МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра «Системний аналіз та обчислювальна математика»
(повне найменування кафедри)

Пояснювальна записка

до дипломної роботи
магістра
(ступінь вищої освіти)

на тему Методи машинного навчання в аналізі дій відвідувачів сайту
(назва теми)

Виконав: студент 2 курсу, групи 813м

Спеціальності 124 Системний аналіз
(код і найменування спеціальності)

Освітня програма (спеціалізація)
«Інтелектуальні технології та прийняття рішень в складних системах»

ПЕРЕТЯТИЙ В.О.
(ПРІЗВИЩЕ та ініціали)

Керівник ШИРОКОРАД Д.В.
(ПРІЗВИЩЕ та ініціали)

Рецензент ДУМІН О.М.
(ПРІЗВИЩЕ та ініціали)

2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут, факультет Факультет комп'ютерних наук і технологій
Кафедра Кафедра «Системний аналіз та обчислювальна математика»
Ступінь вищої освіти магістра
Спеціальність 124 Системний аналіз
(код і найменування)
Освітня програма (спеціалізація) «Інтелектуальні технології та прийняття рішень в складних системах»
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
Еліна ТЕРЕЩЕНКО
«16» грудня 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТА
Перетятому Віталію Олександровичу
(ПРІЗВИЩЕ, ім'я, по батькові)
1. Тема роботи Методи машинного навчання в аналізі дій відвідувачів сайту

керівник роботи Дмитро Вікторович Широкорад
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від «20» листопада 2024 року №480
2. Строк подання студентом роботи
3. Вихідні дані до роботи Дані відвідувачів вебсайту зібрані на офіційному сайті ПрАТ «Аптеки Запоріжжя»

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Задачами дослідження є визначення характеристик, які можна отримати при спостережені за рухом мишки та використання клавіатури відвідувачів вебсайту; зібрати певну кількість даних, щоб можна було створити модель штучного інтелекту для визначення віку та статі людини; запровадити дану модель на сайт та протестувати її точність.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) презентація на 15 слайдів

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Прийняв виконане завдання
Розділ 1, Розділ 2	Широкорад Д.В., доцент каф. САОМ	21.10.2024	20.12.2024
Нормоконтроль	Широкорад Д.В., доцент каф. САОМ	20.12.2024	20.12.2024

7. Дата видачі завдання «21» жовтня 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Визначення можливих даних для збору.	1 тиждень	Розділ 2
4	Збір даних.	2-4 тиждень	Розділ 2
5	Створення моделі для визначення віку та статі.	5 тиждень	Розділ 2
6	Впровадження моделі на сайт.	5 тиждень	Розділ 2
7	Тестування.	5-8 тиждень	Розділ 2
8	Оформлення пояснювальної записки та відповідної документації.	9-10 тиждень	Додатки
9	Нормоконтроль та рецензування.	11 тиждень	
10	Захист дипломної роботи.	12 тиждень	

Студент _____ Перетятий В.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Широкорад Д.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дипломна робота: 68 с., 6 ілюстрації, 1 додаток, 15 джерел.

Об'єкт дослідження – патерни поведінки відвідувачів сайту.

Предмет дослідження – практична реалізація системи для розпізнавання віку та статі відвідувача.

Мета роботи – побудувати модель, яка за допомогою машинного навчання здатна розрізняти відвідувачів за статтю та віком користуючись даними про рух миші та частоту використання клавіатури.

Методи дослідження – методи Deep Learning.

Актуальність проблеми – досягти більш високого рівня персоналізації при мінімальних затратах на збір та обробку інформації за допомогою штучного інтелекту та аналізу поведінки користувача.

Задачі дослідження: аналіз існуючих систем для веб-аналітики та дослідження поведінки відвідувачів сайту; визначення можливих даних для збору; збір даних та створення бази для навчання; створення моделі для визначення віку та статі людини; впровадження та тестування створеної моделі.

Результати роботи та їх новизна – розроблено систему, яка з легкістю буде розпізнавати вік та стать людини, не маючи жодних даних про неї, окрім того як швидко вона рухає мишкою та користується клавіатурою.

Основні висновки – результати роботи є практично корисними, розроблена система дозволяє швидко і без зайвих затрат аналізувати поведінку відвідувачів сайтів та робити висновки щодо їх віку та статі.

Систему можна використовувати, як і на невеликих сайтах так і на крупних онлайн-магазинах та ритейлерах, що робить її універсальною та корисною.

ЦИФРОВА МОДЕЛЬ, ВЕБСАЙТ, JAVASCRIPT, PYTHON, ШТУЧНИЙ ІНТЕЛЕКТ, ВІК ТА СТАТЬ ВІДВІДУВАЧІВ, ПАТЕРНИ ПОВЕДІНКИ.

ЗМІСТ

ВСТУП

Визначення віку та статі користувача має значний вплив на розробку інтерфейсів користувачів, де важливими аспектами є зручність, ефективність та адаптивність. Персоналізація інтерфейсу, орієнтуючись на ці характеристики, дозволяє створювати більш інтуїтивно зрозумілі платформи, які враховують індивідуальні потреби та вподобання. Це особливо важливо в умовах стрімкого розвитку мобільних додатків, вебсайтів та інтерактивних систем, що стають невід'ємною частиною щоденного життя користувачів.

Враховуючи зростаючу кількість даних, які генеруються користувачами під час взаємодії з цифровими платформами, проблема збору та аналізу таких даних набуває особливого значення. Одним із шляхів для визначення віку та статі є використання біометричних даних або аналізу поведінки, який дозволяє отримати значно точніші результати порівняно з традиційними методами. Це дає змогу досягти більш високого рівня персоналізації при мінімальних затратах на збір та обробку інформації.

Однак, з іншого боку, така практика ставить перед нами питання етики та конфіденційності. Використання алгоритмів для визначення віку та статі користувача може бути розцінене як вторгнення в особисте життя. Це питання особливо актуальне для розробників, оскільки потрібно не тільки забезпечити точність та ефективність визначення, а й гарантувати захист персональних даних, відповідно до вимог законодавства, таких як GDPR або інші регіональні стандарти конфіденційності.

Ще однією важливою складовою є соціальні наслідки використання алгоритмів для визначення віку та статі. Вони можуть впливати на формування стереотипів або навіть призвести до дискримінації. Наприклад, в онлайн-рекламі це може призвести до того, що певні групи користувачів будуть відсічені від відповідних маркетингових пропозицій, що обмежить їхні можливості для доступу до різноманітних товарів та послуг. Тому важливою умовою є створення

алгоритмів, які зможуть забезпечити рівність і прозорість у взаємодії з користувачами.

Серед новітніх підходів для визначення віку та статі користувачів на базі рухів миші та введення з клавіатури, важливим є застосування нейронних мереж та машинного навчання. Це дозволяє створювати системи, які вчаться на основі великої кількості даних і можуть з часом покращувати точність своїх прогнозів. Однак, такі підходи вимагають значних ресурсів для збору, обробки та аналізу даних, а також наявності потужних обчислювальних можливостей.

Аналіз рухів миші та натискання клавіш є важливим інструментом не лише для визначення віку та статі, але й для вивчення звичок та поведінкових патернів користувачів. Це дає змогу розробникам створювати не тільки персоналізовані інтерфейси, а й адаптивні системи, які змінюються залежно від зміни поведінки користувача. Такі системи можуть автоматично коригувати свої налаштування під час використання, що сприяє підвищенню ефективності взаємодії.

У сфері електронної комерції персоналізація на основі віку та статі є важливим фактором для створення таргетованих рекламних кампаній. Використання таких даних дозволяє покращити рекомендаційні системи, що, у свою чергу, підвищує рівень продажів та задоволеність користувачів. Однак важливо дотримуватись балансування між персоналізацією та запобіганням надмірному втручанню в особисте життя клієнтів, щоб не втратити їхню довіру.

Враховуючи значення віку та статі для маркетингових стратегій, важливо не лише точно визначати ці параметри, а й використовувати їх з обережністю. Надмірна фокусування на цих характеристиках може призвести до занадто вузької орієнтації на цільову аудиторію, що обмежить потенційні можливості для розвитку брендів та компаній. У цьому контексті важливо розглядати різні підходи до персоналізації, враховуючи не тільки вік і стать, але й інші аспекти, такі як інтереси, поведінкові фактори та культурні відмінності.

У сфері охорони здоров'я застосування алгоритмів для визначення віку та статі може мати значення для створення індивідуальних медичних рекомендацій.

Такі технології дозволяють створювати більш точні прогнози щодо стану здоров'я пацієнтів на основі їхніх специфічних характеристик. Водночас, у цій сфері особливо важливо дотримуватися етичних норм та гарантувати, що персональні дані будуть використовуватись виключно в інтересах пацієнта, зокрема, для підвищення ефективності лікування.

У майбутньому, з розвитком технологій штучного інтелекту та машинного навчання, можна очікувати значного прогресу в автоматизації процесів визначення віку та статі користувачів. Це дозволить ще точніше адаптувати інтерфейси та пропозиції під потреби кожного користувача, а також зробить цифрові платформи більш чутливими до змін у поведінці та перевагах користувачів. Однак, важливо продовжувати вивчення соціальних та етичних аспектів таких технологій, щоб забезпечити їх позитивний вплив на суспільство в цілому.

1 ТЕОРЕТИЧНА ЧАСТИНА

1.1 Проблема збору, обробки, аналізу та захисту інформації

Під час збору інформації про поведінку користувача на сайті, зокрема за допомогою аналізу рухів миші та натискання клавіш, можуть виникати різноманітні проблеми з точністю отриманих даних.

Наприклад, користувач може мати нетипову поведінку через особливості фізіології (наприклад, проблеми з рухомими функціями або використання спеціальних пристроїв для введення даних), що ускладнює коректну інтерпретацію таких дій. Крім того, технічні несправності або затримки на сайті можуть призвести до того, що система не зможе точно відслідковувати всі рухи миші чи натискання клавіатури.

Також поведінка користувача може змінюватися в залежності від різних факторів, які не завжди можна передбачити або врахувати під час збору даних. Наприклад, користувач може використовувати мобільні пристрої замість ПК, що змінює спосіб взаємодії з інтерфейсом. Або ж поведінка може бути змінена під впливом зовнішніх обставин, таких як стрес чи спонтанні рішення, які важко відобразити в стандартних алгоритмах аналізу поведінки.

Ще одна серйозна проблема це ризик збору надмірної кількості даних, що може включати не тільки релевантну інформацію про поведінку, а й зайву, яка може створити додаткові труднощі при її обробці. Наприклад, збирання дрібних рухів миші чи кожного натискання клавіші без достатнього фільтрування може призвести до накопичення великої кількості даних, що не мають суттєвого значення для аналізу і тільки ускладнюють подальшу роботу.

Проблемою є також те, що з часом поведінка користувача може змінюватися через адаптацію до певних інтерфейсів або рекомендацій, що надаються сайтами. Це може призвести до зміщення даних, коли користувач, через механізми персоналізації, починає діяти за шаблоном, орієнтуючись на

алгоритми, що їх рекомендує система. В такому випадку буде важко відрізнити справжню поведінку користувача від адаптованої під алгоритм.

Окрім етапу збору даних не менш важливим є і етап їх обробки. Збір даних про рухи миші та натискання клавіатури призводить до накопичення величезних обсягів інформації, яку необхідно обробити. Це може спричинити значні труднощі у зберіганні та аналізі таких даних, оскільки вони займають багато пам'яті і потребують потужних обчислювальних ресурсів. До того ж обробка великих масивів даних вимагає високої продуктивності алгоритмів для ефективного сортування, класифікації та виявлення важливих патернів.

Поведінка користувача може бути дуже варіативною, що ускладнює її аналіз. Різні користувачі можуть діяти по-різному в схожих ситуаціях, і передбачити їхні реакції на певні елементи інтерфейсу складно. Це створює проблему у визначенні чітких шаблонів, особливо коли йдеться про складні алгоритми машинного навчання.

Оскільки рухи миші та натискання клавіатури не завжди чітко відображають намір користувача, їх точна інтерпретація може бути важкою. Наприклад, випадкові кліки або затримки при введенні тексту можуть бути інтерпретовані як помилки або неправильні дії, хоча насправді це можуть бути ознаки іншого типу взаємодії. В результаті можуть виникнути помилки у прогнозуванні віку, статі чи інших характеристик користувача [1].

Найважливішою проблемою здебільшого є захист отриманої інформації, оскільки інформація про поведінку користувача на сайті є дуже чутливою і може включати персональні дані. Недостатньо захищені сервери або ненадійні системи зберігання можуть стати мішенню для кіберзлочинців, що призведе до витоку особистої інформації користувачів. Такі ситуації можуть мати серйозні наслідки, зокрема для репутації компанії, а також для безпеки самих користувачів.

Для забезпечення безпеки користувачів часто необхідно анонімізувати зібрані дані, щоб вони не могли бути пов'язані з конкретною особою. Однак це може ускладнити точність та ефективність аналізу, оскільки анонімізація може

приховувати важливу інформацію, необхідну для точного прогнозування поведінки. Знайти баланс між збереженням конфіденційності та забезпеченням ефективного аналізу даних є однією з найбільших викликів у цій сфері.

