МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ПрАТ «ПВНЗ «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інформаційних технологій

ДО ЗАХИСТУ ДОПУЩЕНА

Зав. кафедри _____

д.е.н., доц. С.І. Левицький

МАГІСТЕРСЬКА ДИПЛОМНА РОБОТА

МОДЕЛЮВАННЯ СИСТЕМИ АНАЛІЗУ ПОВЕДІНКИ КОРИСТУВАЧІВ ІТ ПРОДУКТУ ЗА ДОПОМОГОЮ МЕТОДІВ МАШИННОГО НАВЧАННЯ

Виконала
ст. гр. ІПЗ-312м _____ Н.О. Іванченко

Керівник
завідувач кафедри
інформаційних технологій _____ С.І. Левицький

Запоріжжя

2024 р

ПрАТ «ПВНЗ «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інформаційних технологій

ЗАТВЕРДЖУЮ

Зав. кафедри _____

д.е.н., доц. С.І. Левицький

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ ДИПЛОМНУ РОБОТУ

студенки гр. ІПЗ-312м, спеціальності 121 «Інженерія програмного забезпечення»
ОП «Інженерія програмного забезпечення»

Іванченко Надії Олександрівни

1.Тема: Моделювання системи аналізу поведінки користувачів ІТ продукту за допомогою методів машинного навчання

затверджена наказом по інституту № 02-30 від 20.10.2023 р.

2. Термін здачі студенткою закінченої роботи: 16.12.2023 р.

3. Перелік питань, що підлягають розробці

1. Визначити сутність причинно-поведінкового каркасу для аналізу великих даних

2. Проаналізувати типи аналітики

3. Здійснити огляд інструментів розробки мікросервісу

4. Обрати кластеризацію як інструмент продуктової аналітики

5. Обгрунтувати модель вирішення задач кластеризації

6. Створити мікросервіс для кластеризації користувачів ІТ продукту

7. Проаналізувати отримані результати

8. Оформити звіт за результатами роботи

4. Календарний графік підготовки магістерської дипломної роботи

№ етапу	Зміст	Термін виконання	Готовність по графіку (%), підпис керівника	Підпис керівника про повну готовність етапу, дата
1.	Формулювання теми магістерської дипломної роботи (збір практичного матеріалу за темою магістерської дипломної роботи)	20.10.23		
2.	I атестація I розділ магістерської дипломної роботи	27.10.23		
3.	II атестація II розділ магістерської дипломної роботи	17.11.23		
4.	III атестація III та IV розділ магістерської дипломної роботи, висновки та рекомендації, додатки, реферат, перевірка програмою «Антиплагіат»	16.12.23		
5.	Доопрацювання магістерської дипломної роботи, підготовка презентації, отримання відгуку керівника і рецензії	08.01.23		
6.	Попередній захист магістерської дипломної роботи	09.01.23		
7.	Подача магістерської дипломної роботи на кафедру	за 3 дні до захисту		
8.	Захист магістерської дипломної роботи	17.01.24		

Дата видачі завдання: 05.10.2023 р.

Керівник магістерської роботи _____ С.І. Левицький
(підпис) (прізвище, ініціали)

Завдання отримала до виконання _____ Н.О. Іванченко
(підпис студента) (прізвище та ініціали)

РЕФЕРАТ

Магістерська дипломна робота містить 98 сторінок, 1 таблицю, 51 рисунок, 2 додатки, 38 бібліографічних посилання.

Об’єктом дослідження є використання методів машинного навчання, а саме кластеризації для аналізу поведінки користувачів IT додатку.

Предметом дослідження є науково-методичні і практичні підходи до моделювання і програмування системи аналізу поведінки користувачів IT додатком.

В першому розділі здійснено детальний огляд предметної області. Досліджено причинно-поведінковий каркас для аналізу великих даних. Розглянуті типи сучасної аналітики. Проведено дослідження системи аналізу даних про поведінку користувачів

У другому розділі проведено огляд архітектури мікросервісів та засобів візуалізації. Обрано мову програмування Python для аналізу даних та A/B тест як спосіб визначення реакції користувачів на зміни в IT продукті

В третьому розділі обгрунтовано використання машинного навчання для аналізу даних. Вибрано і описано кластеризацію як інструмент продуктової аналітики. Розгляноуто методи та моделі вирішення задач кластеризації. Наведено приклад алгоритм к-середніх і особливості його реалізації на Python. Реалізазовано кластерний аналіз на прикладі заданої бази даних.

В четвертому розділі наведено концепцію розробки мікросервісів та впроваджено розроблені моделі у аналітичний мікросервіс. Мікросервіс може знаходитись як на локальному хості та і в мережі інтернет, в модель передаються певні параметри клієнта і вона на цих даних формує прогноз, який по суті є номером ймовірного кластеру до якого треба віднести цього клієнта.Показано методи оцінки ефективності впровадження змін у додатку.

ПРОДУКТОВА АНАЛІТИКА, А/Б ТЕСТУВАННЯ, КЛАСТЕРИЗАЦІЯ, МАШИННЕ НАВЧАННЯ, МІКРОСЕРВІС

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Скорочення	**Повна назва**	**Пояснення/переклад**
IDE	Integrated Development Environment	Інтегроване середовище розробки
API	Application programming interface	інтерфейс прикладного програмування
ESB	Enterprise service bus	Корпоративна шина даних
PYPL	PopularitY of Programming Language	Індекс популярності мов програмування
ML	Machine Learning	Машинне навчання

ВСТУП

Важливість великих даних для компанії важко переоцінити у сучасному бізнес-ландшафті. Ось кілька ключових аспектів, чому великі дані є критичним ресурсом для компаній:

Великі дані надають компаніям великі обсяги інформації, на основі яких можна приймати обґрунтовані стратегічні та операційні рішення. Аналіз даних дозволяє виявляти тенденції, розуміти потреби клієнтів, оптимізувати бізнес-процеси та приймати рішення, що ґрунтуються на фактах, а не на припущеннях.

Великі дані дозволяють компаніям глибше розуміти своїх клієнтів. Аналіз поведінки клієнтів, переваг, відгуків та взаємодії з продуктами чи послугами допомагає створювати спеціальні пропозиції та зміцнювати відносини з клієнтами.

З урахуванням цих факторів великі дані стають невід'ємним інструментом для компаній, які прагнуть бути успішними, інноваційними та адаптованими до умов ринку, що швидко змінюються.

Для аналізу великих даних використовують методи машинного навчання, які дозволяючи отримувати цінну інформацію, виявляти закономірності і прогнозувати майбутні тенденції. В магістерській роботі використано один з методів машинного навчання, а саме кластерний аналіз, який дозволив згрупувати користувачів ІТ додатку на основі їхньої поведінки.

Об’єктом дослідження є використання методів машинного навчання, а саме кластеризації для аналізу поведінки користувачів ІТ додатку.

Предметом дослідження є науково-методичні і практичні підходи до моделювання і програмування системи аналізу поведінки користувачів ІТ додатком.

Для досягнення мети були поставлені і вирішені наступні завдання: - визначено сутність причинно-поведінкового каркасу для аналізу великих даних;- проаналізовано типи аналітики; - здійснено огляд інструментів розробки мікросервісу; - обрано кластеризацію як інструмент продуктової аналітики; - обгрунтувано модель вирішення задач кластеризації; - створено мікросервіс для кластеризації користувачів ІТ продукту.

Наукова новизна. Проведено аналіз основних принципів розробки мікросервісів; обгрунтовано використання машинного навчання для аналізу даних, вибрано і описано кластеризацію як інструмент продуктової аналітики, розгляноуто методи та моделі вирішення задач кластеризації, наведено приклад алгоритм к-середніх і особливості його реалізації на Python, реалізазовано кластерний аналіз на прикладі заданої бази даних та інтегровано в мікросервіс.

Практична цінність роботи полягає в наступному. У роботі наведено приклад роботи розробленого мікросервісу, який може знаходитись як на локальному хості та і в мережі інтернет, в модель передаються певні параметри клієнта і вона на цих даних формує прогноз, який по суті є номером ймовірного кластеру до якого треба віднести цього клієнта.

Апробація теоретичних положень роботи відбулася 6 грудня 2023 року у ході XXV науково-практичної студентської конференції у ПрАТ «ПВНЗ «Запорізький інститут економіки та інформаційних технологій».

Магістерська дипломна робота складається зі вступу, чотирьох розділів та висновків на 99 сторінках машинописного тексту. Робота містить 98 сторінок, 1 таблицю, 51 рисунок, 2 додатки, 38 бібліографічних посилання.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Причинно-поведінковий каркас для аналізу великих даних

У сучасному світі великі дані напряму впливають та змінюють характер бізнесу. Це особливо помітно у діяльності багатьох компаній, особливо тих, чий успіх пов'язаний з генерацією ідей, основним джерелом яких виступають великі дані.

Великі дані – це сфера діяльності, спрямована на аналіз, обробку і зберігання великих обсягів корисних даних, що генеруються різноманітними джерелами. Коли традиційні методи аналізу даних, технології їх обробки і зберігання стають недостатніми, на допомогу приходять рішення і методи для великих даних. Вони відрізняються об'єднанням непов'язаних наборів даних, обробкою великих обсягів неструктурованих даних та виявленням прихованої інформації в оперативному режимі без затримок.

Зазвичай життєвий цикл аналітики великих даних включає ідентифікацію, доставку, підготовку та аналіз великих обсягів сирих даних і неструктурованих даних з метою виявлення корисної і значущої інформації. Ця інформація може служити в якості вхідних даних для визначення моделей, покращення корпоративних даних і виконання масштабних пошукових запитів.

Сучасні великі дані стали різновидом капіталу, особливо для великих технологічних компаній. Цінність їхніх пропозицій значною мірою залежить від даних, які вони постійно аналізують, щоб підвищувати ефективність та розробляти нові продукти.

Завдяки новітнім досягненням у сфері технологій вдалося значно знизити вартість зберігання та обчислень. Це дає можливість зберігати та обробляти обсяги даних, що постійно зростають, за менші витрати. Сучасні технології дозволяють здійснювати більш точні та зважені бізнес-рішення.

Дані накопичуються завдяки їх збору в компанії за допомогою різноманітних додатків, датчиків і зовнішніх джерел. Оброблені дані можуть використовуватися корпоративними додатками та зберігатися у сховищі даних.

Обробка великих даних дає змогу отримати певні переваги представлені на рис.1.1.[36].

Рис.1.1. Переваги від обробки великих даних

Незважаючи на деякі переваги від наявності великих масивів даних, існує ряд проблем, які необхідно враховувати при виборі підходів до аналітики великих даних.

Головною метою аналітики даних є генерація, перевірка та підвищення ефективності управлінських рішень. Для досягнення цієї мети використовуються різноманітні технології, такі як Apache Hadoop та Kafka для збирання та зберігання інформації, Spark та Amazon Kinesis для швидкої аналітичної обробки потокової інформації.

У якості основних інструментів для аналітики даних можна виділити сучасні BI-інструменти, які, крім візуалізації даних, дозволяють робити прогнози. Однак ці інструменти зазвичай є високою вартістю. Також використовуються мови програмування, такі як Python, разом з безліччю різноманітних бібліотек та фреймворків. Це надає можливість використовувати інструментарій для обробки

даних, проведення обчислень, моделювання, включаючи побудову нейромережевих моделей, візуалізації даних, створення дашбордів та інше. Python забезпечує гнучкість та розширюваність для вирішення різноманітних завдань аналітики великих даних.

1.2. Типи аналітики

Аналітика великих даних термін достатньо широкий, який включає і сам аналіз даних (процес дослідження даних для пошуку фактів, відношень, шаблонів, ідей або тенденцій). Аналітика даних охоплює управління повним життєвим циклом даних (збирання, очистка, організація, зберігання, аналіз і регулювання даних) [1].

Різні види організацій по різному використовують інструменти і методи аналітики даних, а саме:

– у бізнесі результати аналітики спрямовані на зниження експлуатаційних витрат і підвищення ефективності прийняття стратегічних рішень;

– в науковій сфері аналітика даних допомагає визначити причини явищ для покращення якості прогнозів;

– у сфері різноманітних сервісів аналітика допомагає підсилити якість послуг при мінімальних витратах.

Як правило, виділяють наступні чотири категорії аналітики представлені на рис.1.2.[34]

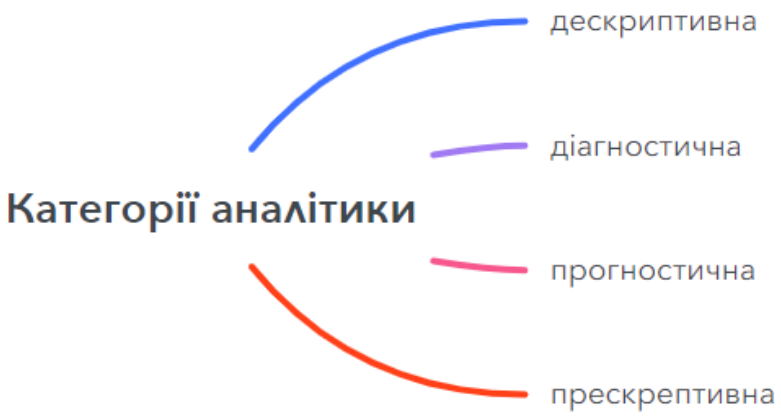


Рис.1.2. Чотири основні категорії аналітики

Основна задача різних типів аналітики підвищувати ефективність різноманітних методів та алгоритмів аналізу.

Так, дескриптивна аналітика використовується для пошуку відповіді на питання про події, які вже відбулися. Ця форма аналітики узгоджує дані з контекстом для генерування інформації. Вона реалізується за допомогою спеціальних звітів або інформаційних панелей представлених на рис. 1.3. Для проведення подібного роду аналізу на промислових підприємствах, досить важливим завданням є безперервний збір інформації з різного роду виробничого обладнання, для цього використовуються smart-датчики та інші IoT-пристрої, які дозволяють зібрати історичні дані для їх подальшого аналізу, виявити проблемні ділянки та неполадки роботи як самого обладнання, так і датчиків.

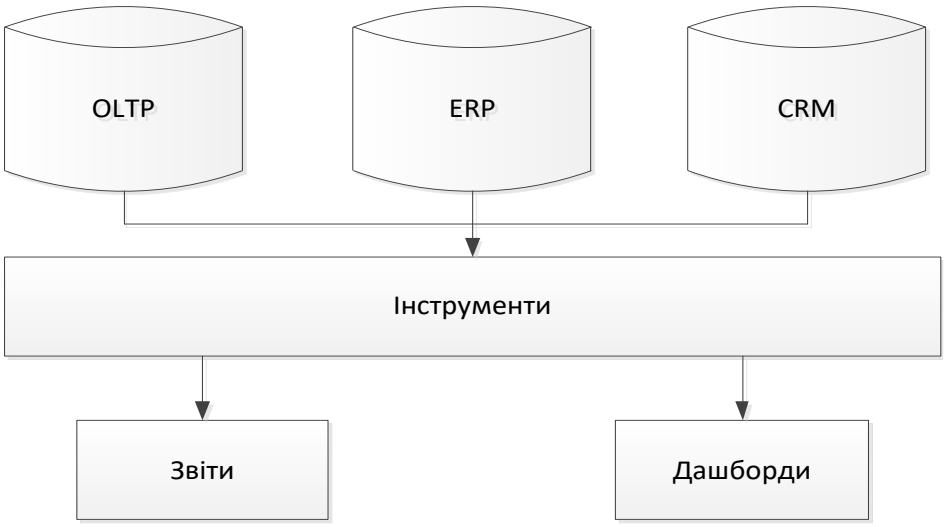


Рис.1.3. Використання дескриптивної аналітики для створення звітів та інформаційних дашбордів

Діагностична аналітика направлена на те, щоб визначити причину події яка виникла. Метою є визначення яка інформація відноситься до даного явища, щоб надати відповіді чому це відбулося.

Зазвичай діагностична аналітика потребує збору даних з різних джерел і зберігання їх в певній структурі, яка піддається аналізу як показано на рис. 1.4.

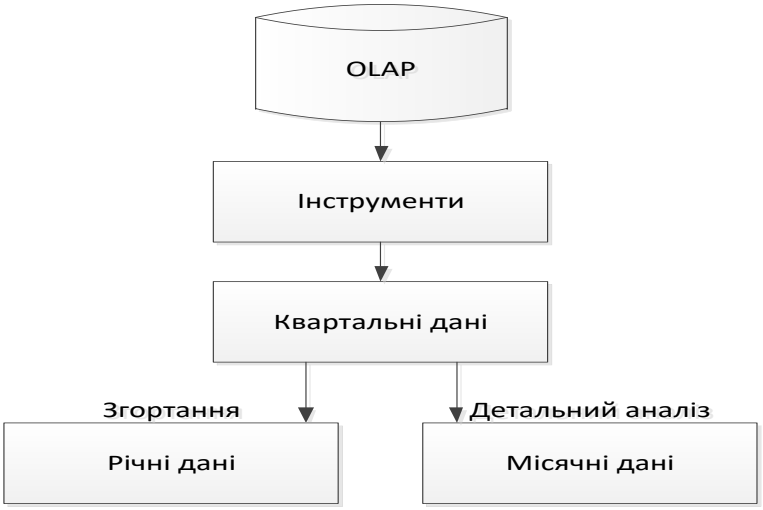


Рис.1.4. Використання діагностичної аналітики для детального вивчення даних

Результати діагностичної аналітики можуть бути представлені за допомогою інструментів інтерактивної візуалізації. Це дозволяє користувачам визначати

тенденції і шаблони. Виконання запитів відбувається на багатовимірних даних, які знаходяться в системі аналітичного опрацювання.

У таблиці 1.1 наведено роботу одного зі смарт-датчиків, при зборі агрегованої інформації з кількох датчиків на етапі дескриптивної аналітики, проблем виявлено не було, однак під час побудови прогнозних моделей і у спробі пояснити їх незадовільну якість, було виявлено, що деякі елементи безперервного збору даних потребують ремонту або заміни. Виправлення зазначеної проблеми суттєво підвищило якість прогностичних моделей.

Таблиця 1.1

Робота одного зі смарт-датчиків

Дата	**Код**	**Добові показники smart-датчику**
01.08.2021	100001	4.2
02.08.2021	100001	4.6
03.08.2021	100001	0
04.08.2021	100001	0
05.08.2021	100001	0
06.08.2021	100001	3.8
07.08.2021	100001	3.6
08.08.2021	100001	3.9
09.08.2021	100001	0
10.08.2021	100001	0
11.08.2021	100001	0
12.08.2021	100001	0
13.08.2021	100001	0
14.08.2021	100001	3.1
15.08.2021	100001	3.4
16.08.2021	100001	4
17.08.2021	100001	3.8
18.08.2021	100001	4.1
19.08.2021	100001	4.1
20.08.2021	100001	4.2
21.08.2021	100001	0
22.08.2021	100001	0

У якості основного інструментарію на цьому етапі можуть застосовуватись кластерний аналіз, кореляційний аналіз, будуватись моделі класифікації тощо.

Прогностична аналітика проводиться з метою визначити результат певної події, яка може виникнути у майбутньому. За допомогою цього виду аналітики інформація підсилюється змістовним навантаженням. Інтенсивність і значимість асоціативних зв'язків формують основу моделей, які використовуються для створення майбутніх прогнозів на основі минулих подій.

