ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ЕКОНОМІЧНОЇ КІБЕРНЕТИКИ І СИСТЕМНОГО АНАЛІЗУ

Пояснювальна записка

до дипломної роботи

МАГІСТРА
(освітній ступінь)

на тему:
«Моделювання поведінки споживача на основі методів машинного навчання»

Виконав: студент 2 року навчання,
групи 8.04.051.020.20.01,
спеціальності 051 «Економіка»
освітньо-професійної програми
«Економічна кібернетика»
Прищепа Д.О.

Керівник: к.е.н., доц. Прокопович С.В.
Рецензент: к.г.н., доц. Ханова О.В.

Харків – 2021 рік

РЕФЕРАТ

Звіт про дипломну роботу: 86 сторінки, 3 розділи, 26 рисунків, 1 таблиця, 57 джерел.

Об’єктом роботи є поведінка користувачів у грі.

Предметом дослідження є дії покупця.

Мета дипломної роботи – з’ясувати яка модель найкраще підійде до існуючих даних та який вплив це може мати на вже функціонуючий застосунок (мобільну гру).

Дипломна робота присвячена розробці моделей оцінки поведінки споживача з використанням машинного навчання. Робота зроблена за власними даними компанії по споживачам. В ній були розглянуті різноманітні моделі машинного навчання та більш детально розібрані моделі бінарної класифікації. Було створено сім моделей бінарної класифікації за нашими даними та для двох найкращих була проведена робота з подбору оптимальних параметрів для покращення роботи моделі.

КЛЮЧОВІ СЛОВА: АНАЛІЗ ПОВЕДІНКИ, МОДЕЛЮВАННЯ, МАШИННЕ НАВЧАННЯ, БІНАРНА КЛАСИФІКАЦІЯ, МОДЕЛЬ ВИПАДКОВИХ ЛІСІВ, ГРАДІЄНТНИЙ БУСТІНГ, КЛАСИФІКАЦІЯ КОРИСТУВАЧЕЙ.

РЕФЕРАТ

Отчет о дипломной работе: 86 страницы, 3 раздела, 26 рисунков, 1 таблица, 57 источников.

Объектом работы является поведение пользователей в игре.
Предметом исследования являются действия покупателя.

Цель дипломной работы – выяснить, какая модель лучше подойдет к существующим данным и какое влияние это может оказать на уже функционирующее приложение (мобильную игру).

Дипломная работа посвящена разработке моделей оценки поведения потребителя с использованием машинного обучения. Работа проделана по собственным данным компании по потребителям. В нем были рассмотрены различные модели машинного обучения и более подробно разобраны модели бинарной классификации. Были созданы семь моделей бинарной классификации по нашим данным и для двух лучших была проведена работа по подбору оптимальных параметров для улучшения работы модели.

КЛЮЧЕВЫЕ СЛОВА: АНАЛИЗ ПОВЕДЕНИЯ, МОДЕЛИРОВАНИЕ, МАШИННАЯ УЧЕБА, БИНАРНАЯ КЛАССИФИКАЦИЯ, МОДЕЛЬ СЛУЧАЙНЫХ ЛЕС, ГРАДИЕНТНЫЙ БУСТИНГ, КЛАССИФИКАЦИЯ ПОЛЬЗОВАТЕЛЕЙ.

ABSTRACT

Thesis of magister degree:: 86 pages, 3 sections, 26 figures, 1 table, 57 sources.

The object of work is the behavior of users in the game.

The subject of the study is the actions of the buyer.

The purpose of the thesis is to find out which model is best suited to existing data and what impact it may have on an already functioning application (mobile game).

Thesis is devoted to the development of models for assessing consumer behavior using machine learning. The work is done according to the company's own data on consumers. It considered various models of machine learning and analyzed models of binary classification in more detail. Seven models of binary classification were created according to our data and for the two best ones work was carried out to select the optimal parameters to improve the performance of the model.

KEY WORDS: BEHAVIOR ANALYSIS, MODELING, MACHINE LEARNING, BINARY CLASSIFICATION, RANDOM FOREST MODEL, GRADIENT BOOSTING, USER CLASSIFICATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 9
ВСТУП 10
РОЗДІЛ 1. РОЛЬ МОДЕЛЮВАННЯ ПОВЕДІНКИ СПОЖИВАЧА У СВІТІ ТА МОЖЛИВОСТІ МАШИННОГО НАВЧАННЯ 12
1.1. Моделювання поведінки споживача 12
1.2. Можливості машинного навчання 17
1.3. Машинне навчання в повсякденному житті 22
1.4. Використання машинного навчання в гейм-деві 25
РОЗДІЛ 2. ІНСТРУМЕНТИ ДОСЛІДЖЕННЯ ТА ОПИС МЕТОДІВ БІНАРНОЇ КЛАСИФІКАЦІЇ 32
2.1. Python, середовища PyCharm і Jupyter, Google Colab та мова SQL 32
2.2. Бібліотеки для обробки даних у Python і моделювання 35
2.3. Класифікація, регресія та кластеризація у Python 40
2.4. Моделі бінарної класифікації в машинному навчанні 45
РОЗДІЛ 3. ПОБУДОВА МОДЕЛІ ПОВЕДІНКИ СПОЖИВАЧА ЗА ДОПОМОГОЮ МЕТОДІВ КЛАСИФІКАЦІЇ МАШИННОГО НАВЧАННЯ 52
3.1. Збір вихідних даних з баз даних за допомогою SQL 52
3.2. Імпорт та обробка даних у Python за допомогою бібліотек 56
3.3. Побудова та оцінка моделей класифікації 62
ВИСНОВКИ 67
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 68
Додаток А. Запит до баз даних BigQuery 76
Додаток Б. Повна версія скрипту визначення моделі бінарної класифікації 85