Регулювання та відповідність нормам. Зібрана інформація про поведінку користувачів має бути захищена відповідно до міжнародних стандартів та законодавства, таких як GDPR. Відсутність відповідних механізмів захисту може призвести до порушення законодавчих вимог, що матиме юридичні та фінансові наслідки. Крім того, важливо постійно оновлювати політики конфіденційності та використовувати новітні технології для захисту даних, щоб уникнути санкцій та штрафів за порушення норм конфіденційності [2].

1.2 Аналіз предметної області

Одна з головних складнощів при визначенні віку та статі невідомого користувача полягає в тому, що поведінка людей є дуже варіативною і не завжди відповідає стереотипним моделям для певної статі чи віку. Наприклад, жінки та чоловіки можуть мати подібний стиль використання клавіатури або миші, особливо якщо вони зосереджені на певному завданні, до того ж, вік не завжди можна точно вирахувати лише через тип взаємодії з інтерфейсом, оскільки як молоді, так і старші користувачі можуть демонструвати схожі патерни поведінки на вебсайтах, особливо якщо мова йде про одні й ті ж технології або додатки.

Хоча сучасні алгоритми машинного навчання можуть навчатися на великих обсягах даних, однією з проблем є те, що вони не завжди здатні правильно узагальнити отримані патерни. Тобто, хоча система може точно вирахувати стать або вік для частини користувачів, вона може допускати значні помилки при розпізнаванні іншої частини, особливо коли мова йде про людей, які активно змінюють свою поведінку в мережі або користуються незвичними

пристроями, наприклад, для забезпечення доступності. Така невизначеність у визначенні статі і віку може знижувати ефективність продукту.

Культурні відмінності та соціальні контексти також можуть впливати на точність визначення віку та статі. Наприклад, в деяких культурах можуть існувати специфічні поведінкові патерни або звички, які можуть бути важко інтерпретовані з точки зору стандартних алгоритмів. Крім того, віртуальна ідентичність користувачів, особливо в онлайн-середовищах, часто може бути сконструйована таким чином, щоб маскувати реальний вік і стать. Наприклад, користувач може навмисно змінювати свої дії на сайті, щоб приховати свою справжню особистість, що також ускладнює точне визначення віку та статі.

Однією з найбільших сфер застосування технології визначення віку та статі є персоналізація досвіду користувача на вебсайтах та в мобільних додатках. Це включає адаптацію контенту, інтерфейсу та реклами відповідно до індивідуальних характеристик користувача. Наприклад, якщо система точно визначає, що користувач — підліток, можна запропонувати йому відповідний контент або знижки, орієнтуючись на його потреби та інтереси. Таке застосування може покращити ефективність реклами і зробити взаємодію з платформою більш зручною.

Технології, які дозволяють визначати стать та вік користувачів, стають важливими інструментами для рекламодавців. Вони дозволяють створювати більш точні таргетовані кампанії, що збільшують ймовірність того, що рекламні матеріали будуть сприйняті цільовою аудиторією. Це дозволяє рекламодавцям економити бюджет на показі оголошень тим, хто не є потенційним споживачем, і досягати більш високого рівня конверсії, знижуючи витрати на неефективну рекламу.

У сфері безпеки та аутентифікації технології визначення віку та статі можуть бути використані для покращення систем перевірки користувачів. Наприклад, в онлайн-іграх або соціальних мережах це дозволяє автоматично обмежувати доступ до певних функцій для неповнолітніх користувачів. Аналогічно, для онлайн-банкінгу або e-commerce платформ такі технології

можуть допомогти в верифікації клієнтів, забезпечуючи захист від шахрайства або несанкціонованого доступу.

У галузі охорони здоров'я та медичних технологій визначення віку та статі може бути корисним для створення індивідуалізованих рекомендацій для пацієнтів. Наприклад, для мобільних додатків, що надають медичні консультації або фітнес-плани, точне розпізнавання віку та статі може допомогти сформувати більш персоналізовані плани, орієнтуючись на фізіологічні особливості кожної людини. Такі системи можуть аналізувати рухи, фізичну активність та інші поведінкові дані для надання відповідних рекомендацій щодо здоров'я.

Одними з основних споживачів таких технологій будуть онлайн-магазини, ритейлери та рекламні агентства, які прагнуть отримати точнішу інформацію про своїх клієнтів для покращення маркетингових стратегій. За допомогою таких продуктів вони зможуть створювати персоналізовані пропозиції, адаптувати контент і рекламу до віку та статі своїх користувачів, що дозволить збільшити ефективність рекламних кампаній і підвищити рівень конверсії.

Іншим важливим споживачем таких технологій є платформи, які пропонують онлайн-послуги, включаючи ігрові платформи, соціальні мережі та платформи для навчання. Технології, що визначають стать та вік, допомагають таким платформам автоматично сегментувати користувачів, адаптувати контент та обмежувати доступ до певних функцій, що особливо важливо для забезпечення безпеки і належного рівня обслуговування користувачів [1].

1.3 Сервіси, які використовують збір та аналіз інформації про користувачів

1.3.1 Google Analytics

Google Analytics (Рис 1.1) — це потужний інструмент для веб-аналітики, створений компанією Google у 2005 році. Він дозволяє веб-майстрам та маркетологам відслідковувати і аналізувати поведінку користувачів на вебсайтах. Інструмент використовує різноманітні патерни поведінки, щоб допомогти користувачам краще розуміти, як відвідувачі взаємодіють з контентом, зокрема, що вони переглядають, скільки часу проводять на сторінках, де залишають сайт тощо.

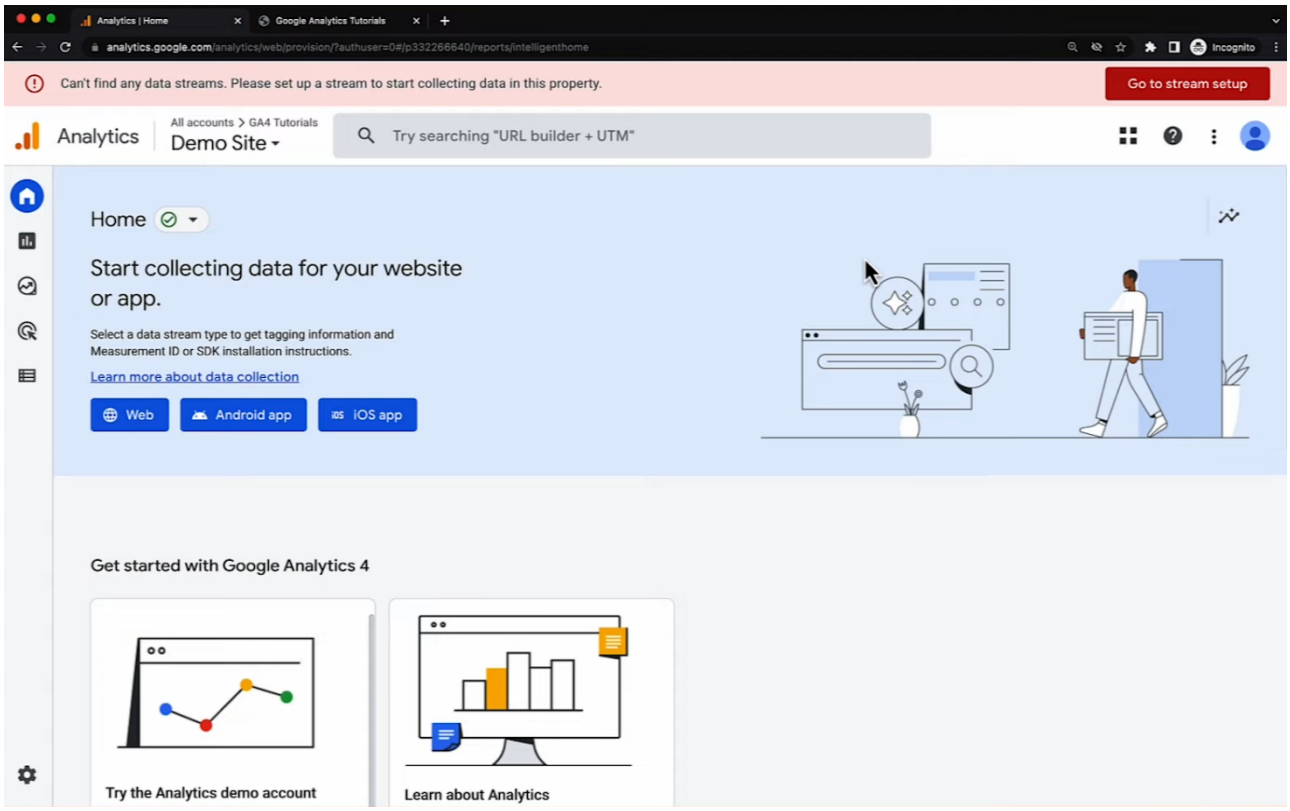


Рисунок 1.1 – Зовнішний вигляд веб-версії Google Analytics

Google Analytics є одним із найбільш популярних інструментів для веб-аналітики, і його використовують як великі компанії, так і малий бізнес. Це

універсальний продукт, що охоплює численні сегменти ринку, від e-commerce платформ до блогерів та стартапів. Інструмент доступний безкоштовно з базовими функціями та в розширеній версії — Google Analytics 360 для великих компаній з більш складними вимогами.

Google Analytics став одним із лідерів на ринку аналітики, оскільки він дозволяє глибоко аналізувати патерни поведінки користувачів і на основі цього приймати рішення щодо оптимізації вебсайтів. Проте в процесі його розвитку виникали труднощі, зокрема з точністю збору даних через блокувальники реклами або обмеження в GDPR, що вимагали адаптацій і постійних змін в політиці збору даних [3].

1.3.2 Netflix

Netflix (Рис. 1.2) — це сервіс для стримінгу відео, заснований у 1997 році. Одна з основних його особливостей — це система рекомендацій, яка використовує алгоритми для вивчення патернів поведінки користувачів (такі як переглянуті фільми, жанри та рейтинги) і пропонує персоналізовані рекомендації. Алгоритми враховують вподобання користувача, а також схожі патерни у поведінці інших користувачів для створення точних рекомендацій.

Netflix має понад 230 мільйонів підписників по всьому світу і є одним із найбільших сервісів для онлайн-стримінгу. Цільова аудиторія охоплює широкий спектр користувачів, від молоді до літніх людей, з різними інтересами та смаками. Платформа адаптована до різних пристроїв і доступна практично в усіх країнах, з різноманітними мовами та культурними відмінностями в контенті.

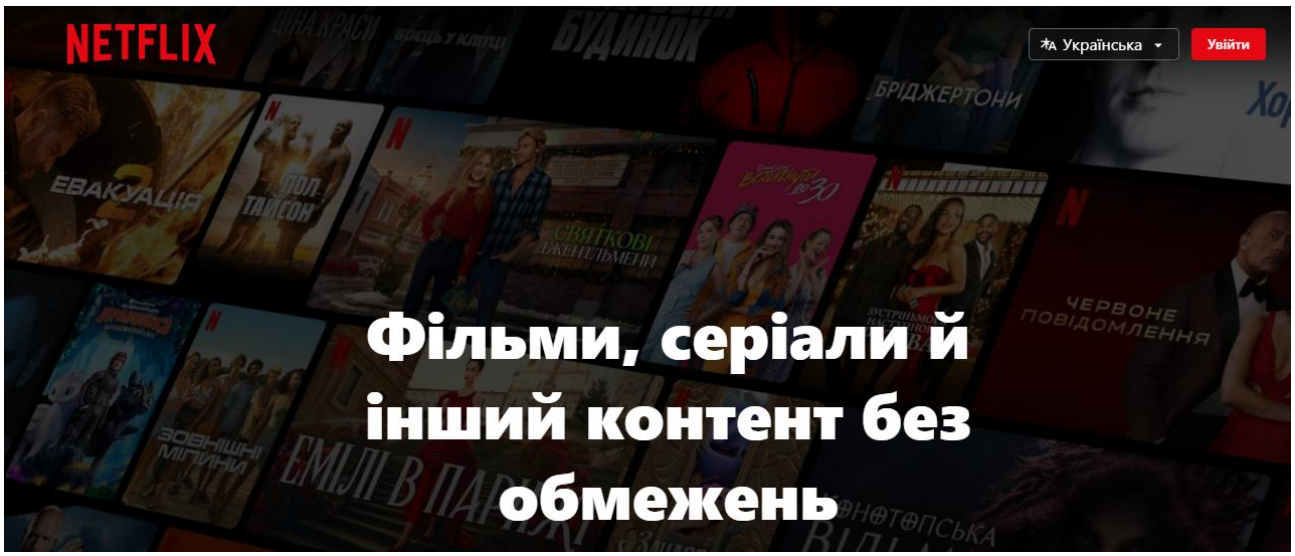


Рисунок 1.2 – Головна сторінка Netflix

Netflix успішно застосував патерни поведінки для створення персоналізованого досвіду користувачів, що дозволило значно збільшити кількість підписників і зберегти лояльність до платформи. Однак компанія зіткнулася з труднощами, такими як питання збереження прав на контент, а також конкуренція з іншими платформами [4].

1.3.3 Amazon

Amazon (Рис. 1.3) — один із найбільших онлайн-ритейлерів у світі, заснований у 1994 році Джеффом Безосом. Інтерфейс сайту і рекомендаційна система Amazon активно використовують аналіз патернів поведінки користувачів для персоналізації покупок. Платформа аналізує історію пошуку, покупки, перегляди товарів та інші фактори для створення індивідуальних пропозицій, що збільшують ймовірність здійснення покупки.