Як правило, інструменти прогностичної аналітики використовують абстрактні способи вирішення статистично складних і неструктурованих задач, надаючи зручні для користувача зовнішні інтерфейси (див. рис. 1.5).

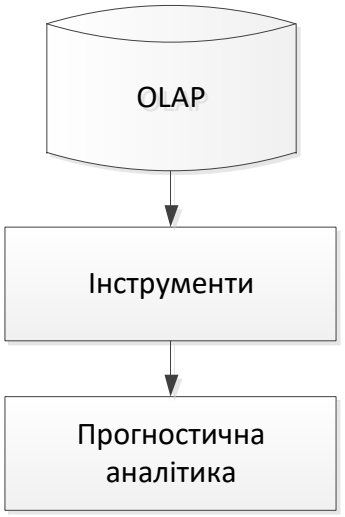


Рис.1.5. Інструменти прогностичної аналітики

На даному етапі, доцільно використовувати математичну статистику, імітаційне моделювання, машинне навчання, а також нейромережеве моделювання. У свою чергу машинне навчання та нейромережеве моделювання потребують великих об'ємів даних, що напряму відобразиться на якості побудованих прогнозів. Слід зазначити, що не завжди доцільно використовувати «складні» моделі і треба дотримуватись паритету між ціною та якістю побудованих прогнозів.

Результатами прогностичної аналітики є прескриптивна, яка пропонує заходи, які потрібно реалізувати. Іншими словами, цей вид аналітики дає результати на основі яких можна робити висновки.

Прескриптивна аналітика є найбільш значимою, оскільки завдячуючи їй розробляються різноманітні сценарії і пропонується оптимальний набір заходів для

кожного з них. Вона передбачає використання бізнес-правил і великих обсягів внутрішніх і зовнішніх даних, у тому числі, отриманих як результат моделювання і описує оптимальний план дій спрямованих на прийняття ефективних управлінських рішень (див. рис. 1.6).

Отримані на етапі прескриптивної аналітики результати, можуть увійти до модельного блоку системи підтримки прийняття рішень, рекомендаційної системи тощо.

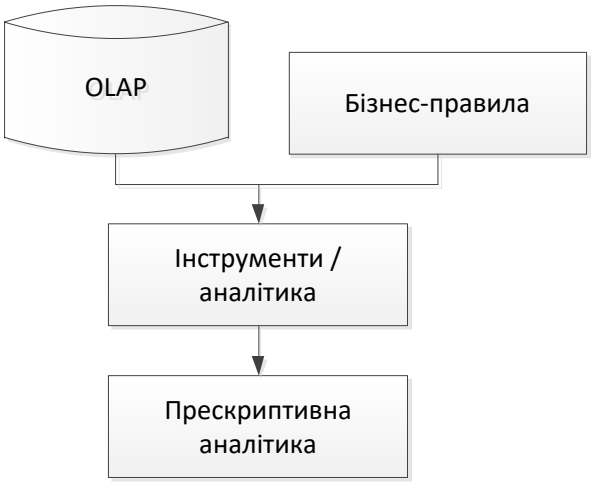


Рис.1.6. Використання прескриптивної аналітики

Даний вид аналітики представляє найбільшу значимість, на відміну від інших видів. Прескриптивна аналітика потребує відповідних навичок та спеціалізованого програмного забезпечення та інструментів. Завдяки їй розраховуються різні результати і пропонується оптимальний перелік дій. Дана тактика переходить від пояснень до консультацій і включає в себе моделювання різних сценаріїв. У магістерській роботі пропонується використовувати саме прескриптивну аналітику для аналізу поведінки користувачів.

1.3. Система аналізу даних про поведінку користувачів

Аналіз поведінки користувачів з використанням прескриптивної аналітики стає надзвичайно важливим, особливо на нових ринках, де умови є непередбачуваними, а стратегії маркетингу можуть бути на етапі раннього

розвитку. Спостерігаючи за кількістю користувачів, які взаємодіють з продуктом (додатком), відвідуванням екранів чи веб-сторінок та кількістю сеансів користування, аналітика допомагає оптимізувати веб-сайт чи маркетингову стратегію, щоб вони краще відповідали потребам і інтересам користувачів на новому ринку.

Аналітика продукту — це процес систематичного моніторингу, збору та обробки даних на рівні користувача у формі статистичних даних (метрик), що вказують на взаємодію клієнтів з конкретними продуктами. Цей процес дозволяє компаніям аналізувати шляхи користувачів — від моменту активації до всіх етапів використання продукту — з метою зрозуміння того, що привертає їх до продукту та заохочує повертатися до нього. Аналітика також допомагає визначити цінність продукту для клієнтів [1].

Відсутність аналітики продукту може призвести до витрат ресурсів на неефективні канали та інструменти, або використання креативних підходів, які не привертають клієнтів, а навпаки, викликають відторгнення.

Основні етапи аналітики продукту включають [2]:

Збір даних: Систематичний збір даних про використання продукту є ключовим етапом. Це включає інформацію про користувачів, їх дії в продукті та взаємодію з функціями та іншими параметрами.

Аналіз даних: Після успішного збору даних важливо провести їх аналіз для виявлення патернів та трендів. Цей етап допомагає зрозуміти, як користувачі взаємодіють з продуктом та використовують його функціонал.

Метрики продукту: Визначення ключових метрик грає важливу роль у відстеженні значущих аспектів продукту. Серед них можуть бути конверсія, утримання користувачів, частота використання функцій та інші важливі параметри.

Тестування гіпотез та експерименти: Аналітика продукту дозволяє проводити тести та експерименти для визначення того, які зміни можуть покращити якість продукту та задовольнення користувачів.

Звітність та рекомендації: На основі результатів аналізу даних аналітична команда готує звіти та рекомендації для розробників та менеджменту, що

допомагає у прийнятті обгрунтованих рішень стосовно подальшого розвитку продукту.

Прескриптивна аналітика продукту є важливою частиною управління продуктом і використовується в різних галузях, таких як інтернет-бізнес, розробка програмного забезпечення, мобільні додатки та інші, де продукти або послуги мають цифровий аспект. Шляхом аналізу історичних даних та використання алгоритмів машинного навчання ця технологія може виявляти закономірності та тенденції, які допомагають організаціям передбачати майбутні результати, особливо в сфері фінансів, де точність прогнозування є вирішальною для прийняття інвестиційних рішень та управління ризиками.

Дані, зібрані за допомогою аналітики продукту, включають інформацію про популярність функцій, середній час витрачений на певні дії, ефективність маркетингових каналів та утримання користувачів. Ця інформація дозволяє компаніям аналізувати взаємодію користувачів та використовувати її для поліпшення продукту.

Аналітика продукту стає важливою практикою управління продуктом, оскільки багато додатків і веб-сайтів не надають достатньо деталізованої звітності. Без аналітики дані про використання можуть бути неправильно відформатованими та суперечливими. Програмне забезпечення для аналізу продуктів робить ці неструктуровані дані корисними, об'єднуючи їх в єдине організоване подання.

Важливо впроваджувати аналітику продукту тільки після досягнення продуктом встановленої мінімальної кількості користувачів. При невеликій клієнтській базі дані, зібрані аналітикою продукту, можуть бути недостатніми для створення значущої вибірки, яка впливає на організаційні рішення.

Процес впровадження системи аналізу даних про поведінку користувачів передбачає рис.1.7 [35]:

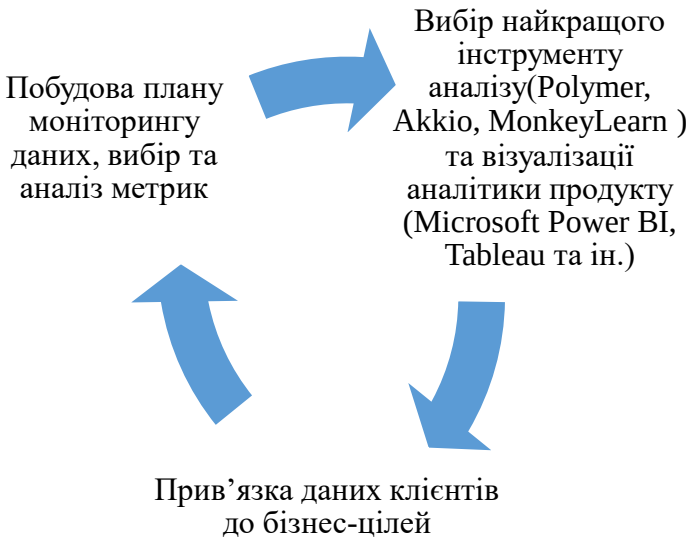


Рис 1.7. Процес впровадження системи аналізу даних про поведінку користувачів

Встановлення зв'язку між даними клієнтів та бізнес-цілями є критичним аспектом. Слід чітко визначити конкретні бізнес-цілі, пов'язані зі збиранням даних. Наприклад, це може включати конвертацію більшої кількості користувачів з безкоштовної пробної версії в платників.

При створенні плану моніторингу даних важливо чітко визначити всі події, які необхідно відстежувати під час взаємодії клієнтів із продуктом. На цьому етапі обрання та аналіз ключових метрик, таких як LTV, ARPU, ARPPU, ARPDAU, DAU, WAU, MAU, Retention, churn rate, відіграє важливу роль у забезпеченні ефективного моніторингу.

Вибір оптимального інструменту для аналізу та візуалізації даних в продуктовій аналітиці вирішується важливим завданням для будь-якої даними керованої організації. Існує багато інструментів, які вирізняються можливістю візуалізації, аналізу та моніторингу даних, сприяючи досягненню бізнес-цілей.

Декілька з найкращих інструментів на сьогодні включають [4]:

Tableau: Даний інструмент підтримує складні обчислення та об'єднання даних, дозволяє створювати інтерактивні візуалізації та легко опрацьовує великі об'єми інформації.

Microsoft Power BI: Цей інструмент інтегрується з існуючими програмами, створює персоналізовані інформаційні панелі, допомагає опубліковувати безпечні звіти та не обмежується обсягами пам'яті та швидкістю.

Polymer: Надійний інструмент штучного інтелекту, який перетворює дані в базу даних без кодування та аналізує їх для поліпшення розуміння користувачами.

Akkio: Ця платформа машинного навчання без коду дозволяє будувати нейронні мережі та має рейтинг точності для моделей.

MonkeyLearn: Інструмент для класифікації тексту за мітками, спрощення очищення, систематизації та візуалізації відгуків, не вимагає кодування і автоматизує бізнес-процеси та аналіз тексту, що призводить до заощадження часу.

Використання візуалізації даних представляє собою надзвичайно важливий інструмент у сфері аналітики та управління продуктом. Значущість візуалізації даних виявляється в наступному:

Розуміння для всіх: Візуалізація робить дані більш доступними і зрозумілими для всіх учасників проекту, навіть для тих, хто не є експертами в області аналітики.

Виявлення патернів: Патерни та тенденції у даних іноді важко виявити, переглядаючи числа. Візуалізація полегшує впізнавання залежностей та аномалій.

Підтримка прийняття рішень: Візуалізація допомагає керівникам та управлінцям швидко розуміти ситуацію та приймати обгрунтовані рішення.

Ефективне поєднання: Значущість візуалізації полягає в здатності ефективно комбінувати складні концепції та інформацію.

Практичні поради щодо візуалізації даних:

Відповідність цільовій аудиторії: Чітко розумійте, кому призначена ваша візуалізація і адаптуйте її відповідно. Різні аудиторії можуть краще розуміти конкретні типи графіків чи діаграм.

Ефективне використання кольорів: Використовуйте кольори обережно, уникайте яскравих відтінків та забезпечте стале маркування.

Інтерактивність: Розгляньте можливість використання інтерактивних елементів у візуалізації, щоб користувачі могли досліджувати дані самостійно.

Спрощення та узагальнення: Уникайте перенасичених діаграм та графіків. Спростіть візуалізацію, фокусуючись на головних точках.

Текстові пояснення: Додавайте текстові елементи для пояснення ключових аспектів візуалізації та надавайте додатковий контекст.

Тестування зрозумілості: Перевірте вашу візуалізацію на представниках цільової аудиторії, щоб забезпечити її зрозумілість та ефективність.

Використання візуалізації даних у продуктовій аналітиці може значно полегшити розуміння та ухвалення рішень з боку всіх учасників проєкту.

Враховуючи все вище сказане розглянемо приклад аналізу великих даних (дата сет) поведінки користувачів додатку. Даний дата сет мість наступнні дані: Users ID, Platform, Device Type, OS Version, App Version, Payment Time, Price Plan, Subscription, Timestamp Event_type, Option, Session_id, Activity. На рис. 1.8 представлено аналіз даних та візуалізацію в Tableau.

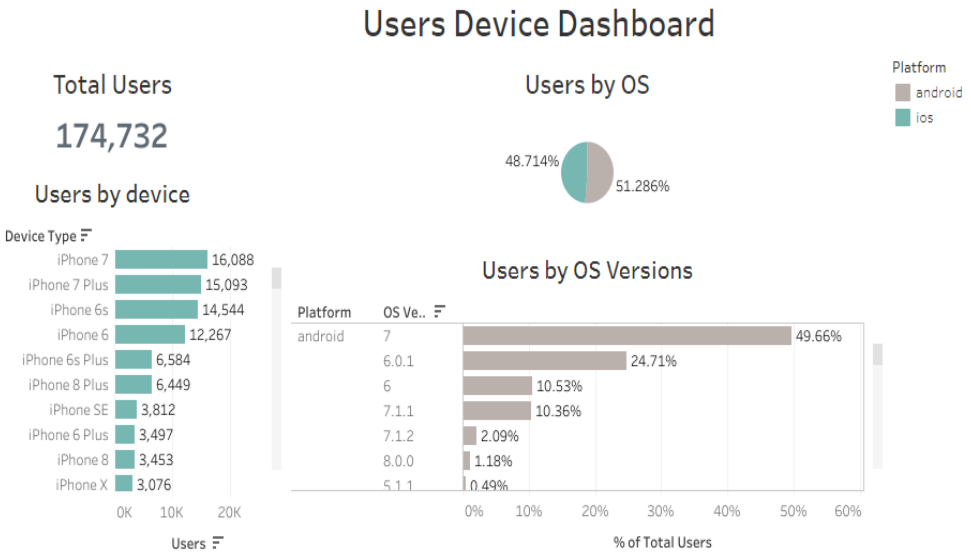


Рис.1.8. Аналіз користувачів додатка за типами девайсів

З дашборду видно аналіз користувачів додатком за типами девайсів, за видами операційних систем, загальну кількість користувачів. Отримані дані нам

необхідні для моніторингу основних метрик користування додатком. На рис.1.9. зображено аналіз основних метрик взаємодії користувачів з продуктом.

Рис.1.9. Аналіз основних метрик взаємодії користувачів з продуктом

Отже, аналіз поведінки користувачів через візуалізацію даних може виявитися викликом, але при належному плануванні, увазі до деталей та фокусу на чіткості та ефективності ці труднощі можна подолати. Важливо обрати відповідні методи візуалізації, адаптувати обробку даних з урахуванням їх складності та обсягу, а також покращити інтерпретацію. Таким чином, компанії можуть отримати цінну інформацію, яка сприяє у прийнятті рішень та оптимізації бізнес-процесів.

1.4. Висновки до розділу

У роботі вказано, що термін "аналітика великих даних" є достатньо широким і охоплює не лише сам процес аналізу даних (який включає в себе дослідження

даних для виявлення фактів, відносин, шаблонів, ідей або тенденцій), але й управління усім життєвим циклом даних (збір, очищення, організація, зберігання, аналіз і регулювання даних). Дані, які збираються за допомогою аналітики продукту, містять інформацію про найпопулярніші функції продукту, середній час, який користувачі витрачають на конкретну дію, ефективність маркетингових каналів у приведенні найкращих користувачів, а також процент повернення користувачів до продукту щоденно, щотижнево чи щомісяця. За допомогою цієї інформації компанії можуть аналізувати, як користувачі взаємодіють із створеним продуктом і використовувати ці дані для поліпшення взаємодії з користувачами.

Прескриптивна аналітика - це форма аналітики, яка використовується для розробки рекомендацій щодо того, що слід робити. Вона використовує моделі машинного навчання для аналізу даних і виявлення закономірностей.

Прескриптивна аналітика є потужним інструментом, який може допомогти організаціям приймати більш обґрунтовані рішення. Вона може використовуватися для виявлення можливостей, вирішення проблем і підвищення ефективності.

Прескриптивна аналітика є відносно новою галуззю, але вона швидко розвивається. Завдяки зростанню доступності даних і потужності машинного навчання прескриптивна аналітика стане ще більш поширеною в майбутньому.

РОЗДІЛ 2
ІНСТРУМЕНТИ РОЗРОБКИ МІКРОСЕРВІСУ

2.1. Архітектура мікросервісів

Архітектура мікросервісів – це підхід до розробки програмного забезпечення, в якому додаток розбивається на невеликі та автономні сервіси, відомі як мікросервіси. Кожен мікросервіс працює незалежно, має свою власну базу коду та базу даних і може бути розгорнутий та масштабований незалежно від інших сервісів. Основні принципи цієї архітектури включають розбиття на сервіси, самостійність, легкість заміни, гнучкість, розподілені дані, масштабованість, зовнішні інтерфейси, керованість, шаровану архітектуру, мінімізацію залежностей [10].

Розбиття на сервіси означає поділ монолітного додатка на невеликі, автономні сервіси, які можуть діяти незалежно. Кожен мікросервіс функціонує самостійно і може бути замінений чи оновлений без впливу на інші частини системи. Гнучкість дозволяє швидко реагувати на зміни та покращення, а масштабованість забезпечує оптимальну продуктивність та відгук системи. Розподілені дані дозволяють кожному мікросервісу мати власну базу даних або використовувати різні підходи до роботи з даними.

Використання мікросервісної архітектури полегшує розробку, тестування та підтримку складних додатків, забезпечуючи гнучкість та швидкість реагування на зміни у вимогах та ринкових умовах. Однак правильне планування, впровадження та управління мікросервісами є ключовим для уникнення складнощів, пов'язаних із зростанням кількості сервісів та їх взаємодією [10].

Підходи до розгортання мікросервісів включають використання контейнерів, таких як Docker, який дозволяє розгортати сервіси в ізольованих контейнерах та використовувати контролери масштабування, наприклад Kubernetes. Це забезпечує надійність, масштабованість та можливість розподілити додаток на різні фізичні та віртуальні машини [9].

Мікросервісна архітектура показано на рисунку 2.1 [11].