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення;

ML – машинне навчання;

ШІ, AI – штучний інтелект;

NPC – неігровий персонаж;

VR – віртуальна реальність;

LiveOps – “живе” оперування процесами гри;

SQL – структурний язик запитів;

PEP 8 – посібник зі стилю коду Python;

JSON – текстовий формат обміну даними між комп'ютерами;

CSV – файловий формат для представлення табличних даних;

REST API – засіб взаємодії і обміну даними серверу;

DF – таблична структура даних в Python.

ВСТУП

На сьогоднішній день про машинне навчання чи нейронні мережі не чув лише ледачий. Та й тлумачень терміну “машинне навчання” існує чимало, але більшість науковців формують його так: машинне навчання – це наука про те, як змусити штучний інтелект вчитися і діяти як людина, а також зробити так, щоб він сам постійно покращував своє навчання та здібності на основі наданих нами даних про реальний світ.

Машинне навчання зараз існує у всіх сферах життя. Коли ми користуємося банківськими послугами, спілкуємося в соцмережах або робимо покупки в інтернеті, алгоритми машинного навчання допомагають зробити взаємодію з усіма сервісами зручніше, ефективніше і безпечніше. Машинне навчання та пов'язані з ним технології швидко розвиваються. А сьогоднішні можливості ML – це тільки вершина айсберга.

У даній роботі буде розказано про те, в яких сферах сьогодні вже існує машинне навчання, в яких тільки зароджується, а в яких використовується вже давно.

В роботі буде розкрито питання машинного навчання взагалі, а також застосування штучного інтелекту конкретно у сфері розробки ігор (гейм-деву): хто і як його використовує для скорочення часу розробки ігор, яка компанія створила гру з нескінченним простором та вільним світом, в якому нові всесвіті генеруються “на льоту”, а хто просто використовує методи та моделі машинного навчання задля того, щоб скоротити відтік користувачів з застосунку, чи заставити їх платити більше, або частіше.

Вздовж роботи буде обговорюватись яким чином збираються дані для подальших маніпуляцій, які існують методи класифікації користувачів. В нашому випадку – це будуть моделі бінарної класифікації, бо треба з’ясувати до якої групи (0 чи 1, неплатящі або платящі) відноситься той або інший юзер.

Будуть побудовані декілька моделей бінарної класифікації за допомогою Python та середовищ розробки Jupyter Notebook та Google Colab, моделі буде порівняно між собою та окремо будуть налаштовані параметри під кожну з моделей.

Також будуть розглянуті інструменти, за допомогою яких можна зекономити багато часу програмування, користуючись вже існуючими функціями та методами. Такими інструментами в роботі будуть бібліотеки для мови програмування Python під назвами pandas, numpy та sklearn, за допомогою яких можна працювати з таблицями, фільтрувати та очищати «сирі» дані, а також будувати моделі та підбирати під них певні параметри запуску.

Об'єктом дослідження є поведінка користувачів у грі. Вздовж роботи буде розглянута поведінка, що завжди має певні наслідки та може бути використана бізнесом для розуміння потреб та мотивацій людини.

Предметом дослідження є дії покупця, його «відповіді» на запропоновані товари чи послуги, які є результатом обмірковувань та інших елементів поведінки.

Мета та завдання роботи – з’ясувати яка модель найкраще підійде до існуючих даних та який вплив це може мати на вже функціонуючий застосунок (мобільну гру).

РОЗДІЛ 1. РОЛЬ МОДЕЛЮВАННЯ ПОВЕДІНКИ СПОЖИВАЧА У СВІТІ ТА МОЖЛИВОСТІ МАШИННОГО НАВЧАННЯ

1.1. Моделювання поведінки споживача

Моделювання поведінки споживачів (клієнтів) у різних джерелах визначається неоднозначно, але основним визначенням моделювання поведінки є створення математичної конструкції для представлення загальної поведінки, що спостерігається серед певних груп клієнтів, щоб передбачити, як подібні клієнти можуть поводитись за певних обставин.

Моделі поведінки клієнтів зазвичай базуються на аналізі даних клієнтів, і кожна модель розроблена для відповіді на одне запитання в певний момент часу. Наприклад, модель клієнта можна використовувати, щоб передбачити, що зробить конкретна група клієнтів у відповідь на певну маркетингову дію. Якщо модель надійна і маркетолог дотримується рекомендацій, які вона генерувала, тоді маркетолог помітить, що більшість клієнтів у групі відповіли так, як передбачала модель [1].

На жаль, створення моделей поведінки клієнтів, як правило, є складним і дорогим завданням. Це пояснюється тим, що розумних і досвідчених експертів з аналітики клієнтів, які знають, як це зробити, дорого і важко знайти, а також тому, що математичні методи, які їм потрібно використовувати, є складними та ризикованими.

Крім того, навіть після того, як модель поведінки покупців була побудована, важко маніпулювати нею для цілей маркетолога, тобто точно визначити, які маркетингові дії слід зробити для кожного клієнта або групи клієнтів.