Amazon є лідером у сфері електронної комерції, з понад 200 мільйонами активних користувачів по всьому світу. Його користувачі — це як звичайні покупці, так і підприємства, що користуються платформою для продажу товарів

через Amazon Marketplace. Цільова аудиторія варіюється від індивідуальних покупців до великих корпорацій, що використовують Amazon для задоволення своїх потреб у постачанні товарів.

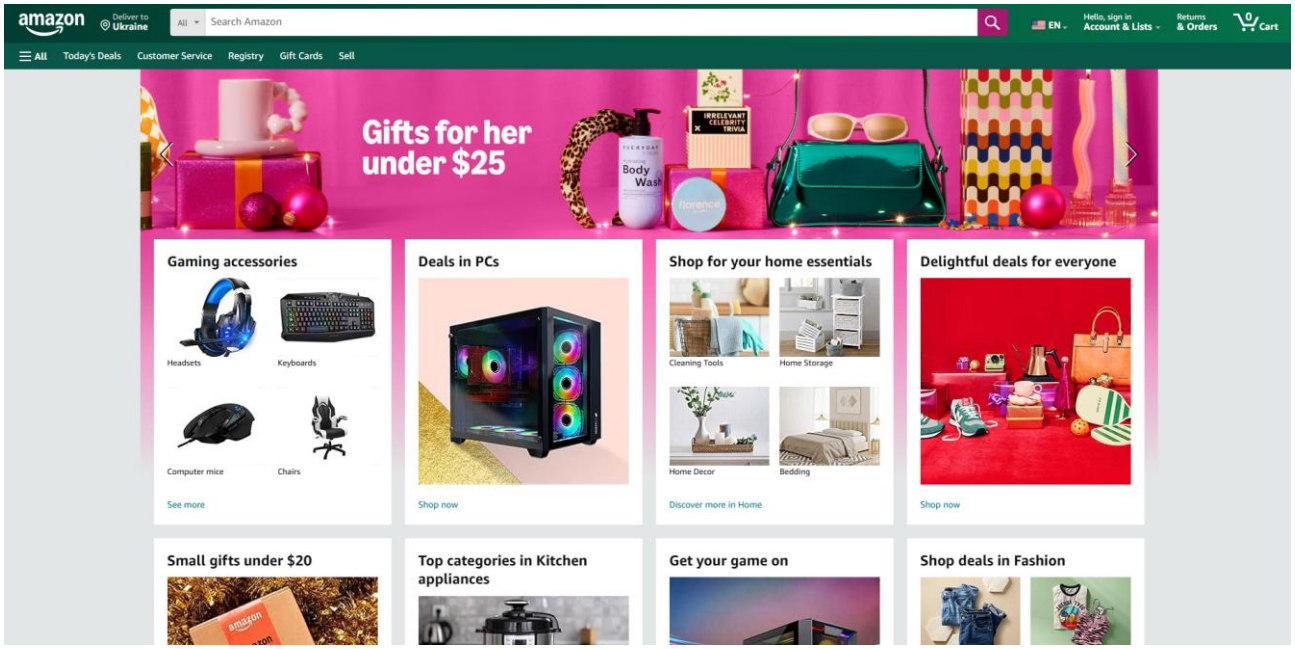


Рисунок 1.3 – Інтернет магазин Amazon

Amazon продовжує бути одним із найбільш успішних прикладів використання аналізу поведінки для персоналізації покупок. Проте компанія зустрілась із критикою через надмірну залежність від алгоритмів, які можуть пропонувати користувачам занадто багато подібних товарів, що обмежує їхній вибір [5].

1.3.4 Fitbit

Fitbit (Рис. 1.4) — це компанія, що спеціалізується на виробництві носимих пристроїв для відстеження фізичної активності, таких як фітнес-трекери і смарт-годинники. Створена в 2007 році Джеймсом Хоскінсоном і Ерном Фрідом, компанія використовує дані про фізичну активність, щоб аналізувати

патерни поведінки користувачів та пропонувати індивідуальні рекомендації для покращення здоров’я, зокрема, щодо кількості кроків, сну, фізичних вправ тощо.

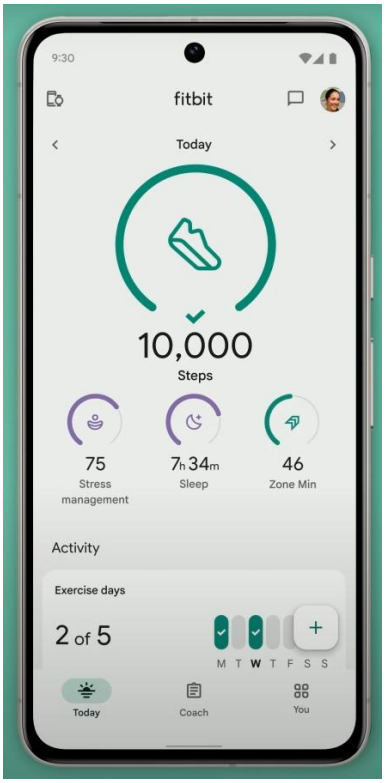


Рисунок 1.4 – Мобільний додаток Fitbit

Fitbit здобув популярність серед людей, які прагнуть контролювати своє здоров'я, зокрема серед любителів спорту, тих, хто слідкує за ваговими показниками або прагне підтримувати загальний фізичний стан. За весь час існування компанії її пристрої використовували мільйони людей по всьому світу. Основною аудиторією є активні люди від молоді до середнього віку, а також люди, що мають проблеми зі здоров’ям і потребують постійного моніторингу.

Fitbit виявився дуже успішним у створенні інноваційного продукту, який дозволяє користувачам не тільки відслідковувати свої фізичні показники, а й отримувати персоналізовані поради. Проте компанія зіткнулася з певними труднощами, такими як конкуренція з іншими виробниками носимих пристроїв (наприклад, Apple Watch) та питання конфіденційності даних, оскільки пристрої збирають велику кількість особистої інформації про здоров’я користувачів [6].

1.3.5 Facebook (Meta)

Facebook (Meta) — це одна з найбільших соціальних мереж у світі, заснована в 2004 році Марком Цукербергом. Платформа активно аналізує патерни поведінки користувачів для таргетування реклами, визначаючи інтереси, звички та демографічні характеристики. Алгоритми Facebook оцінюють взаємодії користувачів, такі як лайки, коментарі, пости, перегляди відео, щоб пропонувати максимально релевантні рекламні оголошення.

Facebook має понад 2,8 мільярда активних користувачів і є найпопулярнішою соціальною мережею у світі. Її цільова аудиторія дуже широка — від підлітків до людей старшого віку, зокрема, активними користувачами є бізнеси, що використовують платформу для реклами своїх товарів та послуг.

Facebook став найбільшим прикладом використання алгоритмів для аналізу поведінки користувачів, що дозволяє ефективно таргетувати рекламу. Однак компанія стикнулася з рядом проблем, таких як скандали з витоком даних, порушення конфіденційності та етичні питання щодо маніпулювання користувацькими вподобаннями. Ці проблеми змусили компанію переглядати свої підходи до збору та використання даних [7].

1.4 Переваги використання нативного JavaScript для збору даних з вебсайтів

JavaScript — це високорівнева мова програмування, яка в основному використовується для створення інтерактивних веб-сторінок. Вона є однією з основних технологій для розробки веб-додатків разом з HTML та CSS.

JavaScript дозволяє розробникам додавати динамічний контент на сайти, та з її допомогою можна створювати як прості інтерактивні елементи, так і

складні веб-додатки, відправляти запити до серверів, обробляти дані в реальному часі та багато іншого.

Важливі технології і концепції в JavaScript:

- DOM (Document Object Model) є інтерфейсом для взаємодії з HTML та XML документами. JavaScript може використовувати DOM для доступу до елементів веб-сторінки, їх змінювання, додавання нових елементів або видалення існуючих.
- JavaScript дозволяє реагувати на події, такі як натискання кнопок, наведення миші, прокручування сторінки та інші. Для цього використовуються обробники подій.
- JavaScript дозволяє обробляти асинхронні операції (наприклад, запити до серверів або зчитування даних з файлів) без блокування основного потоку виконання програми. Для цього використовуються Promises, async/await та callback.
- В JavaScript з ES6 з'явилася підтримка модулів. Це дає змогу організовувати код у окремі файли, що покращує його структуру і зручність підтримки.

Однією з найбільших переваг використання JavaScript є мінімізація залежностей, що дає змогу працювати з меншими обсягами ресурсів, оскільки не потрібно завантажувати зайві скрипти, які можуть лише ускладнити виконання задачі. Зібрані дані можуть бути оброблені на клієнтській стороні без додаткового навантаження на сервер.

JavaScript дозволяє мати повний контроль над процесом збору та обробки даних та надає можливість налаштовувати кожен аспект, від того, як відслідковуються події на сторінці до того, як дані відправляються на сервер або зберігаються локально. Замість того, щоб залежати від обмежених можливостей фреймворків, використовуючи нативний JavaScript можна створювати скрипти, що чітко відповідають вимогам, забезпечуючи точність і гнучкість в зборі даних.

Без фреймворків веб-сторінки завантажуються швидше, оскільки не потрібно завантажувати велику кількість зовнішніх бібліотек. Це особливо

важливо для користувачів з обмеженими швидкостями інтернет-з'єднання, оскільки кожен додатковий запит на завантаження файлів може уповільнити загальну швидкість завантаження сторінки. З нативним JavaScript ви можете максимально зменшити час завантаження сторінки та покращити взаємодію користувача з сайтом, що особливо важливо для збору даних у реальному часі.

JavaScript дозволяє краще контролювати безпеку збору даних. Оскільки багато сторонніх фреймворків можуть мати вразливості або бути частинами більш широких систем збору даних, які можуть потенційно передавати інформацію без відома користувача. За допомогою чистого JavaScript можна впроваджувати кращі практики безпеки, такі як захист від CSRF-атак, шифрування даних або використання безпечних протоколів для передачі інформації.

JavaScript дозволяє вам писати код, який максимально універсальний та сумісний. Це дозволяє збирати дані з користувачів на різних пристроях без потреби в додаткових налаштуваннях чи витратному тестуванні, забезпечуючи стабільну роботу на всіх платформах [8].

1.5 Машинне навчання в обробці даних

Штучний інтелект та машинне навчання є потужними інструментами, які відіграють ключову роль у обробці та аналізі даних, зокрема зібраних з таких джерел, як рухи миші та використання клавіатури. Ці технології здатні не лише автоматично обробляти великі обсяги інформації, а й знаходити закономірності в поведінці користувачів, що можуть бути важливими для персоналізації інтерфейсів, покращення безпеки або маркетингових стратегій.

Машинне навчання дозволяє комп'ютерам адаптуватися до нових даних без явного програмування. В основі цієї технології лежить використання алгоритмів, які навчаються на зібраних даних і з часом покращують свої

прогнози або класифікацію. Зокрема, для аналізу руху миші та натискання клавіатури машинне навчання може бути використане для виявлення патернів поведінки, які допомагають прогнозувати вік, стать, або навіть емоційний стан користувача.

Застосування штучного інтелекту в аналізі руху миші та використання клавіатури має величезний потенціал. Наприклад, за допомогою алгоритмів машинного навчання можна виявити унікальні патерни в поведінці користувачів. Для цього можна використовувати різноманітні методи класифікації та регресії, такі як дерева рішень, підтримка векторних машин або глибоке навчання, яке здатне обробляти великі обсяги даних і знаходити найскладніші закономірності, що не очевидні для людського спостерігача.

Штучний інтелект також дозволяє виявляти аномалії в поведінці користувачів, що може бути корисно для виявлення потенційно небезпечних ситуацій, наприклад, при спробах несанкціонованого доступу до системи. Якщо модель навчена на історії взаємодії з інтерфейсом (рух миші, натискання клавіатури), вона може порівняти поточну поведінку з типовими шаблонами і визначити, чи є відхилення, що можуть вказувати на шахрайські або незвичні дії.

Одним з основних переваг використання штучного інтелекту є його здатність до персоналізації досвіду користувачів. Наприклад, якщо модель розуміє, як конкретний користувач взаємодіє з інтерфейсом, вона може адаптувати навігацію або оформлення сторінок, щоб зробити взаємодію з сайтом або додатком більш зручним. Алгоритм може передбачити, які елементи інтерфейсу найбільш корисні для конкретного користувача, зменшуючи кількість необхідних кліків або рухів миші для виконання завдань.

Розпізнавання емоцій та психологічних станів також може бути досягнуте за допомогою аналізу рухів миші та натискання клавіатури. Наприклад, аналіз швидкості введення тексту, пауз між натисканнями клавіш або руху миші може дати інформацію про те, чи перебуває користувач у стресовому стані, чи він зацікавлений у певному контенті, або, можливо, має фізичні обмеження, які уповільнюють його реакції.

Глибоке навчання є одним з потужних напрямків у штучному інтелекті, що активно застосовується для розпізнавання складних патернів у великих наборах даних. Використовуючи нейронні мережі, що здатні автоматично знаходити ієрархічні зв'язки в даних, можна досягти високої точності в аналізі поведінки користувача. У випадку з рухом миші, нейронна мережа може виявити навіть найнезначніші зміни в траєкторії курсора, які можуть бути важливими для подальших рекомендацій або прогнозів [9].

