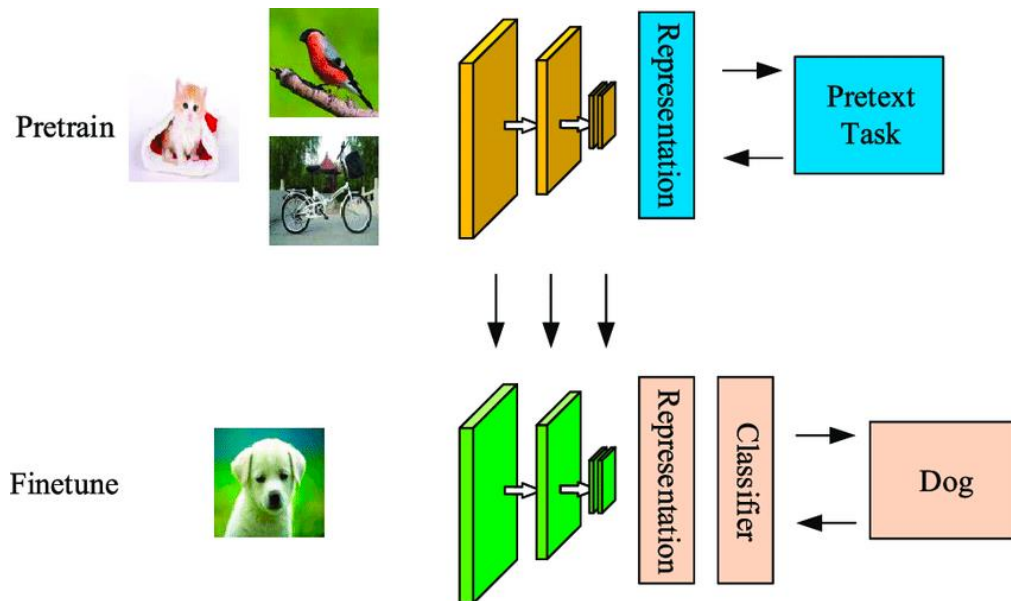


Рис. 1.12 Навчання з самоконтролем

1.3 Аналіз базових технологій веб-розробки

Розглянувши логічну частину для створення веб-додатку у вигляді алгоритмів машинного навчання, слід звернутись до функціональної частини, котра буде головною «обгорткою» нашого застосунку, а саме, до мов програмування та технологій веб-розробки.

Загально веб-розробку можна поділити на дві частини: фронтенд та бекенд.

За допомогою них складається цільна картина застосунку, його властивостей і вигляду.

Загальноприйнятими засобами розробки фронтенд інтерфейсу є набір мов та стилів, а саме:

- HTML (Hypertext Markup Language);
- CSS (Cascading Style Sheets);
- Javascript.

За допомогою цих інструментів можна побудувати будь який сучасний веб-інтерфейс або вебсайт[12].

HTML (мова розмітки гіпертексту) і CSS (каскадні таблиці стилів) є основними технологіями веб-розробки, які використовуються для створення веб-сайтів і веб-додатків. Хоча аналітику даних вони можуть здатися незначними, розуміння того, як працюють HTML і CSS, може бути корисним кількома способами:

1. Розуміння веб-аналітики. Якщо ви працюєте з веб-даними, важливо розуміти, як структуровані веб-сайти та як інструменти веб-аналітики, такі як Google Analytics, відстежують поведінку користувачів. Розуміння HTML і CSS може допомогти вам краще зрозуміти, як створюються веб-сайти та як користувачі з ними взаємодіють.

2. Візуалізація даних: HTML і CSS є будівельними блоками веб-візуалізацій даних, які стають все більш популярними в аналізі даних. Знання того, як користуватися цими технологіями, може допомогти вам створювати власні візуалізації або змінювати існуючі, щоб вони краще відповідали вашим потребам.

3. Співпраця з веб-розробниками: якщо ви працюєте над проектом, який передбачає веб-розробку, може бути корисним мати базове розуміння HTML і CSS, щоб ви могли ефективніше спілкуватися з розробниками та розуміти, над чим вони працюють.

На додаток до цього, великої популярності набирають різноманітні надбудови, що називаються фреймворками. Фреймворки допомагають покращити та полегшити процес розробки, заглибитись у специфічні галузі та сфери, досягти вирішення складніших задач[13].

Розглянемо кожний з приведених прикладів мов програмування детальніше:

HTML, або мова гіпертекстової розмітки, — це мова розмітки для Інтернету, яка визначає структуру веб-сторінок. Це один із найпростіших будівельних блоків кожного веб-сайту. Щоб зрозуміти "HTML" від початку до кінця, давайте подивимося на кожне слово, що входить до складу аббревіатури:

- **Гіпертекст:** текст (часто також із вбудованими зображеннями), організований для з'єднання пов'язаних елементів.
- **Розмітка:** посібник зі стилю для верстки всього, що буде надруковано у форматі паперової чи електронної копії.

- **Мова:** мова, яку комп'ютерна система розуміє та використовує для інтерпретації команд.

HTML визначає структуру веб-сторінок. Однієї цієї структури недостатньо, щоб зробити веб-сторінку гарною та інтерактивною. Таким чином, використовуються допоміжні технології, такі як CSS і JavaScript, щоб зробити HTML код графічно привабливим і додати інтерактивності.

Ви також можете поглянути на HTML, CSS і JavaScript таким чином: HTML — це структура будинку, CSS — це внутрішній і зовнішній декор, а JavaScript — це електрика, система водопостачання та багато інших функціональних особливостей, які роблять будинок придатним для життя[14].

CSS. CSS був розроблений W3C (World Wide Web Consortium) у 1996 році з досить простої причини. Елемент HTML не був розроблений для того, щоб мати теги, які б допомагали форматовувати сторінку. Ви мали лише написати розмітку для веб-сторінки. Такі теги, як `<font>`, були представлені в HTML версії 3.2, і це спричинило чимало проблем для веб-розробників. Через те, що веб-сторінки мають різні шрифти, кольорове тло та кілька стилів, процес переписування коду був тривалим, болісним і дорогим. Таким чином, CSS був створений W3C для вирішення цієї проблеми. Технічно CSS не є обов'язковим, але ви, ймовірно, не захочете дивитися на веб-сторінку, яка містить лише елементи HTML, оскільки це виглядатиме як чернетка сайту без оформлення. Різниця між веб-сторінкою, яка реалізує CSS, і тією, яка його не використовує, величезна й помітна. Можливо, ви бачили веб-сайт, який не завантажується повністю та має білий колір фону, а більша частина тексту синьо-чорна. Це означає, що частина CSS веб-сторінки завантажилася неправильно або її взагалі не існує.

Перш ніж використовувати CSS, усі стилі потрібно було включити в розмітку HTML. Це означає, що веб-розробники повинні були окремо описувати колір фону, розмір шрифту, вирівнювання тощо.

CSS дозволяє стилізувати все в іншому файлі, таким чином створюючи дизайн там і пізніше інтегруючи файли CSS поверх розмітки HTML. Це робить фактичну розмітку HTML набагато чистішою та простішою для обслуговування.

З функціями CSS вам не потрібно постійно описувати, як виглядають окремі елементи. Це економить час, скорочує код і робить його менш схильним до помилок.

CSS дозволяє мати кілька стилів на одній HTML-сторінці, отже, роблячи можливості налаштування майже безмежними. Нині це стає радше необхідністю, ніж додатковою послугою[15].

JavaScript. JavaScript — це динамічна мова програмування, яка використовується для веб-розробки, у веб-додатках, для розробки утиліт, тощо. Це дозволяє реалізувати динамічні функції на веб-сторінках, які неможливо зробити лише за допомогою HTML і CSS. Багато браузерів використовують JavaScript як мову сценаріїв для виконання динамічних завдань в Інтернет просторі. Щоразу, коли ви бачите спадне меню, що відкривається, додатковий вміст, доданий на сторінку, і динамічну зміну кольорів елементів на сторінці, ви бачите ефекти JavaScript.

JavaScript — мова зі слабкою типізацією (динамічно типізована). JavaScript можна використовувати як для розробки на стороні клієнта, так і для розробки на стороні сервера. JavaScript є одночасно імперативним і декларативним типом мови. JavaScript містить стандартну бібліотеку об'єктів, наприклад Array, Date і Math, а також основний набір елементів мови, як оператори, керуючі структури та ствердження[16].

Говорячи про деталі Клієнтської та Серверної частин JavaScript, можна описати підходи наступним чином:

На стороні клієнта: надає об'єкти для керування браузером і його об'єктною моделлю документа (DOM). Наприклад, розширення на стороні клієнта дозволяють програмі розміщувати елементи у формі HTML і реагувати на події користувача, такі як клацання мишею, введення форми та навігація сторінкою. Корисними бібліотеками для клієнта є AngularJS, ReactJS, VueJS та багато інших.

На стороні сервера: він надає об'єкти, що стосуються запуску JavaScript на сервері. Якщо розширення на стороні сервера дозволяють додатку спілкуватися з

базою даних і забезпечують безперервність інформації від одного виклику програми до іншого або виконують маніпуляції з файлами на сервері. Корисний фреймворк, який сьогодні є найвідомішим, — це `node.js`.

Розглянувши фундаментальні інструменти для розробки веб-додатку, можна перейти до раніше зазначеної, але не менш важливої теми – Фреймворків.

У веб розробці існує велика кількість фреймворків для застосування, і, як було зазначено раніше, більшість з них базується на JavaScript.

Фреймворк JavaScript — це набір бібліотек коду JavaScript, які надають веб-розробнику попередньо написаний код для рутинних завдань програмування.

Цілком можливо створювати потужні веб-додатки без фреймворків JavaScript, але фреймворки надають шаблон, який обробляє загальні правила програмування. Щоразу, коли вам потрібно створити програму, вам не потрібно писати код для кожної окремої функції з нуля. Натомість ви можете створити код на основі наявного набору функцій.

Усі фреймворки JavaScript, як і більшість інших фреймворків, надають деякі правила та рекомендації. Використовуючи ці правила та рекомендації, будь-який розробник може створювати складні програми швидше та ефективніше.

Правила та рекомендації також допоможуть сформувати та організувати ваш веб-сайт або веб-додаток, надати йому складнішої архітектури та форми.

Наприклад, подумайте про гончарний круг, на якому можна робити фігурки. Гончарний круг — це ваш Фреймворк; він має певні властивості, з якими вам потрібно працювати. Колесо обертається, і ви можете використовувати це обертання, щоб будувати фігурки різних форм і розмірів.

Ви можете створювати горщики, тарілки, чашки, миски або навіть циліндричні скульптури. Але ви не можете побудувати щось більше, наприклад будинок, для цього вам потрібно знайти зовсім іншу структуру та засіб. Саме тому не менш важливою задачею є обрання вірного фреймворку під ваш проєкт.

Розглянемо популярні фреймворки, що використовуються в нинішній веб розробці:

Angular. Це фреймворк, написаний на TypeScript і розроблений Google. Це фреймворк веб-додатків із відкритим кодом, який використовується для розробки односторінкових додатків (SPA)[17].

Його особливості:

1. Надає вам вільний контроль над адаптивністю та допомагає відповідати величезним вимогам до даних.
2. Завдяки простішому маніпулюванню DOM і двосторонньому зв'язуванню, неймовірній інструментарій забезпечує прості, декларативні та ефективні шаблони.
3. Розробники можуть отримати миттєву допомогу та відгуки щодо Angular майже з кожною IDE.
4. Angular використовує прив'язку даних і підтримує динамічний рендеринг.
5. Має інтуїтивно зрозумілий API і може повторно використовувати код для досягнення конкретної мети розробки.

React. React був представлений у 2013 році та розроблений Facebook. Це структура багаторазового використання, яка використовується для створення інтерактивних інтерфейсів користувача.

Його особливості:

1. Інтеграція React з будь-якою програмою є більш простою завдяки його віртуальній моделі DOM.
2. Він містить багаторазові компоненти, що полегшує розробку веб-сайту чи програми.
3. Завдяки односторонньому потоку даних, це вважається навіть більш інтуїтивно зрозумілим, ніж двонаправлене зв'язування даних.
4. Користувачі можуть одразу побачити застосовані зміни за допомогою функції гарячого перезавантаження React.

Vue.js. Vue.js — це безкоштовна платформа JavaScript із відкритим вихідним кодом, яка є прогресивною інтерфейсною структурою. Він доступний, його легко освоїти і його можна використовувати як для малих, так і для великих програм[18].

Його особливості:

1. Його компоненти містять мультифункціональні коди та попередньо визначені параметри для користувацьких елементів.
2. Фреймворк використовує структурний шаблон MVVM, який відокремлює графічний інтерфейс від бізнес-логіки моделі.
3. Незважаючи на те, що це відносно молодий фреймворк, він уже завоював популярність завдяки численним функціям.
4. Vue.js також можна використовувати для створення висококласного SPA з режимом подвійної інтеграції.

Ember.js Випущений у 2011 році ember.js є безкоштовним фреймворком JavaScript. Від SPA до корпоративних додатків, головною метою є створення інтерактивних інтерфейсів користувача незалежно від розміру веб-сайту.

Його особливості:

1. Має компонентний підхід і його легко освоїти завдяки компонентам, які є центральними для фреймворку.
2. Функції Ember.js підтримують двосторонню прив'язку даних. Це робить його більш придатним для роботи зі складними інтерфейсами користувача та великими проектами.
3. Зосередження на конфігурації, зрештою, зменшує кількість можливих помилок, але лише за рахунок коректного налаштування.

Mithril. Це сучасна структура JavaScript, яка використовується переважно для односторінкових програм або SPA. Він також використовується платформами з відкритим кодом[19].

Його особливості:

1. Mithril має досить маленький і простий API.
2. Проста взаємодія з іншими бібліотеками.
3. Легкий і має хороший механізм створення шаблонів.
4. Має віртуальний DOM і надає компільовані шаблони для розробки веб-сайту чи програми.
5. Забезпечує маршрутизацію та утиліти XHR, що забезпечує стандартну продуктивність.

6. Є декларативним і, таким чином, зменшує складність коду.

Розглянувши найпопулярніші фреймворки можна зробити висновок, що існує безліч варіантів потужних інструментів для конкретних локальних задач, і свідомий підхід до обрання фреймворку є ключовим фактором у створенні додатку.

Дуже важливо переконатися, що JavaScript належно використовується як мова програмування для веб-сайту чи програми. Наприклад, якщо використовується Java або Python для бекенду, потрібно обирати найбільш сумісний фреймворк.

Також, не менш важливо знати масштаби проекту. Це пояснюється тим, що деякі фреймворки можуть підтримувати великі додатки, тоді як деякі можна використовувати лише для створення невеличких SPA. Таким чином, слід враховувати, який фреймворк краще задовольнить потреби вашого проекту.

Швидкість веб-сайту є ще одним важливим фактором, який потрібно враховувати. Через фокус на віртуальному DOM, деякі фреймворки надають пріоритет швидким відповідям. З іншого боку, якщо ваш проект потребує більше вбудованих функцій і контролю, надійніший фреймворк буде кращим варіантом.

Деякі фреймворки JavaScript легко освоїти та реалізувати, тоді як інші можуть бути трохи складнішими, але можуть пропонувати набагато більше функцій.

Не менш важливо, є ознайомлення із засобами розробки серверної частини, розглянемо декілька з них.

PHP – призначений для веб-розробки та відмінно підходить для створення динамічних та інтерактивних веб-сторінок. Він пропонує велику підтримку баз даних, що робить його придатним для створення систем керування контентом, платформ електронної комерції та інших веб-додатків.