Нарешті, незважаючи на свою математичну складність, більшість моделей клієнтів насправді відносно прості. Через цю необхідність більшість моделей поведінки клієнтів ігнорують таку кількість відповідних факторів, що прогнози, які вони створюють, зазвичай не дуже надійні.

Ринкова система на сьогоднішній день влаштована так, що покупець у ній – центральна ланка. Саме він є об'єктом впливу з боку маркетологів, які так чи інакше прагнуть знайти оптимальні способи задоволення споживчих запитів та інтересів. При цьому важливо, щоб маркетингові хитрощі відрізнялися від конкурентних. Для цього створюють, наприклад, УТП – унікальні торгові пропозиції, проводять акції та різні рекламні кампанії.

Стратегія в маркетингу формулюється таким чином, щоб забезпечити споживачеві більше цінностей, ніж конкурентні компанії, але зберегти фінансову стабільність і ефективність. Як правило, це виявляється у таких поняттях та характеристиках: ціни, канали просування та комунікації, сервісне обслуговування [2].

У пострадянському просторі така явна орієнтація на закриття потреб і «болей» стала якимось нововведенням, бо за радянських часів думки та інтереси людей не враховувалися, а вся діяльність будувалася на певному державному плані. В даний час компанії прагнуть привернути увагу до свого продукту, незважаючи на широку асортиментну різноманітність, представлену на ринку конкурентними фірмами.

Важливо також відзначити, що теорія поведінки клієнтів є фундаментальною основою, яка допомагає будувати бізнес у будь-якій сфері. Далі спробуємо розібратися, чим керуються люди після ухвалення рішення про купівлю того чи іншого товару, продукту. Основні характеристики споживчої поведінки:

раціональність – клієнт вибирає товар відповідно до своїх смаків, інтересів, потреб та фінансових можливостей. Саме тому виробники прагнуть якнайбільше розширити асортимент, надати можливість вибору та порівняння продукції однієї категорії;

незалежність вибору – той випадок, коли людина ухвалює рішення про покупку самостійно;

множинність – кількість пропозицій знаходиться у прямій залежності від дій покупця і навпаки. Враховуючи, що сьогодні ринок товарів та послуг

переповнений різними продуктами, які можуть задовольнити майже усі інтереси, поведінка споживачів та фактори, що визначають поведінку, стають з кожним днем дедалі різноманітними [3].

Як приймається рішення про покупку? Як правило, користувачі обирають враховуючи:

власні інтереси, смаки, потреби;

зацікавленість в продуктах;

користь;

фінансові можливості і розцінки.

А на основі перелічених показників формується базова поведінкова модель.

Покупець прагне вибрати саме той товар, який буде відповідати його інтересам, приносити користь, але при цьому враховується ціновий параметр. Якщо є фінансові обмеження, то людина купує лише один продукт або той, що знаходиться нижче в шкалі зменшення корисності та більш доступний за ціною.

Залежно від того, яку продукцію розглядає людина для придбання та споживання – зубну пасту, сезонне взуття, дорогий комп'ютер чи квартиру, його дії відрізнятимуться. Очевидно, що це обумовлено ціною придбання та ризиками. Чим більшу цінність має предмет, тим обережніше і зваженіше клієнт приймає рішення про покупку [4].

Також існують декілька основних груп людей за поведінкою:

складна поведінка – спостерігається у випадках, коли в межах однієї категорії товару пропонується великий вибір. Як приклад можна навести покупку смартфона. Техніка – не найдешевша категорія продукції, тому для покупця важливо оцінити придбання найдрібніших деталей. Він оцінює властивості, аналізує можливості та технічні характеристики, тому головне завдання продавця – це надати найбільш повну інформацію;

невпевнена поведінка – виникає у ситуаціях, коли на обмежену пропозицію є великий попит, але при цьому й величезний ризик. Найчастіше

при виборі людина спирається на власні смакові уподобання, які завжди чітко корелюють з корисністю. Здебільшого це категорія товарів для самореалізації. Завдання, яке стоїть перед продавцем у цьому випадку, переконати клієнта в правильності придбання;

звична поведінка – характерно для споживачів, коли вони купують знайомі, як правило, недорогі продукти. У різних виробників вони не дуже відрізняються, тому не оцінюються властивості або інші характеристики. Швидше, купівля відбувається через постійну звичку купувати саме цей продукт. Завдання продавця привернути увагу, наприклад, за допомогою стимулюючих акцій, знижок, бонусів, а також створення яскравого логотипу;

пошукова поведінка – виникає, коли ринку представлена широка лінійка виробників і продукції потрібної категорії. У клієнта виникає бажання спробувати все, що пропонується, тому часто змінює різні марки. Головне завдання продавця – правильне розміщення акцентів для стимулювання покупців на фіксування бренду в пам'яті, а також створення акцій та вигідних пропозицій [5].

Для того, щоб спробувати посприяти на бажання покупця купити ваш товар, або послугу, треба розуміти, як поетапно формується поведінкова тактика покупця, від чого це залежить, і що передбачає кожен крок:

усвідомлення – здійснення покупки мотивоване впливом певних чинників. Наприклад, фізіологічні чи естетичні. При цьому на вибір того чи іншого варіанта впливають також зовнішні подразники: запахи, смакові характеристики, колір і т.д. Розуміння того, як саме можна використовувати цей вплив, надає перевагу продавцю перед конкурентами;

пошук – подразники змушують людину шукати інформацію про товар. Інформація отримується через різні джерела (реклама, ЗМІ, рекомендації знайомих). Після збору даних споживач проводить порівняння і, ґрунтуючись на отриманому аналізі, робить свій вибір;

оцінка – сформована з урахуванням попереднього етапу модель поведінки у результаті призводить до прийняття конкретного рішення.