Обробка даних в реальному часі є ще однією важливою задачею, яку можна ефективно вирішити за допомогою штучного інтелекту. У реальних системах, де важливо приймати рішення миттєво (наприклад, у випадку аутентифікації або для персоналізації інтерфейсу), алгоритми машинного навчання можуть працювати на основі поточних даних, зібраних про поведінку користувача. Це дозволяє системам адаптуватися до змін у реальному часі, що робить взаємодію більш зручною і безпечною для кінцевого користувача.

1.6 Використання мови Python для створення моделі штучного інтелекту

Важливою частиною цієї задачі є підготовка даних. Перед тим як застосовувати машинне навчання до зібраних даних (наприклад, руху миші або натискання клавіатури), потрібно здійснити їх очистку, нормалізацію та перетворення в зручний для аналізу формат. Цей процес може включати фільтрацію шуму, відокремлення важливих характеристик та створення спеціальних ознак, які підвищать точність моделей машинного навчання [10].

Python є однією з найбільш популярних мов програмування для вирішення задач машинного навчання завдяки наявності потужних бібліотек та інструментів. Такі бібліотеки, як TensorFlow, Keras, scikit-learn, PyTorch та Pandas, дозволяють швидко розробляти, тренувати та тестувати моделі

машинного навчання. Завдяки широкому вибору інструментів для обробки та аналізу даних Python є незамінним для збору та обробки даних.

Оскільки Python є мовою високого рівня з чітким синтаксисом і багатим набором бібліотек, він дозволяє легко реалізувати складні моделі машинного навчання без необхідності глибоких знань у низькорівневих аспектах програмування. Збір, обробка та аналіз даних в Python здійснюються з мінімальними витратами часу, що робить цю мову ідеальним вибором для швидкої розробки прототипів або для реалізації складних, продуктивних систем на основі машинного навчання.

Завдяки можливостям Python для обробки великих обсягів даних і інтеграції з іншими системами, можна створювати потужні платформи для аналізу поведінки користувачів. Наприклад, Python дозволяє зібрані дані з миші та клавіатури обробляти через глибоке навчання, класифікацію або регресію для створення персоналізованих рекомендацій або для виявлення аномалій у поведінці користувачів, що підвищує ефективність роботи системи.

Штучний інтелект і машинне навчання відкривають нові можливості для аналізу та обробки поведінкових даних, дозволяючи створювати більш інтуїтивно зрозумілі та адаптовані до користувачів системи. З використанням таких технологій можна значно підвищити рівень персоналізації та безпеки, що особливо важливо в умовах сучасного цифрового середовища.

1.7 Нечіткологічний підхід

Найбільш вражаючою властивістю людського інтелекту є здатність приймати правильні рішення в умовах неповної і нечіткої інформації. Побудова моделей, які відтворюють мислення людини і використання їх у комп'ютерних системах на сьогодні є однією з найважливіших проблем науки [12].

Основи нечіткої логіки було закладено наприкінці 60-х років у працях відомого американського математика Лотфі Заде. Дослідження подібного роду було викликано зростаючим незадоволенням експертними системами. «Штучний інтелект», що легко справлявся із задачами керування складними технічними комплексами, був безпорадним в простих життєвих ситуаціях, типу "Якщо машиною перед тобою керує недосвідчений водій - тримайся від неї подалі" [12].

Для створення дійсно інтелектуальних систем, здатних адекватно взаємодіяти з людиною, був потрібен новий математичний апарат, який перекладає і неоднозначні життєві твердження на мову чітких математичних формул [12].

Першим серйозним кроком в цьому напрямку була теорія нечітких множин, розроблена доктором Лотфі Заде. Його робота "Fuzzy Sets" з'явилася в 1965 році в журналі "Information and Control". Вона заклала основи моделювання інтелектуальної діяльності людини і стала поштовхом до розвитку нової області науки -"fuzzy logic" (fuzzy - нечіткий, розмитий, м'який) [12].

Нечіткологічний підхід, відомий також як теорія нечіткої логіки або нечіткі системи, є підходом до моделювання та аналізу систем, який дозволяє враховувати нечіткість або неоднозначність в даних та правилах. Цей підхід був розроблений для того, щоб розширити традиційну булеву логіку для обробки нечітких або неоднозначних концепцій.

Замість традиційної бінарної логіки, де елементи належать до множини або ні, нечіткологічний підхід використовує концепцію "ступінь належності". Кожен елемент може мати ступінь приналежності до різних множин, що дозволяє враховувати нечіткість.

Нечіткі правила визначають взаємозв'язки між вхідними та вихідними змінними у вигляді нечітких умов. Наприклад, "Якщо температура висока, то знизити витрату енергії".

Також, вводяться нечіткі множини та відповідні оператори, які дозволяють працювати з нечіткими значеннями та виражати нечіткі взаємозв'язки.

Для отримання конкретних результатів з нечітких вхідних даних використовуються методи агрегації (комбінування нечітких правил) та дефазифікації (перетворення нечіткого виходу в чітке значення).

Такий підхід широко застосовується в областях, де дані або правила можуть бути нечіткими або неоднозначними, таких як керування системами, прийняття рішень, штучний інтелект, обробка сигналів, а також в експертних системах та прогнозуванні.

Нечіткологічний підхід дозволяє моделювати та аналізувати системи, які не піддаються точному визначенню або чітким правилам, що робить його корисним інструментом для розв'язання реальних проблем в різних галузях [2].

Підсумовуючи, можна виокремити головну рису, що відрізняє нечіткологічний підхід від логічного виведення: системи логічного виведення базуються на чітких, бінарних значеннях (правда/неправда, 0/1) і використовуються для виведення точних логічних результатів на основі чітких правил і вхідних даних, а нечіткологічні підхід на відмінну від них використовує нечіткі (або нечітко визначені) значення, що можуть бути у вигляді діапазонів або функцій приналежності, щоб врахувати неоднозначність або нечіткість в даних та правилах. Це дозволяє більш гнучко моделювати реальні ситуації, де відсутня чітка межа між категоріями [12].

1.8 Онтології

Онтологія – це знання, формально відображені на базі концептуалізації. Концептуалізація – опис множини об’єктів і понять, знань про них і зв’язків між ними. Формально онтологія складається з термінів (понять, концептів), організованих у таксономію, їхніх визначень і атрибутів, а також пов’язаних з ними аксіом і правил виведення. Онтологія визначає загальний словник для користувачів, які спільно використовують інформацію про деяку ПО [13].

Враховуючи вищенаведене, під формальною моделлю онтології О розуміють трійку такого вигляду:

$$O = \langle C, R, F \rangle, \quad (1)$$

де С – скінченна множина понять (концептів, термінів) ПО, яку задає онтологія О;

R → C – скінченна множина відношень між концептами (поняттями, термінами) заданої ПО;

F – скінченна множина функцій інтерпретації (аксіоматизація, обмеження), заданих на концептах чи відношеннях онтології О.

Онтологію можна подати у вигляді графу, де вершини – концепти ПО, дуги вказують напрями відношень між концептами (Рис. 1.5). Вершини можуть бути як інтерпретованими (визначені аксіоми понять), так й не інтерпретованими. Інтерпретовані вершини позначені темним кольором. Своєю чергою, відношення діляться на вертикальні (суцільні лінії) та горизонтальні (штрихпунктирні лінії) [13].

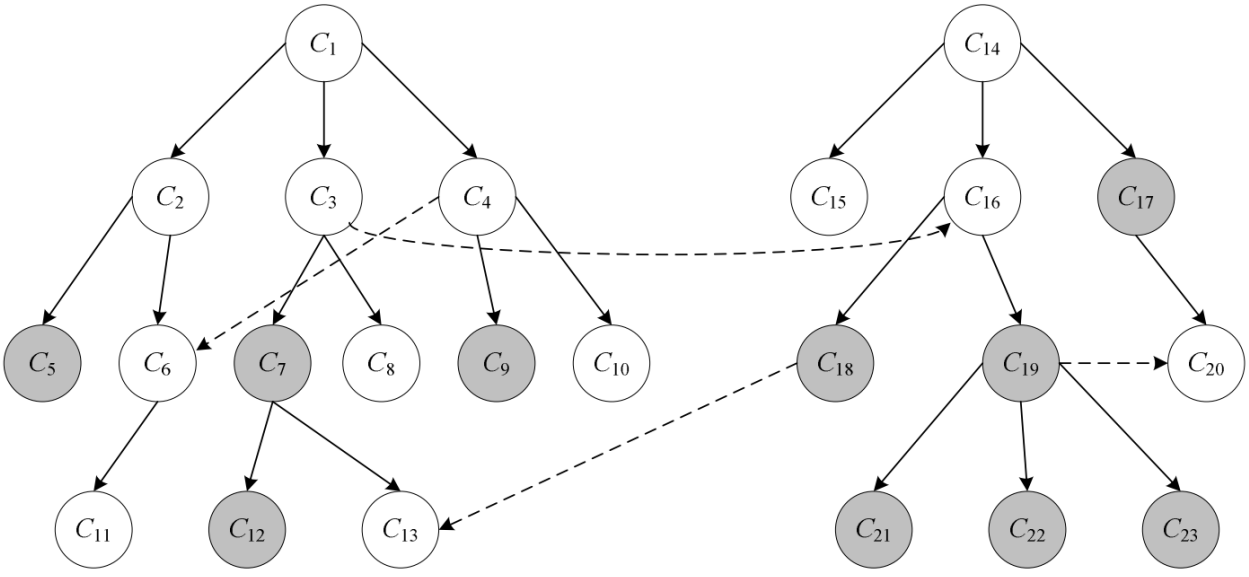


Рисунок 1.5 – Приклад графу онтології

Онтології використовуються для розробки семантичної моделі знань, яка дозволяє комп'ютерам розуміти та обробляти інформацію в певній області. Вони

грають важливу роль у багатьох областях, таких як семантичний веб, штучний інтелект, обробка природної мови, біоінформатика, та інші. Онтології допомагають у створенні стандартизованих та універсальних моделей знань, що полегшує обмін інформацією між різними системами та додатками [13].

Онтологія відноситься до систем підтримки прийняття рішень (СППР), але зазвичай не входить безпосередньо в категорію систем логічного виведення. Вона представляє собою формальну модель, яка використовується для представлення знань про певну область за допомогою концепцій, їх взаємозв'язків та властивостей.

Системи логічного виведення, як правило, використовують правила логічного виведення для аналізу вхідних даних та прийняття рішень. У той час як онтологія служить основою для створення формальної структури знань, яка може використовуватися різними видами СППР, включаючи системи логічного виведення [14].

1.9 TensorFlow.Keras

TensorFlow.Keras — це високорівневий API, який дозволяє будувати, тренувати та оцінювати моделі машинного навчання, що використовують глибокі нейронні мережі. Він заснований на Keras, бібліотеці для глибокого навчання, яка була інтегрована в TensorFlow починаючи з версії 2.0. tensorflow.keras надає потужний інструментарій для розробки нейронних мереж, включаючи підтримку різних типів шарів, оптимізаторів, функцій активації та втрат, що робить його придатним для широкого кола завдань — від класифікації до обробки зображень і тексту.

Однією з ключових переваг tensorflow.keras є його простота в використанні та гнучкість. Модель можна побудувати в кілька рядків коду, використовуючи як функціональний, так і послідовний (sequential) API.

Sequential ідеально підходить для лінійних стеків шарів, де кожен шар приймає на вхід вихід попереднього. Водночас функціональний API надає можливість створювати складніші архітектури, такі як багатоканальні мережі та мережі з кількома входами чи виходами.

Також важливою частиною tensorflow.keras є підтримка різних типів шарів, включаючи повнозв'язні шари (Dense), згорткові шари (Conv2D), рекурентні шари (LSTM, GRU) та інші. Це дозволяє створювати моделі, які можуть вирішувати різноманітні завдання, включаючи класифікацію зображень, обробку тексту та прогнозування часових рядів. У tensorflow.keras легко налаштувати складні архітектури нейронних мереж, які включають такі механізми, як регуляризація, нормалізація та оптимізація.

Крім того, TensorFlow підтримує роботу з різними платформами, такими як CPU, GPU і TPU, що значно прискорює тренування великих моделей. Можливості розподіленого навчання дозволяють обробляти величезні обсяги даних і ефективно тренувати моделі на кількох машинах одночасно. Це робить tensorflow.keras відмінним вибором для роботи з великими датасетами та складними моделями.

Ще однією сильною стороною tensorflow.keras є інтеграція з TensorFlow Hub та TensorFlow Model Garden. Ці інструменти дозволяють використовувати вже навчені моделі для переносу навчання та для прискорення розробки. Наприклад, можна взяти попередньо навчену модель на ImageNet і адаптувати її під своє завдання, що скорочує час тренування і покращує результати на менших датасетах.

scikeras.wrappers — це обгортка, яка дозволяє інтегрувати моделі Keras з бібліотеками та інструментами з екосистеми scikit-learn. Основна мета scikeras — надати простий спосіб використання Keras з такими інструментами, як GridSearchCV, RandomizedSearchCV, а також інтеграцію з іншими компонентами scikit-learn, такими як крос-валідація та оцінка моделей. Це дозволяє легко налаштовувати параметри моделі та виконувати інші завдання, характерні для

машинного навчання, використовуючи можливості, що пропонуються Keras для глибокого навчання.