Node.js – дозволяє розробникам писати серверний код JavaScript - популярній мові програмування з безліччю бібліотек і платформ. Це робить Node.js легко доступним для розробників, які вже мають досвід роботи з JavaScript.

Flask — одна з найпопулярніших платформ розробки веб-додатків, що є модулем мови програмування Python, який використовується для серверної розробки

веб-додатків. Це мікрофреймворк з невеликим ядром, що легко розширюється, не включає диспетчер об'єктних відносин[20].

Django – це середовище веб-розробки на Python. Цей фреймворк пропонує стандартний метод швидкої та ефективної розробки веб-сайтів. Це веб-фреймворк високого рівня, що дозволяє здійснювати швидку розробку. Основною метою цієї веб-інфраструктури є створення складних веб-сайтів, керованих базою даних[21].

Спираючись на, перераховані вище, показники та перелік супутніх факторів, було обрано фреймворк Vue.js як основний для побудови веб-додатку, а саме клієнтської частини, оскільки він відповідає всім зазначеним вимогам та потребам для поставленої задачі. Для серверної частини було обрано фреймворк Flask оскільки, на відміну від інших фреймворків, які потребують складного налаштування та великої кількості бібліотек, Flask спроектований так, щоб бути легким та простим. Це означає, що ми можете створювати програми швидко і легко, без особливих турбот.

1.4 Аналіз мов програмування для створення штучного інтелекту

За словами галузевих експертів Springboard, не існує найкращої мови для машинного навчання, кожна хороша там, де вона підходить найкраще. Так, не існує єдиної мови машинного навчання як найкращої у всіх задачах. Проте, безумовно, є деякі мови програмування, які більш підходять для завдань машинного навчання, ніж інші. Багато інженерів штучного інтелекту обирають мову машинного навчання на основі типу бізнес-проблеми, над якою вони працюють. Наприклад, більшість вважають за краще використовувати Python для проблем NLP, а також віддають перевагу використанню R або Python для завдань аналізу настроїв. Деякі, ймовірно, використовуватимуть Java для інших програм машинного навчання, таких як безпека та виявлення загроз.

Незалежно від індивідуальних уподобань щодо певної мови програмування, можна виділити п'ять мов програмування, що підійдуть до більшості задач, та нададуть універсальне розуміння кращої для конкретного випадку:

Python. Оскільки понад 8,2 мільйона розробників у всьому світі використовують Python для програмування, Python посідає перше місце в останньому щорічному рейтингу популярних мов програмування IEEE Spectrum із результатом 100. Тенденції мов програмування за даними StackOverflow чітко показують(рис.1.13), що це єдина мова, яка набирає обертів для останні п'ять років[22].

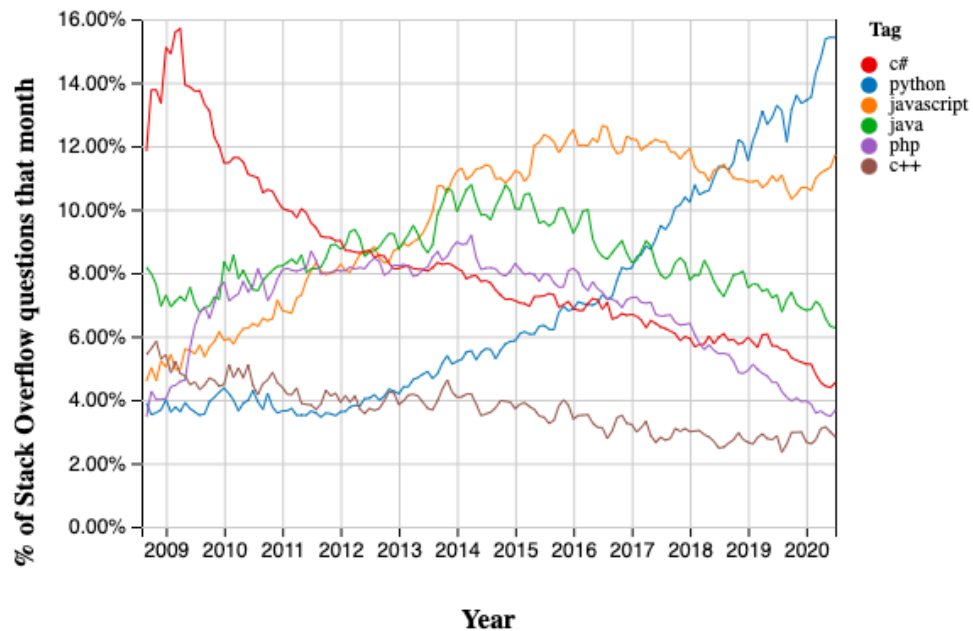


Рис. 1.13 Порівняльна статистика популярності мов програмування

Дедалі більша популярність машинного навчання в усьому світі є головним фактором, що сприяє зростанню популярності Python. У 69% інженерів машинного навчання, Python став улюбленим вибором для аналізу даних, науки про дані, машинного навчання та штучного інтелекту – і все це завдяки його величезній бібліотечній екосистемі, яка дозволяє практикам машинного навчання отримувати доступ, аналізувати, перетворювати та обробляти дані з легкістю.

Python є улюбленою мовою програмування для машинного навчання для деяких гігантів у світі IT, таких як Google, Instagram, Facebook, Dropbox, Netflix, Walt Disney, YouTube, Uber, Amazon і Reddit. Python є беззаперечним лідером і, безумовно, найкращою мовою для машинного навчання на сьогодні, і ось чому:

1. *Велика колекція бібліотек і пакетів.* Вбудовані бібліотеки та пакети Python забезпечують код базового рівня, тому інженерам з машинного навчання не

потрібно починати писати все з нуля кожного разу. Машинне навчання вимагає постійної обробки даних, і Python має вбудовані бібліотеки та пакети майже для кожного завдання. Це допомагає інженерам машинного навчання скоротити час розробки та підвищити продуктивність під час роботи зі складними програмами машинного навчання. Найкраща частина цих бібліотек і пакетів полягає в тому, що не має кривої навчання, як тільки ви дізнаєтеся про основи програмування на Python, ви можете почати використовувати ці бібліотеки одразу.

2. *Читабельність коду.* Зручність використання Python також полягає в тому, що надає можливість використовувати короткі, лаконічні, читабельні класи, які виражають багато дій у невеликій кількості чіткого коду, замість довгих складних мануальних функцій. Математика машинного навчання зазвичай складна й неочевидна. Таким чином, читабельність коду надзвичайно важлива для успішного впровадження складних алгоритмів машинного навчання та різноманітних робочих процесів. Простий синтаксис Python і те значення, яке він надає читабельності коду, дозволяє інженерам машинного навчання легко зосередитися на тому, що писати, замість того, щоб думати про те, як писати. Зручність читання коду полегшує розробникам обмін ідеями, алгоритмами та інструментами зі своїми колегами. Python користується популярністю не тільки серед інженерів машинного навчання, але також є однією з найпопулярніших мов програмування серед науковців з обробки даних.

3. *Гнучкість.* Багатопарадигмальний і гнучкий характер Python дозволяє інженерам машинного навчання легко підходити до проблеми найпростішим способом. Він підтримує процедурний, функціональний, об'єктно-орієнтований та імперативний стиль програмування, що дозволяє експертам з машинного навчання комфортно працювати над тим, який підхід найкраще підходить. Гнучкість, яку пропонує Python, допомагає інженерам машинного навчання вибрати стиль програмування на основі типу проблеми – інколи було б корисно зафіксувати стан об'єкта, а в інших випадках проблема може вимагати передачі функцій як параметрів. Python забезпечує гнучкість у виборі будь-якого з підходів і мінімізує ймовірність помилок[23].

Мова програмування R. Маючи понад 2 мільйони користувачів R, 12000 пакетів у репозиторії з відкритим кодом CRAN, близько 206 груп R Meetup, понад 4000 запитань щодо програмування R щомісяця та понад 40 тисяч учасників у групі R LinkedIn — R — це неймовірна мова програмування для машин навчання, написана фахівцем статистиком для статистиків. Мова R також може використовуватися не лише програмістами, зокрема, майнерами даних, аналітиками даних і статистиками.

Важливою частиною повсякденної роботи інженера з машинного навчання є розуміння статистичних принципів. Мова програмування R є фантастичним вибором, коли справа доходить до обробки великих чисел, і є кращим вибором для програм машинного навчання, які використовують багато статистичних даних. Завдяки зручним IDE, таким як RStudio, і різноманітним інструментам для малювання графіків і керування бібліотеками, R є дуже зручною мовою програмування в інструментарії інженерів штучного інтелекту.

Ось що робить R однією з найефективніших мов машинного навчання:

1. R — це мова програмування з відкритим вихідним кодом, що робить її дуже економічно ефективним вибором для проєктів машинного навчання будь-якого розміру.
2. R підтримує природну реалізацію матричної арифметики та інших структур даних, таких як вектори, які зараз підтримує і Python. Для подібної реалізації на мові програмування Python інженерам машинного навчання доводиться використовувати пакет NumPy, який є більш незграбною реалізацією порівняно з R.
3. R вважається потужним вибором для машинного навчання через широкий спектр методів машинного навчання, який він надає. Будь то візуалізація даних, вибірка даних, аналіз даних, оцінка моделі, кероване/некероване машинне навчання — R пропонує широкий набір методів.
4. Стиль програмування на мові R досить простий.
5. R є дуже гнучким і також пропонує кросплатформену сумісність. R не накладає обмежень під час виконання кожного завдання своєю мовою, практики

машинного навчання можуть змішувати інструменти – вибрати найкращий інструмент для кожного завдання, а також користуватися перевагами інших інструментів разом із R[23].

Java. Хоча Python і R залишаються фаворитами ентузіастів машинного навчання, Java набирає популярності серед інженерів які мають попередній досвід розробки на даній мові програмування, оскільки їм не потрібно вивчати нову мову, щоб реалізувати машинне навчання. Багато організацій уже мають величезні кодові бази Java, і більшість інструментів з відкритим кодом для обробки великих даних, таких як Hadoop або Spark, написані на Java. Використання Java для проектів машинного навчання полегшує інженерам інтеграцію з існуючими сховищами коду.

Такі функції, як простота використання, краща взаємодія з користувачем, легке налагодження та графічне представлення даних роблять цю мову хорошим вибором для машинного навчання:

1. У Java є багато сторонніх бібліотек для машинного навчання. JavaML — це вбудована бібліотека машинного навчання, яка надає набір алгоритмів машинного навчання, реалізованих у Java. Крім того, ви можете використовувати бібліотеку Arbiter Java для налаштування гіперпараметрів, яка є невід’ємною частиною ефективної роботи алгоритмів ШІ, або ви можете використовувати бібліотеку Deeplearning4J, яка підтримує такі популярні алгоритми машинного навчання, як K-Nearest Neighbor і Neuroph, і дозволяє створювати нейронні мережі.

2. Масштабованість є важливою особливістю, яку кожен інженер машинного навчання повинен враховувати перед початком проекту. Java полегшує масштабування програм для інженерів машинного навчання, що робить її чудовим вибором для розробки великих і складних програм машинного навчання з нуля.

3. Віртуальна машина Java є однією з найкращих платформ для машинного навчання, оскільки інженери можуть писати один і той самий код на кількох платформах. JVM також допомагає інженерам машинного навчання швидко створювати спеціальні інструменти та має різні IDE, які допомагають підвищити загальну продуктивність. Java найкраще працює для проектів машинного навчання, які мають критичне значення для швидкості, оскільки вона швидко виконується[23].

Julia. Julia — це високопродуктивна динамічна мова програмування загального призначення, яка може стати потенційним конкурентом для Python і R з багатьма основними функціями виключно для машинного навчання. Зважаючи на те, що це мова програмування загального призначення, і її можна використовувати для розробки будь-яких програм, вона найкраще підходить для високопродуктивного числового аналізу та обчислювальної науки. Завдяки підтримці всіх типів апаратного забезпечення, включаючи TPU та GPU у кожній хмарі, Julia підтримує програми машинного навчання у таких великих корпораціях, як Apple, Disney, Oracle і NASA.

Julia спеціально розроблена для реалізації базових математичних і наукових запитів, які лежать в основі більшості алгоритмів машинного навчання.

Код Julia компілюється в режимі Just-in-Time або під час виконання за допомогою фреймворку LLVM. Це дає інженерам машинного навчання велику швидкість без будь-яких ручних методів профілювання чи методів оптимізації, що вирішує всі проблеми продуктивності.

Також код Julia є універсально виконуваним. Отже, після написання програми машинного навчання її можна скомпілювати в Julia з інших мов, як Python або R, у оболонку, як PyCall або RCall.

Масштабованість, як неодноразово згадувалось раніше, має вирішальне значення для інженерів машинного навчання, і Julia полегшує швидке розгортання у великих кластерах. Завдяки таким потужним інструментам, як TensorFlow, MLBase.jl, Flux.jl, SciKitlearn.jl та багатьом іншим, які використовують масштабованість, надану Julia, це сильний вибір для програм машинного навчання.

Lisp. Заснована в 1958 році Джоном Маккарті, LISP (List Processing) є другою найстарішою мовою програмування, яка все ще використовується, і в основному розроблена для програм, орієнтованих на ШІ. LISP — це динамічно типізована мова програмування, яка вплинула на створення багатьох мов програмування машинного навчання, таких як Python, Julia та Java. LISP працює на циклі Read-Eval-Print-Loop (REPL) і має можливість кодувати, компілювати та запускати код на 30+ мовах програмування.

LISP вважається найефективнішою та найгнучкішою мовою машинного навчання для вирішення конкретних завдань, оскільки він адаптується до рішення, для якого програміст пише код[24].

Саме цим LISP відрізняється від інших мов машинного навчання. Сьогодні він особливо використовується для індуктивних логічних задач і машинного навчання. Перший чат-бот ELIZA зі штучним інтелектом був розроблений за допомогою LISP, і навіть сьогодні спеціалісти з машинного навчання можуть використовувати його для створення чат-ботів для електронної комерції. LISP однозначно заслуговує на, оскільки навіть сьогодні розробники покладаються на LISP для проєктів штучного інтелекту, які потребують важкого машинного навчання.

Незважаючи на те, що LISP є гнучким для машинного навчання, йому бракує підтримки відомих бібліотек для використання у ШІ. Також, LISP досить складна мова для опанування, і не має такої великої спільноти користувачів, як у Python або R.

За вище описаними мовами програмування було обрано Python як найкраще середовище для розгортання машини штучного інтелекту, її навчання та поєднання з веб-додатком, оскільки Python також має широкий спектр бібліотек для веб-розробки.

1.5 Постановка завдання на розробку

Розробка поставленого проєкту складатиметься з двох частин – моделі машинного навчання, та веб-інтерфейсу. Для створення моделі машинного навчання буде використано Python як основне логічне ядро, Flask для зв'язку з веб-інтерфейсом, та ряд бібліотек для машинного навчання.

Побудова моделі складатиметься з таких пунктів:

- Аналіз вхідних даних для розмітки та формування закономірності між функціями.
- Імпорт даних до середовища Python та їх обробка за допомогою бібліотек машинного навчання.
- Нормалізація даних та їх підготовка.

- Розподіл даних на навчальний та тестувальний сегменти.
- Створення моделі та її навчання.
- Аналіз якості прогнозування моделі.
- Розгортання моделі як веб-додатку.