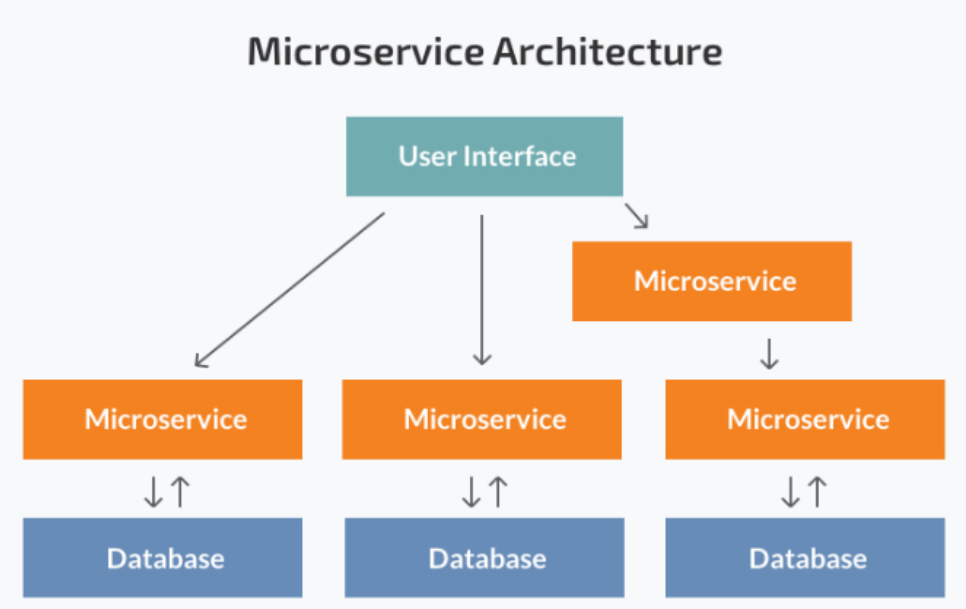


Рис 2.1. Мікросервісна архітектура

При впровадженні мікросервісної архітектури кожен сервіс має власну кодову базу та базу даних, і лише до них має прямий доступ. Оскільки сервіси не можуть безпосередньо взаємодіяти з базами даних інших сервісів, обмін даними здійснюється через виклики ресурсів інших сервісів. Такий підхід призводить до більш складного доступу до ресурсів, що належать іншим сервісам у мікросервісній архітектурі.

Ще однією значущою перевагою мікросервісної архітектури є можливість використання різноманітних мов програмування та бібліотек для створення її компонентів. Взаємодія між різними частинами системи здійснюється за допомогою загальних протоколів зв'язку, які підтримують практично всі актуальні мови програмування. Це відкриває широкий спектр інструментів для вирішення різноманітних завдань і створює безліч можливостей. Наприклад, для створення інтерактивного інтерфейсу користувача можна використовувати JavaScript, для роботи з обсягом великих даних – Python, для обробки подій з мінімальними затримками в реальному часі – мови програмування C та C++.

Архітектура на основі мікросервісів має безліч переваг, але вона також пов'язана з труднощами. Однією з проблем мікросервісів є розподілені журнали, оскільки незалежні сервіси генерують власні журнали. Це становить виклик у порівнянні з централізованими журналами монолітів, які забезпечують єдине джерело правди для розробників та оперативних груп. Моніторинг і управління інфраструктурою також стають складнішими через велику кількість рухомих елементів. Тестування та налагодження стають складнішими, оскільки відсутня інтегрована середовище розробки (IDE), яка притаманна монолітам.

Подібно до відсутності формального визначення терміну "мікросервіси", немає стандартної моделі, яка представлена у кожній системі, ґрунтованій на цьому архітектурному стилі. Однак можна очікувати, що більшість систем, побудованих за принципами мікросервісної архітектури, виявлять кілька загальних рис:

Мультикомпонентність: Програмне забезпечення, розроблене як мікросервіси, може бути розглянуте як набір складових служб. Це дозволяє розгортати, конфігурувати та перевищувати окремі служби незалежно одна від одної, зберігаючи цілісність системи. Цей підхід може вимагати змін лише в окремих службах, замість повного перезапуску всього додатку. Однак це може призводити до витратних віддалених викликів, грубих віддалених API та ускладненого перерозподілу обов'язків між компонентами.

Бізнес-орієнтованість: Мікросервіси часто орієнтовані на бізнес-можливості та пріоритети. На відміну від традиційного монолітного підходу, де різні команди фокусуються на окремих аспектах, у мікросервісній архітектурі використовуються кросфункціональні команди. Кожна команда відповідає за створення конкретних продуктів, базованих на одному чи кількох сервісах, що взаємодіють через шину повідомлень.

Проста маршрутизація: Мікросервіси діють, як і системи UNIX, обробляючи запити, генеруючи відповіді. У відміну від інших продуктів, таких як ESB, де використовуються високотехнологічні системи для маршрутизації повідомлень, мікросервіси мають інтелектуальні кінцеві точки та прості канали, по яких інформація тече.

Децентралізованість: З огляду на використання різних технологій та платформ, мікросервіси віддають перевагу децентралізованому управлінню. Це дозволяє розробникам створювати корисні інструменти для спільноти, які можуть використовуватися іншими для вирішення загальних проблем. Мікросервісна архітектура також підтримує децентралізоване управління даними, де кожна служба управляє своєю унікальною базою даних.

Відмовостійкість: Мікросервіси розроблені так, щоб ефективно впоратися з помилками. З огляду на те, що кілька унікальних та різноманітних сервісів взаємодіють, можливі випадки, коли сервіс вийде з ладу. У таких випадках система повинна дозволити іншим службам працювати далі, доки відновлюється відмовивший сервіс. Моніторинг мікросервісів може допомагати уникнути ризику виникнення неполадок.

Еволюційність: Архітектура мікросервісів є еволюційним дизайном, ідеально підходить для систем, що постійно розвиваються. Це особливо корисно в ситуаціях, коли неможливо повністю передбачити типи пристроїв.

2.2. Засоби віртуалізації

Віртуалізація - це концепція, що передбачає логічне об'єднання обчислювальних ресурсів та їх абстрагування від фізичного обладнання, такого як сервер чи комп'ютер. Віртуальна машина має свій власний пул логічних ресурсів, таких як центральний процесор (CPU), оперативна пам'ять (RAM) і дисковий простір. Завдяки віртуалізації можна одночасно запускати кілька незалежних віртуальних машин на одному фізичному сервері.

На ринку існує безліч платформ, які підтримують різні типи віртуалізації та орієнтовані на вирішення різних завдань. Однак основна область застосування технологій віртуалізації - це хмарні сервіси.

Основна перевага віртуалізації полягає в ефективному зменшенні витрат на підтримку IT-інфраструктури завдяки економії фізичних ресурсів, автоматизації процесів, адаптивності та масштабованості бізнесу.

Надійність є ще однією перевагою цієї технології, оскільки в критичних ситуаціях дані можна легко відновити за допомогою резервного копіювання віртуальних машин. Цей процес можна автоматизувати, щоб система зберігала актуальну інформацію в резервних копіях, зменшуючи ризик простою бізнесу.

Платформи віртуалізації створюють гнучке середовище для тестування різних проектів, зокрема в галузі розробки програмного забезпечення. Віртуалізація також служить основою для впровадження хмарних рішень, які підвищують контроль над критичними даними.

Однією з популярних платформ віртуалізації для ізольованих систем є Docker - інструмент управління контейнерами в середовищі Linux. Docker використовує контейнерну віртуалізацію, де кожна програма запускається в окремому контейнері, повністю ізольованому від інших на рівні файлової системи та внутрішніх процесів. Це дозволяє одному серверу обробляти сотні контейнерів, забезпечуючи масштабованість та рівномірний розподіл навантаження.

Кожен контейнер Docker формується шляхом використання простого текстового файлу, який включає інструкції для створення образу контейнера Docker. Для автоматизації цього процесу використовується Dockerfile, який надає стандартні команди для створення образу через інтерфейс командного рядка (CLI). Образи Docker вміщують в себе виконуваний вихідний код програми та всі необхідні інструменти, бібліотеки та залежності для правильного виконання програмного коду у вигляді контейнера. Сам образ складається з різних шарів, кожен з яких представляє собою версію образу.

Контейнери Docker представляють собою активні екземпляри образів Docker, що виконуються. Вони є живим, ефемерним та виконуваним вмістом, з яким можна взаємодіяти та налаштовувати. Docker Hub, загальнодоступне сховище образів Docker, є важливим ресурсом для обміну зображеннями та є платформою для спільноти Docker.

Крім Docker Hub, існують інші платформи для сховища зображень, такі як GitHub, який використовується для розміщення репозиторіїв та обміну інструментами розробки. Docker Desktop - це програма для операційних систем

Mac або Windows, яка містить Docker Engine, Docker CLI, Docker Compose, Kubernetes та інші компоненти, а також надає доступ до Docker Hub.

Реєстр Docker - це система для зберігання та розповсюдження образів Docker з відкритим кодом. Він дозволяє відстежувати версії зображень, використовуючи теги для їх ідентифікації. Docker Hub і GitHub використовуються для обміну зображеннями та створення репозиторіїв.

Узагальнюючи, віртуалізація та інструменти, такі як Docker, відіграють важливу роль у поліпшенні ефективності, масштабованості та керованості IT-середовищ, роблячи їх більш гнучкими та надійними.

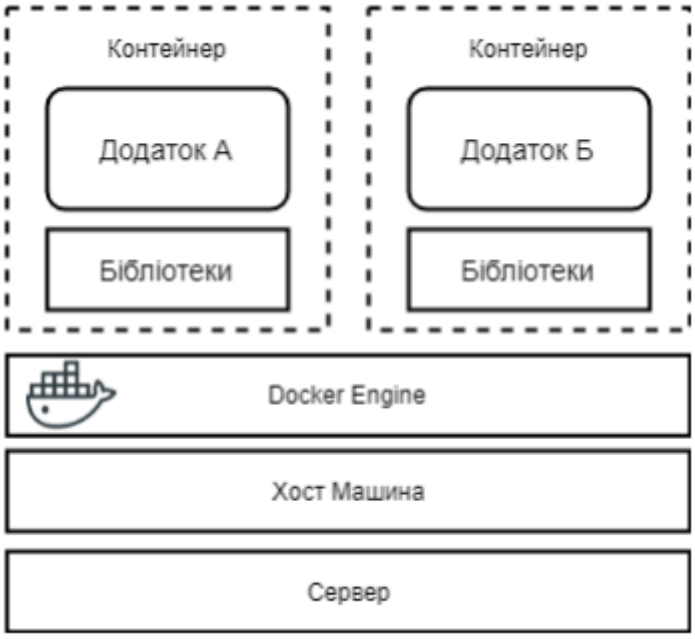


Рис. 2.2. Архітерктура Docker-контейнера

Docker дозволяє запускати різні процеси в ізольованому режимі, нехтуючи їхнім вмістом, і потім легко переносити та клонувати сформовані контейнери на інші сервери. Docker бере на себе всю роботу зі створення, обслуговування та підтримки контейнерів [12]. На рисунку 2.3 зображена структура та основні та основні компоненти Docker.

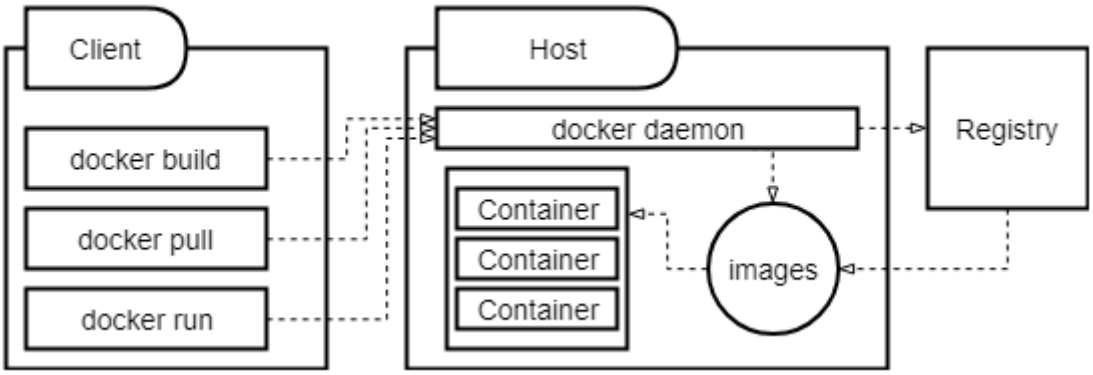


Рис. 2.3. Струкутра та компоненти Docker

Docker Client є інтерфейсом користувача Docker, надаючи набір команд для управління образами та контейнерами [8]. Взаємодія з Docker Client здійснюється через Docker Daemon - демон-процес [9], що відповідає за операційну роботу системи. Docker Host містить контейнери та образи Docker Image (DI). Кожен образ включає операційну систему, застосунок та всі його залежності. Образи в Docker представлені у вигляді шарів. Наприклад, для створення образу з веб-сервером, ми використовуємо образ з операційною системою як базовий, додаємо залежність - веб-сервер і записуємо це як новий образ, який містить два шари: один із операційною системою, інший - із веб-сервером [5]. Зібрані Docker образи зберігаються у Docker Registry (DR) для подальшого використання.

Отже, Docker допомагає:

- мінімізувати використання ресурсів;
- зручно приховувати фонові процеси;
- просто масштабувати;
- зменшити час між написанням і запуском коду;
- швидше тестувати;
- швидко розгортати;
- ефективно створювати додатки.

2.3. Мова програмування Python для аналізу даних

На сучасному IT-ринку існує безліч технологій для розробки програмних продуктів, зокрема на архітектурі клієнт-сервер. Для вибору найкращих та найсучасніших технологій для створення програмного застосунку необхідно вивчити наявні технології, підходи та практики їх використання у сучасних умовах. Згідно з індексом популярності мов програмування PYPL (PopularitY of Programming Language), на сьогодні найбільш популярною є мова програмування Python [28].

Worldwide, May 2021 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	29.9 %	-1.2 %
2		Java	17.72 %	-0.0 %
3		JavaScript	8.31 %	+0.4 %
4		C#	6.9 %	-0.1 %
5	↑	C/C++	6.62 %	+0.9 %
6	↓	PHP	6.15 %	+0.1 %
7		R	3.93 %	+0.0 %

Рис. 2.4. Рейтинг мов програмування

Мова програмування Python завоювала високу популярність відносно недавно, але за цей короткий період вже продемонструвала свою цінність у багатьох галузях розробки програмного забезпечення. Її широка поширеність головним чином обумовлена легкістю вивчення та стрімким зростанням спільноти розробників Python, які неперервно розвивають цю мову.

Python є високорівневою мовою програмування, спрямованою на підвищення продуктивності розробника та полегшення читання коду. Це інтерпретована мова, що дозволяє виконувати код рядок за рядком і виявляється особливо корисною під час налагодження програм. Інтерпретатор написаний мовою C і є відкритим для

усіх; його можна розширювати власними доповненнями для отримання нового функціоналу [29]. Найбільш поширеною реалізацією інтерпретатора є CPython [29].

Python володіє динамічною типізацією, що означає відсутність необхідності попередньо вказувати типи даних. Синтаксис мови є лаконічним, чітким та простим, що дозволяє легко освоювати його та швидко писати програмний код, полегшуючи читання іншими розробниками. Пам'ять управляється автоматично, що звільняє розробника від необхідності вручну керувати використанням пам'яті. Python здатний ефективно працювати зі складними структурами даних високого рівня та використовує динамічне зв'язування, що прискорює процес розробки програмного рішення [29].

Python представляє собою мультипарадигмальну мову програмування, з фокусом на об'єктно-орієнтованій та функціональній парадигмах, а також підтримує інші парадигми, такі як імперативна, процедурна, структурна та аспектно-орієнтована. Python є кросплатформеним, що дозволяє легко переносити програмне забезпечення з однієї платформи на іншу, встановлюючи та компілюючи вихідний код на мові C [29]. Python також дозволяє використовувати компільовані коди C та C++, що розширює можливості мови.

Загалом, Python є надзвичайно потужним і гнучким інструментом для розробників, який знаходить широке застосування у різних областях програмування.

Ще однією значущою перевагою Python є його обширний спектр вбудованих бібліотек, що робить мову ідеальною для використання в різноманітних проектах. Стандартна бібліотека Python містить широкий набір інструментів та компонентів для роботи з мережевими протоколами, різними форматами представлення даних, створення HTTP серверів та клієнтів, обробки поштових повідомлень, маніпуляції даними JSON, XML, CSV та іншими. Крім того, є модулі для взаємодії з операційною системою, що дозволяють легко створювати кросплатформні додатки.

Завдяки широкій підтримці спільноти розробників, Python насичений різноманітними бібліотеками, які покривають практично всі аспекти

програмування. Ці бібліотеки включають інструменти для обробки зображень, роботи з базами даних, веб-розробки, чисельних методів, створення штучного інтелекту, обробки тексту, розробки ігор та багато інших. Більшість цих бібліотек є вільнодоступними і мають високий рівень документації.

Невід'ємною перевагою Python є його ліцензія, яка не обмежує комерційне використання. Це робить Python привабливим для розробників, які працюють над бізнес-рішеннями. Крім того, велика кількість розширень та бібліотек Python доступні у вільному доступі, що сприяє швидкому та ефективному розвитку програм.

Однак важливо враховувати, що Python, як інтерпретована мова, може демонструвати невисоку швидкість виконання програм, особливо для великих обчислювальних завдань. Це обмеження вирішується в більшості випадків меншими витратами часу на розробку програмного забезпечення. Загалом, Python є ефективним та потужним інструментом, зокрема для проектів, де важлива не критична швидкість виконання.

2.4 Розробка додатків з використанням фреймворків

Flask - це ліцензований мікрофреймворк на базі Werkzeug та Jinja2, розроблений Pocoo і на сьогоднішній день підтримуваний проектом The Pallets Project. Однією з особливостей мікрофреймворків, до яких належить Flask, є їх фокус на наданні лише необхідних компонентів для вирішення конкретної задачі [38].

Flask дозволяє розробникам використовувати лише необхідні конфігурації, роблячи його простим та гнучким. Це особливо вигідно для розробників, які можуть легко налаштовувати і використовувати тільки ті функції, які їм потрібні, спрощуючи процес розробки програм та використання плагінів.

Основні компоненти Flask - це Werkzeug і Jinja2. Werkzeug відповідає за маршрутизацію, налагодження та інтерфейс шлюзу веб-сервера (WSGI), в той час

як Jinja2 виступає в якості движка для шаблонів. На відміну від деяких інших фреймворків, Flask не вбудовує в себе підтримку баз даних, автентифікації користувачів чи інших високорівневих утиліт. Однак він має широкий вибір розширень, які реалізують ці функціональні можливості.

Структура Flask є модульною, що полегшує організацію великих застосунків. Цей фреймворк надає надійну основу для додатків, а функціонал верхнього рівня виконується за допомогою розширень. Flask пропонує зручний інструментарій для створення як простих, так і складних веб-додатків, забезпечуючи при цьому гнучкість та легкість використання.