Тут важливо розуміти, через які канали відомості про товар доходять до покупця, якою є його оцінка, адже для клієнта це не просто предмет, а набір певних характеристик, що задовольняють його запити;

прийняття рішення – клієнт вже склав якийсь рейтинг, на вершині якого розташовується найбільш бажане придбання, а внизу – найменш;

реакція – формується на основі переваг окремо взятого покупця, а також реальних властивостей та якостей придбаного товару. При цьому людина може залишитися задоволеною або навпаки розчаруватися.

Мотивація покупця часто ґрунтується на потребах, бажаннях і можливостях людини. В зв'язку з цим у світі вже давно існують основні наукові теорії у цій сфері, наприклад:

піраміда Маслоу – знаменита ієрархія потреб, в основі якої стоять фізіологічні потреби (голод, спрага, статевий потяг тощо.), але в вершині – духовні (компетентність, досягнення успіху, приналежність до спільності). Як тільки задовольняються базові бажання, вони перестають мотивувати людину, також це працює і у зворотному напрямку – ті, хто стурбований пошуком їжі, не замислюється над духовним та культурним розвитком;

теорія МакКлелланда, за якою головна мотивація для індивіда – здобуття влади. На її основі владних прагнень та формується тактика, що впливає на вибір [6].

Аналітика допомагає розібратися в уподобаннях людей, їх вимогах, відношенні до інших виробників. Також за допомогою оцінки можна відстежити рівень лояльності та задоволеності продукцією. Це дозволяє адаптувати маркетингову стратегію таким чином, щоб вона була найефективнішою: можливо, варто покращити ціноутворення, оптимізувати канали просування чи змінити рекламну політику.

До основних напрямів комплексного аналізу покупців, а також до факторів, що впливають на їх поведінку в системі маркетингу, можна віднести:

аналіз потреб – головна мета це розуміння того, що потрібно покупцю

зараз, які його бажання не задоволені;

аналіз очікувань – тут відбувається вивчення стандартів обслуговування, яке клієнти очікують побачити;

дослідження сприйняття;

якість обслуговування – наприклад, це можуть бути технічні чи організаційні аспекти;

скарги – часто це джерело інформації про роботу компанії сприймається як позитивне, оскільки допомагає зрозуміти основні проблеми, усунути їх або по можливості уникнути їх повторення.

Розвиток нових сфер економіки сприяє зміні купівельних пріоритетів у бік покращення способу життя, мотивації, переконання. Так зростає не лише різноманітність та загальний обсяг пропозицій на ринку, а й потреби стають більш різноманітними, що грає на руку бізнесу, пов'язаному з інноваціями [7].

1.2. Можливості машинного навчання

Машинне навчання – це клас методів штучного інтелекту, який орієнтований на створення системи, які навчаються, або вдосконалюють продуктивність завдяки аналізу даних. Штучний інтелект – це широке поняття, яке відноситься до системи або машини з «людським» мисленням. Поняття «машинне навчання» та «штучний інтелект» часто використовуються в одному контексті, іноді як взаємозамінні, однак вони мають різне значення. Різниця полягає в тому, що машинне навчання завжди має на увазі використання штучного інтелекту, однак штучний інтелект не завжди має на увазі машинне навчання [8].

Машинне навчання вважається підмножиною штучного інтелекту. "Інтелектуальний" комп'ютер мислить як людина і самостійно виконує завдання. Один із способів навчити комп'ютер імітувати мислення людини –

використовувати нейронну мережу. Це серія алгоритмів, змодельованих за принципом людського мозку.

Машинне навчання може використовуватись як тип прогнозної аналітики. Але є одна важлива особливість. Машинне навчання значно простіше реалізувати з оновленням у реальному часі, оскільки воно забезпечує більше даних. Прогнозна аналітика зазвичай працює зі статичним набором даних та потребує регулярного оновлення [9].

Глибоке навчання в машинному навчанні – це спеціалізована форма, яка використовує нейронні мережі для надання відповідей. Алгоритми глибокого навчання можуть самостійно визначати точність. Вони дозволяють класифікувати інформацію так само, як людський мозок. Такі алгоритми лежать в основі одних із найближчих до мислення людини систем штучного інтелекту.

Машинне навчання сьогодні існує у всіх сферах життя. Кожен раз, коли ми користуємося банківськими послугами, робимо покупки в інтернеті або спілкуємося в соцмережах, алгоритми машинного навчання допомагають зробити взаємодію з усіма сервісами зручніше, ефективніше і безпечніше. Машинне навчання та пов'язані з ним технології швидко розвиваються. А сьогоднішні можливості ML – це тільки вершина айсберга [10].

Серед сфер, в яких може використовуватися машинне навчання сьогодні:

банківська справа та фінанси – управління ризиками та запобігання шахрайству – це найважливіші напрями, рентабельність яких можна значно підвищити за допомогою машинного навчання;

охорона здоров'я – діагностика, моніторинг стану пацієнтів та прогнозування епідемій – це лише небагато прикладів областей охорони здоров'я, в яких можна досягти високих результатів з використанням машинного навчання;

транспортування – виявлення аномалій дорожнього руху, оптимізація маршрутів доставки та автономне керування – це лише кілька напрямків

транспортної галузі, до яких машинне навчання може внести значні покращення;

обслуговування клієнтів – відповіді на запитання, визначення намірів клієнтів та надання віртуальної допомоги є прикладами того, як машинне навчання підтримує сферу обслуговування клієнтів;

роздрібна торгівля – машинне навчання допомагає роздрібним продавцям аналізувати закономірності при купівлі, оптимізувати пропозиції та ціни, а також підвищити загальний рівень обслуговування клієнтів на основі отриманих даних [11].