Друга частина розробки додатку, це створення веб-інтерфейсу для моделі машинного навчання. В першу чергу, веб-інтерфейс має бути інтуїтивно зрозумілим, приємним для користування, та змістовним. Створення веб-інтерфейсу складається з програмування фронтенд та бекенд частин.

Розробка фронтенд частини включає в себе створення компонентів, які будуть взаємодіяти з користувачем, інформаційних панелей, тощо. Реалізація простого та привабливого інтерфейсу є важливою частиною для майбутнього процесу користування веб-додатком, оскільки будь-який користувач в першу чергу буде оцінювати зовнішній вигляд застосунку, та простоту навігації у ньому.

Розробка бекенд частини міститиме всю технічну складову веб-додатку. Це буде основний механізм взаємодії користувацького веб-інтерфейсу та моделі машинного навчання за лаштунками. В бекенд частині розробляються відповідні засоби обробки даних, структуризації, інтеграції з веб-інтерфейсом, та зв'язок з моделлю машинного навчання. Відповідно, користувач не матиме доступу до серверно-логічної частини, проте буде отримувати бажаний результат від платформи.

Після розробки додатку, він пройде відповідне тестування за загальноприйнятим протоколом з шести пунктів:

- функціональність;
- використовність;
- інтерфейс;
- сумісність;
- продуктивність;
- безпека.

Висновки до 1-го розділу:

В результаті аналізу алгоритмів машинного навчання та мов програмування веб-додатків, було визначено основні фактори для створення веб-додатку на основі ШІ, та обрано найбільш доцільні засоби для створення відповідного проєкту.

Проаналізовано представників безпекової аналітичної сфери, котрі використовують технології штучного інтелекту у своїй діяльності, за аналізом приведено актуальність машинного навчання у сферах безпеки та дата аналізу.

Було розглянуто принципи роботи з технологіями фронтенд та бекенд розробки у напрямку машинного навчання та дата аналізу. Проведено порівняльні характеристики.

РОЗДІЛ 2 РОЗРОБКА ВЕБ-ДОДАТКУ АНАЛІЗУ СТАТИЧНОГО МЕРЕЖЕВОГО ТРАФІКУ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ

2.1 Аналіз вхідних даних

Як і в будь-якому застосунку для дата аналізу, початок роботи завжди ведеться з отримання оригінального набору даних, над яким будуть проводитись усі необхідні операції.

Вибір для нашого застосунку був зроблений з оглядом на задачі безпеки та машинного навчання, і, саме тому було обрано набір даних KDD'99.

Це славнозвісний набір даних, який використовувався для Третього міжнародного конкурсу інструментів дата майнінгу та дата аналізу, який проводився разом із П'ятою міжнародною конференцією KDD-99.

Конкурсне завдання полягало в створенні детектора мережевого вторгнення, прогнозної моделі, здатної розрізняти «погані» з'єднання, які називаються вторгненнями або атаками, і «хороші» звичайні з'єднання. Ця база даних містить стандартний набір даних, які підлягають перевірці, включаючи широкий спектр вторгнень, змодельованих у військовому мережевому середовищі.

Залучений набір даних підходить найкращим чином під поставлені задачі проєкту, та є міжнародно затвердженим орієнтиром для тренування на навчання машин штучного інтелекту.

Початковий файл було обрано у форматі CSV задля найкращої сумісності з сучасними аналітичними бібліотеками та зручного використання кінцевим користувачем. Перед початком огляду практичного вигляду файлу слід зазначити, що таке CSV.

Comma-separated values (CSV) – це формат текстового файлу, у якому для розділення значень використовуються коми. Файл CSV зберігає табличні дані (числа та текст) у вигляді звичайного тексту, де кожен рядок файлу зазвичай представляє один запис даних. Кожен запис складається з однакової кількості полів, розділених комами у файлі CSV. Якщо сам роздільник поля може з'являтися в полі, поля можна взяти в лапки.

Формат файлу CSV є одним із типів формату файлу, з роздільниками. До роздільників, які часто використовуються, належать кома, табуляція, пробіл і крапка з комою. Файли, розділені роздільниками, часто мають розширення ".csv", навіть якщо роздільник полів не є комою. Багато програм або бібліотек, які використовують або створюють файли CSV, мають параметри визначення альтернативного розділювача.

Відсутність дотримання стандарту CSV RFC 4180 вимагає підтримки різноманітних форматів CSV у програмному забезпеченні для введення даних. Незважаючи на цей недолік, CSV залишається широко поширеним у програмах обробки даних і широко підтримується різноманітним програмним забезпеченням, включаючи звичайні програми для роботи з електронними таблицями, такі як Microsoft Excel. Переваги CSV включають легкість читання та простоту формату.

Після обрання початкового набору даних та формату файлу, слід провести ознайомлення з його вмістом та властивостями.

Початковий файл містить 494 021 рядок, кожний з яких відповідає 41 стовпцю властивостей, що у сумі надає нам 20 254 861 змінних для навчання та тренування моделі. Цей набір даних буде розділено на навчальну та тренувальну частини, задля підвищення якості моделі(рис.2.1).

kddcup_data_10_percent_corrected																																							
0	tcp	http	SF	181	5450	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	239	486	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	235	1337	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	219	1337	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	217	2032	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	217	2032	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	212	1940	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	159	4087	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	210	151	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	212	786	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	210	624	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	177	1985	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	222	773	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	256	1169	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	241	259	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	260	1837	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	241	261	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	257	818	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	233	255	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	233	504	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	256	1273	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	234	255	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	241	259	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	239	968	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	245	1919	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	248	2129	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		
0	tcp	http	SF	354	1752	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	normal.		

Рис. 2.1 Початковий CSV файл з необробленим набором даних

Після встановлення бібліотек слід розібрати властивості файлу, за якими буде проводитись аналіз та прогнозування даних(рис.2.3).

```
col_names = ['duration','protocol_type','service','flag','src_bytes','dst_bytes','land','wrong_fragment','urgent','hot',
            'num_failed_logins','logged_in','num_compromised','root_shell','su_attempted','num_root',
            'num_file_creations','num_shells','num_access_files','num_outbound_cmds','is_host_login',
            'is_guest_login','count','srv_count','error_rate','srv_error_rate','error_rate','srv_error_rate',
            'same_srv_rate','diff_srv_rate','srv_diff_host_rate','dst_host_count','dst_host_srv_count',
            'dst_host_same_srv_rate','dst_host_diff_srv_rate','dst_host_same_src_port_rate','dst_host_srv_diff_host_rate',
            'dst_host_error_rate','dst_host_srv_error_rate','dst_host_error_rate','dst_host_srv_error_rate']
]
```

Рис. 2.3 Стовпці властивостей для полів інтернет трафіку

Наступним кроком є трансформація даних до бінарного вигляду, оскільки наша модель працює за методом класифікації(рис.2.4).

```
[ ] dataset = pd.read_csv('traffic_data_10_percent_corrected.csv', sep=",")

dataset['normal.'] = dataset['normal.'].replace(
    ['back.', 'buffer_overflow.', 'ftp_write.', 'guess_passwd.', 'imap.', 'ipsweep.', 'land.', 'loadmodule.',
    'multihop.', 'neptune.', 'nmap.', 'perl.', 'phf.', 'pod.', 'portsweep.', 'rootkit.', 'satan.', 'smurf.', 'spy.',
    'teardrop.', 'warezclient.', 'warezmaster.'], 'attack')
```

Рис. 2.4 Трансформація даних до бінарного вигляду

Заключним етапом обробки даних є їх маркування та трансформація для введення у навчальний та тренувальний набори(рис.2.5). Таким чином дані будуть заздалегідь підготовані до навчання у моделі.

```
[ ] x = dataset.iloc[:, :-1].values
    y = dataset.iloc[:, 41].values

[ ] labelencoder_x_1 = LabelEncoder()
    labelencoder_x_2 = LabelEncoder()
    labelencoder_x_3 = LabelEncoder()
    x[:, 1] = labelencoder_x_1.fit_transform(x[:, 1])
    x[:, 2] = labelencoder_x_2.fit_transform(x[:, 2])
    x[:, 3] = labelencoder_x_3.fit_transform(x[:, 3])
    # onehotencoder_1 = ColumnTransformer([('oh_enc', OneHotEncoder(), [1,2,3])], remainder='passthrough')
    # x = onehotencoder_1.fit_transform(x)
    # onehotencoder_2 = ColumnTransformer([('oh_enc', OneHotEncoder(), [4])], remainder='passthrough')
    # x = onehotencoder_2.fit_transform(x)
    # onehotencoder_3 = ColumnTransformer([('oh_enc', OneHotEncoder(), [7])], remainder='passthrough')
    # x = onehotencoder_3.fit_transform(x)
    labelencoder_y = LabelEncoder()
    y = 1 - labelencoder_y.fit_transform(y)

pd.DataFrame(x)
```

Рис. 2.5 Маркування та трансформація даних у навчальний та тренувальний набори

Для зручності перегляду, трансформовані дані виводяться у вигляді проміжної таблиці(рис.2.6).



	0	1	2	3	4	5	6	7	8	9	...	31	32	33	34	35	36	37	38	39	40
0	0	1	22	9	239	486	0	0	0	0	...	19	19	1.0	0.0	0.05	0.0	0.0	0.0	0.0	0.0
1	0	1	22	9	235	1337	0	0	0	0	...	29	29	1.0	0.0	0.03	0.0	0.0	0.0	0.0	0.0
2	0	1	22	9	219	1337	0	0	0	0	...	39	39	1.0	0.0	0.03	0.0	0.0	0.0	0.0	0.0
3	0	1	22	9	217	2032	0	0	0	0	...	49	49	1.0	0.0	0.02	0.0	0.0	0.0	0.0	0.0
4	0	1	22	9	217	2032	0	0	0	0	...	59	59	1.0	0.0	0.02	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
494015	0	1	22	9	310	1881	0	0	0	0	...	86	255	1.0	0.0	0.01	0.05	0.0	0.01	0.0	0.0
494016	0	1	22	9	282	2286	0	0	0	0	...	6	255	1.0	0.0	0.17	0.05	0.0	0.01	0.0	0.0
494017	0	1	22	9	203	1200	0	0	0	0	...	16	255	1.0	0.0	0.06	0.05	0.06	0.01	0.0	0.0
494018	0	1	22	9	291	1200	0	0	0	0	...	26	255	1.0	0.0	0.04	0.05	0.04	0.01	0.0	0.0
494019	0	1	22	9	219	1234	0	0	0	0	...	6	255	1.0	0.0	0.17	0.05	0.0	0.01	0.0	0.0

494020 rows x 41 columns

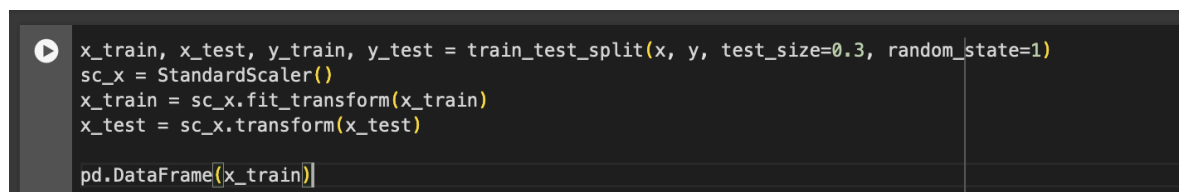
Рис. 2.6 Таблиця з трансформованими проміжними даними

2.2 Розробка та навчання моделі штучного інтелекту

Основним діючим механізмом роботи веб-додатку є модель машинного навчання, котра працює за алгоритмом дерева рішень. Для її створення було обрано комбінацію мови програмування Python, додатку Flask та вбудованих бібліотек машинного навчання.

Створення моделі машинного навчання є наступним етапом після аналізу вхідних даних, оскільки на попередньо оброблених даних модель і буде навчатись.

Першим кроком після аналізу та підготовки даних, є їх розбиття на навчальні та тренувальні набори(рис.2.7). Таким чином модель зможе навчитись на меншій частині файлу розпізнавати загрози, а на більшій проведе фінальне випробування якості роботи.



```

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1)
sc_x = StandardScaler()
x_train = sc_x.fit_transform(x_train)
x_test = sc_x.transform(x_test)

pd.DataFrame(x_train)

```

Рис. 2.7 Трансформація та розбиття даних на навчальний та тренувальний набори

Також, для зручності перегляду було виведено проміжкову таблицю з розбитими трансформованими групами даних(рис.2.8).

Рис. 2.8 Таблиця проміжкових трансформованих даних розбитих на групи

Після розбиття даних можна почати розгортання моделі машинного навчання. Як раніше і було зазначено, модель буде базуватись на алгоритмі дерева рішень(рис.2.9).

```
[ ] classifier = tree.DecisionTreeClassifier()
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
```

Рис. 2.9 Розгортання моделі дерева рішень

Даний уривок коду розгортає логічну частину машини, та надає можливість аналізувати та прогнозувати результат.

Дані передаються з тренувального набору до машини, де вона навчається про- ставляти відповідні прогнози та мітки, а після направляються у тестовий набір для виведення фінальних показників.

Фінальним етапом розгортання моделі є відстеження якості роботи моделі. На- ступний уривок коду виводить усі стандартні метрики моделі(рис.2.10).

```
[ ] cm = confusion_matrix(y_test, y_pred)

accuracies = cross_val_score(estimator=classifier, X=x_train, y=y_train,
cv=10)
print("CV Accuracy: %0.4f (+/- %0.4f)" % (accuracies.mean(), accuracies.
std() * 2))

# the performance of the classification model
print("Test Accuracy is : %0.4f" % ((cm[0, 0] + cm[1, 1]) /
(cm[0, 0] + cm[0, 1] + cm[1, 0] + cm[1, 1])))
recall = cm[1, 1] / (cm[0, 1] + cm[1, 1])
print("Recall is : %0.4f" % (recall))
print("False Positive rate: %0.4f" % (cm[1, 0] / (cm[0, 0] + cm[1, 0])))
precision = cm[1, 1] / (cm[1, 0] + cm[1, 1])
print("Precision is: %0.4f" % (precision))
print("F-measure is: %0.4f" % (2 * ((precision * recall) /
(precision + recall))))

print("Entropy is: %0.4f" % (-precision * log(precision)))
print("\nConfusion Matrix:\n", cm)
```

Рис. 2.10 Виведення оцінки якост роботи моделі машинного навчання

Окрім текстових метрик модель також виводить графічні представлення, такі як матриця неточностей та ROC крива(рис.2.11-2.12).

```
[ ] ax = sns.heatmap(cm / np.sum(cm), annot=True,
                    fmt='.2%', cmap='Blues')
ax.set_title('Percentage Confusion Matrix\n')
ax.set_xlabel('Predicted Category')
ax.set_ylabel('Actual Category ')
sns.set(rc={'figure.figsize': (10, 7)})
plt.show()
```

Рис. 2.11 Виведення матриці неточностей

```
def plot_roc_curve(fpr, tpr):
    plt.figure(figsize=(10, 8), dpi=100)
    plt.axis('scaled')
    plt.xlim([0, 1])
    plt.ylim([0, 1])
    plt.title("AUC & ROC Curve")
    plt.plot(fpr, tpr, 'r')
    plt.fill_between(fpr, tpr, facecolor='lightblue', alpha=0.7)
    plt.text(0.95, 0.05, 'AUC = %0.4f' % auc, ha='right', fontsize=12, weight='bold', color='blue')
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.show()

y_pred = classifier.predict_proba(x_test)
y_pred = y_pred[:, 1]

auc = roc_auc_score(y_test, y_pred)
print('AUC: ', auc)

fpr, tpr, thresholds = roc_curve(y_test, y_pred)

plot_roc_curve(fpr, tpr)
```

Рис. 2.12 Виведення ROC кривої

2.3 Проектування та розробка веб-додатку

Фінальним етапом створення проекту є створення веб-додатку для кінцевого користувача. Попередні етапи розробки надали нам весь необхідний функціонал для роботи з даними та машинним навчанням, проте користувачу потрібне зручне та просте середовище для роботи.