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, From Flask!'

if __name__ == '__main__':
      app.run()
```

Цей код використовує бібліотеку Flask для створення простого веб-додатка. Він ініціює додаток, оголошує маршрут та визначає функцію для виконання при виклику маршруту. Однак вбудований сервер, який використовується в цьому коді, призначений переважно для тестування і не підходить для введення додатку в експлуатацію.

Для роботи з базою даних у Flask часто використовують розширення Flask-SQLAlchemy, яке надає підтримку для бібліотеки SQLAlchemy. SQLAlchemy пропонує інструменти для роботи з SQL та Object Relational Mapping (ORM).

Для розширення функціональності додатку, розробники часто додають модулі аутентифікації користувача та операцій CRUD (створення, читання, оновлення та видалення даних). Flask також дозволяє інтегрувати Swagger для автоматичної генерації документації API.

У тестуванні зазвичай використовують pytest для перевірки функціональності. Це повнофункціональний інструмент для тестування Python,

який спрощує розробку тестів та масштабується для складних випадків використання.

Postman є важливим інструментом для роботи з API REST, надаючи повноцінну платформу для розробки та інтеграції API на всіх етапах життєвого циклу. Це полегшує розробку та робить її більш надійною.

Стрімліт (Streamlit) - це відкрите програмне забезпечення для швидкої розробки веб-застосунків з використанням мови програмування Python. Основна ідея за Streamlit полягає в тому, щоб дозволити розробникам легко створювати веб-застосунки з мінімальною кількістю коду. Застосунки, створені за допомогою Streamlit, призначені для швидкої прототипізації, аналізу даних та візуалізації [37].

Основні особливості Streamlit включають:

Простота використання: Streamlit розроблено з фокусом на простоту використання. Розробники можуть створювати веб-застосунки, використовуючи всього кілька строк коду.

Динамічна оновлюваність: Streamlit дозволяє автоматично оновлювати вміст віджетів та графіків при зміні параметрів, без необхідності перезавантаження сторінки.

Велика кількість вбудованих віджетів: Streamlit має набір вбудованих віджетів, таких як кнопки, поля введення, графіки та інші, що дозволяє розробникам легко взаємодіяти з користувачами та візуалізувати дані.

Підтримка вбудованого рендерингу: Streamlit автоматично розпізнає та рендерить різні типи даних, такі як тексти, зображення, графіки та інші.

Підтримка аналізу даних: Streamlit дозволяє розробникам легко використовувати Python-бібліотеки для аналізу даних, такі як Pandas та NumPy, для обробки та візуалізації даних в їхніх веб-застосунках.

Підтримка розгортання в хмарі: Розроблені за допомогою Streamlit веб-застосунки можна легко розгортати в хмарних сервісах, таких як Streamlit Sharing, Heroku, або AWS.

Стрімліт робить процес створення веб-застосунків на мові Python більш доступним і зручним для розробників, що спеціалізуються на аналізі даних та штучному інтелекті.

2.5. A/B тест як спосіб тестування реакції користувачів на зміни в IT продукті

A/B-тестування – це процес, під час якого одночасно показують два варіанти одного продукту двом тестовим групам споживачів. Засновано на поведінковій активності кожної групи, можна зробити висновки щодо ефективності тестованих змін [32].

Зазвичай A/B-тести використовуються для точкового підвищення конверсії, таких як перевірка невеликих змін інтерфейсу, дизайну елементів продукту, розташування контенту тощо, де користувачі взаємодіють з новими функціями. Кількість користувачів вашого продукту визначає кількість можливостей для отримання прибутку, тому постійна робота над конверсійністю продукту є важливою для максимізації цих можливостей.

В A/B тестуванні визначається варіант оформлення, який призводить до більш високої конверсії, і цей варіант допомагає оптимізувати продукт для досягнення кращих результатів. Результат – підвищення конверсії користувачів, що призводить до привертання більше клієнтів без збільшення витрат на залучення відвідувачів.

Щоб забезпечити вірогідність результатів, тест слід проводити із контрольною вибіркою. Рекомендується розділити аудиторію на три групи (A, A, B) замість двох (A та B). Це дозволяє виявити вплив зовнішніх чинників на результат тесту та визначити, чи не виникли помилки у зборі метрик.

Наприклад, якщо інший відділ запускає рекламну кампанію, що привертає багато користувачів, які можуть вплинути на результати тесту, контрольна вибірка допоможе визначити, чи рівномірно розподілені учасники A/B-тесту за групами.

За результатами контрольної вибірки можна визначити, чи вплинули зовнішні чинники на результат тесту. Якщо показники груп А та А не відрізняються, можна довіряти даним. У разі відмінностей, цифри спотворені, і рішення на основі тесту слід утримати.

Рис. 2.5. Принцип A/B тестування

Якщо у вас існують більше ніж два варіанти для порівняння, можна використовувати мультиваріантне тестування. Принцип є схожим до A/B-тесту, але ви порівнюєте одночасно більше ніж дві версії однієї зміни. Кожному варіанту виділяється певна частина аудиторії для показу, і на завершення тесту результати порівнюються. Перемагає версія, яка показала найкращі метрики.

Мультиваріантне тестування дозволяє вам оцінити вплив кількох різних змін одночасно, що може бути корисним в складних сценаріях. Забезпечуючи аудиторії різними комбінаціями змін, ви можете визначити оптимальну конфігурацію, яка максимізує бажані метрики.

Процес мультиваріантного тестування включає в себе вибір змін, розподіл аудиторії між різними варіантами, виконання тесту та аналіз результатів. Пам'ятайте, що важливо враховувати статистичну значущість і контролювати можливі фактори спотворення результатів тесту.

Мультиваріантне тестування є потужним інструментом для оптимізації різноманітних елементів веб-сайту або продукту, але важливо правильно планувати та виконувати його для отримання надійних результатів.

Рис. 2.6. Мультиваріантне тестування

Метод мультиваріантного тестування ефективно дозволяє перевіряти кілька версій з невеликими змінами. Наприклад, можна провести тестування чотирьох різних фраз "виклик до дії" для однієї кнопки. Крім того, цей метод зручно застосовувати для оцінки метрик при різних комбінаціях. Наприклад, якщо у вас є чотири варіанти тексту для кнопки та два кольори, мультиваріантним тестуванням можна порівняти вісім можливих версій та визначити, яка комбінація показала найкращі результати. Щодо A/B-тестування, процес включає в себе вісім кроків:

Сформулювати гіпотези: Гіпотеза допомагає визначити цілі експерименту, вибрати метрики та інтерпретувати результати.

Визначити метрики: Метрика A/B-тесту – це показник, за яким оцінюється, чи підтвердилася гіпотеза чи ні.

Визначення критеріїв успіху та прийняття рішень на основі результатів тестування: Критерій успіху – це очікуваний вихід, на основі якого можна зробити висновок про успішність проведеного тесту.

Віддати колегам на крос-рев'ю: Крос-рев'ю допомагає оцінити сформульовані гіпотези, врахувати всі метрики та перевірити правильність рішень.

Підготувати експеримент: Створення версії зі змінами для тестування, визначення контрольної та експериментальної вибірки, мінімального розміру вибірки, показника статистичної значущості та тривалості тестування.

Запустити експеримент: Перевірка коректності роботи A/B-тесту та уникнення виявлення значущих різниць вже на початкових етапах.

Проаналізувати результати та зробити висновки: Порівняння отриманих даних з критеріями успіху, визначеними на етапі готування тесту.

Зробити звіт та поділитися ним із колегами: Розповідь про результати тестування для розуміння всією командою.

Інші методи для перевірки гіпотез включають юзабіліті-тестування, яке оцінює, наскільки інтерфейс є зручним для користувачів, та fake door тест, який дозволяє перевірити, чи потрібна користувачам нова функція, додаючи її на рівні макетів. Також можна використовувати реліз нового товару на обмежену аудиторію для тестування великих змін бізнес-моделі або нового продукту. A/B-тести дозволяють швидко тестувати гіпотези та розвивати продукт.

2.6. Висновки до розділу

У другому розділі досліджено використання мікросервісного підходу, який надає інженерам можливість працювати незалежно один від одного та розробляти окремі сервіси з використанням різних підходів та мов програмування. Цей метод проектування додатків як сукупності мікросервісів забезпечує масштабованість системи та розподіл додатка на різні фізичні та віртуальні машини, що також підвищує надійність. Зазвичай, мікросервіси запускаються в окремих ізольованих контейнерах та використовують контролер масштабованості, наприклад, Kubernetes, для управління та масштабування окремих сервісів згідно з робочими навантаженнями системи.

Як приклад платформи контейнеризації, був наведений Docker – інструментарій для управління ізольованими контейнерами. Такий архітектурний

підхід, що базується на незалежних службах та контейнеризації, набирає популярність серед розробників.

Вказано, що технології, такі як Docker та Kubernetes, надають інженерам можливість розробляти надійні, гнучкі та масштабовані системи. Однією з переваг є можливість розгортання та перезапуску мікросервісів незалежно один від одного, що робить цей підхід більш гнучким і ефективним в порівнянні з монолітними системами.

Зазначено, що стрімліт робить процес створення веб-застосунків на мові Python більш доступним і зручним для розробників, що спеціалізуються на аналізі даних та штучному інтелекті.

Подальше розглядання зазначає доцільність проведення A/B-тестування мікросервісу для кластеризації користувачів у додатку. Цей метод є ефективним при локальному застосуванні продукту та випробуванні значних змін у бізнес-моделі чи випробуванні нового продукту.

РОЗДІЛ 3.
МЕТОДИ ТА МОДЕЛІ МАШИННОГО НАВЧАННЯ

3.1. Машинне навчання для аналізу даних

Технологічно ринок машинного навчання швидко розширюється. Починаючи з 2016 року, його розмір перевищив позначку у $1 млрд, і прогнозується, що до 2025 року він може збільшитися до $39,98 млрд. В кінці 2016 року MIT Technology Review та Google Cloud провели спільне дослідження на тему "Машинне навчання: новий спосіб отримати конкурентну перевагу". У цьому опитуванні брали участь 375 кваліфікованих респондентів з різних країн світу, представників як дрібних, так і великих компаній у сферах промисловості, послуг та фінансів. Згідно з результатами дослідження, 60% компаній вже використовують машинне навчання (ML), і третина з них перейшла від стадії інновацій до стадії зрілості. Додатково, 26% компаній стверджують, що вони вже отримують конкурентну перевагу завдяки ML. Чверть компаній вкладає понад 15% від коштів, призначених на розвиток IT, в машинне навчання, і виявляє значну віддачу від цих інвестицій.

Машинне навчання, зокрема, нейронні мережі, виявляється ефективним інструментом для розв'язання бізнес-завдань у випадках, коли:

Великий обсяг різноманітних даних: застосування машинного навчання виправдано, коли є обширні та різнорідні дані, для яких важко розробити традиційні програми обробки та систематизації.

Спотворені, неповні або не систематизовані дані: машинне навчання може впоратися з викликами спотворених чи неповних даних, вдосконалюючи їх та витягаючи цінну інформацію.

Різноманітність даних: коли дані настільки різноманітні, що складно визначити зв'язки і закономірності, машинне навчання може виявити непередбачувані взаємозв'язки.

Машинне навчання дозволяє ефективно вирішувати різноманітні бізнес-завдання, такі як [31]:

Прогнозування: прогнозування попиту, обсягу продажів, наповнення складу, завантаження устаткування та інших ресурсів для покращення стратегій розвитку підприємства.

Виявлення: виявлення тенденцій, прихованих взаємозв'язків, аномалій та повторюваних елементів для забезпечення кращого прийняття рішень.

Використання машинного навчання в бізнесі дозволяє реалізовувати різноманітні завдання, такі як розпізнавання фото-, відео-, аудіоконтенту, а також виявлення шахрайства, брехні, внутрішніх загроз та зовнішніх атак на систему безпеки.

Основні застосування машинного навчання включають:

Розпізнавання: Машинне навчання дозволяє ефективно розпізнавати фото-, відео-, та аудіоматеріали. Також воно застосовується для виявлення спроб шахрайства, брехні, інтернальних загроз та зовнішніх атак на систему безпеки.

Автоматизація: Машинне навчання використовується для автоматизації роботи операторів в онлайн-чатах та телефонних операторів, щоб покращити ефективність обслуговування клієнтів.

Класифікація: Важливим використанням є класифікація та сегментація покупців, клієнтів та замовників за різними параметрами для отримання більш точного розуміння аудиторії.

Кластеризація: Машинне навчання допомагає в кластеризації за параметрами, які з самого початку не були відомі, щоб виявити нові підходи до категоризації.

Розробка: В компаніях активно використовується машинне навчання для розробки чат-ботів, що поліпшують комунікацію з клієнтами та оптимізують обслуговування.

Прикладами успішного застосування машинного навчання є технологічний гігант Alibaba, який широко використовує інтелектуальний аналіз для персоналізації віртуальних вітрин та оптимізації пошуку. Також, американська

роздрібна мережа Target використовує машинне навчання для передбачення поведінки покупців, а платформа Neuromation з України працює над створенням штучного навчального середовища для глибокого навчання нейронних мереж.

Усі моделі машинного навчання поділяються на контрольоване, неконтрольоване та навчання з підкріпленням. Контрольоване та неконтрольоване навчання далі класифікуються як різні терміни. Розглянемо кожен із них детально [33].

1. **Контрольоване навчання:**

Навчання під наглядом: Проста модель машинного навчання, яка вивчає базову функцію, що відображає вхідні дані на вихідні.

Приклад: Прогнозування зросту людини на основі віку.

Класифікація:

Опис: Задача прогнозного моделювання, де мітка передбачається для заданих вхідних даних.

Приклади застосування: Фільтрація спаму, розпізнавання рукописних символів, аналіз настроїв.

Регресія:

Опис: Модель, де вихід завжди безперервний. Використовується для прогнозування зарплат, віку, температури тощо.

Приклад: Прогнозування ціни авіаквитка.

2. **Навчання без контролю:**

Опис: Використовується для виявлення шаблонів вхідних даних без позначених результатів. Застосовується для сегментації клієнтів, дослідницького аналізу даних.

Приклад: Зменшення розмірності, кластеризація.

Кластеризація:

Опис: Групування точок даних для виявлення шаблонів. Застосовується для класифікації документів, сегментації клієнтів.

Приклад: Алгоритми кластеризації: ієрархічна кластеризація, кластеризація на основі щільності.

Зменшення розмірності:

Опис: Зменшення випадкових величин для отримання головних змінних. Алгоритм аналізу головних компонент використовується для зменшення розмірності.

Ці моделі вирізняються за своїми завданнями та методами навчання, що робить їх ефективними для вирішення різноманітних проблем в галузі машинного навчання.

3. Підкріплення машинного навчання:

Опис: Схожа на кероване машинне навчання, але використовує метод проб і помилок для навчання. Це часто використовується в іграх, навігації, робототехніці тощо.

Типи алгоритмів машинного навчання:

Лінійна регресія:

Опис: Знаходження рядка, який найкращим чином відповідає даним. Включає множинну лінійну регресію та поліноміальну регресію.

Логістична регресія:

Опис: Використовується для отримання кінцевої кількості результатів (зазвичай двох) замість лінійної регресії. Результати в діапазоні від 0 до 1.

Дерево рішень:

Опис: Використовується в стратегічному плануванні та дослідженні операцій. Складається з вузлів, які утворюють дерево рішень.

Випадковий ліс:

Опис: Ансамблевий метод, заснований на деревах рішень. Випадково вибирає підмножину змінних на кожному кроці дерева.

Підтримуюча векторна машина:

Опис: Знаходження межі між класами даних для максимізації запасу між ними.

Аналіз основних компонентів (PCA):

Опис: Зменшення розмірності шляхом проектування більш вимірних даних у менший простір. Використовується для збереження вихідних значень в моделі.

Ці алгоритми використовуються для вирішення різноманітних завдань у сфері машинного навчання, забезпечуючи ефективні та точні результати в різних контекстах.

Наприклад, метод головних компонент (PCA) є корисним інструментом для інтерпретації опитувань з великою кількістю запитань або змінних, таких як ті, що стосуються добробуту, культури навчання чи поведінки. (techukraine.net) Використання моделі PCA дозволяє зменшити кількість змінних до мінімуму, при цьому зберігаючи значущість інформації.

Наївний Байєс:

Опис: Використовується в науці про дані та інших галузях. Заснований на теоремі Байєса, визначає ймовірність вихідної змінної при вході P.

Нейронна мережа:

Опис: Мережа математичних рівнянь, що приймає вхідні дані, проходить через різні шари та повертає результати в одній або кількох вихідних змінних.

Алгоритм K-найближчих сусідів (KNN):

Опис: Використовується для класифікації та регресії. Класифікує точки даних за допомогою голосування їхніх k найближчих сусідів.

Кластеризація K-Means:

Опис: Модель неконтрольованого машинного навчання для кластеризації даних. Утворює K кластерів, вибираючи центроїди та класифікуючи точки навколо них.

Ці алгоритми, такі як Наївний Байєс, Нейронна мережа, KNN та K-Means, різноманітні та використовуються для різних завдань у науці про дані та машинному навчанні. Вони вирішують класифікаційні, регресійні та кластеризаційні завдання, роблячи процеси більш ефективними та точними.

3.2. Кластеризація як інструмент продуктової аналітики

Кластерний аналіз є потужним інструментом у сфері аналізу даних, широко використовується в бізнесі. В основному, кластерний аналіз застосовується для

групування схожих об'єктів або осіб, базуючись на їхніх характеристиках чи поведінці. Ось кілька прикладів використання кластерного аналізу в бізнесі:

1. Кластерний аналіз допомагає визначити групи схожих між собою клієнтів за різними характеристиками на ринку. Це дозволяє компаніям ліпше розуміти свою цільову аудиторію, налаштовувати маркетингові стратегії та персоналізувати продукти та послуги для кожного сегмента.

2. Кластерний аналіз допомагає компаніям класифікувати свої продукти або послуги на основі спільних характеристик або властивостей. Це може допомогти в управлінні асортиментом, позиціонуванні продуктів та розробці цілеспрямованих маркетингових стратегій для кожного кластеру.

3. Кластерний аналіз дозволяє ідентифікувати групи споживачів зі схожою поведінкою, покупками чи споживацькими звичками. Це може бути корисним для передбачення майбутніх тенденцій, розробки персоналізованих пропозицій, вдосконалення стратегій управління клієнтами та розуміння того, як можна підвищити продажі.

4. Кластерний аналіз можна використовувати для виявлення незвичайних або аномальних моделей, що можуть свідчити про зловмисницьку чи шкідливу діяльність або помилки в процесах бізнесу.

5. Кластеризація може бути використана для аналізу фінансових даних та виявлення ризиків та можливостей. Наприклад, кластеризація може бути використана для класифікації компаній за різними показниками, такими як обсяги продажів, витрати на рекламу, рентабельність тощо.

6. Кластеризація дозволяє групувати користувачів за спільними інтересами, допомагаючи забезпечити більш ефективну співпрацю та спілкування.

7. Кластерний аналіз знаходить застосування в різних сферах, таких як сегментація ринку, аналіз соціальних мереж, групування результатів пошуку, медична візуалізація, сегментація зображень та виявлення аномалій. В бізнесі він допомагає компаніям краще розуміти своїх клієнтів та оптимізувати стратегії для досягнення конкурентної переваги.

Кластеризація, або кластерний аналіз, представляє собою статистичну процедуру, спрямовану на розбиття вибірки об'єктів на підмножини, які не перетинаються і називаються кластерами. Кожен кластер об'єднує схожі об'єкти, тоді як об'єкти різних кластерів виявляють суттєві відмінності один від одного. Кластерний аналіз входить до класу завдань навчання без вчителя, де не фіксується приналежність об'єктів до конкретних класів [31].

Методи створення кластерів можуть використовувати різні підходи, такі як відстань між об'єктами, щільність ділянок у просторі даних, інтервали або конкретні статистичні розподіли. Вибір конкретного методу залежить від характеристик конкретного датасету та визначеної мети використання отриманих результатів.

Кластерний аналіз не є автоматизованим процесом, але ітераційним, оскільки часто необхідно змінювати метод обробки даних та параметри моделі. Існують декілька причин неоднозначності рішення в задачі кластеризації, таких як відсутність єдиного критерію якості, необхідність попереднього визначення кількості кластерів і суб'єктивний вибір метрики.

Враховуючи ці особливості, кластерний аналіз залишається потужним інструментом для розгляду різноманітних даних у бізнесі та інших галузях.

Щодо формальної постановки задачі: Маємо вибірку $X\ell = \{x1, \ldots, x\ell\} \subset X$ і функцію відстані між об'єктами $\rho(x, x')$. Треба розбити вибірку на підмножини, які не будуть перетинатися і щоб кожен кластер складався з об'єктів, близьких по метриці ρ, а об'єкти різних кластерів істотно відрізнятись. При цьому кожному об'єкту $xi \in X\ell$ приписується мітка (номер) кластера yi. Алгоритм кластеризації — це функція a: $X \to Y$, що будь-якому об'єкту $x \in X$ ставить у відповідність мітку кластера $y \in Y$. Множина міток Y в деяких випадках відома заздалегідь, однак частіше завдання полягає в тому, щоб визначити оптимальне число кластерів з точки зору того чи іншого критерію якості кластеризації. Задача групування набору об'єктів полягає в тому, що об'єкти в одному кластері більш схожі один на одного, ніж об'єкти в інших кластерах [31].

Подібність визначає ступінь взаємозв'язку між об'єктами, і кластеризація виступає як ефективний інструмент для видобутку даних, знайдення застосувань у машинному навчанні, розпізнаванні образів, аналізі зображень, пошуку інформації, біоінформатиці, стисненні даних та комп'ютерній графіці. Існують два основних типи кластеризації: жорстка і м'яка.

У жорсткій кластеризації кожен об'єкт повністю належить одному кластеру або не належить жодному. У м'якій кластеризації об'єкт може мати ймовірність належати більше ніж одному кластеру.

Фактори для визначення кластеризації включають типи вхідних даних та проблему виміру відстані:

Об'єкти описуються набором характеристик, що можуть бути числовими або нечисловими.

Матриця відстаней між об'єктами визначається відстанями між об'єктами навчальної вибірки.

Дані, які використовуються в кластерному аналізі, можуть бути інтервального, порядкового або категоріального типу.

Алгоритми кластеризації є суб'єктивними, і може існувати більше одного правильного підходу для їх вибору. Кожен алгоритм має свій набір правил для визначення "подібності" між об'єктами даних. Вибір належного алгоритму для конкретної задачі часто вимагає експериментального підходу, оскільки немає одного математично обґрунтованого варіанту вибору алгоритму кластеризації.

Алгоритми кластеризації класифікуються за кластерною моделлю, яка ґрунтується на тому, як вони формують кластери. Наприклад, кластеризація на основі зв'язку використовує відстань між точками даних для визначення, наскільки вони споріднені. Кластери формуються з'єднанням точок даних відповідно до їх відстані. Кластеризація на основі центроїда використовує центральний вектор для представлення кластерів, і цей центроїд визначається в ітеративних алгоритмах, де подібність визначається близькістю точок даних до центроїда кластера.

3.3. Методи та моделі вирішення задач класифікації

Аналіз математичних моделей та методів кластерного аналізу є важливою складовою обробки даних та вирішення заданої проблеми. Існує багато моделей та методів кластеризації, які використовуються в дослідженнях та практичних застосуваннях. Деякі з них включають [33]:

– Агломеративне згортання: Цей метод починає з кожного об'єкта як окремого кластера, а потім послідовно зливає найбільш схожі кластери, поки не буде отримано один загальний кластер або досягнуто певної умови зупинки. Даний метод має здатність розпізнавати кластери кластери різної форми та розміру, проте його обчислювальна складність є високою, а масштабованість для великих наборів даних недостатньою.

– DBSCAN (Density-Based Spatial Clustering of Applications with Noise): Цей метод заснований на щільності даних і визначає кластери, виявляючи області високої щільності. Він визначає ядро кластера, об'єкти в межах якого мають достатньо близьких сусідів, та розріджені області між кластерами. Даний метод автоматично визначає шумові (аномальні) точки, проте є чутливим до параметрів, таких як радіус та мінімальна кількість сусідів, а також не підходить для даних з високою розмірністю.

– Ієрархічний кластерний аналіз: Цей метод будує ієрархію кластерів шляхом послідовного об'єднання або розбиття кластерів на основі відстаней або схожості між ними. Результатом є дерево кластерів, яке можна візуалізувати у вигляді дендрограми, що є досить зручним. В залежності від особливостей досліджуваних даних можна обрати один із двох підходів: агломеративний або дивізивний. Але даний метод має високу обчислювальну складність та чутливість до шуму, а також відсутність можливості змінити рішення щодо кластеризації.

– Метод k-середніх (k-means): детальніше про цей метод у наступному підрозділі.

Кожен з цих методів має свої переваги та недоліки і підходить для різних типів даних та завдань аналізу. Вибір певного методу залежить від конкретної ситуації, обсягу даних, характеристик досліджуваних об'єктів та мети дослідження.

Ієрархічна кластеризація – це метод кластерного аналізу, спрямований на створення ієрархії кластерів, існують два типи стратегій цього методу:

Дивізійна стратегія («згори донизу»): У цьому підході всі спостереження спочатку об'єднуються в одному кластері, а потім рекурсивно розщеплюються вниз по ієрархії.

Агломеративна стратегія («знизу догори»): Кожне спостереження спочатку розташовується у власному кластері, а потім пари кластерів об'єднуються, пересуваючись вгору по ієрархії.

Алгоритм агломеративної кластеризації включає такі етапи:

Кожна точка розташовується у своєму кластері.

1. Попарні відстані між центрами кластерів упорядковуються за зростанням.
2. Пара найближчих кластерів об'єднується в один, перераховується центр кластера.
3. Процедура повторюється, доки всі дані не об'єднаються в один кластер.

Процес об'єднання кластерів може використовувати різні методи:

Single linkage: Мінімум попарних відстаней між точками.

Complete linkage: Максимум попарних відстаней між точками.

Average linkage: Середнє значення попарних відстаней між точками.

Centroid linkage: Відстань між центроїдами двох кластерів.

Перевага перших трьох методів полягає в тому, що для них не потрібно перераховувати відстані кожен раз після об'єднання, що знижує обчислювальну складність алгоритму. За результатами виконання алгоритму будується дендрограма, яка вказує, на якому етапі найкраще зупинити алгоритм [31].

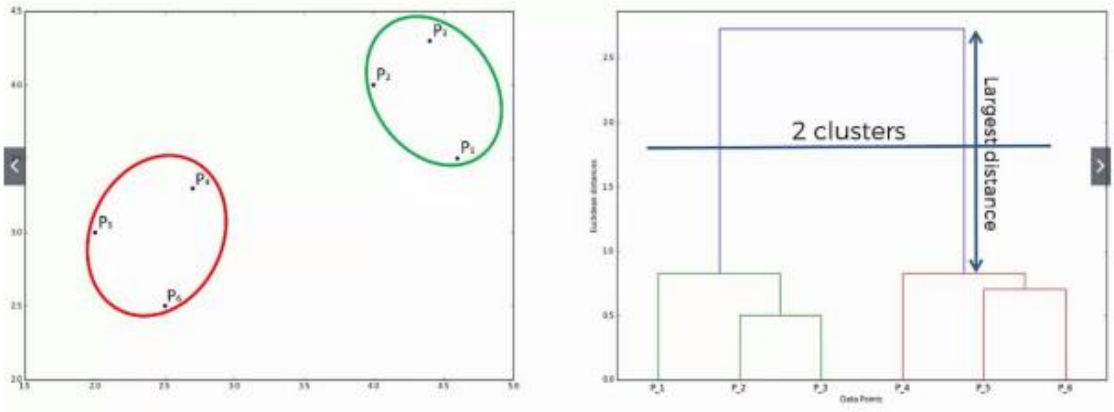


Рис.3.1. Результати виконання алгоритму

Основним принципом роботи алгоритму k-середніх (k-means) є оптимальне розбиття множини даних на k кластерів. Алгоритм намагається згрупувати дані в кластери так, щоб досягти екстремуму цільової функції розбиття.

Для пояснення роботи алгоритму, нехай об'єкти, які підлягають кластеризації, описуються вектором параметрів *Xr*. Ми визначаємо множину кластерів та їхні ядра *Ck* як типові об'єкти свого кластера.

Оцінка близькості об'єкта до ядра використовує евклідову міру близькості *D(xp,ck).* Ця міра є тим менше, чим більше об'єкт схожий на ядро класу. Таким чином, під час розбиття на кластери ми мінімізуємо сумарну міру близькості для всієї множини вхідних об'єктів.

Алгоритм k-середніх складається з наступних етапів:

Ініціалізація:

Випадковим чином вибрати k об'єктів як початкові центри кластерів (ядра).

Присвоєння кластерів:

Для кожного об'єкта знаходити найближче ядро і призначати йому відповідний кластер.

Оновлення центрів кластерів:

Перерахувати центри кластерів як середнє арифметичне всіх об'єктів у кожному кластері.

Повторення:

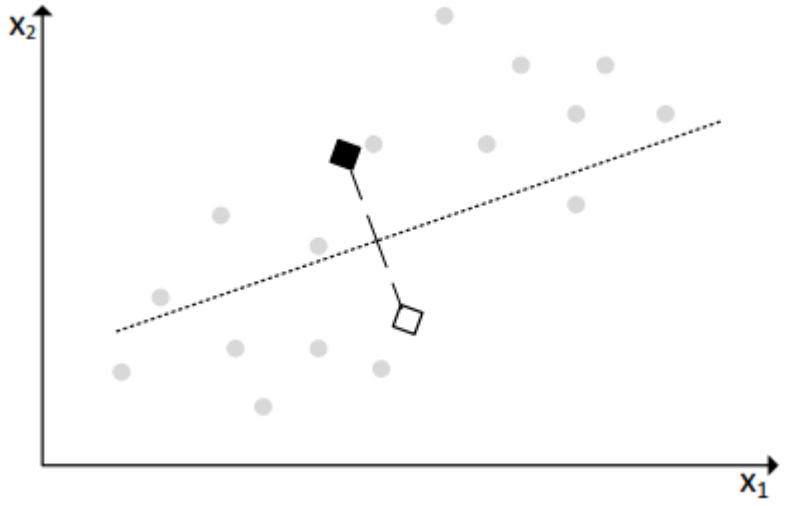
Повторювати кроки 2-3 до збіжності (коли кластери залишаються стабільними).

На кожному кроці алгоритму прагнемо мінімізувати сумарну квадратичну відстань між кожним об'єктом та центром його призначеного кластера. Ця мета визначається цільовою функцією, яка під час виконання алгоритму поступово зменшується.

Цей процес триває до тих пір, поки кластери збігнуться і об'єкти в межах кожного кластера будуть максимально схожими між собою, що відповідає досягненню екстремуму цільової функції.

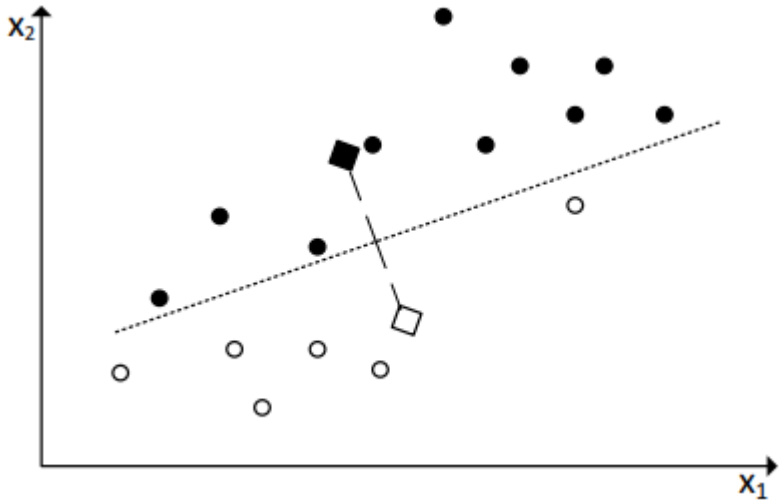
$$D(x^p, c^k) = \sum_p \sum_i (x_i^p - c_i^k)^2 \to min$$

Розглянемо загальний процес з визначеною кількістю груп k (кількість груп визначається заздалегідь відповідно до конкретної задачі, і початкові значення груп можуть бути вибрані випадковим чином, ідентичними або відповідно до інших евристичних правил) [31].

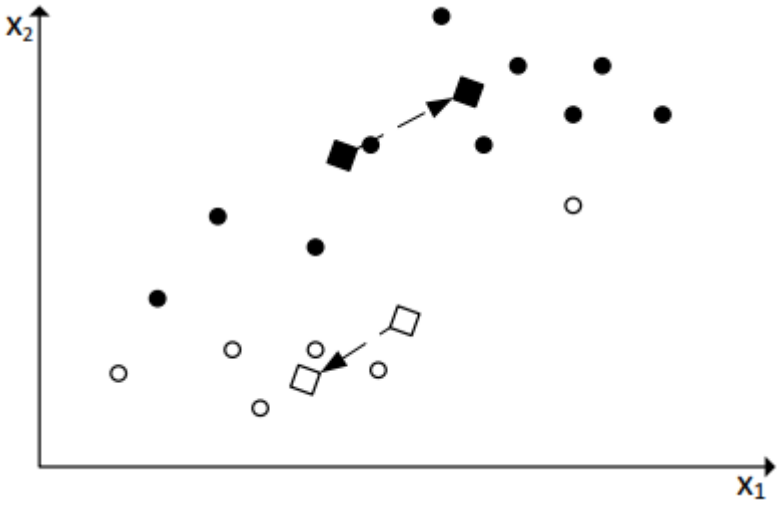


Кожна ітерація алгоритму складається з двох етапів [31]:

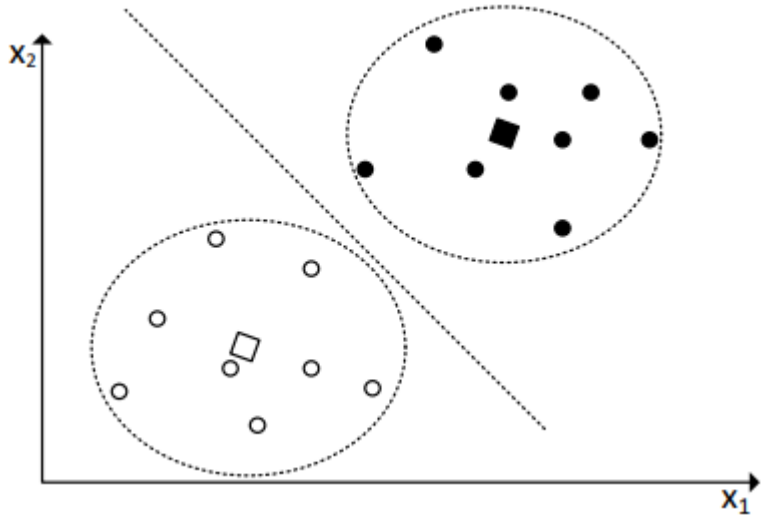
1. За незмінних ядер шукається таке розбиття об’єктів на класи, щоб мінімізувати сумарну міру близькості між об’єктами та ядрами класів: min *{Σ D (xp , ck)}*. Результат етапу – створення функції, що розбиває об’єкти на класи.



2. Під час налаштування класів за незмінного розбиття ядра, стараємося забезпечити такі параметри, щоб в межах кожного класу сумарна міра близькості між ядром цього класу та об'єктами, що до нього належать, була мінімальною. На виході цього етапу формується новий набір ядер.



Ітерації повторюються доти допоки розбиття не стабілізується.

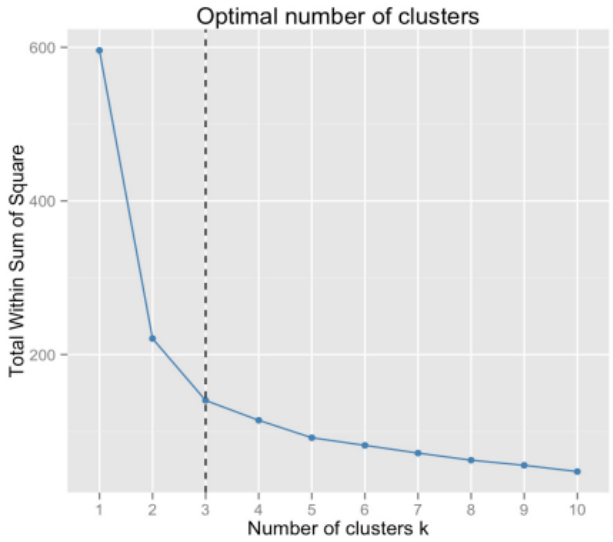


На кожному етапі розбиття, для визначення оптимальної кількості кластерів, проводиться послідовне збільшення їх кількості. Для кожного кластера оцінюється сумарна відстань від його ядра до об’єктів, що до нього відносяться. Після цього розраховується сумарна міра щільності кластерів.

$$WCSS = \sum_{cluster_i} \sum_{object_j} dist(c_i, o_j)$$

Сумарна міра щільності кластерів зменшується при збільшенні кількості кластерів. Оптимальна кількість кластерів визначається як точка, де швидкість цього зменшення досягає максимального значення (застосовується правило ліктя).

$$\frac{WCSS_t - WCSS_{t+1}}{WCSS_{t-1} - WCSS_t} \rightarrow min$$



Приклад визначення фінальних центроїдів для алгоритму k-means.