Можливості машинного навчання лаконічно описав глава Amazon Джефф Безос: «За останні десятиліття комп'ютери автоматизували багато процесів, які програмісти могли описати через точні правила та алгоритми. Сучасні техніки машинного навчання дозволяють нам робити те саме із завданнями, для яких набагато складніше задати чіткі правила» [12].

Twitter навчив нейромережі так публікувати зображення, щоб у центрі виявився самий важливий та цікавий елемент. Apple Watch має намір рекомендувати плейлисти з iTunes, які будуть враховувати серцевий ритм власника гаджета. Віртуальні помічники Siri, Cortana або Google Now вже давно допомагають у вирішенні повсякденних питань, наприклад, який фільм переглянути [13].

Машинне навчання має на увазі використання математичних моделей даних, які допомагають комп'ютеру навчатися без людських інструкцій. При машинному навчанні за допомогою алгоритмів виявляються закономірності в даних. На основі цих закономірностей створюється модель даних для прогнозування. Чим більше даних обробляє така модель і чим більше вона використовується, тим точніше становляться результати. Це дуже схоже на те, як людина відточує навички на практиці. А завдяки адаптивному характеру машинного навчання воно відмінно підходить для сценаріїв, в яких дані постійно змінюються, властивості запитів чи задач нестабільні або написати код для рішень фактично неможливо.

Машинне навчання застосовується у багатьох областях. Його можливості постійно розширюються. Ось деякі з основних переваг, які компанії набули завдяки проектам машинного навчання:

отримання аналітичних відомостей – машинне навчання допомагає визначати закономірності чи структури з урахуванням як структурованих, і неструктурованих даних, щоб отримати важливі аналітичні відомості;

підвищення рівня цілісності даних – машинне навчання - ідеальний варіант для інтелектуального аналізу даних. Технологія підвищує його точність та розширює можливості в динаміці;

розширення можливостей користувачів – адаптивні інтерфейси, цільовий вміст, чат-боти та віртуальні помічники з підтримкою голосу – це приклади того, як машинне навчання розширює можливості користувачів;

зменшення ризику – тактики шахраїв постійно змінюється. Машинне навчання дозволяє відстежувати та визначати нові прийоми зловмисників, щоб можна було вжити заходів, перш ніж буде завдано будь-якої шкоди;

прогнозування поведінки клієнтів – за допомогою машинного навчання можна виконувати інтелектуальний аналіз даних, пов'язаних із клієнтами. Це дозволяє виявляти закономірності та особливості поведінки, щоб оптимізувати рекомендації щодо продукту та максимально підвищити рівень обслуговування клієнтів;

зниження витрат – одна із сфер застосування машинного навчання – автоматизація процесів, що звільняє час та ресурси для виконання найважливіших завдань.

Зараз існують три основні напрямки машинного навчання (рис. 1.1):

контрольоване навчання – ця методика підходить для наборів даних із мітками або структурою. Дані виступають як викладач. Вони "навчають" комп'ютер, розширюючи його можливості прогнозування чи ухвалення рішення;

неконтрольоване навчання – ця методика підходить для наборів даних без міток чи структури. Щоб визначити закономірності та зв'язки, дані

групуються у кластери;

навчання з підкріпленням – замінюючий оператор-агент (програма, яка діє від чийогось імені) допомагає визначити результат на основі циклу зворотного зв'язку [14].

Який підхід найкраще відповідає потребам залежить від структури та обсягу даних, а також сценарію використання. Машинне навчання вже з успіхом застосовується в різних галузях для різних цілей і сценаріїв використання, включаючи наступні:

визначення цінності клієнта;

виявлення аномалій;

динамічне ціноутворення;

предиктивне керування обслуговуванням;

класифікація образів;

розробка рекомендацій.

Рис. 1.1. Класифікація методів машинного навчання

Типовий алгоритм машинного навчання звичайно складається з

чотирьох пунктів:

1) збір та підготовка даних – після визначення джерел даних, доступні дані компілюються. Тип даних, що використовуються, допоможе визначити, які алгоритми машинного навчання ви можете застосовувати. При перевірці даних виявляються аномалії, розробляється структура та усуваються проблеми з цілісністю даних;
2) навчання моделі – підготовлені дані поділяються на дві групи: набір для навчання та набір для перевірки. Набір для навчання – це більшість даних, за допомогою яких моделі машинного навчання налаштовуються з максимальною точністю;
3) перевірка моделі – при виборі кінцевої моделі даних продуктивність та точність оцінюється за допомогою набору для перевірки;
4) інтерпретація результатів – отримані дані вивчаються, для того щоб отримати аналітичні відомості, сформувати висновки та спрогнозувати результати [15].

Машинне навчання допомагає отримувати додаткову вартість із величезних обсягів даних, доступних сьогодні компаніям. Проте неефективні процеси можуть перешкодити компанії реалізувати його повний потенціал. Щоб машинне навчання приносило користь компанії, потрібна комплексна платформа, яка спростить виконання операцій та розгортання масштабних моделей. Правильно підібране рішення допомагає організаціям об'єднувати всі процеси з data science на єдиній платформі та прискорювати застосування інструментів, середовищ та інфраструктур на основі відкритого коду [16].