Першим кроком створення користувацького веб-додатку є розгортання серверної частини(рис.2.13).

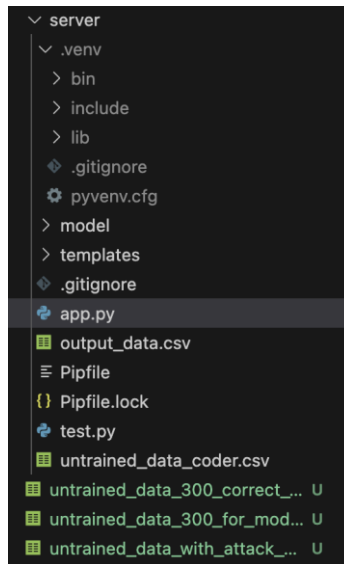


Рис. 2.13 Загальна структура серверної частини додатку

Для початку ми додаємо бібліотеки для роботи з вхідними даними від моделі машинного навчання, та їх передачі до клієнтської частини(рис.2.14).

```
from flask import Flask, request, jsonify, json, make_response
from flask_cors import CORS, cross_origin
import pickle
import numpy as np
import pandas as pd
import os
import csv
import codecs, json
from json import JSONEncoder
from sklearn.preprocessing import LabelEncoder
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
```

Рис. 2.14 Додавання бібліотек для серверної частини

Після успішного встановлення бібліотек, слід налаштувати зв'язок між серверною частиною та клієнтською за допомогою CORS політик. Це допоможе нам передавати файл для обробки машиною безшовно та зручно для користувача(рис.2.15).

```
19 app = Flask(__name__)
20 # CORS(app)
21 # CORS(app, resources={r'/*': {'origins': 'http://localhost:8080'}}, CORS_SUPPORTS_CREDENTIALS = True)
22 # app.config['CORS_HEADERS'] = 'Content-Type'
23 CORS(app, resources={r"/prediction": {"origins": "http://localhost:8080"}}, supports_credentials = True)
24 # CORS(app, origins=["http://localhost:8080", "http://localhost:8080"])
25
```

Рис. 2.15 Розгортання CORS політик

Наступним кроком є імпорт навченої моделі для подальшого прогнозування міток для вхідних даних(рис.2.16).

```

26 APP_ROOT = os.path.dirname(os.path.abspath(__file__))
27 MODEL_PATH = os.path.join(APP_ROOT, "../model/attacks_classifier_model.pkl")
28 model = pickle.load(open(MODEL_PATH, 'rb'))
29

```

Рис. 2.16 Передавання даних від моделі до веб-додатку

Після отримання даних від моделі, вони обробляються та трансформуються перед початком прогнозування(рис.2.17).

```

40 def transformData(data):
41     labelencoder_x_1 = LabelEncoder()
42     labelencoder_x_2 = LabelEncoder()
43     labelencoder_x_3 = LabelEncoder()
44     data[:, 1] = labelencoder_x_1.fit_transform(data[:, 1])
45     data[:, 2] = labelencoder_x_2.fit_transform(data[:, 2])
46     data[:, 3] = labelencoder_x_3.fit_transform(data[:, 3])
47
48     sc_x = StandardScaler()
49     data_prepared = sc_x.fit_transform(data)
50     return data_prepared
51

```

Рис. 2.17 Перетворення даних

Для початку, нам потрібно закодувати категоріальні змінні, оскільки наша модель розпізнає тільки чисельні значення. Останнім кроком перетворення буде стандартизація вхідних даних.

Сервер буде приймати дані за маршрутом «/prediction». Далі нам потрібно прочитати вхідні дані з переданого файлу. Для подальшої роботи, з отриманого файлу ми формуємо масив даних, який передаємо в описану вище функцію трансформації даних. Функція повертає готовий набір даних для прогнозування.

Спрогнозований набір даних ми формуємо масив об'єктів, який повертаємо у клієнтську частину в json форматі(рис.2.18).

```

53 @app.route('/prediction', methods = ['POST', 'OPTIONS'])
54 def upload_file():
55     if request.method == "OPTIONS":
56         return _build_cors_preflight_response()
57     if request.method == 'POST':
58         try:
59             file = request.files['csvFile']
60             print(file)
61             # file.save(file.filename)
62             so = csv.reader(codecs.iterdecode(file, 'utf-8'), delimiter=',', quotechar='')
63             so_data = []
64             for row in so:
65                 so_data.append(row)
66             so_data = pd.DataFrame(so_data)
67             so_data = so_data.iloc[:, :].values
68             transformed_data = transformData(so_data)
69             predicted_data = model.predict(transformed_data)
70             arrData = []
71             i = 1
72             for item in predicted_data:
73                 arrData.append({'item': i, 'predicted': item})
74                 i += 1
75             json_str = predicted_data.tolist()
76             return json.dumps(arrData, cls=NpEncoder)
77             # return _corsify_actual_response(jsonify(arrData.tolist()))
78         except Exception as e:
79             return jsonify({'error': str(e)}), 400
80
81

```

Рис. 2.18 Функція обробки вхідних даних на серверній частині

Після розгортання серверної частини слід розробити клієнтську, яка буде головним інтерфейсом взаємодії кінцевого користувача з нашою моделлю.

У розробці користувацького інтерфейсу задіяно загальноприйняті технології front-end розробки та зв'язку між серверною та клієнтською частинами.

Структура користувацької частини виглядатиме наступним чином(рис.2.19):

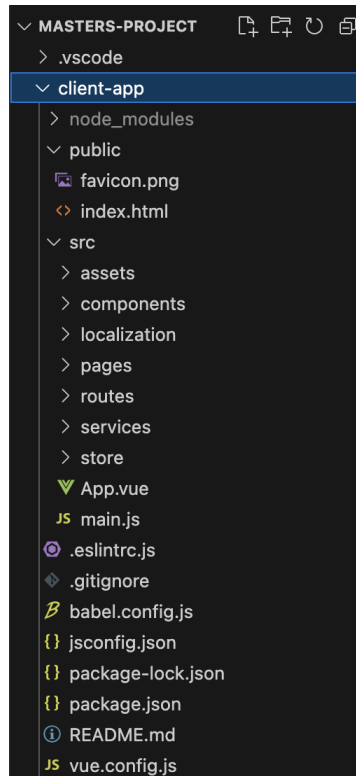


Рис. 2.19 Структура клієнтського інтерфейсу додатку

Першим основним блоком логіки та взаємодії є папка “Routes”, де прописано усі маршрути зв'язку між сторінками, за якими буде переходити користувач(рис.2.20).

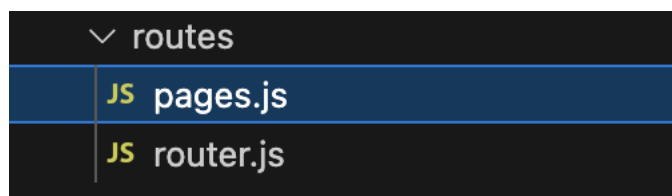


Рис. 2.20 Структура Папки “Routes”

Дана папка відповідає за коректне перенаправлення між сторінками, взаємодію користувача та серверної частини додатку, відправленні та отриманні наборів даних.

У першому файлі “Pages.js” описано маршрути переходу між сторінками користувацького інтерфейсу. Це є одним з ключових файлів, оскільки без прокладання вірних маршрутів, перехід між сторінками та взаємодія даних між серверною та клієнтською частинами будуть неможливі(рис.2.21).

```
client-app > src > routes > JS pages.js > ...
1  import LayoutPage from '@pages/Layout';
2
3  const routes = [
4    {
5      path: 'home',
6      name: 'home',
7      component: () => import('@pages/children/Home'),
8    },
9    {
10     path: 'about-us',
11     name: 'about-us',
12     component: () => import('@pages/children/AboutUs'),
13   },
14   {
15     path: 'analytics-dashboard',
16     name: 'analytics-dashboard',
17     component: () => import('@pages/children/AnalyticsDashboard'),
18   },
19   {
20     path: 'evaluation',
21     name: 'evaluation',
22     component: () => import('@pages/children/Evaluation'),
23   },
24   {
25     path: 'contact-us',
26     name: 'contact-us',
27     component: () => import('@pages/children/ContactUs'),
28   },
29 ]
30
31 const pagesRoutes = [
32   {
33     path: '/',
34     component: LayoutPage,
35     name: 'layout',
36     redirect: '/home',
37     children: routes,
38   },
39   {path: '*', component: LayoutPage},
40 ];
41
```

Рис. 2.21 Прокладання маршрутів між сторінками веб-додатку

Наступним ключовим етапом після налаштування маршрутів є підключення зв'язку між серверною та користувацькою частинами для передачі даних.

Оскільки суть додатку у постійному обміні наборами даних для їх аналізу, це є життєво важливою функцією.

Даний функціонал описаний у дерикторії “Services”, та має наступну структуру(рис.2.22):



Рис. 2.22 Структура дерикторії Services

У даному файлі “api.js” прописано налаштування API інтерфейсу, котрий дозволяє обмін даними між моделлю машинного навчання та користувацьким інтерфейсом.

За загальною логікою, користувач відправляє необроблений файл з даними до серверної частини, де модель машинного навчання обробляє дані та прогнозує результати, і після обробки відправляє прогнозований файл назад до користувацької сторінки як фінальний результат(рис.2.23).

```
client-app > src > services > JS api.js > ...
1  const HOST = 'http://localhost:5000';
2
3  async function getPrediction(data) {
4    let formData = new FormData();
5    formData.append("csvFile", data);
6    console.log(data);
7    const response = await fetch(`${HOST}/prediction`, {
8      method: "POST",
9      body: formData,
10   });
11
12   let result;
13
14   try {
15     result = await response.json();
16   } catch(err) {
17     console.log(err)
18   }
19
20   return result;
21 }
22
23
24 export {
25   getPrediction
26 }
```

Рис. 2.23 Функція обміну даними між серверною та користувацькою частинами

Останнім підготовчим елементом перед розробкою користувацьких сторінок веб-додатку є підключення усіх необхідних бібліотек та елементів для графічного представлення.

Ці дії описані у файлі “main.js”(рис.2.24), котрий налічує перелік підключених бібліотек для виведення таблиць, графіків, статистик, локалізації та ін. Він є ключовим файлом у проекті.

```

client-app > src > JS main.js > ...
1  import Vue from 'vue'
2  import App from './App.vue'
3  import VueRouter from 'vue-router';
4  import VueI18n from 'vue-i18n';
5  import VueCookies from 'vue-cookies';
6  import SmartTable from './vuejs-smart-table';
7  import VueGraph from './vue-graph';
8  import Donut from './vue-css-donut-chart';
9  import 'vue-css-donut-chart/dist/vcdonut.css';
10 import { library } from '@fortawesome/fontawesome-svg-core';
11 import { FontAwesomeIcon } from '@fortawesome/vue-fontawesome';
12 import { faUserSecret, faHouse, faTable, faEnvelope, faEarthAmerica, faStar } from '@fortawesome/free-solid-svg-icons';
13 // import GraphLine3D from 'vue-graph/src/components/line3d.js';
14 // import NoteWidget from 'vue-graph/src/widgets/note.js';
15 // import LegendWidget from 'vue-graph/src/widgets/legends.js';
16 import store from './store'
17
18 import localizationEN from './localization/en.js';
19
20 // Style
21 import './assets/scss/reset.scss'
22 import './assets/scss/main.scss'
23
24 Vue.config.productionTip = false
25
26 import router from './routes/router';
27
28
29 library.add(faUserSecret, faHouse, faTable, faEnvelope, faEarthAmerica, faStar)
30
31 Vue.use(VueRouter);
32 Vue.use(VueI18n);
33 Vue.use(VueCookies);
34 Vue.use(SmartTable);
35 Vue.use(VueGraph);
36 Vue.use(Donut);
37 Vue.component('FontAwesomeIcon', FontAwesomeIcon);
38 // Vue.component(GraphLine3D.name, GraphLine3D);
39 // Vue.component(NoteWidget.name, NoteWidget);
40 // Vue.component(LegendWidget.name, LegendWidget);

```

Рис. 2.24 Підключення бібліотек та функцій до веб-додатку у файлі main.js

Тепер, коли усі попередні етапи підготовки проведено, і наша серверна частина має зв'язок з користувацькою, можна перейти до розробки відповідних сторінок, з якими буде напряду працювати кінцевий користувач.

Усі відповідні сторінки знаходяться у дерикторії “Pages”(рис.2.25):

- Home – Перша сторінка веб-додатку, яка містить загальну інформацію та орієнтує користувача щодо функцій додатку (додаток А);
- Analytics Dashboard – Головна аналітична сторінка з графічними та табличними прогнозованими даними (додаток Б);
- Evaluation – Статистична сторінка з оцінкою якості роботи моделі (додаток В);
- Contact Us – Контактна інформація додатку для зворотнього зв'язку (додаток Г);

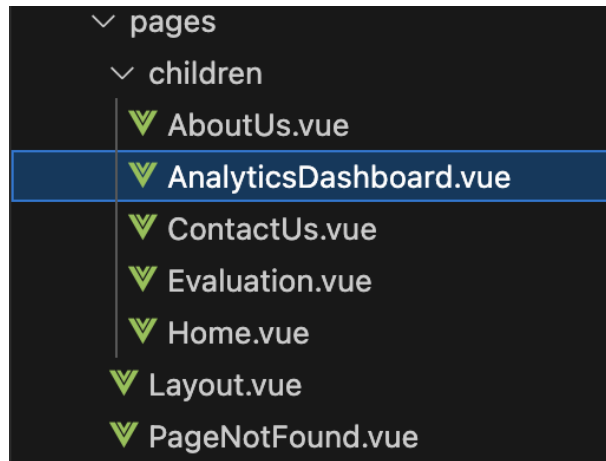


Рис. 2.25 Структура дерикторії Pages

Головною інформаційною сторінкою для користувача буде Analytics Dashboard, де буде виведено усю інформацію щодо прогнозованих даних, таку як графіки, табличні дані, та статистика.

Початком роботи з аналітичним дешбордом є завантаження файлу з трафіком користувачем. На етапі коли користувач завантажив файл, починає свою роботу метод “handleFileUpload”, який приймає вхідні дані, перетворює їх на масив об’єктів, та локально зберігає(рис.2.26).

```
handleFileUpload($event) {
  this.file = $event.target.files[0];
  const reader = new FileReader();
  if (this.file) {
    reader.onload = (e) => {
      this.createData(e);
      this.saveToLocalStorage();
    };
    reader.readAsText(this.file);
  }

  if(this.checkLocalStorageSuccessfull) {
    this.successfullPrediction = localStorage.removeItem('isSuccessfull');
  }
},
selectRow(item) {
  this.selectedRows = this.selectedRows !== item ? item : null;
},
createData(e) {
  let content = [];
  let splitedContent = e.target.result.split(/\r\n|\n/);
  content.push(splitedContent);

  this.tableContent = [];

  content[0].forEach((item, index) => {
    this.tableContent.push({id: ++index, item: item.split(',') , predicted: ''});
  });
},
saveToLocalStorage() {
  if(this.checkLocalStorageData) {
    localStorage.removeItem('storedData');
  }
  localStorage.setItem('storedData', JSON.stringify(this.tableContent));
},
```

Рис. 2.26 Метод обробки даних з файлу “handleFileUpload”

Таким чином ми матимемо підготовані розбиті дані, які можна буде використовувати для графічного та статистичного представлення на сторінці у зручному вигляді.