```
    def final_centroids(self):
        """
        Повертає:
        clusters -- список масивів, що містять точки кожного кластеру
        centroids -- масив розміру (кількість_кластерів, кількість_факторів),
            що містить фінальні центроїди
        """

        c = self.initialize_centroids()
        mvc = self.move_centroids(c)
        while np.any(c != mvc):
          c = mvc
          mvc = self.move_centroids(mvc)
        centroids = mvc

        cc = self.closest_centroid(centroids)
        clusters = []
        for i in range(self.k):
          p9 = []
          for j in range(len(cc)):
            if cc[j] == i:
              p9.append(self.X[j])
          clusters.append(np.array(p9))

        return clusters, centroids
```

Для реалізації алгоритму k-means важливо визначити значення k, що відповідає кількості кластерів. Хоча візуальний аналіз у деяких випадках може надати інформацію щодо кількості кластерів, цей метод не завжди є достатньо

надійним. Для визначення оптимальної кількості кластерів часто використовується метод "ліктя" - графічне відображення оптимального значення k при використанні алгоритму k-середніх.

Візуальний аналіз "ліктя" допомагає визначити оптимальну кількість кластерів, розраховуючи середню суму квадратів відстаней між точками та центроїдами всередині кожного кластеру.

$$W_k = \frac{1}{k}\sum_{i=1}^{k}\sum_{j}^{n_s} || \, x_{ij} \, - \, c_i \, ||^2$$

де k – кількість кластерів, n_S – кількість точок в кластері S.

Приклад реалізації функції середніх відстаней для побудови графіка візуального «ліктя»

```
def mean_distances(k, X):
Аргументи:
k
--
X
int, кількість кластерів
np.array, матриця вхідних даних
Повертає:
Масив розміру (k, ), що містить середні значеннння сум відстаней від центроїда до кожної точки кластеру для k кластерів
re = []
for ki in range(1, k+1):
s = 0
model =
KMeans (X, ki)
cl, fc =
model.final_centroids()
for n in range(ki):
s1 = 0
for i in cl[n]:
s1 += (np.linalg.norm(i - fc[n]))**2
s += s1
re.append(s / ki)
return np.array(re)
```

Якщо визначення кількості кластерів ускладнене, можна використовувати алгоритм g-means. Цей алгоритм визначає кількість кластерів в моделі, використовуючи послідовний статистичний тест на те, чи дані всередині кластера підпорядковуються гауссівському (Gaussian, звідси назва алгоритму) закону розподілу. Процес роботи алгоритму такий:

Початкове розбиття: Починаємо зі створення початкового розбиття, наприклад, використовуючи k-means.

Статистичний тест: Для кожного кластера виконується статистичний тест на гауссівський розподіл даних всередині кластера.

Поділ кластера: Якщо статистичний тест вказує на те, що дані не підпорядковуються гауссівському розподілу, кластер розбивається на два нових кластера. Ці нові кластери мають центри, розташовані на осі головних компонент попереднього кластера.

Повторення тесту: Процес статистичного тесту та поділу кластера повторюється для кожного кластера, що пройшов тест.

Важливо пам'ятати, що алгоритми k-means та g-means спрямовані на гіпотезу про компактність, що передбачає, що дані навчальної вибірки утворюють компактні області. В іншому випадку кластери, знайдені цими алгоритмами, можуть бути малоінформативними.

Для дослідження було обрано набір даних «Customer segmentation dataset», який містить 541909 записів покупок додатку більше ніж 2000 покупців та 8 характеристик цих покупок (номер , унікальний код для кожної трансакції, опис кожної фічі, кількість придбаних додатків, дата покупки, ціна одної фічі, унікальне ID кожного покупця та країна проживання) між 01.12.2010 та 09.12.2011. Таким чином, обсяг даних є занадто великим, щоб проаналізувати його неозброєним оком. Саме тому доцільним буде використання алгоритмів машинного навчання для знаходження певних висновків.

Спершу варто загально проаналізувати базу даних.