Однією з ключових можливостей scikeras є надання інтерфейсу для навчання та оцінки моделей Keras як для стандартних моделей машинного навчання. Наприклад, можна використовувати KerasClassifier або KerasRegressor для класифікації та регресії відповідно, що дозволяє інтегрувати нейронні мережі з рештою частиною машинного навчання, наданого scikit-learn, і використовувати такі алгоритми, як крос-валідація та пошук гіперпараметрів.

sklearn.model_selection — це підмодуль бібліотеки scikit-learn, який надає інструменти для поділу даних на навчальні та тестові вибірки, а також для проведення крос-валідації та пошуку гіперпараметрів моделей. Серед найбільш популярних функцій цього модуля — train_test_split, який допомагає поділити дані на навчальну та тестову вибірки, та GridSearchCV, який автоматично підбирає найкращі параметри для моделі на основі крос-валідації. Важливою функцією також є cross_val_score, яка дозволяє оцінювати продуктивність моделі на різних підмножинах даних, мінімізуючи ймовірність переобучення і підвищуючи точність результатів [15].

2 ПРАКТИЧНА ЧАСТИНА

2.1 Постановка завдання

Першим етапом процесу є збір даних про взаємодію користувача з веб-сторінкою за допомогою нативного JavaScript. Завданням є відслідковування рухів миші, натискань клавіатури та часу взаємодії з різними елементами інтерфейсу. Для цього використовуватимуться стандартні методи обробки подій, такі як mousemove, keydown, keyup, click та інші. JavaScript буде записувати ці події у вигляді метрик: координати курсора, частота натискання клавіш, інтервали між подіями, а також інші аспекти поведінки, такі як швидкість переміщення миші та кількість повторюваних натискань. Всі ці дані будуть зібрані в реальному часі і передані до клієнтської частини для попередньої обробки.

Після збору даних вони підлягають попередній обробці на клієнтській частині за допомогою JavaScript. На цьому етапі здійснюється нормалізація та очищення даних, тобто перетворення всіх показників у зручний для подальшого аналізу формат. Наприклад, можна обчислити середні значення швидкості руху миші, інтервали між натисканнями клавіатури та інші параметри. Також на клієнтській частині можна агрегувати та зберігати лише необхідні дані, щоб мінімізувати обсяг інформації, яка буде передаватися на сервер. Зібрані та оброблені дані упаковуються в JSON-формат і відправляються на сервер через HTTP-запит.

На серверній стороні дані обробляються за допомогою методів машинного навчання, реалізованих на мові Python. Після отримання даних, сервер застосовує попередньо натреновані моделі для класифікації віку та статі користувача на основі їх поведінки. Для цієї мети можуть бути використані алгоритми класифікації, які навчалися на великих наборах даних про взаємодію користувачів. Моделі отримують входи у вигляді числових параметрів

(швидкість руху миші, частота натискання клавіатури, час взаємодії з елементами інтерфейсу тощо) і на основі цих ознак прогнозують вік і стать користувача.

Після того, як модель визначить вік та стать користувача, сервер формує персоналізований рекомендаційний список. Це може бути, наприклад, список товарів, контенту чи послуг, які найбільше підходять для даної вікової категорії та статі. Окрім простого відображення характеристик користувача, модель може враховувати додаткові аспекти взаємодії, такі як інтереси користувача або схильність до певних типів контенту. Отриманий список рекомендацій відправляється назад на клієнтську частину, де він відображається у вигляді динамічних елементів інтерфейсу. Таке рішення дозволяє адаптувати досвід користувача в реальному часі, підвищуючи ефективність взаємодії з вебсайтом або додатком.

2.2 Визначення можливих даних для збору

Одним з основних параметрів, який можна відправляти на сервер для аналізу, є швидкість руху миші. Це значення вказує на те, як швидко користувач переміщає курсор по екрану, що може свідчити про рівень його досвіду з інтерфейсом або навіть про фізичний стан (наприклад, стомленість). Збираючи ці дані, можна визначити характер взаємодії користувача з інтерфейсом і використовувати їх для персоналізації або навіть для безпеки (виявлення аномальних швидкостей руху, що можуть вказувати на спробу атаки).

Іншим важливим параметром є відношення реальної траєкторії руху миші до найкоротшого шляху між двома точками. Це дозволяє оцінити точність і прямолінійність руху курсора. Високе значення цього відношення може свідчити про неефективну або заплутану навігацію, яка може бути корисною для аналізу звичок користувачів або для виявлення помилок у дизайні інтерфейсу, що змушують користувача здійснювати зайві рухи миші.

Частота зміни напрямків руху миші також є важливим параметром для збору даних. Якщо користувач постійно змінює напрямок руху миші, це може вказувати на невизначеність або сумніви у виборі елементів інтерфейсу, що дозволяє оптимізувати інтерфейс під потреби користувачів. Дані про частоту змін напрямків можуть бути корисні для оцінки легкості або складності навігації на сайті, а також для виявлення проблем з юзабіліті.

Дані про час натискання клавіші миші можуть бути корисними для аналізу користувацької поведінки, особливо у випадку, коли миша використовується для вибору елементів на сторінці. Це може включати середній час між кліками або затримки, які можуть вказувати на невизначеність користувача при виборі елемента. Для застосування в реальному часі, ці дані можуть допомогти адаптувати інтерфейс або навіть створити спеціальні рекомендації.

Швидкість подвійного кліку є важливим параметром для визначення швидкості взаємодії користувача з елементами інтерфейсу. Якщо користувач робить подвійний клік швидше, це може свідчити про досвідченість або впевненість у виборі елементів. Повільний подвійний клік може вказувати на те, що користувач не зовсім знайомий з функціоналом інтерфейсу або має проблеми з точністю при наведенні миші. Ці дані допомагають розуміти рівень комфорту користувача з інтерфейсами та можуть бути використані для їх поліпшення.

Час затискання клавіші (наприклад, натискання клавіші "Enter" або "Backspace") також може дати багато корисної інформації. Якщо користувач довго затримує певну клавішу, це може свідчити про те, що він обмірковує наступний крок або часто використовує цю клавішу для виконання певних дій. Наприклад, довге натискання клавіші "Shift" або "Ctrl" може вказувати на використання комбінацій клавіш або специфічних функцій програми, що також може бути корисно для аналізу поведінки.

Час пошуку клавіші — це параметр, що вказує на те, скільки часу користувач витрачає на пошук конкретної клавіші на клавіатурі, особливо якщо він не впевнений у своїх діях або ж використовує нестандартну розкладку клавіатури. Це може виявитися корисним для аналізу того, як швидко користувачі

можуть взаємодіяти з інтерфейсами на основі їх рівня знайомства з платформою або пристроєм.

Тривалість набору тексту для певного блоку на сторінці є ще одним важливим індикатором. Це дозволяє аналізувати, як швидко користувачі заповнюють форми, анкети або інші текстові поля. Затримки можуть свідчити про те, що користувач має труднощі з розумінням запиту, що може вимагати поліпшення інтерфейсу або надання додаткових пояснень. Ці дані також можна використовувати для персоналізації — наприклад, для налаштування підказок або допоміжних текстів в реальному часі.

Вибір даних, які варто відправляти на сервер, залежить від специфіки завдання та цілей збору. Дані, які дозволяють зрозуміти, як користувач взаємодіє з інтерфейсом, зокрема швидкість руху миші, частота змін напрямку та час натискання клавіші, можуть бути використані для покращення досвіду користувача. Однак важливо враховувати, що надмірний збір персональних або чутливих даних може порушувати конфіденційність користувачів. Тому вибір параметрів для збору даних має бути збалансований, з акцентом на їхню корисність і мінімізацію впливу на конфіденційність.

2.3 Збір даних

Для збору даних про рух миші ми будемо використовувати функцію слухача подій, яка буде реєструвати координати курсора на екрані з частотою кожні 0,25 секунди. Це дозволить нам отримати досить точні дані про траєкторію руху миші, не перевантажуючи систему занадто частими запитами. Функція слухача буде реєструвати координати x та y (положення миші по горизонталі та вертикалі) кожного руху миші. Ці координати зберігатимуться в масиві, де кожен запис буде супроводжуватись часом події (часова мітка), що дозволить створити точну хронологію руху. Оскільки час затримки між записами становить 0,25

секунди, ми отримаємо достатньо точний набір даних для подальшої обробки, але без надмірного навантаження на ресурс користувача.

Лістинг 2.1

```
document.addEventListener("mousemove", function(event) {
   if (Date.now() - lastMouseMove >= 25) {
      [код запису і обробки подій]
      lastMouseMove = Date.now();
   }
});
```

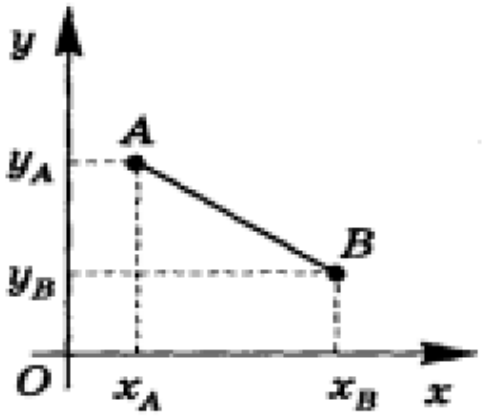
Щоб забезпечити коректну обробку даних, після кожного збору координат миші будемо фіксувати три точки на основі поточної та попередньої позицій миші. Це дозволить отримати три ключові елементи: стартову точку, кінцеву та посередню, що дає можливість точно оцінити швидкість руху.

Лістинг 2.2

```
if (dots.length < 3) {
   dots.unshift(new Dots(event.clientX, event.clientY, Date.now()));
} else {
   dots.pop();
   dots.unshift(new Dots(event.clientX, event.clientY, Date.now()));
}
```

Використовуючи ці точки, можна обчислити відстань між ними за допомогою стандартної формули для відстані між двома точками на площині (формула Евкліда (Рис. 2.1)). Після цього порівнюється реальна пройдена відстань з теоретичною мінімальною відстанню між стартовою та кінцевою точкою (лінійна відстань між ними), що дає нам коефіцієнт відношення пройденого шляху до мінімального. Чим більша різниця між реальним і

мінімальним шляхом, тим більше інформації про складність руху, що може бути корисним для виявлення ускладнень в навігації або для подальшої персоналізації інтерфейсу.



$$AB = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

Рисунок 2.1 – Евклідова відстань

Лістинг 2.3

```
if (dots.length === 3) {
    let dist1 = Math.sqrt(Math.pow(dots[2].x - dots[1].x, 2) + Math.pow(dots[2].y - dots[1].y, 2));
    let dist2 = Math.sqrt(Math.pow(dots[1].x - dots[0].x, 2) + Math.pow(dots[1].y - dots[0].y, 2));
    let dist3 = Math.sqrt(Math.pow(dots[2].x - dots[0].x, 2) + Math.pow(dots[2].y - dots[0].y, 2));
    let timeDiff = (dots[0].time - dots[2].time) / 1000;

    if (timeDiff > 0 && dist1 > 0 && dist2 > 0 && dist3 > 0) {
      speed.push((dist1 + dist2) / timeDiff);
      kBtw2Dots.push((dist1 + dist2) / dist3 – 1);
    }
}
```

Крім того, ми будемо визначати кількість різких змін напрямку, тобто моментів, коли траєкторія різко змінюється (кути між векторами напрямку більше за 90°), що може вказувати на неузгодженість у поведінці або на потенційні проблеми в юзабіліті.

Лістинг 2.4

```
if ((dots[0].x < dots[1].x && dots[0].x > dots[2].x) || (dots[0].y < dots[1].y && dots[0].y > dots[2].y) || (dots[0].x < dots[2].x && dots[0].x > dots[1].x) || (dots[0].y < dots[2].y && dots[0].y > dots[1].y)) {
    gerz += 1;
}
```

Наступним важливим етапом є відстеження кліків миші. Для цього створимо окремий слухач подій, який буде фіксувати всі натискання кнопок миші — як одиночні, так і подвійні. Для одиночних кліків будемо зберігати час події, що дозволить аналізувати швидкість виконання дій користувачем. Окрему увагу приділимо подвійним клікам, які є важливим індикатором активності користувача.

Лістинг 2.5

```
document.addEventListener("click", function(event) {
    if (clickCount === 0){
        clickTime1 = Date.now();
        clickCount += 1;
    }else{
        clickTime2 = Date.now();
        dblClick = clickTime2 - clickTime1;
        clickCount = 0;
    }
});
```

У випадку подвійного кліку ми будемо зберігати час між двома кліками, що дає можливість оцінити, наскільки швидко користувач виконує подвійний клік, а отже — визначити його рівень досвіду або звички у використанні інтерфейсу. Зібрані дані допоможуть виявити можливі аномалії у поведінці (наприклад, якщо час між кліками занадто великий або занадто малий, це може вказувати на технічні проблеми або проблеми з навчанням користувача).

Лістинг 2.6

```
document.addEventListener("dblclick", function(event) {
   doubleClick.push(dblClick);
});
```

Паралельно з рухом миші ми будемо також зберігати дані про взаємодію користувача з клавіатурою. Це включатиме фіксацію часу натискання окремих клавіш, таких як Shift, Ctrl, Alt, а також тривалість затискання клавіші або час між натисканням і відпусканням клавіші.