Локальне збереження даних дозволить нам зберігати усі результати після закриття або перезавантаження сторінки веб-додатку.

Після відправки даних користувачем, починає свою роботу метод “submitData”, який призначає відповідні спрогнозовані мітки до об’єктів. Таким чином користувач буде бачити та відрізняти шкідливий трафік від нормального за числовим значенням(рис.2.27).

```
submitData() {  
  getPrediction(this.file).then((res) => {  
    this.result = res;  
  }).then(() => {  
    this.addPrediction();  
    this.showSuccessMsg();  
    this.successfullPrediction = true;  
    localStorage.setItem('isSuccessfull', this.successfullPrediction);  
  })  
  .catch((e) => {  
    console.log('Something went wrong...', e);  
  })  
},
```

Рис. 2.27 Метод “submitData”

Після того, як наші дані успішно передані та спрогнозовані, можна виводити графічні та статистичні представлення для кінцевого користувача.

Спочатку виводиться статистична інформація щодо файлу, яка містить загальну кількість пакетів трафіку, кількість шкідливих та нормальних пакетів, а також кількість сервісів та протоколів(рис.2.28).

На рисунку 2.29 ми можемо побачити як дані відображаються у нашому веб-додатку.

```

5      <div
6        class="statistic"
7        v-if="successfullPrediction"
8      >
9        <div>
10         <p>{{ $('analyticsDashboard.text1') }}</p>
11         <p class="count">
12           {{ tableContent.length }}
13         </p>
14       </div>
15       <div>
16         <p>{{ $('analyticsDashboard.text2') }}</p>
17         <p class="count">
18           {{ getHarmfulTraffic }}
19         </p>
20       </div>
21       <div>
22         <p>{{ $('analyticsDashboard.text3') }}</p>
23         <p class="count">
24           {{ getUsefulTraffic }}
25         </p>
26       </div>
27       <div>
28         <p>{{ $('analyticsDashboard.text4') }}</p>
29         <p class="count">
30           {{ namesProtocol.length }}
31         </p>
32       </div>
33       <div>
34         <p>{{ $('analyticsDashboard.text5') }}</p>
35         <p class="count">
36           {{ nameServices.length }}
37       </div>

```

Рис. 2.28 Функція виведення статистичних даних



Рис. 2.29 Відображення статистичних даних у веб-додатку

Після загальної числової статистики користувачу пропонується дві кругові діаграми з відсотковим співвідношенням шкідливого та нормального трафіку до усього набору даних. Код реалізації ми можемо побачити на рисунку 2.30. У компонент графіка ми передаємо функцію, яка фільтрує наші дані відповідно вимогами, а результат ми передаємо в іншу функцію, яка перетворює отримані дані у відсотковий формат.

На рисунку 2.31 ми бачимо результат виведення вище зазначеного коду.

```

<div class="first-line-graph">
  <div class="donut-graph">
    <vc-donut
      :sections="sectionHarmful"
      :total="tableContent.length"
    >
      <span>Harmful Traffic<br></span>
      <span class="number">{{ dataFormat(getHarmfulTraffic) }}</span>
    </vc-donut>
    <vc-donut
      :sections="sectionUseful"
      :total="tableContent.length"
    >
      <span>Useful Traffic<br></span>
      <span class="number">{{ dataFormat(getUsefulTraffic) }}</span>
    </vc-donut>
  </div>
  <graph-bar
    :width="600"
    :height="300"
    :axis-min="0"
    :axis-reverse="true"
    :axis-max="Math.max(...valueProtocol) + 20"
    :axis-step="10"
    :text-rotate-x="90"
    :labels="['All protocols']"
    :values="valueProtocol"
  >
    <note :text="'Show each protocol'" />
    <tooltip
      :names="namesProtocol"
      :position="'left'"
    />
    <legends
      :names="namesProtocol"
      :filter="true"
    />
  </graph-bar>
</div>

```

Рис. 2.30 Функція виведення кругових діаграм

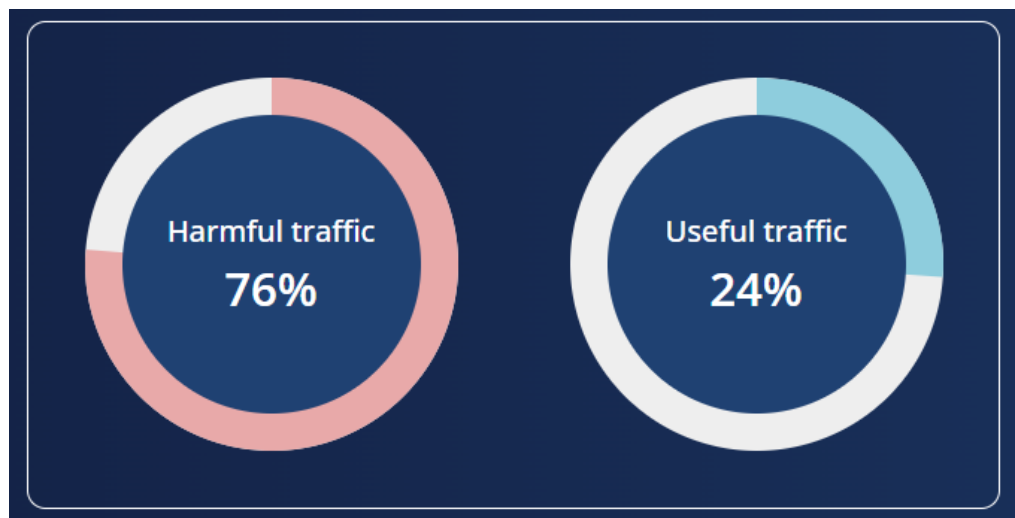


Рис. 2.31 Відображення кругових діаграм у веб-додатку

Поруч із круговими діаграмами розташовано стовпчиковий графік зі статистикою використаних протоколів підключення(рис.2.33). За допомогою цієї інформації можна буде відстежити найчастіший спосіб підключення відповідно. Лістинг коду ми бачимо на рисунку 2.32.

```
<graph-bar
  :width="600"
  :height="300"
  :axis-min="0"
  :axis-reverse="true"
  :axis-max="Math.max(...valueProtocol) + 20"
  :axis-step="10"
  :text-rotate-x="90"
  :labels="['All protocols']"
  :values="valueProtocol"
>
  <note :text="'Show each protocol'" />
  <tooltip
    :names="namesProtocol"
    :position="'left'"
  />
  <legends
    :names="namesProtocol"
    :filter="true"
  />
</graph-bar>
</div>
```

Рис. 2.32 Функція виведення протоколів підключення

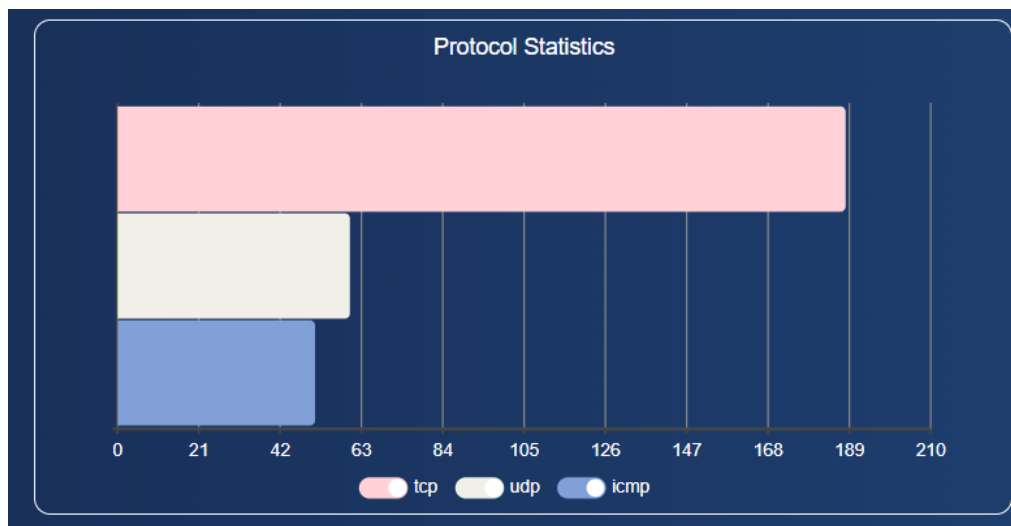


Рис. 2.33 Відображення протоколів підключення у веб-додатку

Наступним не менш важливим графічним представленням є виведення усіх сервісів з'єднання наявних у наборі даних(рис.2.34). Цей метод за своєю логікою є

схожим на метод виведення протоколів підключення, та також зображений у вигляді стовпчикової діаграми(рис.2.35).

```
<div>
  <graph-bar
    :width="900"
    :height="400"
    :axis-min="0"
    :padding-bottom="100"
    :axis-max="Math.max(...valueServices) + 20"
    :labels=["All Services"]
    :values="valueServices"
  >
    <note :text="'Show each services'" />
    <tooltip
      :names="nameServices"
      :position="'left'"
    />
    <legends
      :names="nameServices"
      :filter="true"
    />
  </graph-bar>
</div>
```

Рис. 2.34 Функція виведення сервісів з'єднання

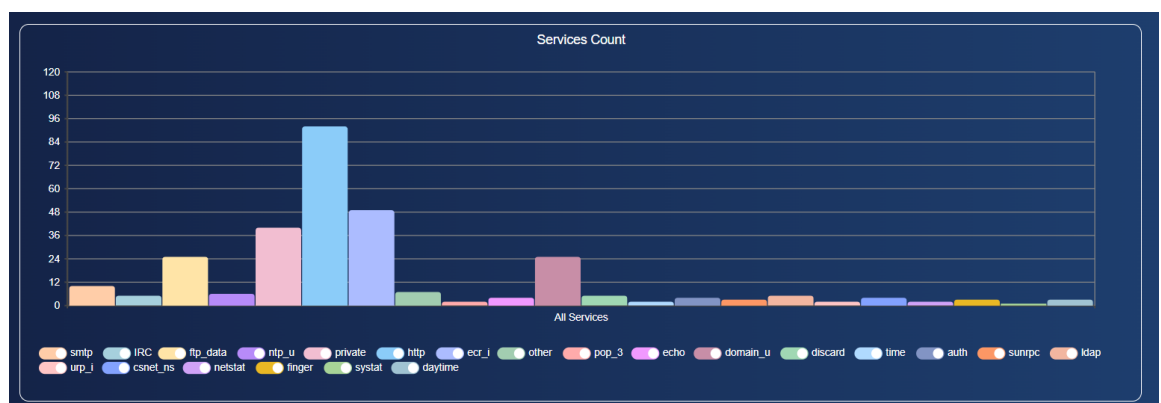


Рис. 2.35 Відображення сервісів з'єднання у веб-додатку

Останнім графічним представленням у дашборді є гістограма(рис.2.37) з відображенням концентрації шкідливого трафіку на довжині усього набору даних. За цим графіком можна відстежити найактивніші точки під час атак.

Лістинг коду гістограму ми можемо побачити на рисунку 2.36.

```

<graph-line
  :width="1200"
  :height="400"
  :shape="'curve'"
  :clip="true"
  :axis-min="0"
  :axis-max="Math.max(...predictedHarmfulArray)"
  :axis-full-mode="false"
  :labels="getHarmfulTrafficForLineGraph"
  :styles="{
    gridXFontColor : '#fff',
  }"
  :names="['All harmful traffic']"
  :values="predictedHarmfulArray"
>
  <note :text="'Line Chart'" />
  <legends :names="['All harmful traffic']" />
  <guideline :tooltip-y="true" />
</graph-line>
</div>
</section>
</div>
</template>

```

Рис. 2.36 Функція виведення гістограми

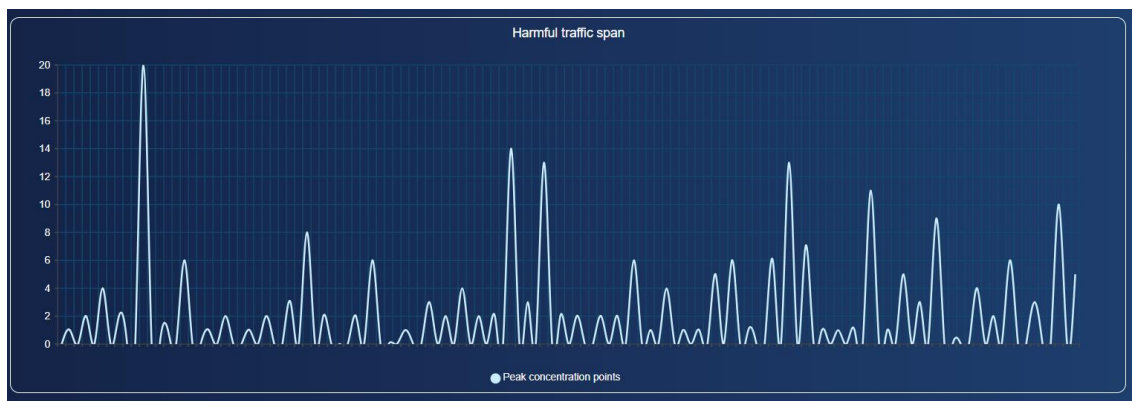


Рис. 2.37 Відображення гістограми

Ключовим і заключним елементом аналітичного дашборду є таблиця усіх прогнозованих даних (рис. 2.39), яка виводиться наприкінці сторінки. Дана таблиця виводить кожен рядок трафіку у наборі даних, відповідні властивості та прогнозовані мітки. За допомогою таблиці можна відстежити будь-який рядок файлу, переглянути усі наявні детальні властивості, фільтрувати шкідливий трафік, а також здійснити пошук по типу протоколу або сервісу.

Лістинг коду таблиці зображений на рисунку 2.38.

```

<v-table
  class="table"
  :data="tableContent"
  :current-page.sync="currentPage"
  :page-size="20"
  :filters="filters"
  @totalPagesChanged="totalPages = $event"
>
  <thead slot="head">
    <tr>
      <th
        v-for="(header, index) in colNames"
        :key="index"
      >
        {{ header }}
      </th>
    </tr>
  </thead>
  <tbody
    slot="body"
    slot-scope="{displayData}"
  >
    <tr>
      v-for="(row, rowKey) in displayData"
      :key="row.id"
      :class="{ 'active-row': selectedRows === row.id }"
      @click="selectRow(row.id)"
    >
      <td>{{ row.id }}</td>
      <td>
        v-for="(column, columnKey) in row.item"
        :key="'row-'+rowKey+'-column-'+columnKey"
      >
        {{ column }}
      </td>
      <td class="predicted">
        {{ row.predicted }}
      </td>
    </tr>
  </tbody>
</v-table>

```

Рис. 2.38 Функція виведення таблиці прогнозованих даних

No	duration	protocol	type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	root_shell
1	35	tcp		smtp	SF	2383	477	0	0	0	0	0	1	0	0
2	3572	tcp		irc	RSTR	402	4657	0	0	0	0	0	1	0	0
3	0	tcp		ftp_data	SF	334	0	0	0	0	0	0	1	0	0
4	0	udp		ntp_u	SF	48	48	0	0	0	0	0	0	0	0
5	0	udp		private	SF	28	0	0	3	0	0	0	0	0	0
6	0	tcp		http	SF	229	304	0	0	0	0	0	1	0	0
7	0	tcp		http	SF	302	2164	0	0	0	0	0	1	0	0
8	0	icmp		ecr_j	SF	1032	0	0	0	0	0	0	0	0	0
9	0	tcp		http	SF	227	1247	0	0	0	0	0	1	0	0
10	0	udp		private	SF	28	0	0	3	0	0	0	0	0	0
11	0	tcp		http	SF	222	773	0	0	0	0	0	1	0	0
12	5374	udp		other	SF	147	105	0	0	0	0	0	0	0	0
13	0	tcp		pop_3	SO	0	0	0	0	0	0	0	0	0	0
14	0	tcp		ftp_data	SF	334	0	0	0	0	0	0	1	0	0
15	0	tcp		ftp_data	SF	641	0	0	0	0	0	0	1	0	0
16	0	tcp		echo	SO	0	0	0	0	0	0	0	0	0	0
17	0	tcp		http	SF	217	2032	0	0	0	0	0	1	0	0
18	0	tcp		http	SF	240	2164	0	0	0	0	0	1	0	0
19	0	udp		domain_u	SF	42	42	0	0	0	0	0	0	0	0
20	0	tcp		discard	SO	0	0	0	0	0	0	0	0	0	0

Рис. 2.39 Відображення таблиці прогнозованих даних

Повний лістинг програмного коду надано у [додатку Д](#).