```
import pandas as pd
import numpy as np
from datetime import timedelta
import matplotlib.pyplot as pit
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
```

Рис. 3.2. Імпорт бібліотек Python, необхідних для використання

```
df = pd.read_csv('OnlineRetail.csv')
df.drop('Unnamed: 0', axis = 1, inplace = True)
df.head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

Рис. 3.3. Перші п’ять спостережень заданої бази даних

```
df.describe()
```

	Quantity	UnitPrice	CustomerID
count	541909.000000	541909.000000	406829.000000
mean	9.552250	4.611114	15287.690570
std	218.081158	96.759853	1713.600303
min	-80995.000000	-11062.060000	12346.000000
25%	1.000000	1.250000	13953.000000
50%	3.000000	2.080000	15152.000000
75%	10.000000	4.130000	16791.000000
max	80995.000000	38970.000000	18287.000000

Рис. 3.4. Опис заданої бази даних

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  object
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

Рис. 3.5. Інформація про задану базу даних

На рис 3.5. видно, що в базі даних відсутні деякі значення в стовпцях Description і CustomerID. Враховуючи унікальність ID кожного покупця, немає сенсу заповнювати пропущені значення, тому ці рядки можна просто викинути. Відсутні значення стовпця Description можна замінити на пусте текстове поле, щоб зберегти інші дані щодо цих записів.

```
df.drop(df[df["CustomerID"].isnull()].index, axis=0, inplace=True)
df["Description"] = df["Description"].fillna("")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 406829 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   InvoiceNo    406829 non-null  object 
 1   StockCode    406829 non-null  object 
 2   Description  406829 non-null  object 
 3   Quantity     406829 non-null  int64  
 4   InvoiceDate  406829 non-null  object 
 5   UnitPrice    406829 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      406829 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 27.9+ MB
```

Рис. 3.6. Інформація про оновлену базу даних після усунення відсутніх значень

У стовпці Quantity також наявні від'ємні значення. Це свідчить про скасовані замовлення, тому варто забрати їх разом із відповідними додатними замовленнями.

```
size_before = len(df)
neg_quantity = df[df["Quantity"] < 0][["CustomerID", "StockCode", "Quantity"]].sort_val
print(f"Negative number of objects: {len(neg_quantity)}")
filters = df[df["CustomerID"].isin(neg_quantity["CustomerID"]) & df["StockCode"].isin(n
p_count = []
for index, series in neg_quantity.iterrows():
  customer = series["CustomerID"]
  code = series["StockCode"]
  quant  = -1 * series["Quantity"]
  counterpart = filters[(filters["CustomerID"] == customer) & (filters["StockCode"] ==
  p_count.extend(counterpart.index.to_list())

to_drop = neg_quantity.index.to_list() + p_count
df.drop(to_drop, axis=0, inplace=True)
print(f"Deleted {size_before - len(df)} raws from DB")
```

```
Negative number of objects: 8905
Deleted 14468 raws from DB
```

Рис. 3.7. Видалення з бази даних записів про скасовані замовлення

Завдання полягає в тому, щоб допомогти компанії збільшити їхню конкурентоспроможність за рахунок проведення сегментації покупців. Як було описано в попередньому розділі, кластерний аналіз ефективно працює з таким видом проблем. Маючи саме такий набір даних, доцільно буде провести RFM-аналіз (Recency, Frequency, Monetary value), щоб розділити покупців на групи зі схожою споживацькою поведінкою, а оскільки дані будуть представлені саме у числовому форматі та структуру кластерів можна представити лінійно, найбільш оптимальним рішенням буде використати саме алгоритм k-середніх.

На Рис. 3.8. показано створення нових пов'язаних із часом характеристик шляхом вилучення їх із стовпця InvoiceDate. Перед цим, треба перетворити стовпець InvoiceDate з типу object у тип datetime

```
df["InvoiceDate"] = pd.to_datetime(df["InvoiceDate"])

# extracting time related features from InvoiceDate

df["InvoiceDateDay"] = df["InvoiceDate"].dt.date
df["InvoiceDateTime"] = df["InvoiceDate"].dt.time
df["InvoiceYear"] = df["InvoiceDate"].dt.year
df["InvoiceMonth"] = df["InvoiceDate"].dt.month
df["InvoiceMonthName"] = df["InvoiceDate"].dt.month_name()
df["InvoiceDay"] = df["InvoiceDate"].dt.day
df["InvoiceDayName" ] = df["InvoiceDate"].dt.day_name()
df["InvoiceDayOfWeek"] = df["InvoiceDate"].dt.day_of_week
df["InvoiceWeekOfYea r"] = df["InvoiceDate"].dt.isocalendar().week
df["InvoiceHour"] = df["InvoiceDate"].dt.hour
```

Рис. 3.8. Створення нових часових характеристик даних

Для того, щоб провести RFM-аналіз, потрібно створити нову базу даних, де кожен рядок відповідає окремому покупцю, а стовпці – це Recency, Frequency, Monetary.

```
r_date = df["InvoiceDateDay"].max() + timedelta(days=1)
df_customers = df.groupby('CustomerID').agg({
  "InvoiceDateDay": lambda sv: (r_date - sv.max()).days,
  "InvoiceNo": "count",
  "TotalValue": "sum"
}).rename(columns = {
  "InvoiceDateDay": 'Recency',
  "InvoiceNo": 'Frequency',
  "TotalValue": 'Monetary'
})
df_customers.head()
```

CustomerID	Recency	Frequency	Monetary
12347.0	3	182	4310.00
12348.0	76	31	1797.24
12349.0	19	73	1757.55
12350.0	311	17	334.40
12352.0	37	67	1405.28

Рис. 3.9. Побудова нової бази даних на основі RFM-аналізу

На Рис. 3.10. показано усунення зайвих викидів, щоб зменшити шум у свторенному датафреймі.

```
def remove_outliers(df, col, threshold=1.5):
    Q1 = np.quantile(df[col], .25)
    Q3 = np.quantile(df[col], .75)
    IQR = Q3 - Q1
    df.drop(df[(df[col] < (Q1 - threshold * IQR)) | (df[col] > (Q3 + threshold * IQR))

    return df

for col in df_customers.columns:
    size_before = len(df_customers)
    df_customers = remove_outliers(df_customers, col)
    print(f"Removed {size_before - len(df_customers)} outliers from {col}")
```

```
Removed 140 outliers from Recency
Removed 363 outliers from Frequency
Removed 316 outliers from Monetary
```

Рис. 3.10. Видалення шуму з бази даних

Дані готові до аналізу.

3.4. Алгоритм к-середніх і особливості його реалізації на Python

Основна ідея алгоритму k-середніх полягає в тому, щоб згрупувати спостереження в k кластерів, де k – попередньо визначене число в залежності від набору даних. Далі наведено поступові кроки алгоритму та приклад їх виконання за допомогою Python.

Загальна структура алгоритму виглядатиме так:

– Клас KMeans:
  - Ініціалізація центроїдів;
  - Знайти найближчі центроїди для кожної точки;
  - Знайти оптимальні центроїди.

– Пошук оптимального значення k:

○ Метод «ліктя».

1. Клас KMeans

Припустимо, що ми маємо вхідні дані $x_1, x_2, x_3, \dots, x_n \in X$ і значення k:

1.1. Створюємо клас KMeans з параметррами:

де x – Матриця вхідних даних;

к – кількість кластерів.

```
class KMeans(object):


  def init  (self, X, k) :
    self.X = X
    self.k = k
```

Рис. 3.11. Створення класу KMeans

1.2. Обираємо k випадкових точок, які будуть кластерними центроїдами Нехай C – набір центроїдів, тоді c_1, c_2, c_3, … , $c_k \in C$ – центроїди. Ініціалізація центроїдів відіграє важливу роль і напряму впливає на швидкість роботи цього алгоритму.

```
    def initialize_centroids(self):
        """
        Повертає:

        Масив розміру (k, кількість_факторів),
            який містить k центроїдів з початкових точок
        """

        np.random.seed(512)
        ic = np.random.permutation(self.X)
        return ic[:self.k]
```

Рис. 3.12. Створення класу KMeans

1.3. Віднесемо кожен x_i до найближчого кластеру, визначивши відстань від нього до кожного центроїда. Для цього розрахуємо квадрат Евклідової норми (або як її називають евклідову метрику) різниці між точкою та кожним центроїдом:

$$|| x - c_i ||^2, c_i \in C$$

```
def close_centroid(self, centroids):



  idx = np . zeros((self.X.shape[0],), dtype=np.int8)
  for i in range(self.X.shape[0]):
    dist = np.linalg.norm(self.X[i] - centroids, axis=1)
    min_dist = np.argmin(dist)
    idx[i] = min_dist
  return idx
```

Рис. 3.13. Знаходження найближчого центроїда для кожної точки

1.4.Пошук нових центроїдів, розраховуючи середнє серед точок кожного кластеру:

$$c_i = \frac{1}{|S_i|} \sum_{x_i \in S_i} x_i$$

де S_i – це набір усіх точок, що належать i-ому кластеру.

```
def move_centroids(self, centroids):


  cc = self.close_centroid(centroids)
  rr = []
  for i in range(self.к):
    pp = []
    for k in range(len(cc)):
      if cc[k] == i:
        pp.append(self.X[k])
    rr.append([sum(x)/len(x) for x in zip(*pp)])
  return np.array(rr)
```

Рис. 3.14. Пошук нових центроїдів

1.5. Повтор кроків 3 та 4, поки усі центроїди не перестануть змінювати свої координати. Це означає, що ми повторюємо алгоритм, поки кластери не стабілізуються.

```
def final_centroids(self) :

  c = self.initialize_centroids()
  mapc = self.move_centroids(c)
  while np.any(c != mc):
    c = mc
    mc = self.move_centroids(mc)
  centroids = mc
  cc = self.closest_centroid(centroids)
  clusters = []

  for i in range(self.k):
    p1 = []
    for k in range(len(cc)):
        if cc[k] == i:
          p1.append(self.X[k])
    clusters.append(np.array(p1))
  return clusters, centroids
```

Рис. 3.15. Виведення фінальних центроїдів

2. Пошук оптимального значення k

Для того, щоб реалізувати даний алгоритм, потрібно мати значення k. Іноді, дивлячись на певну множину, можна наочно зрозуміти, яка кількість кластерів буде найбільш оптимальною. Але це не завжди можна побачити. У таких випадках одним із варіантів є метод «ліктя». Метод «ліктя» - це графічне відображення оптимального значення k для кластеризації методом k-середніх.

Щоб знайти візуальний «лікоть», який вказує на оптимальну кількість кластерів, потрібно розрахувати середню суму квадратів відстаней між точками та центроїдом всередині кожного кластеру або скористатись бібліотекою, яку буде описано пізніше:

$$W_k = \frac{1}{k}\sum_{i=1}^{k}\sum_{j}^{n_s} || \, x_{ij} \, - \, c_i \, ||^2$$

де k – кількість кластерів, n_S – кількість точок в кластері S.

```
def mean_distances(k, X):


  re = []
  for ine in range(1, k+1):
    s = 0
    model = KMeans(X, ine)
    cl, fc = model.final_centroids()
    for n in range(ine):
      s1 = 0
      for i in cl[n]:
        s1 += (np.linalg.norm(i - fc[n]))**2
      s += s1
    re.append(s / ine)
  return np.array(re)
```

Рис. 3.16. Функція середніх відстаней для побудови графіка візуального «ліктя»

3.5. Реалізація кластерного аналізу на прикладі заданої бази даних

Далі наведено реалізацію описаного алгоритму на прикладі заданої бази даних.

Для того, щоб побудувати модель, потрібно нормалізувати дані:

```
scaler = MinMaxScaler()
X = scaler.fit_transform(df_customers)
X
```

```
array([[0.22590361, 0.13215859, 0.62772958],
       [0.05421687, 0.31718062, 0.61386688],
       [0.93373494, 0.07048458, 0.1167973 ],
       ...,
       [0.54216867, 0.02643172, 0.02822834],
       [0.02108434, 0.04845815, 0.06218827],
       [0.12650602, 0.30396476, 0.64171452]])
```

Рис. 3.17. Нормалізація даних

```
plt.plot(K_clusters, distances)
plt.xlabel('k-clusters')
plt.ylabel(f'dist score')
plt.title("Finding the optimal number of clusters with elbow method");
```

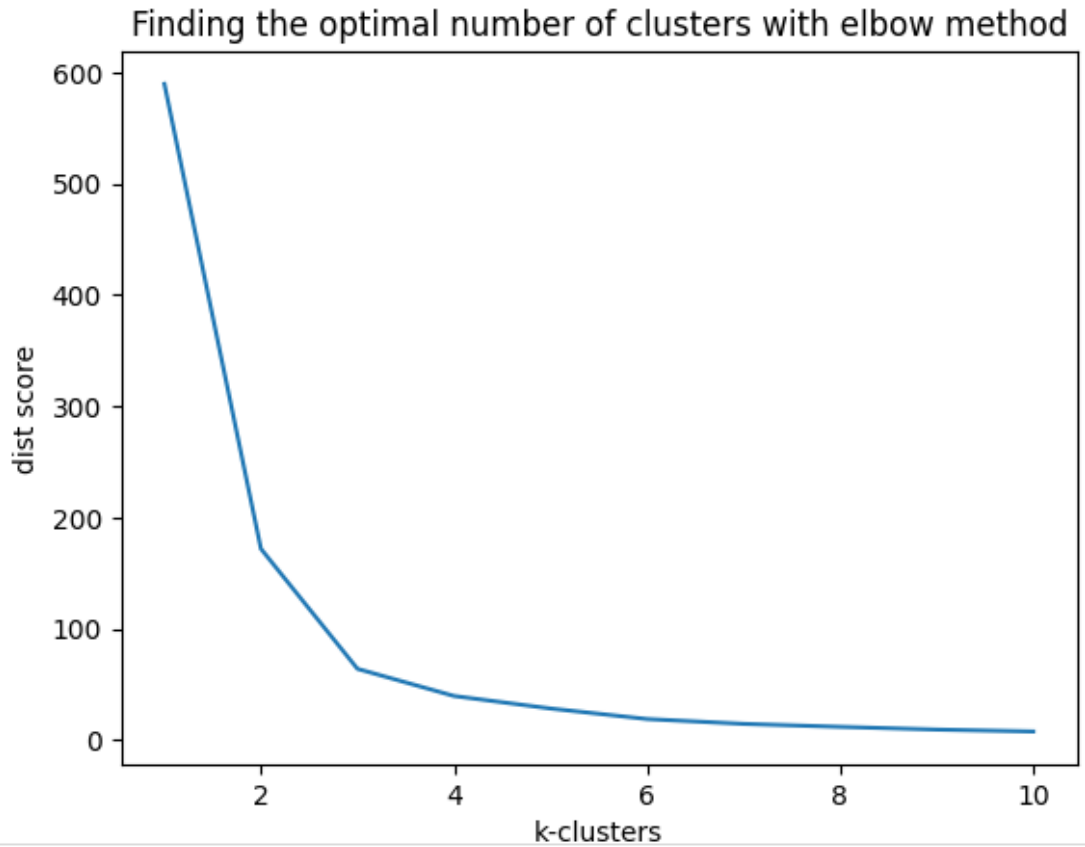


Рис. 3.18. Пошук оптимального значення k для заданої бази даних

```
model = KMeans(X, 3)
```

```
clusters, final_centrs = model.final_centroids()
print('Фінальні центроїди: ', final_centrs)
```

```
Фінальні центроїди:  [[0.12458384 0.47928642 0.59195919]
 [0.69849666 0.10070838 0.1357616 ]
 [0.14950028 0.14436424 0.18444729]]
```

Рис. 3.19. Знаходження центроїдів

```
# створення 3D графіка
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# побудова кластерів
for cluster in clusters:
    ax.scatter(cluster[:, 0], cluster[:, 1], cluster[:, 2])

ax.set_xlabel('Recency')
ax.set_ylabel('Frequency')
ax.set_zlabel('Monetary Value')
ax.set_title('Результати кластеризації')
plt.show()
```

Рис. 3.20. Побудова графіку даних, поділених на кластери

```
cl_count = []
for i, cluster in enumerate(clusters):
    num_points = len(cluster)
    cl_count.append(num_points)
import plotly.express as px
plt.rcParams['figure.figsize'] = [10,6]
customers_df = pd.DataFrame(X, columns=df_customers.columns)
fig = px.pie(data, values = cl_count,
             names = ['1', '2', '3'],
             title = 'Прогнозований розподіл кластерів')
fig.show()
```

Рис. 3.21. Прогнозований розподіл кластерів

На Рис. 3.18. видно, що 3 є хорошим значенням для кількості кластерів у даному випадку, але це значення може змінюватися залежно від потреб бізнесу. За допомогою моделі, отриманої в результаті аналізу, компанія може розробляти індивідуальні маркетингові стратегії та рекламні акції, спрямовані на клієнтів, враховуючи групу, до якої вони належать.