Лістинг 2.7

```
document.addEventListener("keydown", function(event) {
   if (textStart === 0){
      textStart = Date.now(); // запам’ятовуємо час початку вводу тексту
   }
   if (event.key === 'Shift'){
      keyDownS = Date.now(); // запам’ятовуємо час натискання Shift
   }else if (event.key === 'Ctrl'){
      keyDownC = Date.now(); // запам’ятовуємо час натискання Ctrl
   }else if (event.key === 'Alt'){
      keyDownA = Date.now(); // запам’ятовуємо час натискання Alt
   }else {
```

```
        keyDown = Date.now(); // запам’ятовуємо час іншої клавіші
    }
    if (keyUp !== 0 && Date.now() - keyUp < 1000) {
        search.push(Date.now() – keyUp);
        keyUp = 0;
    }
    if (keyUp !== 0 && Date.now() - keyUp > 1000){
        text.push(Date.now() – textStart);
        textStart = 0;
    }
});
document.addEventListener("keyup", function(event) {
    keyUp = Date.now();
    if (keyDown !== 0) {
        tap.push(Date.now() – keyDown);
        keyDown = 0;
    }
    if (keyDownS !== 0 && event.key === 'Shift') {
        shifts.push(Date.now() – keyDownS);
        keyDownS = 0;
    }
    if (keyDownC !== 0 && event.key === 'Ctrl') {
        shifts.push(Date.now() – keyDownC);
        keyDownC = 0;
    }
    if (keyDownA !== 0 && event.key === 'Alt') {
        shifts.push(Date.now() – keyDownA);
        keyDownA = 0;
    }
});
```

Це дасть нам змогу проаналізувати, як активно користувач взаємодіє з клавіатурою, чи використовує комбінації клавіш або здійснює часті перемикання між різними функціями. Наприклад, довге затискання клавіші може вказувати на її використання для активації певної функції або скорочення. Ці дані можуть бути важливими для аналізу звичок користувача, а також для покращення персоналізації — наприклад, для автоматичного налаштування інтерфейсу або додавання підказок для нових користувачів.

Для того щоб зберегти ресурси і не перевантажувати сервер надмірною кількістю даних, будемо оптимізувати обсяг інформації, яку відправляємо на сервер. Один із способів зменшення обсягу даних — це обчислення середніх значень кожного зібраного масиву характеристик. Для цього будемо створювати функцію, яка буде вираховувати середні значення по кожному параметру (наприклад, середня швидкість руху миші, середній час натискання клавіші, середня кількість різких змін напрямку). Це дозволить скоротити обсяг переданої інформації, зберігаючи при цьому достатньо важливих характеристик для аналізу. Такий підхід дозволяє мінімізувати навантаження на сервер та мережу, забезпечуючи при цьому ефективну обробку даних.

Лістинг 2.8

```
function AVG(arr) {
   if (arr.length === 0) return 0;
   let sum = arr.reduce((acc, val) => acc + val, 0);
   return sum / arr.length;
}
```

Щоб не перевантажувати систему постійним обміном даними, будемо здійснювати відправку оброблених характеристик на сервер з певним інтервалом. Наприклад, кожні 5 хвилин будуть передаватися агреговані дані про рух миші, натискання клавіш і кліки. Це дозволить серверу отримувати достатньо інформації для обробки та аналізу поведінки користувачів без зайвих затримок

або переписів. Використання такого інтервалу також дозволить виконувати аналіз в реальному часі, створюючи динамічні профілі користувачів і адаптуючи інтерфейс або рекомендації до їх поведінки.

Лістинг 2.9

```
setInterval(function() {
    let data = {
        mouse: {
            speed: AVG(speed).toFixed(3),
            k: AVG(kBtw2Dots).toFixed(3),
            gerz: gerz,
            doubleClick: AVG(doubleClick).toFixed(3)
        },
        keyboard: {
            tap: AVG(tap).toFixed(3),
            shift: AVG(shifts).toFixed(3),
            search: AVG(search).toFixed(3),
            time: AVG(text).toFixed(3)
        }
    };
    let jsonData = JSON.stringify(data);
    hiddenInput.value = jsonData;
}, 300000);
```

2.4 Аналіз отриманого датасету

Початкові дані, що надходять на вхід системи, включають різноманітні параметри, які описують взаємодію користувача з комп'ютерними пристроями. Зокрема, це швидкість руху миші, коефіцієнт ефективності її руху, частота змін напряму, подвійний клік, а також деталі про взаємодію з клавіатурою: швидкість набору тексту, час затримки між натисканнями клавіші shift, ctrl, alt і час, витрачений на пошук наступної літери. Ці параметри формують комплексну картину взаємодії користувача з комп'ютером, що може бути корисно для визначення певних характеристик, таких як стать та вік користувача, на основі аналізу його поведінки.

Для того, щоб навчити штучний інтелект на цих даних, необхідно привести їх до зручного формату, який полегшить обробку та забезпечить швидкість аналізу. У випадку з наведеними даними, формат, що використовується для навчання, має бути значно спрощений, прибравши всі зайві структурні елементи. Це означає, що на вхід повинні подаватись лише числа у вигляді однотипних значень, без вкладених об'єктів або складних ієрархій. Наприклад, дані про швидкість миші, коефіцієнт ефективності, час затримки та інші параметри мають бути представлені як один рядок з числовими значеннями, що дозволяє системі легше обробляти ці вхідні дані.

Щоб підготувати дані до навчання, їх перетворюють у формат, де кожен параметр розташований в одному рядку без ієрархічних структур. Приклад такого перетворення виглядає так: "220.630,0.038,6,0.000,71.678,0.000,205.088,2211.922,24", де всі числові значення та вік користувача (24 роки) об'єднані в один рядок. Для моделі, яка повинна визначати стать, замість віку вказується значення "m" для чоловічої статі або "f" для жіночої. Наприклад: "220.630,0.038,6,0.000,71.678,0.000,205.088,2211.922,m". Такий підхід значно

спрощує обробку даних і дає змогу штучному інтелекту ефективніше використовувати ці дані для навчання.

Зменшення даних до спрощеного формату дозволяє суттєво покращити швидкість обробки та навчання моделі, оскільки складні структури, які містять багато мета-даних або додаткової інформації, значно уповільнюють процес. Спрощення даних до простих числових значень забезпечує більш ефективне використання обчислювальних ресурсів, що знижує час на підготовку і сам процес навчання. Крім того, такий формат дозволяє моделі зосередитися виключно на важливих характеристиках, таких як швидкість миші чи час набору тексту, що є найбільш релевантними для визначення таких параметрів, як стать або вік.

В результаті спрощення даних зменшується кількість можливих помилок, а також зростає точність передбачень, оскільки модель отримує чіткі, стандартизовані значення, що не потребують додаткової обробки або аналізу. Цей підхід також полегшує інтеграцію нових даних та масштабування системи, дозволяючи застосовувати її для більш широкого кола завдань. Таким чином, спрощення та стандартизація даних не тільки пришвидшують навчання штучного інтелекту, але й забезпечують більш точні та стабільні результати.

2.5 Альтернативний підхід до вирішення завдання

Нечіткологічний підхід до формування вихідних даних для визначення віку та статі дозволяє створювати більш гнучкі та адаптивні системи, які можуть враховувати неповні, неточні або не зовсім чіткі дані, що надходять з різних джерел. У контексті визначення таких характеристик, як стать або вік, ці правила можуть бути створені на основі досвіду або експертних знань, що враховують типові тенденції поведінки користувачів. При цьому нечіткість дозволяє побудувати систему, яка може зробити припущення навіть у разі наявності

"незрозумілих" або суперечливих даних. Ключова перевага такого підходу — це можливість працювати з нечисловими або нечіткими вхідними параметрами, що неможливо без втрати точності у традиційних моделях.

Використання нечіткого логічного підходу для визначення статі може ґрунтуватися на аналізі типових патернів поведінки користувачів, таких як швидкість набору тексту, частота натискання певних клавіш або рух мишки. В основі правил можуть бути не тільки статистичні дані, але й знання про те, як різні гендери зазвичай взаємодіють з пристроями. Наприклад, чоловіки можуть бути більш схильними до швидшого набору тексту або мати інший стиль взаємодії з інтерфейсом. Оскільки нечіткий підхід дозволяє працювати з "розмитими" даними, такі правила можуть враховувати різні варіації у поведінці, зокрема, коли це стосується критеріїв, які не є чітко визначеними або суворо певними.

Проаналізувавши 200 перших зібраних записів про рух миші та використання клавіатури, можна сформулювати низку важливих правил, які дозволяють з'ясувати звички користувачів при взаємодії з інтерфейсом та виокремити певні патерни поведінки, які характерні для відвідувачів різних вікових груп та статі.

Для визначення статі було сформовано наступні 5 правил:

- Якщо швидкість набору тексту більше 80 символів за хвилину, то ймовірність того, що користувач — чоловік, більше 70%.
- Якщо швидкість руху мишки перевищує 300 пікселів за секунду і частота змін напряму більше 5, то ймовірність чоловічої статі — 60%.
- Якщо користувач часто натискає клавіші "shift" і "ctrl" в комбінаціях з літерами, це вказує на більш технічну поведінку, що, з ймовірністю 65%, може бути ознакою чоловічої статі.
- Якщо користувач витрачає більше 2 секунд на пошук літери на екрані, це може свідчити про те, що користувач — жінка, із ймовірністю 50%.

- Якщо під час набору тексту є кілька затримок понад 1 секунду між натисканнями, це може вказувати на жіночу поведінку, і ймовірність становить 55%.

Визначення віку за допомогою правил нечіткої логіки здійснювалось на основі категоризації вікових груп, де були визначені наступні категорії: дитина (менше 12 років), підліток (від 13 до 19 років), дорослий (від 20 до 39 років), середнього віку (від 40 до 59 років) та літнього віку (від 60 років і більше). Кожна з цих категорій відображала діапазон віку, до якого належить користувач, з урахуванням нечітких меж між групами, що дозволяло більш гнучко та адаптивно визначати відповідний вік. Для визначення віку було створено наступні правила:

- Якщо швидкість набору тексту більше 100 символів за хвилину, то ймовірність того, що користувач молодший за 30 років, становить 80%.
- Якщо час пошуку наступної літери більше 2 секунд, то ймовірність того, що користувач старший за 50 років, становить 70%.
- Якщо рух миші плавний і без різких змін напряму, то ймовірність того, що користувач старший за 40 років, становить 60%.
- Якщо під час набору тексту використовуються комбінації клавіш, типові для швидкого доступу до системних функцій, ймовірність того, що користувач молодший за 25 років, становить 75%.
- Якщо частота подвійних кліків перевищує 2 кліки на секунду, то ймовірність того, що користувач молодший за 35 років, становить 65%.
- Якщо користувач витрачає більше 3 секунд на перегляд елементів на екрані, то ймовірність того, що він старший за 45 років, становить 50%.
- Якщо швидкість руху мишки низька і частота змін напрямку рідка, ймовірність того, що користувач старший за 60 років, становить 70%.

Нечіткий підхід дозволяє комбінувати ці правила, використовуючи операції типу "якщо-то", але з урахуванням нечітких значень (наприклад, "більше" або "менше" ніж певний поріг). Це дозволяє системі робити м'якші висновки і враховувати варіативність у поведінці користувачів, не орієнтуючись

на жорстко визначені межі. Такі правила можуть бути налаштовані так, щоб максимізувати точність результату, поки ці правила не почнуть поступатися в точності моделям штучного інтелекту.

Під час додавання нових записів проводилась перевірка та формування статистики, і коли їх кількість досягла 400, було виявлено, що точність прогнозування статі склала 70%, а віку — 58%. Це свідчить про те, що правила, розроблені на основі аналізу первинних даних, здатні з певною мірою успішності визначати стать користувачів, проте їх здатність передбачати вік є менш точною.

На відміну від нечіткої логіки, онтологічний підхід спрямований на створення чітких правил порівняння вхідних даних для визначення віку та статі користувачів. Онтологія забезпечує формалізацію понять, таких як "швидкість руху миші", "тип натискання клавіатури" або "частота взаємодії", що дозволяє чітко визначати, які саме патерни поведінки характерні для різних вікових або статевих груп. Хоча структура правил в онтологічному підході залишатиметься подібною до нечіткої логіки, ключова відмінність полягає у більш строгому та формальному визначенні критеріїв, що використовуються для прийняття рішень.

У рамках онтологічного підходу зміни в правилах, як правило, стосуються визначення обмежень для вірогідності логічних висновків, які впливають на точність класифікації користувача. Наприклад, якщо для нечіткої логіки досить вказати, що "швидкість руху миші може бути великою або помірною", то в онтології це правило має більш строгі параметри, такі як конкретні межі швидкості та чіткі критерії для визначення того, який користувач належить до певної групи. Цей підхід дозволяє більш точно порівнювати дані та приймати логічно обґрунтовані рішення, що сприяє високій точності у визначенні статі та віку.

Таким чином, онтологічний підхід надає можливість не тільки створити чіткі правила для порівняння даних, але й адаптувати їх до конкретних умов, що виникають під час взаємодії користувачів з вебсайтами. Хоча правила в обох підходах можуть бути схожими, основна різниця полягає в ступені деталізації і точності цих правил. Онтологія дозволяє створювати більш структуровані,

прогнозовані та інтерпретовані моделі, що в кінцевому підсумку підвищує ефективність і точність системи рекомендацій.