Висновки до 2-го розділу:

В даному розділі було розглянуто спосіб аналізу та підготовки набору даних для машинного навчання. способи та методи розробки та навчання моделі ШІ, а також способи та підходи до розробки користувацького веб-інтерфейсу.

Наведено приклади та функції графічного представлення прогнозованих даних, їх цілі та задачі, проведено огляд структури додатку та його побудови.

РОЗДІЛ 3 ПЕРЕВІРКА ПРАЦЕЗДАТНОСТІ ТА ЕФЕКТИВНОСТІ ДЕТЕКТОРА АТАК НА ОСНОВІ АЛГОРИТМІВ ДЕРЕВ РІШЕНЬ

3.1 Тестування та оцінка якості моделі машинного навчання

Після основного етапу розробки моделі та її полірування, настає етап перевірки якості та працездатності навченої машини. Цей етап є ключовим перед застосуванням моделі у веб-додатку, і саме тому слід розглянути його детальніше.

Проблема алгоритму класифікації є однією з найбільш широко досліджуваних областей у світі. Варіанти використання присутні майже в усіх виробничих і промислових середовищах. Розпізнавання мовлення, розпізнавання обличчя, класифікація тексту – список нескінченний.

Моделі класифікації мають дискретний результат, тому нам потрібна метрика, яка порівнює дискретні класи в певній формі. Показники класифікації оцінюють продуктивність моделі та повідомляють, наскільки класифікація хороша чи погана, але кожна з них оцінює її по-різному.

Отже, щоб оцінити модель класифікації, ми детально обговоримо наступні показники:

- Точність;
- Матриця невідповідності;
- Акуратність і Повнота;
- F1-оцінка;
- AUC-ROC Крива;

Метрика Акуратності є, мабуть, найпростішим показником для використання та впровадження, який визначається як кількість правильних прогнозів, формула (3.1), поділена на загальну кількість прогнозів, помножену на 100.

$$ACC = \frac{tp + tn}{tp + fp + tn + fn} \quad (3.1)$$

Матриця невідповідностей – це таблична візуалізація міток істинних значень у порівнянні з прогнозами моделі. Кожен рядок матриці представляє екземпляри в

передбаченому класі, а кожен стовпець представляє екземпляри в фактичному класі[25].

Матриця невідповідностей(рис.3.1) — це не зовсім показник ефективності, а свого роду основа, на якій інші метрики виводять свою оцінку.

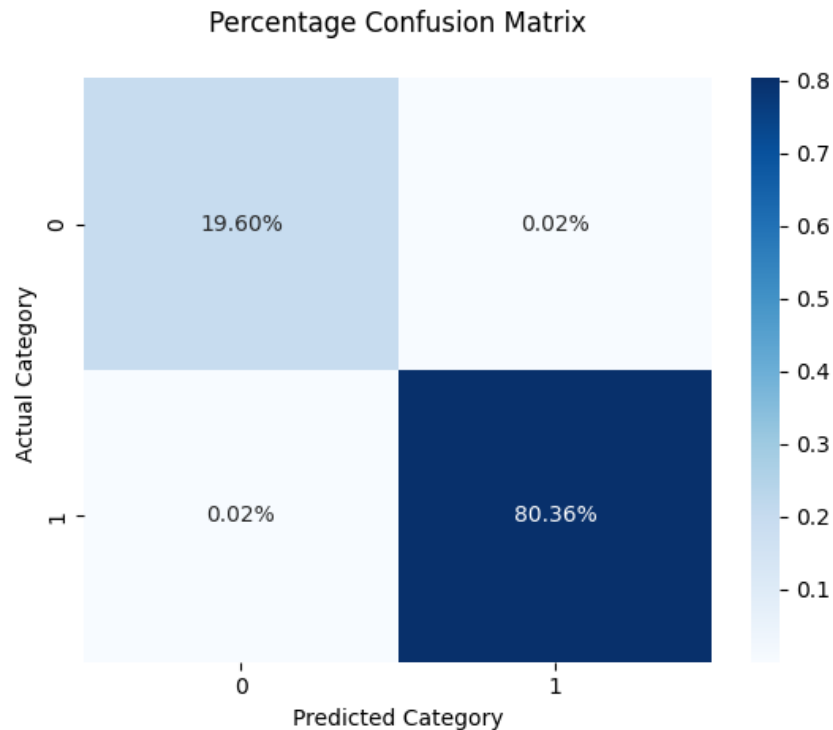


Рис. 3.1 Матриця Невідповідностей

Кожна клітинка в матриці представляє фактор оцінки. Ці фактори є показниками усіх вказаних формулах(3.1-3.5). Розберемо ці фактори один за іншим:

True Positive (TP) означає, скільки зразків позитивного класу правильно передбачила ваша модель.

True Negative (TN) означає, скільки зразків негативного класу правильно передбачила ваша модель.

False Positive (FP) означає, скільки негативних зразків класу ваша модель передбачила неправильно. Цей фактор представляє помилку I типу в статистичній номенклатурі. Позиціонування цієї помилки в матриці залежить від вибору нульової гіпотези.

False Negative (FN) означає, скільки позитивних зразків класу ваша модель передбачила неправильно. Цей фактор представляє помилку II типу в статистичній

номенклатурі. Позиціонування цієї помилки в матриці також залежить від вибору нульової гіпотези[26].

Вище вказана матриця навченої моделі (рис.3.1) вказує на 80.36% відсотків успішно спрогнозованих даних щодо шкідливого трафіку. Відповідно, 19.6% трафіку було спрогнозовано як корисний.

Точність – це відношення справжніх позитивних результатів до загальної прогнозованої кількості позитивних результатів, формула(3.2):

$$P = \frac{TP}{TP+FP} = \frac{\text{Cancer patients correctly identified}}{\text{Cancer patients correctly identified+incorrectly labelled cancer patients as non-cancerous}} \quad (3.2)$$

Показник точності зосереджується на помилках типу I (FP). Помилка типу I виникає, коли ми відхиляємо справжню нульову гіпотезу (H^0). Отже, у цьому випадку помилка типу I полягає в тому, що вірні відповіді позначаються як хибні.

Оцінка точності близький до 1 означатиме, що ваша модель не пропустила жодного справжнього позитивного результату та здатна добре класифікувати правильне та неправильне маркування даних.

Низька оцінка точності ($<0,5$) означає, що ваш класифікатор має велику кількість помилкових спрацьовувань, що може бути наслідком незбалансованого класу або неналаштованих гіперпараметрів моделі. У проблемі незбалансованого класу слід заздалегідь підготувати дані з надлишковою/недостатньою дискретизацією або фокусними втратами, щоб налаштувати відношення FP/FN.

Повнота – це, по суті, відношення істинно позитивних до всіх позитивних відповідей у визначенні, формула (3.3).

$$R = \frac{TP}{TP+FN} = \frac{\text{Cancer patients correctly identified}}{\text{Cancer patients correctly identified+incorrectly labelled non-cancer patients as cancerous}} \quad (3.3)$$

Метрика повноти фокусується на помилках типу II (FN). Помилка типу II виникає, коли ми приймаємо хибну нульову гіпотезу (H^0). Отже, у цьому випадку помилка II типу полягає в неправильному позначенні хибних даних як позитивних[27].

Результат близький до 1 означатиме, що ваша модель не пропустила жодного справжнього позитивного результату та здатна добре класифікувати правильне та неправильне позначення даних.

Низький показник результату ($<0,5$) означає, що ваш класифікатор має велику кількість помилкових негативів, що може бути наслідком незбалансованого класу або неналаштованих гіперпараметрів моделі. У проблемі незбалансованого класу слід заздалегідь підготувати дані з надлишковою/недостатньою дискретизацією або фокусними втратами, щоб нормалізувати FP/FN.

Показник F1 використовує комбінацію акуратності та повноти. Насправді оцінка F1 є середнім гармонійним з двох. Формула (3.4) є наступною:

$$F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} \quad (3.4)$$

Висока оцінка F1 вказує на високу акуратність та повноту. Це забезпечує гарний баланс і дає хороші результати в незбалансованих проблемах класифікації.

Низький показник F1 говорить про продуктивність на пороговому рівні. Низька повнота означає, що ми не намагалися добре виконати більшу частину всього тестового набору.

Низька акуратність означає, що серед випадків, які ми визначили як позитивні, ми не врахували багато з них правильно.

Але низький F1 не вказує точно, які саме випадки були невірними та в чому полягає проблема (низька акуратність чи низька повнота?), і чи модель страждає від помилки типу I чи типу II.

Використання FPR (хибнопозитивних показників) разом із F1 допоможе зменшити помилки типу I, і надасть уявлення про помилку, яка стоїть за низьким результатом F1.

AUROC (Area under Receiver operating characteristics curve), або, більш відома як крива AUC-ROC. Дана графічна метрика використовує істинні позитивні показники (TPR) і помилкові позитивні показники (FPR), формули (3.5).

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \qquad \text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \quad (3.5)$$

TPR відповідає частці позитивних точок даних, які правильно вважаються позитивними, відносно всіх позитивних точок даних. Іншими словами, чим вищий TPR, тим менше позитивних точок даних ми пропустимо.

FPR відповідає частці негативних точок даних, які помилково вважаються позитивними, відносно всіх негативних точок даних. Іншими словами, чим вище FPR, тим більше негативних точок даних ми неправильно класифікуємо[27].

Щоб об'єднати FPR і TPR в одну метрику, ми спочатку обчислюємо дві попередні метрики з різними пороговими значеннями для логістичної регресії, а потім наносимо їх на єдиний графік. Отримана крива називається кривою ROC, а показник, який ми розглядаємо, є площею під цією кривою, яку ми називаємо AUC/ROC[28].

Високий ROC означає, що ймовірність навмання вибраного позитивного прикладу справді позитивна. Високий ROC також означає, що ваш алгоритм добре справляється з ранжуванням тестових даних, з більшістю негативних випадків на одному кінці шкали та позитивних випадків на іншому.

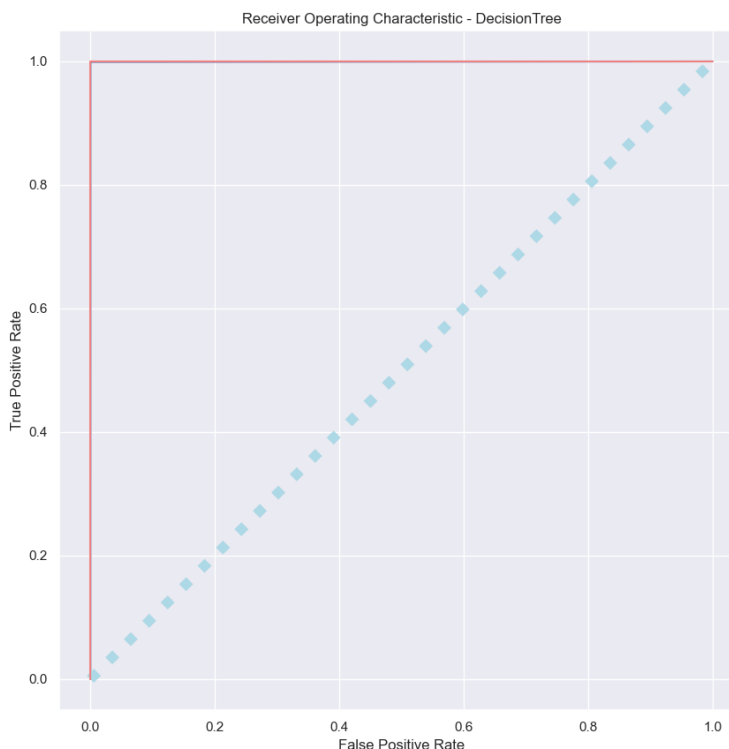


Рис. 3.2 ROC/AUC Крива

Оскільки модель має високу точність прогнозів, даних графік(рис.3.2) максимально наближений до 1 і мінімальна похибка не є помітною.

Ознайомившись з основними метриками для оцінки нашої моделі, можемо дедалініше ознайомитись з її показниками(рис.3.3):

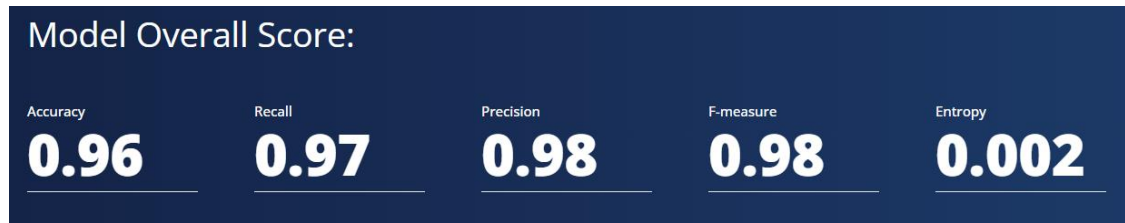


Рис. 3.3 Метрики навченої моделі

Акуратність – 0.96

Повнота – 0.97

Точність – 0.98

Показник F1 – 0.98

Спираючись на дані метрики, можна зробити висновок щодо високої якості показників прогнозування створеної моделі машинного навчання.

3.2 Тестування веб-додатку

Як і в будь-якому проєкті, тестування веб-додатків спрямоване на перевірку можливих дефектів перед їх публічним випуском.

Веб-тестування передбачає перевірку дизайну веб-додатку, його сумісності та можливостей, щоб забезпечити якісну взаємодію з користувачем із високим рівнем безпеки та продуктивності.

Оскільки веб-додатки запускаються у веб-браузерах (це означає, що користувачі можуть отримати до них доступ з будь-якого пристрою, підключеного до Інтернету), розробникам програмного забезпечення більше не потрібно створювати окрему програму для кожної платформи. Крім того, простіше оновлювати веб-додатки: щойно розробники розгортають нові функції або оновлену версію додатка у виробництві, оновлення миттєво стає доступним для всіх користувачів.

Саме тому етап тестування веб-додатку є ключовим перед кінцевим розгортанням продукту.

Для початку, протестуємо завантаження файлу у веб-додаток. Для цього натискаємо кнопку «Upload file»(рис.3.4), обираємо локальний csv-файл(рис.3.5) та відкриваємо його.