1.3. Машинне навчання в повсякденному житті

Машинне навчання знаходить собі застосування у житті кожної людини. Використовуючи дані про місцезнаходження зі смартфонів, Google Maps може в будь-який час перевіряти рухливість руху транспорту, крім

того, карта може організовувати дані про дорожній рух, про які повідомляють користувачі, як-от будівництво, затори та аварії [17].

Маючи доступ до відповідних даних і відповідних алгоритмів подачі, Google Maps може скоротити час у дорозі, вказуючи найшвидший маршрут.

Від того, як встановити ціну на поїздку та як мінімізувати час очікування, до того, як автомобілі, що їздять, регулюють поїздку з іншими пасажирами, щоб зменшити відволікання. Так, рішення – машинне навчання. ML допомагає компанії оцінити ціну поїздки, обчислюючи оптимальне місце відправлення та забезпечуючи найкоротший маршрут поїздки, а також для виявлення шахрайства. Наприклад, Uber використовує машинне навчання для оптимізації своїх послуг [18].

Деякі спам-фільтри засновані на правилах машинного навчання, вони не пропускають більшість спам повідомлень, наприклад, якщо повідомлення приходить зі словами «онлайн-консультація», «онлайн-аптека» або «невідома адреса».

Або авто-відповіді? Gmail, наприклад, підказує прості фрази для відповіді на електронні листи: «Дякую», «Гаразд», «Так, мені цікаво». Ці відповіді налаштовуються на електронну пошту, коли ML та AI розуміють, оцінюють і розмірковують про те, як вони протистоять з часом.

Найбільш актуальна для нас тема – антиплагіат. ML можна використовувати для створення детектора плагіату. Багато шкіл та університетів вимагають, щоб перевірки на плагіат аналізували навички письма учнів.

Алгоритмічна суть плагіату полягає у функціях подібності, які призводять до чисельної оцінки того, наскільки ідентичні два документи.

А під час завантаження фотографії у Facebook, вона автоматично відображає обличчя та пропонує теги друзів. Facebook використовує AI та ML для ідентифікації обличь.

Також давно знайома всім тема – маски. Instagram або Snapchat пропонує фільтри для обличчя, які фільтрують і відстежують активність

обличчя, дозволяє користувачам позначати анімовані зображення або цифрові маски, які змінюються, коли їх обличчя рухаються [19].

У банківській сфері, під час подання заявки на отримання кредитної картки чи позики, штучний інтелект допомагає швидко визначити, давати цій людині позику, чи ні. І, якщо надати пропозицію, то які конкретні умови можна запропонувати щодо процентної ставки, суми кредитної лінії тощо.

Також останнім часом набуває популярності розпізнавання мовлення – це переклад вимовлених слів у текст, також відоме як комп'ютерне розпізнавання мовлення. Тут програма може розпізнавати слова, вимовлені в аудіоформаті або файлі, а потім конвертувати аудіо в текстовий файл. Вимірюванням у цій програмі може бути набір чисел, які представляють мовний сигнал [20].

Розпізнавання мовлення використовується в таких програмах, як голосовий інтерфейс користувача, голосовий пошук тощо. Голосові інтерфейси користувача включають голосовий набір, маршрутизацію викликів та керування приладами. Також можна використовувати простий введення даних і підготовку структурованих документів.

Комп'ютерний зір – це взагалі окремий всесвіт, в якому машинне навчання максимально розкривається. Станом на 2021 рік навіть у нас в Україні вже широко використовують технологію розпізнання обличь або об'єктів камерою, чи, якщо точніше сказати, штучним інтелектом. Є також всім знайомі приклади, такі як "листи щастя" в Дію або додому в поштову скриньку. Якщо вам хоч раз приходив штраф за перевищення або інше правопорушення, яке автоматично зафіксувала камера, то є можливості комп'ютерного зору.

Pinterest, наприклад використовує комп'ютерний зір, щоб автоматично розпізнавати об'єкти на зображеннях або «закріпити», а потім рекомендувати подібні зображення. Інші програми охоплюють запобігання пам'яті, пошук і виявлення, маркетинг електронною поштою, ефективність реклами тощо за допомогою машинного навчання [21].

Щодо користі працівникам з даними, за допомогою машинного навчання можна витягнути структуровану інформацію з неструктурованих даних. Організації накопичують величезні обсяги даних від клієнтів. Алгоритм машинного навчання автоматизує процес очищення наборів даних для інструментів прогнозної аналітики.

Моделі машинного навчання можна створювати, обробляти, затверджувати, покращувати та випробовувати з використанням останніх пристроїв і досягнень. Це гарантує швидшу динамічну, розширену прибутковість, автоматизацію бізнес-вимірів та швидшу ідентифікацію аномалій для організацій [22].

1.4. Використання машинного навчання в гейм-деві

Розробка гри – це дуже складний процес, який включає в себе ігрове середовище, сюжети, особливості кожного персонажа тощо. Безсумнівно, що це процес трудомісткий та вимагає великих витрат. Однак, коли це виконується ефективно, результати бувають дуже грандіозні. Наприклад, поєднання доповненої реальності та розробки ігор створило “Pokemon Go”, ігрову сенсацію, яка за перший місяць залучила більш, ніж 200 мільйонів активних користувачів.