В той же час, онтологічний підхід може бути неефективним у системах, де дані мають високу взаємозалежність або суттєву варіативність, що може призвести до неправильних або занадто спрощених висновків. Якщо, наприклад, система формулює правило, що швидкість набору тексту понад 80 символів за хвилину свідчить про те, що користувач — чоловік, це правило може не враховувати індивідуальні відмінності, наприклад, коли жінка здатна друкувати швидше, ніж це передбачає правило. В такому випадку онтологія, яка базується на чітких категоріях і визначеннях, може призвести до помилок через свою неадаптивність до подібних винятків. Вона не враховує можливість перекриття або перетину між різними класами даних, що вимагає більш гнучкого підходу, здатного враховувати непередбачувані зміни поведінки або зовнішні фактори. Тому для таких ситуацій, де дані можуть бути складними та суперечливими, краще застосовувати методи, які дозволяють працювати з невизначеністю та варіативністю, як-от нечітка логіка або машинне навчання.

З часом, коли точність прогнозування за допомогою нечітких правил буде порівняно з точністю прогнозування за допомогою машинного навчання, можна почати порівнювати ефективність цих підходів. Це дозволить визначити, на яких етапах навчання ШІ можуть бути перевершені за точністю нечіткі правила, і коли потрібно почати використовувати більш складні моделі штучного інтелекту, які можуть працювати з величезними обсягами даних і складними взаємозв'язками.

2.6 Тренування моделі машинного навчання

Щоб почати працювати з моделями машинного навчання, спершу потрібно встановити необхідні бібліотеки. Для цього в даному випадку використовуються TensorFlow для побудови нейронних мереж та Scikeras для

інтеграції з бібліотекою Scikit-learn, яка дозволяє проводити пошук по параметрах моделі. Команди для встановлення цих бібліотек виглядають так:

Лістинг 2.10

```
!pip install --upgrade tensorflow
!pip install scikeras
```

Ці команди забезпечують встановлення та оновлення TensorFlow та Scikeras до останніх доступних версій, що є необхідним для подальшої роботи.

Після встановлення бібліотек потрібно підключити їх до середовища для подальшої роботи. Для цього імпортуємо необхідні модулі:

Лістинг 2.11

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Input
from scikeras.wrappers import KerasClassifier
from sklearn.model_selection import GridSearchCV
```

Цей код підключає NumPy для роботи з масивами даних, Keras для побудови нейронних мереж, а також Scikit-learn для використання методів оптимізації параметрів та пошуку по сітці (Grid Search).

Наступний крок — зчитування даних. Для цього відкривається файл clear_data.txt, в якому зберігаються вхідні дані. Кожен запис у файлі розділений комами, де останні два значення — це стать та вік користувача. Після зчитування даних, вони перетворюються в масиви чисел, де «m» та «f» замінюються на 1 та 0 відповідно. Інші значення залишаються числовими.

Лістинг 2.12

```
with open('/content/clear_data.txt', 'r') as file:
```

```
  lines = file.readlines()
data = []
for line in lines:
  variables = line.strip().split(',')
  numbers = [float(1) if var == 'm' else float(0) if var == 'f' else float(var) for var in variables]
  data.append(numbers)
```

Далі, розбиваємо дані на вхідні характеристики X та цільові змінні Y та Y2:

Лістинг 2.13

```
X = []
Y = []
Y2 = []
for dat in data:
  X.append(dat[:8])
  Y.append(dat[8])
  Y2.append(dat[9])
X = np.array(X).T
Y = np.array(Y)
Y2 = np.array(Y2)
```

Для покращення роботи нейронної мережі, дані потрібно масштабувати та нормалізувати. Це забезпечить стабільність та швидкість навчання. У даному випадку для Y2 (віку) здійснюється нормалізація значень, а для X (вхідних ознак) виконується їх масштабування:

Лістинг 2.14

```
min_Y2 = min(Y2)
```

```
diff_Y2 = max(Y2) - min_Y2
for i in range(Y2.shape[0]):
  Y2[i] = (Y2[i] - min_Y2) / diff_Y2
for i in range(X.shape[0]):
  max_X = max(abs(X[i]))
  for j in range(X.shape[1]):
    X[i][j] = abs(X[i][j]/max_X)
```

Далі дані розділяються на тренувальні та тестові набори, де тренувальний набір складає 200 записів:

Лістинг 2.15

```
train_x = X.T[:200].T
train_y = Y[:200]
train_y2 = Y2[:200]
train_y = train_y.reshape((1, train_y.shape[0]))
train_y2 = train_y2.reshape((1, train_y2.shape[0]))
test_x = X.T[200:].T
test_y = Y[200:]
test_y2 = Y2[200:]
test_y = test_y.reshape((1, test_y.shape[0]))
test_y2 = test_y2.reshape((1, test_y2.shape[0]))
```

Для створення моделі використовуємо бібліотеку Keras. Модель складається з кількох шарів, де перший шар має 64 нейрони з функцією активації ReLU, другий — 32 нейрони з такою ж активацією, а вихідний шар — один нейрон з активацією sigmoid для задачі бінарної класифікації.

Лістинг 2.16

```
def create_model(optimizer='adam', dropout_rate=0.0, learning_rate=0.001):
```

```
    model = Sequential()
    model.add(InputLayer(input_shape=(8,)))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(32, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    if optimizer == 'adam':
        optimizer = Adam(learning_rate=learning_rate)
    elif optimizer == 'sgd':
        optimizer = SGD(learning_rate=learning_rate)
    elif optimizer == 'rmsprop':
        optimizer = RMSprop(learning_rate=learning_rate)
    elif optimizer == 'nadam':
        optimizer = Nadam(learning_rate=learning_rate)
    elif optimizer == 'adagrad':
        optimizer = Adagrad(learning_rate=learning_rate)
    model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])
    return model
```

Модель необхідно обгорнути в клас KerasClassifier, щоб інтегрувати її з бібліотекою Scikit-learn для подальшої оптимізації та пошуку по параметрах.

Для покращення роботи моделі використовувався пошук по параметрах (Grid Search) для визначення оптимальних значень для таких параметрів, як розмір пакета (batch_size), кількість епох (epochs), оптимізатор (optimizer), швидкість навчання (learning_rate) та регуляризація через dropout (dropout_rate). Для більшої варіативності було збільшено кількість значень у параметрах пошуку. Крім того, використовувалось 3 фолда для кросвальдаційного тестування, що дозволило забезпечити більш стабільну оцінку моделі.

Лістинг 2.17

```
model = KerasClassifier(build_fn=create_model, verbose=0)
param_grid = {
   'batch_size': [4, 8, 16, 32, 64, 128],
   'epochs': [10, 20, 30, 40, 50, 60, 70],
   'optimizer': ['adam', 'sgd', 'rmsprop', 'nadam', 'adagrad'],
   'learning_rate': [0.001, 0.01, 0.1, 0.0001],
   'dropout_rate': [0.0, 0.2, 0.3, 0.4]
}
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=3, scoring='accuracy')
grid_result = grid.fit(train_x.T, train_y.T)
print("Best Parameters:", grid_result.best_params_)
```

Після проведення пошуку найкращих параметрів за допомогою методу GridSearchCV, було отримано оптимальні значення batch_size = 16, epochs = 50, optimizer = 'adam'.

Ці параметри виявилися найбільш ефективними для даної задачі, оскільки оптимізатор "adam" зазвичай працює добре при різних умовах і є здатним до швидкої збіжності. Кількість епох була вибрана оптимальною, оскільки модель не потребує надмірного навчання для досягнення стабільних результатів, а batch size був підібраний так, щоб забезпечити баланс між швидкістю навчання та стабільністю оновлень ваг.

Після вибору найкращих параметрів, ми використовуємо їх для компіляції моделі. Компіляція моделі відбувається за допомогою оптимізатора "adam" та функції втрат binary_crossentropy, яка підходить для задач бінарної класифікації. Також ми визначаємо метрику accuracy для оцінки точності моделі під час навчання.

Наступним кроком є навчання моделі з використанням вибраних параметрів. Для цього ми задаємо кількість епох, розмір партії (batch size) та

використовуємо 10% даних для валідації, щоб перевіряти точність моделі на кожному етапі навчання.

Лістинг 2.18

```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
model.fit(train_x.T, train_y2.T, epochs=50, batch_size=16, validation_split=0.2)
```

Для того, щоб зберегти всю модель (архітектуру, ваги та компіляцію), можна використовувати метод save(). Це дозволить швидко завантажити модель на сервер за допомогою методу load_model().

Лістинг 2.19