3.6 Висновки до розділу

Кластеризація - це метод машинного навчання, який використовується для групування схожих об'єктів чи точок даних в кластери або групи. Мета полягає в тому, щоб об'єкти в одному кластері були більш схожими один на одного, ніж на ті, які належать до інших кластерів. Кластеризація є неконтрольованим методом, оскільки в ньому немає попередньо визначених категорій для об'єктів.

Наведено алгоритм к-середніх і особливості його реалізації на Python. Існує багато бібліотек для реалізації алгоритму k-середніх на Python, серед них scikit-learn, який надає простий інтерфейс та гнучкість для використання.

Для дослідження було обрано набір даних «Customer segmentation dataset», який містить 541909 записів покупок додатку більше ніж 2000 покупців та 8 характеристик цих покупок (номер , унікальний код для кожної трансакції, опис кожної фічі, кількість придбаних додатків, дата покупки, ціна одної фічі, унікальне ID кожного покупця та країна проживання) між 01.12.2010 та 09.12.2011. Таким чином, обсяг даних є занадто великим, щоб проаналізувати його неозброєним оком. Саме тому доцільним буде використання алгоритмів машинного навчання для знаходження певних висновків.

РОЗДІЛ 4.
РОЗРОБКА МІКРОСЕРВІСУ АНАЛІЗУ ПОВЕДІНКИ КОРИСТУВАЧІВ (РОЗРОБКА АНАЛІТИЧНОГО ДОДАТКУ)

4.1. Концепція розробки мікросервісів та структура додатку

Концепція розробки мікросервісів – це архітектурний підхід до розробки програмного забезпечення, при якому програма розбивається на невеликі, автономні служби, які називаються мікросервісами. Кожен мікросервіс виконує конкретну функцію або набір пов'язаних функцій і може бути розгорнутий та масштабований незалежно від інших мікросервісів.

Основні принципи концепції мікросервісів:

Розбиття на служби (Decomposition): Розділення програмного забезпечення на невеликі служби, які можуть бути розроблені та підтримувані незалежно.

Автономія (Autonomy): Кожен мікросервіс є незалежним, має свою базу даних та може бути розгорнутий окремо.

Резервуар мов (Polyglot): Використання різних технологій та мов програмування для кращого вирішення конкретних завдань.

Самостійне масштабування (Independent Scaling): Можливість масштабування кожного мікросервісу окремо в залежності від навантаження.

Різновидність інтерфейсів (Diversity of Interfaces): Кожен мікросервіс має свій власний API, що забезпечує доступ до його функціональності.

Регенерація (Resilience): Здатність системи відновлюватися в разі виникнення помилок або збоїв в одному з мікросервісів.

Структура додатку на основі мікросервісів може виглядати наступним чином:

Сервіс автентифікації і авторизації: Забезпечує управління користувачами, реєстрацію, вход та контроль доступу.

Сервіс керування користувачами: Надає можливість зміни профілю користувача, встановлення параметрів безпеки та інші функції, пов'язані з особистими обліковими записами.

Сервіс обробки платежів: Забезпечує обробку транзакцій, розрахунок цін та оплату послуг або товарів.

Сервіс збереження даних: Може включати в себе бази даних, файлові сховища та інші механізми для зберігання даних.

Сервіс нотифікацій: Відповідає за відправку повідомлень, електронних листів, а також повідомлень через платформи сповіщень.

Сервіс аналітики: Забезпечує збір та обробку даних для подальшого аналізу та прийняття рішень.

Сервіс інтеграції з зовнішніми сервісами: Дозволяє взаємодіяти з іншими системами та сервісами через API.

Кожен з цих мікросервісів може бути незалежно розгорнутий, масштабований та оновлюваний, що робить систему більш гнучкою та легше підтримуваною. Однак важливо правильно вибрати границі між мікросервісами та дотримуватися принципів, щоб уникнути зайвої складності та забезпечити ефективну взаємодію між компонентами системи.

4.2. Впровадження (інтеграція) розроблених моделей у аналітичний додаток

Впровадження (інтеграція) розроблених моделей у аналітичний додаток є важливим етапом, що дозволяє використовувати передові аналітичні рішення та машинне навчання для отримання цінної інформації. Нижче наведено загальні кроки для інтеграції моделей у аналітичний додаток:

Визначення бізнес-задачі:

З'ясуйте, яку саме бізнес-проблему або завдання модель повинна вирішити.

Визначте, які величини або параметри є ключовими для вирішення цієї проблеми.

Розробка моделі:

Розробіть та тренуйте модель, яка найкраще вирішує визначену бізнес-задачу.

Перевірте точність та ефективність моделі на тестових данних.

Експорт моделі:

Збережіть навчену модель у відповідний формат (наприклад, TensorFlow SavedModel або ONNX для моделей, створених у PyTorch).

Створення API:

Розробіть API для взаємодії з моделлю. Можна використовувати фреймворки, такі як Flask чи FastAPI для створення RESTful API.

Розгортання моделі:

Розгорніть API та модель на сервері чи в хмарному середовищі. Використовуйте інструменти, такі як Docker для контейнеризації.

Інтеграція з аналітичним додатком:

Забезпечте можливість аналітичного додатку взаємодіяти з розгорнутою моделлю через API.

Вбудуйте відповідні запити та логіку для обробки результатів моделі в аналітичний додаток.

Забезпечення безпеки та забезпечення якості:

Захистіть API ключами, обмеженнями швидкості запитів та іншими методами безпеки.

Впевніться, що інтеграція не впливає на продуктивність та стабільність аналітичного додатку.

Моніторинг та підтримка:

Встановіть систему моніторингу для відстеження продуктивності моделі та виявлення можливих проблем.

Надайте підтримку та відслідковуйте результати, щоб періодично переглядати та вдосконалювати модель.

Інтеграція моделей у аналітичний додаток дозволить вам використовувати передові аналітичні методи для покращення прийняття рішень та вибудування цінних функцій для користувачів.

Розробка мікросервіси є дуже потужним інструментом, які дають змогу використовувати різного роду інструментарій всім у кого є до нього доступ. Окрім того, що це дуже зручно це ще і практично. Наявність мікросервісів дозволяє швидко і без додаткової підготовки вирішувати найскладніші задачі у тому числі і ті, які ефективно вирішуються за допомоги методів машинного навчання.

Кластеризація достатньо складна задача, існує багато різноманітних алгоритмів для її вирішення. Одним із популярних алгоритмів, про який вже згадувалось раніше є K-means. Незважаючи на те, що він має один суттєвий недолік, а саме одним з параметрів цього алгоритму є кількість кластерів. Тобто, аналітик повинен знати скільки кластерів можна отримати на нерозмічених даних.

Існують декілька методів визначення кількості кластерів, одним з таких методів є метод ліктя. Технології і аналітичні інструменти постійно розвиваються, згаданий метод можна реалізувати як самостійно, так і за допомоги спеціальних бібліотек. Однією з таких бібліотек є «yellowbrick».

У якості дата сету для проведення кластеризації були використанні дані з Kaggle [https://www.kaggle.com/datasets/imakash3011/customer-personality-analysis] про маркетингову кампанію (див. рисунок), які мають відкриту ліцензію. Перед початком роботи з дата сетом, треба зробити деякі перетворення даних.

```
df.head()
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumWebVisitsMonth
0	5524	1957	Graduation	Single	58138.0	0	0	04-09-2012	58	635	...	7
1	2174	1954	Graduation	Single	46344.0	1	1	08-03-2014	38	11	...	5
2	4141	1965	Graduation	Together	71613.0	0	0	21-08-2013	26	426	...	4
3	6182	1984	Graduation	Together	26646.0	1	0	10-02-2014	26	11	...	6
4	5324	1981	PhD	Married	58293.0	1	0	19-01-2014	94	173	...	5

5 rows × 29 columns

Рис.4.1. Приклад даних для сегментації клієнтів

В дата сеті наявні дані про рік народження клієнта, зазвичай вік клієнта є дуже важливою ознакою, тому для перетворення віку, а також дати реєстрації клієнта в число скористаємось наступною функцією.

```
from typing import Tuple

def get_diff_date(end_data:pd.Timestamp,
                  data: Tuple[str, int],
                  type_date: str = 'year'):
  if type_date == 'year':
    time_date = int(end_data.year - data)
  elif type_date == 'month':
    time_data = end_data.to_period('M') - pd.to_datetime(data, format = '%d-%m-%Y').to_period('M')
    time_data = time_data.n
  else:
    time_data = end_data - pd.to_datetime(data)
    time_data = int(time_data.days)
  return time_data
```

Рис. 4.2. Функція для перетворення дати народження клієнта і дати реєстрації у числа

Таким чином, були створені дві додаткові ознаки (див. рисунок) Age Month_register для подальшого їх використання при вирішенні задачі кластерного аналізу.

```
df['Age'] = df.Year_Birth.apply(lambda x: end_data.year - x)
```

```
df['Month_regist'] = df.Dt_Customer.apply(
     lambda x: get_diff_date(end_data, x, type_date ='month'))
```

Рис.4.3. Додаткові ознаки, які будуть використанні для кластерізації (поділу на групи) клієнтів

Після проведення всіх необхідних маніпуляцій з даним для проведення кластерного аналізу та побудови моделі було відібрані наступні фактори.

```
num_cols = [
    'Age',
    'Income',
    'Month_regist',
    'Sum_Purchsed',
    'Count_Campaings',
    'Recency',
    'NumDealsPurchases',
    'NumCatalogPurchases',
    'NumStorePurchases',
    'NumWebVisitsMonth',
    'NumWebPurchases',
    'MntWines',
    'MntFruits',
    'MntMeatProducts',
    'MntFishProducts',
    'MntSweetProducts',
    'MntGoldProds'
]
```

Рис.4.4. Відібрані ознаки для проведення кластерного аналізу

Перед безпосередньо побудовою моделі, треба провести певну додаткову підготовку даних, а саме їх перетворення за допомоги інструменту StandardScaler.

```
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler

X_scaled = StandardScaler().fit_transform(df_new)
```

Рис.4.5. Препроцесінг даних за допомоги бібліотеки sklearn та інструменту StandardScaler

StandardScaler дозволяє стандартизувати дані, х середнім яке дорівнює нулю та одиничною дисперсією. Це один з ефективних інструментів, який застосовується для вирішення задач кластерізації.

Як зазначалось раніше, перед побудовою моделі треба встановити кількість кластерів. Для цього скористаємось бібліотекою yellowbrick і наступним кодом.

```
from yellowbrick.cluster import KElbowVisualizer

model = KMeans(random_state = 42, n_init= 'auto')

visual = KElbowVisualizer(model, k = (1, 10), timings = False)

visual.fit(X_scaled) #(X_pca)
visual.show();
```

Рис.4.6. Код, який доволяє встановити кількість кластерів за допомоги методу ліктя

Для встановлення ефективної кількості кластерів треба вказати модель, яка буде використовуватись, вказати діапазон кількості кластерів, в даному прикладі це від 1 до 10, а так передати безпосередньо дані. Результат виконання коду наведено на рисунку.

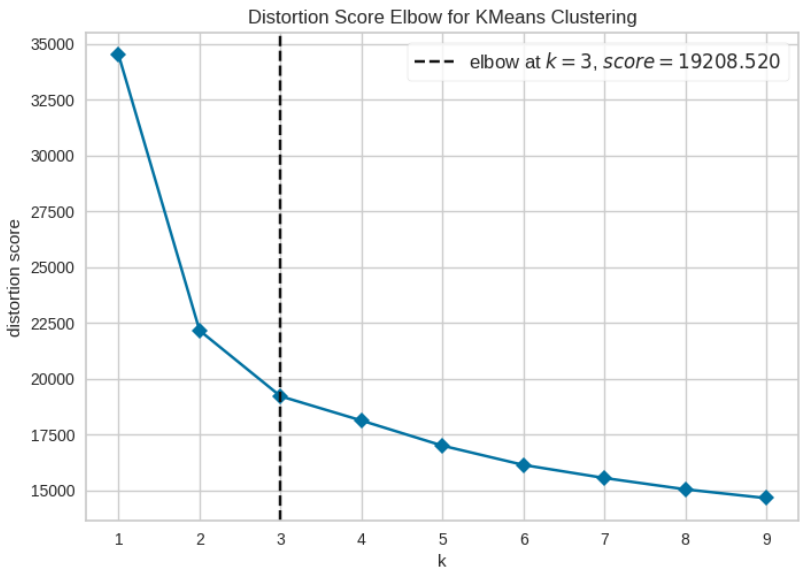


Рис. 4.7. Встановлення ефективної кількості кластерів за допомоги бібліотеки yellowbrick

Таким чином, оптимальною кількістю кластерів є 3 зі score 19209.

Після встановлення кількості кластерів можна будувати модель. Код побудови моделі наведено на рисунку.

```
from sklearn.cluster import KMeans
model = KMeans(n_clusters= 3,
               random_state = 42)

model.fit(X_scaled) # df_new
```

Рис. 4.8. Побудова моделі K-means з трьома кластерами

Результатом побудованою моделі та код отримання номерів кластерів для кожного клієнта наведено на рисунку.

```
labels = model.predict(X_scaled) # df_new
labels[:20]
```

```
array([2, 2, 1, 0, 2, 2, 2, 0, 0, 0, 0, 2, 2, 0, 1, 2, 0, 1, 0, 0],
      dtype=int32)
```

Рис. 4.9. Результатом роботи моделі та код отримання номерів кластерів для кожного клієнта

Отримані labels для кожного клієнта тепер можна використовувати для розробки мікросервісу, який для кожного нового клієнту буде надавати номер кластеру.

Для розробки мікросервісу скористаємось зручною та ефективною бібліотекою «streamlit». Ця бібліотека містить багато вбудованих елементів і дозволяє швидко розробляти мікросервіси.

Для роботи з streamlit і побудови мікросервісу зручно скористатись такою IDE як Pycharm.

Після завантаження даних та імпорту необхідних бібліотек, необхідно створити модель і зберегти її для подальшого використання і створення мікросервісу.

```
preprocessing = make_column_transformer(
    (StandardScaler(), numerical_features),
)

clf = make_pipeline(preprocessing, LogisticRegression())

clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
y_pred_proba = clf.predict_proba(X_test)
#print(y_pred)
# print(y_pred_proba)

# Save the model
joblib.dump(clf, PATH_MODEL)
```

Рис.4.10. Створення та збереження моделі для подальшого використання

Створимо функції для завантаження даних та збереженої моделі. Застосуємо кешування, для цього треба скористатись декоратором.

```
10  @st.cache_data
11  def load_data(path):
12      data = pd.read_csv(path)
13      data = data.sample(100)
14      return data
15
16  @st.cache_data
17  def load_model(path):
18      model = joblib.load(path)
19      return model
```

Рис. 4.11. Функції для завантаження даних та збереженої раніше моделі

Необхідно завантажити дані, створити заголовок, також додамо перші декілька записів з дата сету, створимо markdown з описом всіх змінних.

```
22  df = load_data(PATH_DATA)
23
24  st.header('Cluster Analisys')
25
26  st.write(df[:4])
27
28  st.markdown(
29      """
30      ### Description of data
31          - People
32          -- ID: Customer's unique identifier
33          -- Year_Birth: Customer's age
34          -- Income: Customer's yearly household income
35          -- Recency: Number of days since customer's last purchase
36          - Products
```

Рис. 4.12. Завантаження дата сету, створення заголовку, markdown

Створюємо для зручності необхідні додаткові елементи, а саме selectbox та різноманітні slider’и, що будуть розтошовані на спеціальній панелі з ліва (sidebar), які відображають різні параметри клієнта, ці параметри будуть використанні для встановлення номера кластера для кожного окремого клієнта.

Рис.4.13. Приклад selectbox та різноманітні slider'ів на боковій панелі

За допомоги коду (див. рисунок), занатажуємо раніше збережену модель, зберігаємо параметри клієнта в DataFrame і додаємо кнопку «Predict». У разі натиснення на кнопку «Predict», запускається сценарій в рамках якого данні для прогнозування передаються в модель, яка у якості результату, на основі переданих даних, повертає прогноз, а саме номер кластеру до якого відноситься клієнт.

```
data_predict = pd.DataFrame([dict_data])
model = load_model(PATH_MODEL)

button = st.button('Predict')

if button:
    output = model.predict(data_predict)[0]
    st.success(f'{output + 1} cluster')
```

Рис. 4.14. Код реалізації збереження даних для прогнозування та сценарій функціонування кнопки Predict

На рисунку наведено приклад роботи розробленого мікросервісу та результат, який повертає модель для клієнта з урахуванням певних параметрів. Тобто, цей конкретний клієнт відноситься до другого кластеру.

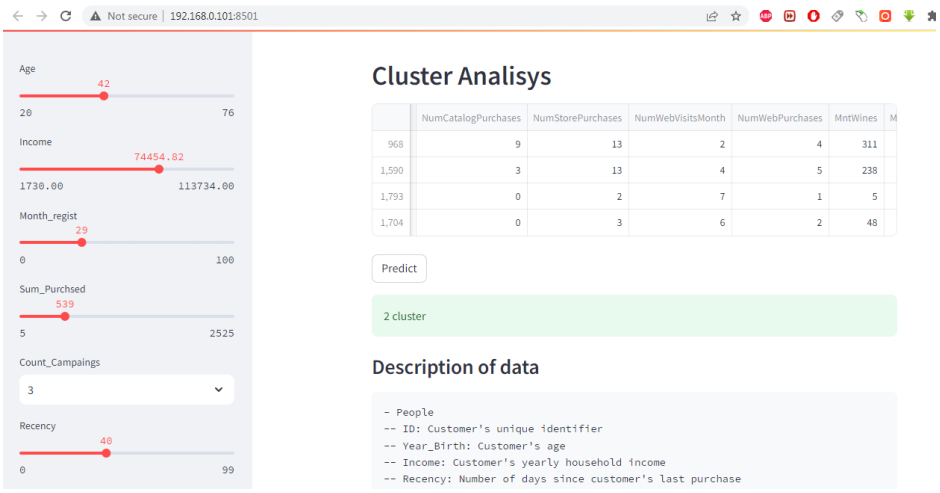


Рис. 4.15. Приклад роботи мікросервісу з поверненням номеру кластера для клієнта за певними параметрами

Таким чином, за допомоги мікросервісу, який може знаходитись як на локальному хості та і в мережі інтернет, в модель передаються певні параметри клієнта і вона на цих даних формує прогноз, який по суті є номером ймовірного кластеру до якого треба віднести цього клієнта.

4.3. Методи оцінки ефективності впровадження змін у додатку

Оцінка ефективності впровадження змін в IT додатку може включати різноманітні методи, а A/B тестування є одним із ключових інструментів для цього. A/B тестування (або split testing) дозволяє порівнювати дві версії продукту або веб-сайту для визначення, яка зміна є більш ефективною. Ось деякі методи оцінки ефективності:

A/B тестування:

Розробка гіпотези: Визначте мету змін та створіть гіпотезу, яка описує, як ці зміни повинні поліпшити продукт.

Вибір метрик: Оберіть ключові показники ефективності (KPI), які ви будете вимірювати для оцінки впливу змін.

Розділіть аудиторію: Рандомізуйте користувачів на дві групи – контрольну (A) та експериментальну (B), і впроваджуйте зміни лише для експериментальної групи.

Збір даних та аналіз: Збирайте дані про взаємодію користувачів з продуктом та аналізуйте їх для визначення статистично значущих різниць між двома групами.

Контроль за метриками продуктивності:

Слідкуйте за основними метриками продуктивності, такими як конверсія, залучення користувачів, збереження та інші.

Порівнюйте метрики до та після впровадження змін, а також між контрольною та експериментальною групами.

Оцінка впливу на користувачів:

Здійснюйте зворотний зв'язок з користувачами для оцінки їхнього сприйняття нововведень та реакції на зміни.

Retention Analysis (Аналіз утримання):

Вивчайте, які зміни впливають на утримання користувачів на платформі чи сервісі.

Customer Satisfaction Surveys (Опитування задоволеності користувачів):

Проводьте опитування для вимірювання задоволеності користувачів і виявлення їхніх пріоритетів.

Time Series Analysis (Аналіз часових рядів):

Аналізуйте часові ряди метрик для виявлення трендів та змін в часі після впровадження нововведень.

Кластерний аналіз користувачів:

Групуйте користувачів за певними характеристиками та спостерігайте за тим, як впроваджені зміни впливають на різні

В нашому випадку ми будемо тестувати мікросервіс кластериції користуачів нашого додатку. Реалізації процедури проведення A/B тестування на Python.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import binom
from scipy.stats import norm
import seaborn as sns
from statsmodels.stats.proportion import proportion_effectsize
from statsmodels.stats import power
from scipy.stats import shapiro
from scipy.stats import levene
from scipy.stats import ttest_ind
```