Віртуальна реальність є дуже захоплюючою сферою досліджень, вчені прогнозують, що вплив штучного інтелекту виявиться набагато більш корисним. Перш ніж досліджувати можливості успіху, важливо отримати критичне розуміння того, як працює штучний інтелект і як його можна застосувати.

Доктор Джуліан Тогеліус, професор інженерної школи Тандон, Нью-Йорк, пояснив, що штучний інтелект є одним із способів зробити ігри більш прибутковими. Незважаючи на те, що деякі люди стверджують, що його можна використовувати, тільки щоб краще кинути виклик гравцям і створити

більш реалістичні ігри, або складніших супротивників для справжніх гравців, але Джуліан пропонує простіший підхід [23].

Існують такі ігри, як Spelunkyy, які можуть змінювати складність кожного рівня під час гри, No Man’s Sky, яка використовує майже нескінченну кількість планет для створення цілого всесвіту. Алгоритми пошуку шляху, які дозволяють персонажам визначати альтернативні маршрути, є одними з розроблених областей штучного інтелекту. Крім того, “розумний” дизайн персонажів у таких іграх, як Creatures, Skyrim, Galactic Arms і Assault, є важливими етапами [24].

Таким чином, машинне навчання може мати величезний вплив на те, як розробляються ігри. У пошуках більш реалістичних світів, захоплюючих завдань та унікального контенту магазини розробників відеоігор все частіше звертаються до машинного навчання як до корисної зброї в розробці ігор. Алгоритми машинного навчання можуть динамічно реагувати на дії гравця. У той час як все в сучасних відеоіграх має бути написане вручну, відео ігри з механізмом машинного навчання можуть відреагувати і змінити поведінку світу, неігрових персонажів (NPC) або об’єктів в режимі реального часу на основі дій і рішень гравця [25].

Розглянемо як машинне навчання та на яких етапах може використовуватись безпосередньо на самому етапі розробки гри.

Алгоритми гри неігрових персонажів. NPC, яких можна навчати, є нетривіальним удосконаленням для розробки ігор. В даний час ігрові студії витрачають сотні людино-годин на написання сценаріїв NPC. NPC без жорсткого кодування можуть значно скоротити цикл розробки гри. Від тижнів до годин.

Робити ігри більш красивими. Розробники ігор також використовують машинне навчання на цьому фронті. У відеоіграх часто все виглядає добре здалеку, але коли ви наближаєтеся, об’єкти відображаються погано і стануть піксельними. Microsoft співпрацює з Nvidia над цією проблемою. Вони використовують машинне навчання для динамічного покращення зображень і

візуалізації. У реальному житті, коли ви знаходитесь далеко від об’єкта, деталі неясні, але коли ви наближається, ви можете помітити більш дрібні деталі. Таке динамічне відтворення дрібних деталей є проблемою, з якою можуть допомогти алгоритми комп’ютерного зору [26].

Більш реалістичні взаємодії. Використання обробки природної мови може дозволити вам власним голосом розмовляти з ігровими персонажами та отримувати реальні відповіді, подібно до розмови з Siri, Alexa або Google Assistant. Крім того, ігри, які включають тактику VR або зображення гравця, можуть дозволити алгоритмам комп’ютерного зору виявляти мову тіла та наміри, ще більше покращуючи досвід взаємодії з NPC.

Створення всесвіту на льоту. Одним з найбільш перспективних застосувань машинного навчання в розробці ігор є створення світу на льоту. На сьогоднішній день одними з найпопулярніших відеоігор є ігри з відкритими картами, які дозволяють досліджувати величезний ландшафт. Ці ігри вимагають тисячі годин часу розробника та виконавця для візуалізації. Однак алгоритми машинного навчання можуть допомогти у пошуку шляху та створенні світу. Прикладом є така гра, як No Man’s Sky, з нескінченною кількістю нових світів, які можна відкрити, і всі вони створюються на льоту, коли ви досліджуєте [27].

Більш захоплюючі мобільні ігри. Мобільні ігри складають 50% доходів від ігор у всій галузі. Ігри на вашому телефоні чи планшеті легко підібрати та грати, коли у вас простої, без спеціального пристрою. У минулому мобільні ігри були обмежені за обсягом, оскільки ваш телефон не мав процесорної потужності та графіки, як у консолі чи ПК. Однак ці обмеження починають змінюватися з використанням мікросхем штучного інтелекту в новітніх смартфонах, які додають спеціалізовану потужність обробки. Багато переваг машинного навчання, про які йшлося вище, стануть доступними для мобільних ігор, а також апаратне забезпечення продовжує вдосконалюватися, роблячи мобільні ігри більш реалістичними, інтерактивними та захоплюючими [28].

Досі існують серйозні проблеми, з якими стикаються програми машинного навчання в іграх. Однією з основних проблем є відсутність даних, з яких можна вчитися. Ці алгоритми будуть моделювати складні системи та дії, і ми не маємо хороших історичних даних про ці складні взаємодії. Крім того, алгоритми машинного навчання, розроблені для ігрової індустрії, мають бути надійними. Вони не можуть зламати гру чи досвід гравця. Це означає, що алгоритми мають бути правильними, але вони також мають бути швидкими та безперебійними з точки зору гравця. Все, що уповільнює або порушує гру, виводить гравця із занурення у світ, який створила гра.