```
model.save('my_model.h5')
```

Спершу навчання проходило на 200 записах, але точність не була задовільною, а потім на 600. Навчання на 200 записах давало результати для віку — 43%, для статі — 58%, а при використанні 600 записів точність підвищується: для віку — 54%, для статі — 71%. Це вказує на те, що модель потребує донавчання, оскільки точність все ще недостатньо висока для практичного використання.

Необхідність донавчання моделі пояснюється тим, що з більшою кількістю даних модель здатна виявляти більш складні патерни у поведінці користувачів, що покращує точність. Водночас для підвищення точності можна використовувати правила нечіткої логіки, які дозволяють враховувати невизначеність у даних та коригувати прогнози в умовах нечітких або суперечливих результатів, що характерно для задач, пов'язаних з поведінкою користувачів.

2.7 Формування та повернення рекомендаційного списку

Оскільки дані про вік та стать відвідувачів сайту є чутливими та потребують захисту відповідно до законодавства про захист персональних даних, було прийнято рішення змінити підхід до персоналізації на сайті. Замість того, щоб прямо визначати стать та вік користувачів, було вирішено використовувати більш загальні підходи, щоб зберегти конфіденційність і уникнути порушень. Одним із таких підходів стало формування списку рекомендацій на основі поведінки користувача та його переваг. Наприклад, аналізуючи рухи миші, кліки, швидкість набору тексту та інші ознаки, можна зробити висновки про тип користувача без прямого розкриття його особистої інформації.

Одним з таких рішень є надання персоналізованих рекомендацій товарів, зокрема фармацевтичних продуктів, які можуть бути корисними для різних вікових груп. Наприклад, для молодших користувачів, таких як підлітки або молодь, можна рекомендувати вітаміни та добавки для підтримки імунної системи, засоби для покращення шкіри, а також препарати, які сприяють здоровому розвитку в період активного росту. Для середнього віку, наприклад, для осіб від 30 до 50 років, рекомендації можуть включати препарати для підтримки серцево-судинної системи, комплекси для зниження стресу або засоби для поліпшення енергії та концентрації.

Для старших вікових груп, від 50 років і старше, особливу увагу слід приділяти препаратам, що підтримують здоров'я суглобів, артеріальний тиск, а також засобам для боротьби з хронічними захворюваннями, такими як діабет або остеопороз. Рекомендації для цієї категорії можуть включати вітаміни D та кальцій для підтримки здоров'я кісток, препарати для нормалізації рівня холестерину та антиоксиданти для уповільнення процесів старіння.

Такий підхід дозволяє не лише надавати релевантні пропозиції, але й зберігати конфіденційність користувачів. Це дозволяє максимально точно передбачити потреби користувача, зберігаючи при цьому його конфіденційність.

Цей спосіб надання рекомендацій товарів, що базуються на аналізі поведінки користувача, був реалізований у співпраці з компанією ПрАТ "Аптеки Запоріжжя". Компанія є одним з лідерів на фармацевтичному ринку України, з багаторічним досвідом у наданні фармацевтичних послуг та рекомендацій. Спільно з експертами компанії, було розроблено набір рекомендацій для різних вікових груп, що допомагає ефективно підібрати продукти, спрямовані на покращення здоров’я користувачів.

Завдяки партнерству з ПрАТ "Аптеки Запоріжжя", користувачі можуть отримувати точні, науково обґрунтовані рекомендації, орієнтовані на їхній вік і потенційні потреби у фармацевтичних засобах. Це дозволяє не лише покращити якість життя користувачів, але й підвищити їхню довіру до онлайн-платформи. Такі рекомендації допомагають користувачам більш свідомо підходити до вибору лікарських засобів та вітамінів, оскільки вони базуються на персоналізованих даних про їх поведінку.

Також важливим аспектом співпраці стало дотримання високих стандартів конфіденційності. Всі дані користувачів обробляються анонімно, без розкриття особистої інформації, що відповідає вимогам чинного законодавства про захист персональних даних. Крім того, партнерство з ПрАТ "Аптеки Запоріжжя" гарантує, що рекомендації не тільки є персоналізованими, але й надаються на основі рекомендацій лікарів та фармацевтів, що гарантує їх високу якість.

ВИСНОВКИ

Розробка системи розпізнавання віку та статі відвідувачів на основі патернів поведінки виявилася успішною і відповідає актуальним вимогам сучасної веб-аналітики. Використання методів глибокого навчання для аналізу руху миші та частоти використання клавіатури дозволяє досягти високої точності в розпізнаванні віку та статі користувачів без необхідності збору традиційних особистих даних.

Ідея використання патернів поведінки для визначення демографічних характеристик відвідувачів є інноваційною і має значний потенціал для застосування в різних сферах. Система, розроблена в рамках цієї роботи, здатна безпосередньо адаптувати контент вебсайтів до характеристик користувачів, що значно підвищує рівень персоналізації.

Методи Deep Learning, що були застосовані в дослідженні, продемонстрували свою ефективність при роботі з великими масивами даних. Використання цих методів дозволяє не тільки аналізувати поведінку відвідувачів, але й прогнозувати їхні можливі демографічні характеристики, що відкриває нові можливості для персоналізації користувацького досвіду на сайтах.

Розроблена система здатна працювати без збору особистих даних про користувачів, що відповідає сучасним вимогам конфіденційності і захисту персональних даних. Вона використовує лише інформацію про поведінку відвідувача на сайті, що значно знижує ризики порушення приватності користувачів.

Збір та обробка даних для навчання системи здійснювався шляхом аналізу взаємодії користувача з веб-сторінками через рухи миші та натискання клавіш. Цей підхід дозволяє створити ефективну базу для навчання штучного інтелекту, що є основою для подальшого розвитку системи та її вдосконалення.

Результати тестування моделі показали високу точність у визначенні віку та статі відвідувачів на основі поведінкових патернів. Це підтверджує практичну

цінність розробленої системи для аналізу користувацької взаємодії з веб-ресурсами без необхідності додаткових витрат на отримання традиційних персональних даних.

Універсальність системи дозволяє застосовувати її не тільки на невеликих вебсайтах, але й на великих платформах, таких як онлайн-магазини або ритейлери. Це робить її корисним інструментом для компаній, що хочуть персоналізувати досвід своїх користувачів і підвищити рівень залучення.

Програмні засоби, які використовувалися для створення системи (JavaScript, Python), забезпечують високу гнучкість і сумісність з різними веб-технологіями, що дозволяє легко інтегрувати систему в існуючі онлайн-платформи.

Важливим результатом роботи є створення технології, яка дозволяє за мінімальних витрат збирати дані про користувачів, здійснювати їх аналіз і на основі цього робити висновки про їхні характеристики, такі як вік та стать. Це дозволяє значно спростити процес персоналізації контенту на веб-ресурсах.

Практична значущість розробленої системи полягає в тому, що вона може бути використана для покращення користувацького досвіду. Наприклад, розпізнавання статі та віку відвідувачів дозволяє сайтам адаптувати контент під конкретну аудиторію, що може сприяти підвищенню рівня конверсії та залучення клієнтів.

Недоліками системи є необхідність наявності достатньої кількості даних для навчання моделі, а також можливі неточності у розпізнаванні, якщо користувачі мають незвичні патерни поведінки. Однак ці проблеми можуть бути вирішені шляхом подальшого вдосконалення алгоритмів і збирання більшої кількості даних.

Подальші напрямки розвитку дослідження включають вдосконалення моделі для визначення інших демографічних характеристик, таких як соціальний статус або інтереси користувачів, а також оптимізацію роботи системи для роботи з мобільними пристроями, що дозволить зробити технологію ще більш універсальною та доступною для широкого кола користувачів.

Інтеграція розробленої системи в реальні веб-платформи може значно покращити стратегії маркетингу та таргетування реклами. Використання патернів поведінки для розпізнавання віку та статі дає змогу більш точно визначати цільову аудиторію і пропонувати їй персоналізовані рекламні кампанії. Це підвищує ефективність рекламних витрат і дозволяє бізнесам краще взаємодіяти з потенційними клієнтами, забезпечуючи їм більш релевантний контент.

Загалом, робота демонструє потенціал використання штучного інтелекту та глибокого навчання для вирішення задач аналізу поведінки користувачів в реальному часі. Можливість розпізнавання віку та статі на основі простих даних про поведінку користувача відкриває нові горизонти в сфері персоналізації веб-досвіду та оптимізації взаємодії з відвідувачами сайтів, що, в свою чергу, сприяє розвитку більш ефективних і зручних онлайн-сервісів.

ПЕРЕЛІК ПОСИЛАНЬ

1. **Pental A.** Predicting Age and Gender by Keystroke Dynamics and Mouse Patterns [Текст] / Pental A. // Conference: 25th Conference on User Modeling, Adaptation and Personalization. – 2017. – 11 p.

2. Загальний регламент про захист даних (GDPR) [Електронний ресурс]. – Режим доступу: https://gdpr-text.com/uk/ (дата звернення: 21.12.2024)

3. Google Marketing Platform Overview [Електронний ресурс]. – Режим доступу: https://sites.google.com/view/login-now-access/access?gad_source=1&gclid=CjwKCAiAgoq7BhBxEiwAVcW0LMdL43Hua6gU_h2sLkbgaF1e1bHnCkMQEdBS-drk5wddf-BZ-fYlDBoC8oEQAvD_BwE (дата звернення: 21.12.2024)

4. Netflix [Електронний ресурс] – Режим доступу: https://www.netflix.com/ua/ (дата звернення: 21.12.2024)

5. Learn about Amazon [Електронний ресурс] – Режим доступу: https://www.aboutamazon.com/?utm_source=gateway&utm_medium=footer (дата звернення: 21.12.2024)

6. Сайт офіційного постачальника та піар-компанії Fitbit в Україні [Електронний ресурс] – Режим доступу: https://icases.ua/ua/brand/fitbit (дата звернення: 21.12.2024)

7. Facebook: історія створення та успіху Фейсбук [Електронний ресурс] – Режим доступу: https://worldbank.org.ua/4685-facebook.html (дата звернення: 21.12.2024)

8. Переваги вивчення JavaScript [Електронний ресурс] – Режим доступу: https://formula.kr.ua/tsikavinki/perevagi-vivchennya-javascript.html (дата звернення: 21.12.2024)

9. **Блазнов, В. М.** Модифікований метод машинного навчання для обробки даних в мережі MicroGrid [Текст] / В. М. Блазнов. – Київ, 2020. – 80 с.

10. **Swamynathan, M.** Mastering Machine Learning with Python in Six Steps [Текст] / S. Manohar. – CA: Apress Berkeley, 2017. – 358 p.

11. **Duda, R. O.** Pattern Classification [Текст] / R. O. Duda, P. E. Hart, D. G. Stork. – Wiley, 2001. – 654 p.

12. **Kim, P.** MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence [Текст] / P. Kim – CA: Apress Berkeley, 2017. – 151 p.

13. **Литвин, В.В.** Бази знань інтелектуальних систем підтримки прийняття рішень [Текст] / В.В. Литвин. – Львів: Видавництво Львівської політехніки, 2011. – 240 с.

14. **Оробчук, О.** Поняття онтології у філософії та науках про штучний інтелект: порівняльний аналіз [Текст] / О. Оробчук – Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2023. – 50 с.

15. **Aurélien, G.** Hands-On Machine Learning with ScikitLearn and TensorFlow [Текст] / A. Géron – O'Reilly Media, 2017. – 751 p.

ДОДАТОК А.

Програмні коди

```
document.addEventListener("mousemove", function(event) {
   if (Date.now() - lastMouseMove >= 25) {
      [код запису і обробки подій]
      lastMouseMove = Date.now();
   }
});

if (dots.length < 3) {
   dots.unshift(new Dots(event.clientX, event.clientY, Date.now()));
} else {
   dots.pop();
   dots.unshift(new Dots(event.clientX, event.clientY, Date.now()));
}

if (dots.length === 3) {
   let dist1 = Math.sqrt(Math.pow(dots[2].x - dots[1].x, 2) +
Math.pow(dots[2].y - dots[1].y, 2));
   let dist2 = Math.sqrt(Math.pow(dots[1].x - dots[0].x, 2) +
Math.pow(dots[1].y - dots[0].y, 2));
   let dist3 = Math.sqrt(Math.pow(dots[2].x - dots[0].x, 2) +
Math.pow(dots[2].y - dots[0].y, 2));
   let timeDiff = (dots[0].time - dots[2].time) / 1000;

   if (timeDiff > 0 && dist1 > 0 && dist2 > 0 && dist3 > 0) {
      speed.push((dist1 + dist2) / timeDiff);
      kBtw2Dots.push((dist1 + dist2) / dist3 – 1);
   }
```

```
}

if ((dots[0].x < dots[1].x && dots[0].x > dots[2].x) || (dots[0].y < dots[1].y &&
dots[0].y > dots[2].y) || (dots[0].x < dots[2].x && dots[0].x > dots[1].x) || (dots[0].y <
dots[2].y && dots[0].y > dots[1].y)) {
   gerz += 1;
}

document.addEventListener("click", function(event) {
   if (clickCount === 0){
      clickTime1 = Date.now();
      clickCount += 1;
   }else{
      clickTime2 = Date.now();
      dblClick = clickTime2 - clickTime1;
      clickCount = 0;
   }
});
document.addEventListener("dblclick", function(event) {
   doubleClick.push(dblClick);
});

document.addEventListener("keydown", function(event) {
   if (textStart === 0){
      textStart = Date.now(); // запам’ятовуємо час початку вводу тексту
   }
   if (event.key === 'Shift'){
      keyDownS = Date.now(); // запам’ятовуємо час натискання Shift
   }else if (event.key === 'Ctrl'){
      keyDownC = Date.now(); // запам’ятовуємо час натискання Ctrl
```

```
    }else if (event.key === 'Alt'){
        keyDownA = Date.now(); // запам’ятовуємо час натискання Alt
    }else {
        keyDown = Date.now(); // запам’ятовуємо час іншої клавіші
    }
    if (keyUp !== 0 && Date.now() - keyUp < 1000) {
        search.push(Date.now() – keyUp);
        keyUp = 0;
    }
    if (keyUp !== 0 && Date.now() - keyUp > 1000){
        text.push(Date.now() – textStart);
        textStart = 0;
    }
});

document.addEventListener("keyup", function(event) {
    keyUp = Date.now();
    if (keyDown !== 0) {
        tap.push(Date.now() – keyDown);
        keyDown = 0;
    }
    if (keyDownS !== 0 && event.key === 'Shift') {
        shifts.push(Date.now() – keyDownS);
        keyDownS = 0;
    }
    if (keyDownC !== 0 && event.key === 'Ctrl') {
        shifts.push(Date.now() – keyDownC);
        keyDownC = 0;
    }
    if (keyDownA !== 0 && event.key === 'Alt') {
```

```
        shifts.push(Date.now() – keyDownA);
        keyDownA = 0;
    }
});

function AVG(arr) {
    if (arr.length === 0) return 0;
    let sum = arr.reduce((acc, val) => acc + val, 0);
    return sum / arr.length;
}

setInterval(function() {
    let data = {
        mouse: {
            speed: AVG(speed).toFixed(3),
            k: AVG(kBtw2Dots).toFixed(3),
            gerz: gerz,
            doubleClick: AVG(doubleClick).toFixed(3)
        },
        keyboard: {
            tap: AVG(tap).toFixed(3),
            shift: AVG(shifts).toFixed(3),
            search: AVG(search).toFixed(3),
            time: AVG(text).toFixed(3)
        }
    };
    let jsonData = JSON.stringify(data);
    hiddenInput.value = jsonData;
}, 300000);
```

```
!pip install --upgrade tensorflow
!pip install scikeras
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Input
from scikeras.wrappers import KerasClassifier
from sklearn.model_selection import GridSearchCV

with open('/content/clear_data.txt', 'r') as file:
  lines = file.readlines()
data = []
for line in lines:
  variables = line.strip().split(',')
  numbers = [float(1) if var == 'm' else float(0) if var == 'f' else float(var) for var in variables]
  data.append(numbers)
X = []
Y = []
Y2 = []
for dat in data:
  X.append(dat[:8])
  Y.append(dat[8])
  Y2.append(dat[9])
X = np.array(X).T
Y = np.array(Y)
Y2 = np.array(Y2)

min_Y2 = min(Y2)
diff_Y2 = max(Y2) - min_Y2
for i in range(Y2.shape[0]):
```

```
  Y2[i] = (Y2[i] - min_Y2) / diff_Y2
for i in range(X.shape[0]):
  max_X = max(abs(X[i]))
  for j in range(X.shape[1]):
    X[i][j] = abs(X[i][j]/max_X)

train_x = X.T[:200].T
train_y = Y[:200]
train_y2 = Y2[:200]
train_y = train_y.reshape((1, train_y.shape[0]))
train_y2 = train_y2.reshape((1, train_y2.shape[0]))
test_x = X.T[200:].T
test_y = Y[200:]
test_y2 = Y2[200:]
test_y = test_y.reshape((1, test_y.shape[0]))
test_y2 = test_y2.reshape((1, test_y2.shape[0]))

def create_model(optimizer='adam', dropout_rate=0.0, learning_rate=0.001):
    model = Sequential()
    model.add(InputLayer(input_shape=(8,)))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(32, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    if optimizer == 'adam':
        optimizer = Adam(learning_rate=learning_rate)
    elif optimizer == 'sgd':
        optimizer = SGD(learning_rate=learning_rate)
    elif optimizer == 'rmsprop':
        optimizer = RMSprop(learning_rate=learning_rate)
    elif optimizer == 'nadam':
```

```
            optimizer = Nadam(learning_rate=learning_rate)
        elif optimizer == 'adagrad':
            optimizer = Adagrad(learning_rate=learning_rate)
        model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])
        return model

    model = KerasClassifier(build_fn=create_model, verbose=0)
    param_grid = {
        'batch_size': [4, 8, 16, 32, 64, 128],
        'epochs': [10, 20, 30, 40, 50, 60, 70],
        'optimizer': ['adam', 'sgd', 'rmsprop', 'nadam', 'adagrad'],
        'learning_rate': [0.001, 0.01, 0.1, 0.0001],
        'dropout_rate': [0.0, 0.2, 0.3, 0.4]
    }
    grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=3, scoring='accuracy')
    grid_result = grid.fit(train_x.T, train_y.T)
    print("Best Parameters:", grid_result.best_params_)

    model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
    model.fit(train_x.T, train_y2.T, epochs=50, batch_size=16, validation_split=0.2)
    model.save('my_model.h5')
```