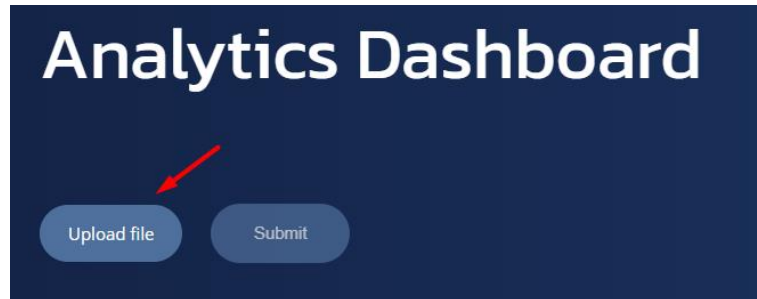


Рис. 3.4 Завантаження файлу

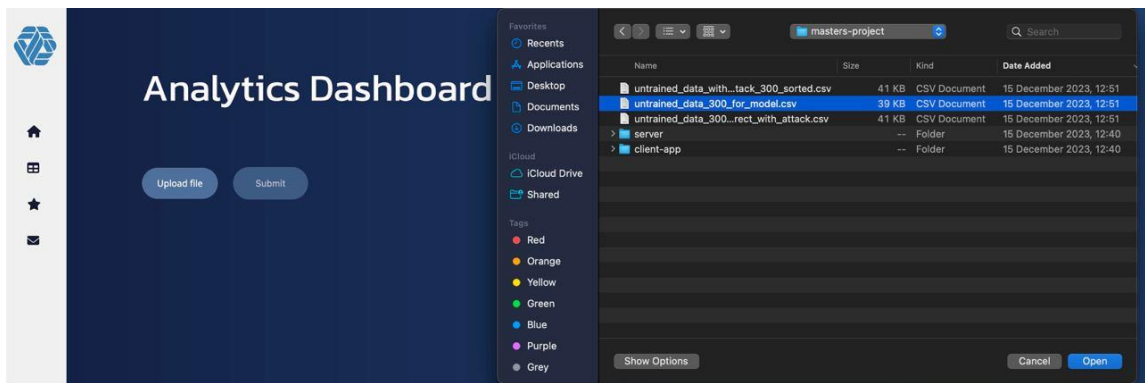


Рис. 3.5 Завантаження файлу

Після завантаження файлу дані виводяться у вигляді таблиці та зберігаються у localStorage(рис.3.6).

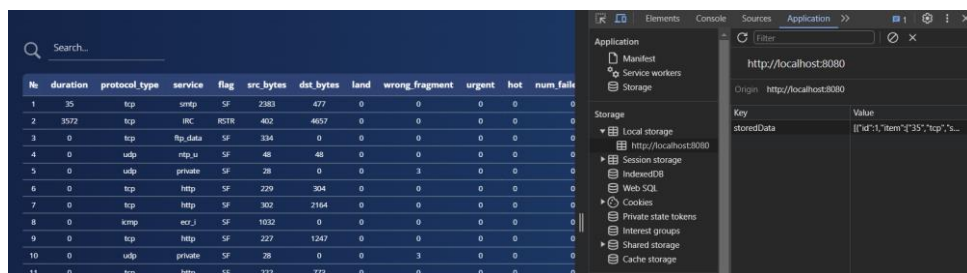


Рис. 3.6 Відображення та збереження даних з файлу

Кнопка «Submit» стає активною, тому ми можемо відправити наші дані на сервер для прогнозування шкідливого трафіку(рис.3.7).

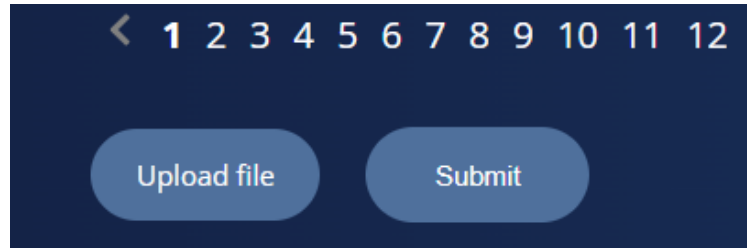


Рис. 3.7 Відправка даних

Після відправки даних, на екран виводиться сповіщення(рис.3.8) про успішне прогнозування даних.

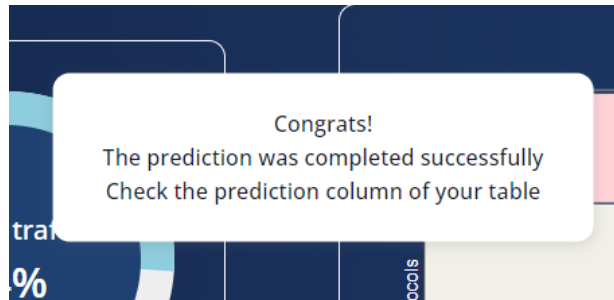


Рис. 3.8 Сповіщення про успішне прогнозування даних

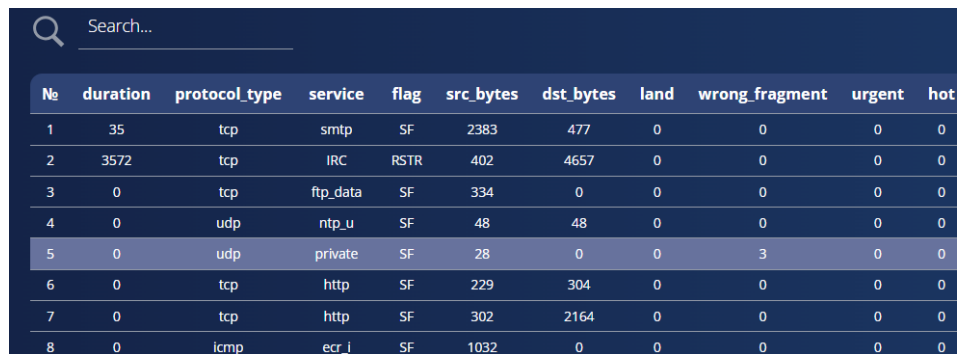
Також колонка «prediction» заповнюється відповідними даними(рис.3.9)

error_rate	prediction
	0
	1
	0
	1
	1
	0
	1
	1
	1

Рис. 3.9 Виведення прогнозу

Після успішного прогнозу, у нас є змога переглянути виведені графіки та діаграми(рис.3.10).

Також для зручності перегляду, користувач може виділити певний потрібний рядок просто натиснувши на нього(рис.3.13).



№	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot
1	35	tcp	smtp	SF	2383	477	0	0	0	0
2	3572	tcp	IRC	RSTR	402	4657	0	0	0	0
3	0	tcp	ftp_data	SF	334	0	0	0	0	0
4	0	udp	ntp_u	SF	48	48	0	0	0	0
5	0	udp	private	SF	28	0	0	3	0	0
6	0	tcp	http	SF	229	304	0	0	0	0
7	0	tcp	http	SF	302	2164	0	0	0	0
8	0	icmp	ecr_l	SF	1032	0	0	0	0	0

Рис. 3.13 Виділення рядка

Тест пройдено успішно.

Висновки до 3-го розділу:

Було проаналізовано метрики якості для моделей маштнного навчання за методом класифікації. На основі розглянутих показників якості було проведено оцінку створеної моделі. За результатом оцінювання моделі, було отримано задовільний результат з високими показниками.

Було проведено тестування розробленого веб-додатку, а саме швидкість завантаження та відображення даних, збереження даних у localStorage, пошук та фільтрація таблиці, виведення графіків та діаграм. Тест був пройдений успішно.

За результатами тестування моделі МН та розробленого веб-додатку, можна зробити висновок, про високу якість роботи та готовність використання нашого застосунку кінцевим користувачем.

ВИСНОВКИ

У кваліфікаційній магістерській роботі було досліджено актуальні методи та інструменти аналізу статичного мережевого трафіку на основі штучного інтелекту. На основі досліджень було розроблено веб-додаток з дослідження та детектування шкідливого трафіку на основі МН.

Обґрунтування актуальності теми дало змогу впевнитись в необхідності дослідження та розробки інструменту аналізу даних на основі штучного інтелекту, а саме машинного навчання.

Було розглянуто технології машинного навчання. За результатами дослідження, обрано метод контрольованого машинного навчання, зокрема класифікації. Також було проаналізовано алгоритми МН, та обрано алгоритм Дерева прийняття рішень для вирішення поставленої задачі магістерської роботи. Основними перевагами даного алгоритму є легке розуміння, інтерпретуваність та гнучкість.

Було проведено аналіз базових та сучасних технологій веб-розробки, а також досліджено їх властивості відповідно до поставленого завдання. Для клієнтської частини було обрано фреймворк Vue.js як основний для побудови веб-додатку, оскільки він відповідає всім зазначеним вимогам, а саме, простота, гнучкість та легка інтеграція з іншими технологіями. Для серверної частини було обрано фреймворк Flask за його легке середовище веб-розробки, яке робить його ідеальним для швидкого та ефективного створення додатків невеликого та середнього розміру.

Дослідивши переваги та недоліки різних мов програмування для розробки моделі машинного навчання, було обрано мову Python. Перевагою стала величезна бібліотечна екосистема, яка дозволяє практикам машинного навчання отримувати доступ, аналізувати, перетворювати та обробляти дані з легкістю. Також сама мова гнучка та зручна для використання.

Проаналізувавши та обравши необхідні технології для розробки, було розпочато створення моделі МН. Розробка включає в себе наступні етапи: імпорт даних та їх підготовка (позначення категоріальних стовпців, кодування та масштабування даних), навчання та оцінка моделі машинного навчання.

Наступним кроком було оцінка якості моделі, за допомогою наступних метрик: акуратність, точність, повнота, показник F1, матриця неточностей та AUC-ROC крива. Дані метрики показали хороший результат, що дало змогу використувати навчену модель для кінцевого продукту.

Після збереження моделі, було розпочато розробку веб-застосунку. Першим кроком було підготовка вимог до продукту. Далі було створено архітектуру проекту, розроблено дизайн, програмування, яке включало написання клієнтської та серверної частини, та налаштування зв'язку між ними.

Останнім та важливим етапом було тестування кінцевого продукту. Даний крок включав в себе завантаження файлу з даними, відправка їх до навченої моделі та отримання результатів прогнозу шкідливого трафіку. Також було протестовано такі функції як збереження даних, побудова діаграм, пошук та фільтрація. Тест пройдено успішно.

Отже, веб-додаток відповідає всім початковим вимогам та готовий до використання кінцевим користувачем. Дана розробка допоможе аналітикам якісно виявляти шкідливий трафік на великих наборах даних та покращувати захист мережі. Візуальна дошка з діаграмами та графіками дасть змогу легше аналізувати трафік та знаходити закономірності.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Tyler York. The Cybersecurity Analytics Ultimate Guide, 2023. URL: <https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML>
2. Al Perlman. The Growing Role of Machine Learning in Cybersecurity, 2022. URL: <https://www.securityroundtable.org/the-growing-role-of-machine-learning-in-cybersecurity/>
3. Alyssa Schroer. AI Cybersecurity: 27 Companies to Know, 2022. URL: <https://builtin.com/artificial-intelligence/artificial-intelligence-cybersecurity>
4. Флах. П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных/ пер. с англ. А. А. Слинкина. - М.: ДМК Пресс, 2015. - 400 с.
5. Мерков А. Б. Распознавание образов. Введение в методы статистического обучения. 2011. 256 с.
6. Sunil Ray. Top 10 Machine Learning Algorithms to Use in 2024, 2023. URL: <https://levity.ai/blog/difference-machine-learning-deep-learning>
7. Katrina Wakefield. A guide to the types of machine learning algorithms and their applications. URL: https://www.sas.com/en_gb/insights/articles/analytics/machine-learning-algorithms.html
8. Pethuru Raj, Preetha Evangeline. Advances in Computers. Machine learning and deep learning algorithms on the Industrial Internet of Things (IIoT), 2020, p. 321-338.
9. Samadrita Ghosh. How to Compare Machine Learning Models and Algorithms, 2023. URL: <https://neptune.ai/blog/how-to-compare-machine-learning-models-and-algorithms>
10. Shahadat Uddin, Arif Khan, Md Ekramul Hossain & Mohammad Ali Moni. Comparing different supervised machine learning algorithms for disease prediction, 2019. URL: <https://bmcmmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-019-1004-8>
11. Coursera Staff. 10 Machine Learning Algorithms to Know in 2024, 2023. URL: <https://www.coursera.org/articles/machine-learning-algorithms>

12. What Is Web Development? 2023. URL: <https://brainstation.io/career-guides/what-is-web-development>
13. What Is Web Development and How Does It Work? 2023. URL: <https://www.webfx.com/web-development/glossary/what-is-web-development/>
14. Allie Decker. The Beginner's Guide to Website Development, 2023. URL: <https://blog.hubspot.com/website/website-development>
15. Cem Eygi. CSS for Beginners: What is CSS and How to Use it in Web Development? 2019. URL: <https://medium.com/swlh/css-for-beginners-what-is-css-and-how-to-use-it-in-web-development-5985afe53096>
16. Mohit Joshi. A detailed guide on JavaScript Web Development, 2023. URL: <https://www.browserstack.com/guide/javascript-web-development>
17. Robert Johns. Want To Level Up In Web Development? Use Web Development Frameworks! 2023. URL: <https://hackr.io/blog/web-development-frameworks>
18. Artem Arkhipov. 10 Best Frameworks for Web Development in 2023, 2023. URL: <https://www.techmagic.co/blog/best-frameworks-for-web-development/>
19. Roger James. Top Web Development Frameworks Comparison: All You Need To Know, 2022. URL: <https://www.pixelcrayons.com/blog/software-development/best-web-development-frameworks-comparison/>
20. Trung Tran. Top 7 Server-side Scripting Languages, 2021. URL: <https://www.orientsoftware.com/blog/server-side-scripting-languages/>
21. Katie Horne. Languages and Frameworks for Programming in 2024, 2023. URL: <https://digital.com/programming-language-frameworks/>
22. Nazar Kvartalnyi. Why Use Python for Machine Learning?, 2022. URL: <https://inoxoft.com/blog/why-use-python-for-machine-learning/>
23. Sakshi Gupta. What Is the Best Language for Machine Learning? 2021. URL: <https://www.springboard.com/blog/data-science/best-language-for-machine-learning/>
24. Nataliia Huivan. Best Language for Machine Learning 2023, 2023. URL: <https://atlasiko.com/blog/ai/best-language-for-machine-learning/>