Імпорт необхідних бібліотек для проведення A/B тестування на Python

```
    print('SRM likely not present')
#Testing normality Shapiro- Wilk test
  stat_A, p_value_A=shapiro(control[metric])
  stat_B, p_value_B=shapiro(test[metric])
  if p_value_A > alpha:
    print("The control data appears to be normally distributed (Fail to reject H0)")
  else:
    print("The control data does not appear to be normally distributed")
  if p_value_B > alpha:
    print("The test data appears to be normally distributed (Fail to reject H0)")
  else:
    print("The test data does not appear to be normally distributed ")
  if ((p_value_A> alpha) & (p_value_B < alpha)) | ((p_value_A < alpha) & (p_value_B > alpha) ) | ((p_value_A < alpha) & (p_value_B
    print(f"One of the control or test data does not seem normally distributed so we go to non parametric test")
    # Mann Whitney U - test
    from scipy.stats import mannwhitneyu
    u_value, mannw_test_p=mannwhitneyu(control[metric],test[metric],alternative='two-sided',nan_policy='omit')
    if mannw_test_p > alpha:
      print('H0 can not be rejected')
    else:
      print('H0 is rejected')
  if (p_value_A > alpha) & (p_value_B> alpha):
```

Фрагмент коду по реалізації функції на Python для проведення тестування гіпотез

4.4. Висновки до розділу

Розробка мікросервісу є дуже потужним інструментом, який дає змогу використовувати різного роду інструментарій всім у кого є до нього доступ. Окрім того, що це дуже зручно це ще і практично. Наявність мікросервісів дозволяє

швидко і без додаткової підготовки вирішувати найскладніші задачі у тому числі і ті, які ефективно вирішуються за допомоги методів машинного навчання.

У якості дата сету для проведення сегментація були використанні дані з Kaggle [https://www.kaggle.com/datasets/imakash3011/customer-personality-analysis] про маркетингову кампанію, які мають відкриту ліцензію.

Для розробки мікросервісу використали зручну та ефективну бібліотеку «streamlit». Ця бібліотека містить багато вбудованих елементів і дозволяє швидко розробляти мікросервіси.

У роботі наведено приклад роботи розробленого мікросервісу та результат, який повертає модель для клієнта з урахуванням певних параметрів.

Таким чином, за допомогою мікросервісу, який може знаходитись як на локальному хості та і в мережі інтернет, в модель передаються певні параметри клієнта і вона на цих даних формує прогноз, який по суті є номером ймовірного кластеру до якого треба віднести цього клієнта.

ВИСНОВКИ

У магістерській роботі об’єктом дослідження є методи машинного навчання, а саме кластеризації для аналізу поведінки користувачів ІТ додатку. Предметом дослідження є науково-методичні і практичні підходи до моделювання і програмування системи аналізу поведінки користувачів ІТ додатком.

В першому розділі здійснено детальний огляд предметної області. Досліджено причинно-поведінковий каркас для аналізу великих даних. Оглянуті типи сучасної аналітики. Проведено дослідження системи аналізу даних про поведінку користувачів

У другому розділі проведено огляд архітектури мікросервісів та засобів візуалізації. Обрано мову програмування Python для аналізу даних та A/B тест як спосіб визначення реакції користувачів на зміни в ІТ продукті.

В третьому розділі обгрунтовано використання машинного навчання для аналізу даних. Вибрано і описано кластеризацію як інструмент продуктової аналітики. Розгляноуто методи та моделі вирішення задач кластеризації. Наведено приклад алгоритм к-середніх і особливості його реалізації на Python. Реалізазовано кластерний аналіз на прикладі заданої бази даних.

В четвертому розділі наведено концепцію розробки мікросервісів та впроваджено розроблених моделей у аналітичний мікросервіс. Мікросервіс може знаходитись як на локальному хості та і в мережі інтернет, в модель передаються певні параметри клієнта і вона на цих даних формує прогноз, який по суті є номером ймовірного кластеру до якого треба віднести цього клієнта.Показано методи оцінки ефективності впровадження змін у додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Product Analytics: A Comprehensive Guide. Режим доступу до ресурсу:https://cxl.com/blog/product-analytics/

2. A MIXPANEL GUIDE: The Guide to Product Analytics. A book of questions and answers. Режим доступу до ресурсу: https://mixpanel.com/content/guide-to-product-analytics/report/

3. Марець О.Р., Панчишин Т.В., Прокопович-Павлюк І.В. Сучасні бізнес-метрики оцінки ефективності маркетингових заходів Режим доступу до ресурсу: http://www.visnyk-onu.od.ua/journal/2021_26_1/24.pdf

4. 5 найкращих інструментів AI для аналітиків даних. (жовтень 2023 р.) Режим доступу до ресурсу: https://www.unite.ai/uk/5-best-ai-tools-for-data-analysts/

5. Віртуалізація [Електронний ресурс] / Wikipedia – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Апаратна_віртуалізація.

6. Namespace [Електронний ресурс] / Wikipedia – Режим доступу до ресурсу: http://man7.org/linux/man-pages/man7/namespaces.7.html.

7. W. Felter, A. Ferreira, R. Rajamony, J. Rubio. “An updated performance comparison of virtual machines and linux containers.” In: Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium on. IEEE. 2015, pp. 171– 172 (cit. on pp. 5, 6).

8. Fowler M. Microservices [Електронний ресурс] / Martin Fowler – Режим доступу до ресурсу: https://martinfowler.com/articles/microservices.html.

9. Docker [Електронний ресурс] / Wikipedia – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Docker.

10. Microservices are hard [Електронний ресурс] – Режим доступу до ресурсу: https://hackernoon.com/microservices-are-hard-an-invaluable-guide-tomicroservices-2d06bd7bcf5d.

11. What is Microservices Architecture? [Електронний ресурс] – Режим доступу до ресурсу: https://smartbear.com/learn/api-design/what-are-microservices/

12. Сайт компанії Docker.io. Docker [Електронний ресурс] / Docker.io – Режим доступу до ресурсу: https://www.docker.com/.

13. Daemon [Електронний ресурс] / Wikipedia – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Daemon_(computing).

14. Сайт компанії Kubernetes.io. Kubernetes [Електронний ресурс] / Kubernetes.io – Режим доступу до ресурсу: https://kubernetes.io.

15. Molyneaux I. The Art of Application Performance Testing / Ian Molyneaux.. – 209 с. – (O'Reilly Media).

16. Сайт компанії Software Performance [Електронний ресурс] / Wikipedia – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Software_performance_testing. 99

17. Automated Verification of Load Tests Using Control Charts / T.H.D. Nguyen, A. Bram, J. Zhen Ming, H. Ahmed E.. // Asia-Pacific Software Engineering Conference. – 2011. – С. 282–289.

18. The httperf HTTP load generator [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/httperf/httperf.

19. Rational Performance Tester [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Rational_Performance_Tester.

20. Сайт компанії System Performance Monitor [Електронний ресурс] – Режим доступу до ресурсу: https://www.techopedia.com/definition/12397/system-performance-monitor-spm.

21. Сайт компанії Linux proc [Електронний ресурс] – Режим доступу до ресурсу: http://tldp.org/LDP/Linux-Filesystem-Hierarchy/html/proc.html.

22. Performance Testing Guidance for Web Applications by Microsoft Corporation / Microsoft Corporation., 2007. – 221 с. – (Microsoft Corporation).

23. Automated Detection of Performance Regressions Using Regression Models on Clustered Performance Counters / W. Shang, A.E. Hassan, M. Nasser, F. Parminder // Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering – Austin, 2015. – С. 15-26.

24. Student t-Test [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Student%27s_t-test.

25. Thanh H.D. Nguyen Automated detection of performance regressions using statistical process control techniques / T. H. Nguyen, B. Adams, Z. M. Jiang, A. E. Hassan, M. Nasser, P. Flora // Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering – Boston, 2012. – С. 299-310.

26. An Industrial Case Study on the Automated Detection of Performance Regressions in Heterogeneous Environments / A.Bram, M. Zhen, C. King, H. Ahmed E.. // IEEE/ACM 37th IEEE International Conference on Software Engineering. – 2015. – С. 49–58.

27. Association Rule [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Association_rule_learning. 100

28. PYPL Popularity of Programming Language. Режим доступу до ресурсу: https://pypl.github.io/PYPL.html

29. Python 3.8.10 documentation. URL: https://docs.python.org/3.8/.

30. Mark Lutz Learning Python. O'Reilly. Режим доступу до ресурсу: https://liteka.ru/english/library/2974-learning-python-5e#1

31. Кононова К. Ю. Машинне навчання: методи та моделі: підручник для бакалаврів, магістрів та докторів філософії спеціальності 051 «Економіка» / К. Ю. Кононова. – Харків: ХНУ імені В. Н. Каразіна, 2020. – 301 с.

32. A/B-тестування: що це, навіщо потрібне і як його проводити. Режим доступу до ресурсу: https://skillsetter.io/blog/AB-test-questions-ua

33. Пояснення найпопулярніших моделей машинного навчання Режим доступу до ресурсу: https://techukraine.net/%D0%BF%D0%BE%D1%8F%D1%81%D0%BD%D0%B5%D0%BD%D0%BD%D1%8F-%D0%BD%D0%B0%D0%B9%D0%BF%D0%BE%D0%BF%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D1%96%D1%88%D0%B8%D1%85-%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B5%D0%B9-%D0%BC/

34. O. Liashenko, O. Podskrebko, N. Ivanchenko. The Impact of Data Analytics on the Nature of Doing Business. 12th International Conference on Advanced Computer Information Technologies (ACIT), Conference, Spišská Kapitula, .2022, P. 331-334.

35. Іванченко Н.О., Подскребко О.С. Інструменти аналізу ефективності взаємодії користувача з продуктом. Modernization of science and its influence on global processes: collection of scientific papers «SCIENTIA» with Proceedings of the IV International Scientific and Theoretical Conference, November 3, 2023. Bern, Swiss Confederation: International Center of Scientific Research, P. 21-24.

36. Концептуальний підхід бізнесу до Data Science / Н.О. Іванченко, О.С. Подскребко Proceedings of the III International Scientific and Theoretical Conference, August 19, 2022. Tel Aviv, State of Israel: 2022. P. 134-135.

37. Сайт компанії A faster way to build and share data apps. Режим доступу до ресурсу: https://streamlit.io/

38. Сайт компанії Flask. Режим доступу до ресурсу: https://flask.palletsprojects.com/en/3.0.x/

ДОДАТОК А

Фрагмент коду файлу train.py

```
import json
import joblib
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.compose import make_column_transformer
from sklearn.linear_model import LogisticRegression



PATH_DATA = "../data/cluster.csv"
PATH_MODEL = "../models/lr_pipeline.sav"
PATH_UNIQUE_VALUES = "../data/unique_value.json"
numerical_features = ['Age', 'Income', 'Month_regist', 'Sum_Purchsed',
      'Count_Campaings', 'Recency', 'NumDealsPurchases',
      'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
      'NumWebPurchases', 'MntWines', 'MntFruits', 'MntMeatProducts',
      'MntFishProducts', 'MntSweetProducts', 'MntGoldProds']

df = pd.read_csv(PATH_DATA)
print(df.columns)
y = df['Clusters']
X = df.drop('Clusters', axis = 1)
X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size = 0.2, random_state = 42
)
print(X_train.shape, y_train.shape)
preprocessing = make_column_transformer(
   (StandardScaler(), numerical_features),
)
clf = make_pipeline(preprocessing, LogisticRegression())
```

```
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
y_pred_proba = clf.predict_proba(X_test)
#print(y_pred)
# print(y_pred_proba)
# Save the model
joblib.dump(clf, PATH_MODEL)
# Save unique values
dict_unique = {key: X[key].unique().tolist() for key in X.columns}
with open(PATH_UNIQUE_VALUES, 'w') as file:
    json.dump(dict_unique, file)
```

ДОДАТОК Б

Фрагмент коду файлу main.py

```
import pandas as pd
import streamlit as st
import json
import joblib

PATH_DATA = "../data/cluster.csv"
PATH_MODEL = "../models/lr_pipeline.sav"
PATH_UNIQUE_VALUES = "../data/unique_value.json"

@st.cache_data
def load_data(path):
    data = pd.read_csv(path)
    data = data.sample(100)
    return data

@st.cache_data
def load_model(path):
    model = joblib.load(path)
    return model

df = load_data(PATH_DATA)

st.header('Cluster Analisys')

st.write(df[:4])

with open(PATH_UNIQUE_VALUES) as file:
    dict_unique = json.load(file)

numerical_features = ['Age', 'Income', 'Month_regist', 'Sum_Purchsed',
      'Count_Campaings', 'Recency', 'NumDealsPurchases',
```

```
    'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
    'NumWebPurchases', 'MntWines', 'MntFruits', 'MntMeatProducts',
    'MntFishProducts', 'MntSweetProducts', 'MntGoldProds']

Age = st.sidebar.slider("Age", min_value = min(dict_unique["Age"]),
                        max_value = max(dict_unique["Age"]))

Income = st.sidebar.slider("Income", min_value = min(dict_unique["Income"]),
                        max_value = max(dict_unique["Income"]))

Month_regist = st.sidebar.slider("Month_regist", min_value = 0,
                        max_value = 100)

Sum_Purchsed = st.sidebar.slider("Sum_Purchsed", min_value = min(dict_unique["Sum_Purchsed"]),
                        max_value = max(dict_unique["Sum_Purchsed"]))


Count_Campaings = st.sidebar.selectbox('Count_Campaings', dict_unique['Count_Campaings'])

Recency = st.sidebar.slider("Recency", min_value = min(dict_unique["Recency"]),
                        max_value = max(dict_unique["Recency"]))

NumDealsPurchases = st.sidebar.slider("NumDealsPurchases", min_value =
min(dict_unique["NumDealsPurchases"]),
                        max_value = max(dict_unique["NumDealsPurchases"]))

NumCatalogPurchases = st.sidebar.slider("NumCatalogPurchases", min_value =
min(dict_unique["NumCatalogPurchases"]),
                        max_value = max(dict_unique["NumCatalogPurchases"]))

NumStorePurchases = st.sidebar.slider("NumStorePurchases", min_value =
min(dict_unique["NumStorePurchases"]),
                        max_value = max(dict_unique["NumStorePurchases"]))
```

```
NumWebPurchases = st.sidebar.slider("NumWebPurchases", min_value =
min(dict_unique["NumWebPurchases"]),
                max_value = max(dict_unique["NumWebPurchases"]))

NumWebVisitsMonth = st.sidebar.slider("NumWebVisitsMonth", min_value =
min(dict_unique["NumWebVisitsMonth"]),
                max_value = max(dict_unique["NumWebVisitsMonth"]))

MntWines = st.sidebar.slider("MntWines", min_value = min(dict_unique["MntWines"]),
                max_value = max(dict_unique["MntWines"]))

MntFruits = st.sidebar.slider("MntFruits", min_value = min(dict_unique["MntFruits"]),
                max_value = max(dict_unique["MntFruits"]))

MntMeatProducts = st.sidebar.slider("MntMeatProducts", min_value =
min(dict_unique["MntMeatProducts"]),
                max_value = max(dict_unique["MntMeatProducts"]))

MntFishProducts = st.sidebar.slider("MntFishProducts", min_value =
min(dict_unique["MntFishProducts"]),
                max_value = max(dict_unique["MntFishProducts"]))

MntSweetProducts = st.sidebar.slider("MntSweetProducts", min_value =
min(dict_unique["MntSweetProducts"]),
                max_value = max(dict_unique["MntSweetProducts"]))

MntGoldProds = st.sidebar.slider("MntGoldProds", min_value = min(dict_unique["MntGoldProds"]),
                max_value = max(dict_unique["MntGoldProds"]))

dict_data = {
    'Age': Age,
    'Income': Income,
    'Month_regist': Month_regist,
    'Sum_Purchsed': Sum_Purchsed,
    'Count_Campaings': Count_Campaings,
```

```
    'Recency': Recency,
    'NumDealsPurchases': NumDealsPurchases,
    'NumCatalogPurchases': NumCatalogPurchases,
    'NumStorePurchases': NumStorePurchases,
    'NumWebVisitsMonth': NumWebVisitsMonth,
    'NumWebPurchases': NumWebPurchases,
    'MntWines': MntWines,
    'MntFruits': MntFruits,
    'MntMeatProducts': MntMeatProducts,
    'MntFishProducts': MntFishProducts,
    'MntSweetProducts': MntSweetProducts,
    'MntGoldProds': MntGoldProds
}

data_predict = pd.DataFrame([dict_data])
model = load_model(PATH_MODEL)

button = st.button('Predict')

if button:
    output = model.predict(data_predict)[0]
    st.success(f'{output + 1} cluster')

st.markdown(
    """
    ### Description of data
        - People
        -- ID: Customer's unique identifier
        -- Year_Birth: Customer's age
        -- Income: Customer's yearly household income
        -- Recency: Number of days since customer's last purchase
        - Products
        -- MntWines: Amount spent on wine in last 2 years
        -- MntFruits: Amount spent on fruits in last 2 years
        -- MntMeatProducts: Amount spent on meat in last 2 years
```

-- MntFishProducts: Amount spent on fish in last 2 years

-- MntSweetProducts: Amount spent on sweets in last 2 years

-- MntGoldProds: Amount spent on gold in last 2 years

- Promotion

-- NumDealsPurchases: Number of purchases made with a discount

- Place

-- NumWebPurchases: Number of purchases made through the company's website

-- NumCatalogPurchases: Number of purchases made using a catalogue

-- NumStorePurchases: Number of purchases made directly in stores

-- NumWebVisitsMonth: Number of visits to company's website in the last month

"""

)