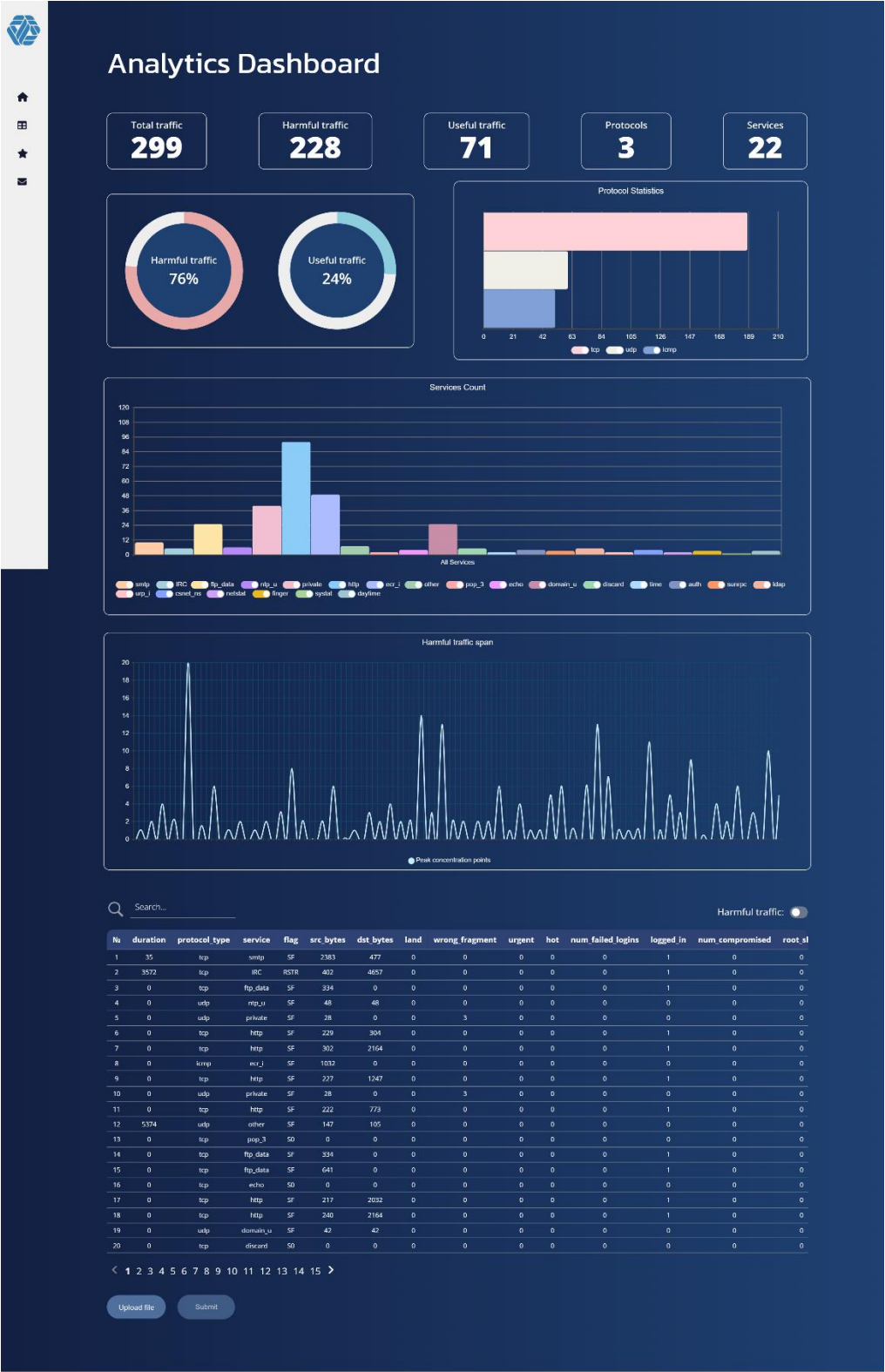
25. Aayush Bajaj. Performance Metrics in Machine Learning, 2023. URL: <https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide>
26. Tavish Srivastava. 12 Important Model Evaluation Metrics for Machine Learning Everyone Should Know, 2023. URL: <https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/>
27. Data Science. Machine Learning Metrics: How to Measure the Performance of a Machine Learning Model, 2022. URL: <https://www.altexsoft.com/blog/machine-learning-metrics/>
28. Sarang Narkhede. Understanding AUC - ROC Curve, 2018. URL: <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>

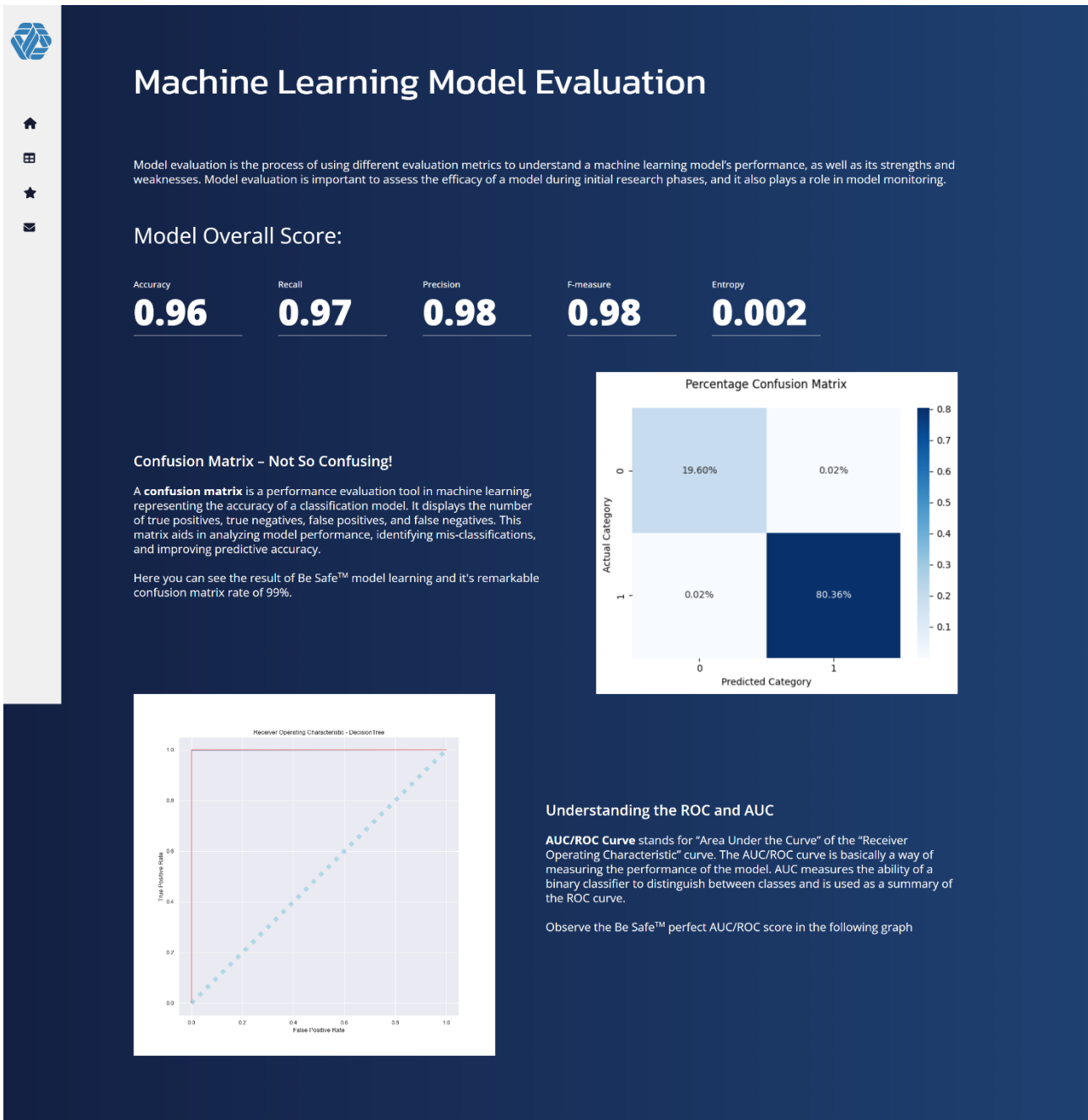
ДОДАТКИ

Сторінка Home

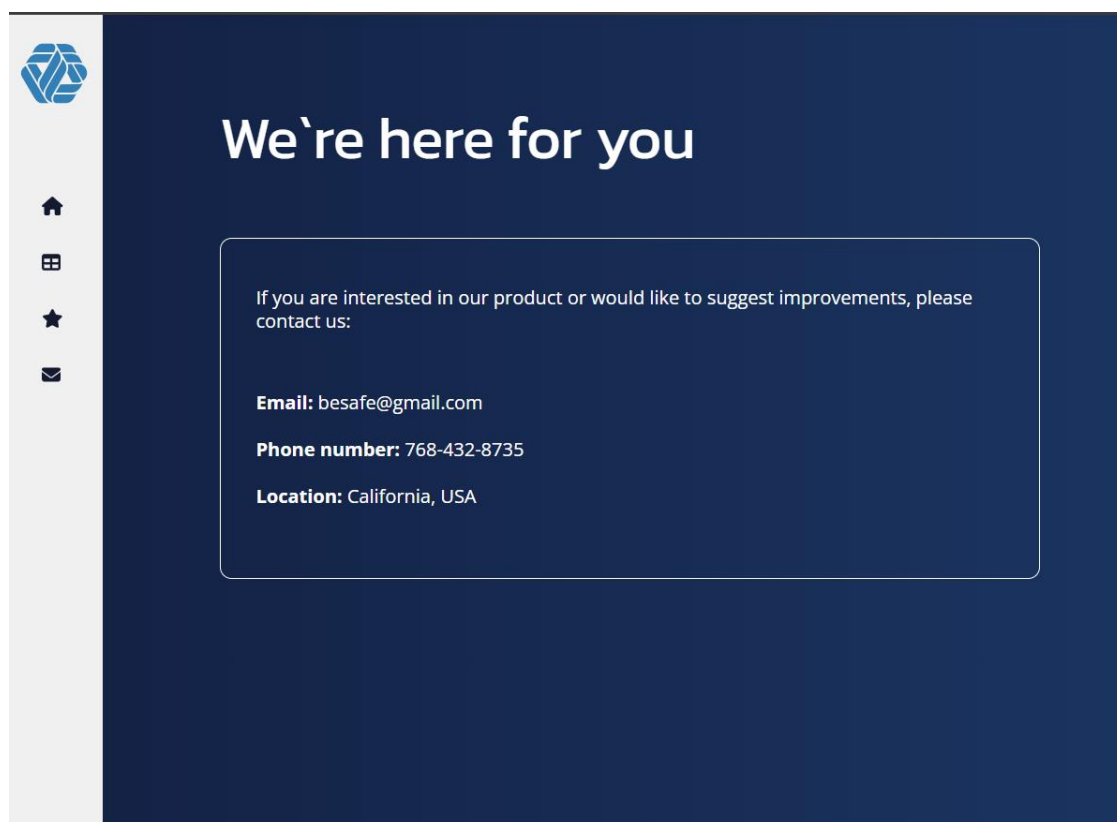


Сторінка Analytics Dashboard





Сторінка Contact us



Лістинг коду app.py

```

from flask import Flask, request, jsonify, json, make_response
from flask_cors import CORS, cross_origin
import pickle
import numpy as np
import pandas as pd
import os
import csv
import codecs, json
from json import JSONEncoder
from sklearn.preprocessing import LabelEncoder
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler

app = Flask(__name__)
CORS(app, resources={r"/prediction": {"origins": "http://localhost:8080"}}, supports_credentials = True)
APP_ROOT = os.path.dirname(os.path.abspath(__file__))
MODEL_PATH = os.path.join(APP_ROOT, "./model/attacks_classifier_model.pkl")
model = pickle.load(open(MODEL_PATH, 'rb'))

class NpEncoder(json.JSONEncoder):
    def default(self, obj):
        if isinstance(obj, np.integer):
            return int(obj)
        if isinstance(obj, np.floating):
            return float(obj)
        if isinstance(obj, np.ndarray):
            return obj.tolist()
        return super(NpEncoder, self).default(obj)

def transformData(data):
    labelencoder_x_1 = LabelEncoder()
    labelencoder_x_2 = LabelEncoder()
    labelencoder_x_3 = LabelEncoder()
    data[:, 1] = labelencoder_x_1.fit_transform(data[:, 1])
    data[:, 2] = labelencoder_x_2.fit_transform(data[:, 2])
    data[:, 3] = labelencoder_x_3.fit_transform(data[:, 3])

    sc_x = StandardScaler()
    data_prepared = sc_x.fit_transform(data)
    return data_prepared

@app.route('/prediction', methods = ['POST', 'OPTIONS'])
def upload_file():
    if request.method == "OPTIONS":
        return _build_cors_preflight_response()
    if request.method == 'POST':

```

```

        try:
            file = request.files['csvFile']
            so = csv.reader(codecs.iterdecode(file, 'utf-8'), delimiter=',',
quotechar='"')
            so_data = []
            for row in so:
                so_data.append(row)
            so_data = pd.DataFrame(so_data)
            so_data = so_data.iloc[:,].values
            transformed_data = transformData(so_data)
            predicted_data = model.predict(transformed_data)
            arrData = []
            i = 1
            for item in predicted_data:
                arrData.append({'item': i, 'predicted': item})
                i += 1
            return json.dumps(arrData, cls=NpEncoder)

        except Exception as e:
            return jsonify({'error': str(e)}), 400

def _build_cors_preflight_response():
    response = make_response()
    response.headers.add("Access-Control-Allow-Origin", "*")
    response.headers.add('Access-Control-Allow-Headers', "*")
    response.headers.add('Access-Control-Allow-Methods', "*")
    return response

if __name__ == '__main__':
    app.run(debug=True)

```

Лістинг коду api.js

```

const HOST = 'http://localhost:5000';

async function getPrediction(data) {
    let formData = new FormData();
    formData.append("csvFile", data);
    console.log(data);
    const response = await fetch(`${HOST}/prediction`, {
        method: "POST",
        body: formData,
    });

    let result;

    try {
        result = await response.json();
    } catch(err) {
        console.log(err)
    }
}

```

```

    }

    return result;
}

export {
  getPrediction
}

```

Лістинг коду router.js

```

import Vue from "vue";
import VueRouter from 'vue-router';
import PagesRoutes from '@routes/pages';

Vue.use(VueRouter);

const pageNotFoundRoutes = [
  {
    path: '/404',
    name: 'PageNotFound',
    component: () => import('@pages/PageNotFound')
  },
];

const router = new VueRouter({
  routes: [
    ...PagesRoutes,
    ...pageNotFoundRoutes,
    {
      path: "/",
      redirect: "/patients/home",
    },
    { path: '/*', redirect: "/404", }
  ],
  mode: 'history',
  linkActiveClass: 'active',
  scrollBehavior(to, from, savedPosition) {
    if (savedPosition) {
      return savedPosition
    } else {
      return {x: 0, y: 0}
    }
  }
});

export default router;

```

Лістинг коду main.js

```

import Vue from 'vue'

```

```

import App from './App.vue'
import VueRouter from 'vue-router';
import VueI18n from 'vue-i18n';
import VueCookies from 'vue-cookies';
import SmartTable from 'vuejs-smart-table';
import VueGraph from 'vue-graph';
import Donut from 'vue-css-donut-chart';
import 'vue-css-donut-chart/dist/vcdonut.css';
import { library } from '@fortawesome/fontawesome-svg-core';
import { FontAwesomeIcon } from '@fortawesome/vue-fontawesome';
import { faUserSecret, faHouse, faTable, faEnvelope, faEarthAmerica, faStar } from
 '@fortawesome/free-solid-svg-icons';

import localizationEN from './localization/en.js';

// Style
import './assets/scss/reset.scss'
import './assets/scss/main.scss'

Vue.config.productionTip = false

import router from './routes/router';

library.add(faUserSecret, faHouse, faTable, faEnvelope, faEarthAmerica, faStar)

Vue.use(VueRouter);
Vue.use(VueI18n);
Vue.use(VueCookies);
Vue.use(SmartTable);
Vue.use(VueGraph);
Vue.use(Donut);
Vue.component('FontAwesomeIcon', FontAwesomeIcon);

const i18n = new VueI18n({
  locale: 'en',
  messages: {
    "en" : localizationEN,
  },
});

new Vue({
  i18n,
  router,
  render: h => h(App),
}).$mount('#app')
```