Тим не менш, більшість великих студій розробки ігор мають команди, які досліджують, уточнюють та застосовують AI у своїх іграх. Це виклик, над яким працюють багато компаній, оскільки він дає таку захоплюючу можливість розширити відеоігри на нові горизонти, надаючи гравцям ще більш реалістичні враження та більше ігрового контенту [29].

Але машинне навчання в гейм-деві використовується не тільки в момент розробки самої гри, але й процесах підтримки самої гри, так званого LiveOps (Live Operations).

Пошук схожих користувачів. Наприклад, компанія за допомогою інструментів машинного навчання може вирішувати питання залучення нових клієнтів, що приносять більш прибутку, ніж інші. Кожен бізнес має групу клієнтів, яка приносить грошей більше, ніж інші. Логічно, що бізнес зацікавлений у розширенні бази таких клієнтів.

Цього можна досягти за допомогою інструменту look-alike, що базується на алгоритмах машинного навчання. Його роботу можна описати так: алгоритми вивчають найбільш прибуткових клієнтів, відокремлюють їх загальні риси і моделі поведінки, а потім складають портрет ідеального замовника. Після цього цей анонімний портрет завантажується в рекламні мережі, які згодом показують рекламу бізнесу тим людям, які максимально схожі на ідеального користувача.

У гейм-деві таким чином розширюється група гравців, які приносять

найбільше прибутку. Цей інструмент працює і для інших бізнесів, які збирають та систематизують інформацію про своїх клієнтів.

Динамічна зміна ціни. Напевно ви помічали, що в різних країнах, містах або навіть районах, в одних й тих же магазинах ціна на один товар може сильно відрізнятися. Приклад терміналу який показує ціну на товар з динамічною ціною можна зустріти в магазинах Amazon (рис. 1.2).

Таким же чином відповідно до країни, в якій ви знаходитесь, можуть відрізнятися ціни не лише на ігри, а й на внутрішньо ігрові продукти, наприклад алмази чи підписку на повний контент.

Рис. 1.2. Термінал з динамічною ціною на обраний товар

Динамічні ціни активно використовують авіакомпанії щодо вартості квитків. Саме цим пояснюються різні ціни на той самий рейс залежно від

того, з якого пристрою ви зайшли на сайт авіакомпанії. Не відстають Uber та Airbnb, які змінюють ціни залежно від поточного рівня попиту.

Щоб динамічне визначення ціни працювало правильно, потрібно навчати алгоритми враховувати та аналізувати інформацію на двох рівнях: макро (ціна у конкурентів, сезонність, зміни на ринку тощо) та мікро (поведінка користувача та його переваги). І глобальним лідером тут є Amazon, який здатний алгоритмічно, з урахуванням усіх факторів, змінювати ціни на сотні тисяч товарів за лічені хвилини.

Передбачення відтоку користувачів. Машинне навчання можна використовувати і для передбачення догляду клієнтів, що уходять. У гейм-деві це виглядає приблизно так: система зауважує, що гравця, наприклад, занадто часто вбивають - це веде до того, що він все рідше заходить у гру і проводить у ній дедалі менше часу. По суті, це вже перша ознака того, що незабаром він піде. Побачивши це, ми можемо знизити рівень складності, щоб відродити інтерес користувача до гри [30].

Аналогічно і для інших сфер бізнесу: клієнт став рідше замовляти таксі, розплачуватись карткою, менше купувати чи користуватися сервісом – привід задуматися про те, як його утримати чи повернути. Знижка, подарунок, ексклюзивні умови - це перше, що в такій ситуації спадає на думку, але рішення можуть бути і більш витонченими.

Персональні пропозиції. В іграх алгоритми навчилися відстежувати та аналізувати поведінку гравців для того, щоб робити їм індивідуальні пропозиції. Наприклад, якщо гравець активно “прокачує” у свого персонажа навички стрільби з лука, він швидше купить ексклюзивний лук, ніж потужний дворучний меч. Завдяки такому підходу прибуток від продажів внутрішньоігрового контенту може зрости на 20-40%.

Той самий підхід може використовуватися і в інших сферах бізнесу. Алгоритми вже вміють аналізувати поведінку користувачів та знаходити приховані моделі поведінки. Отже, вони розуміють, коли і що можна запропонувати клієнту. Якщо взяти як приклад рітейл, то лідерами у

використанні цих технологій є Amazon та Alibaba [31].

РОЗДІЛ 2. ІНСТРУМЕНТИ ДОСЛІДЖЕННЯ ТА ОПИС МЕТОДІВ БІНАРНОЇ КЛАСИФІКАЦІЇ

2.1. Python, середовища PyCharm і Jupyter, Google Colab та мова SQL

Для реалізації моделі була обрана інтерпретована об’єктно-орієнтована мова програмування Python. Python є мовою високого рівня зі строгою динамічною типізацією. Плюсом інтерпретованої мови програмування є відсутність необхідності переведення коду програми до машинного. Це дозволяє скоріше запускати програмні продукти і легше встановлювати необхідні для мови налаштування. Python може бути встановлений з різною версією в любому каталозі комп’ютера, що робить можливим контролювати та використовувати версії як самої мови, так і швидко змінювати версії пакетів. Раніше недоліком була повільна робота самих програмних продуктів на Python, якщо він би вийшов до світу раніше, то люди не змогли його використовувати, коли на першому місці була швидкість алгоритмів. Але зараз багато модулів переписані на мовах C++ та C, що дозволяє виконувати швидкі обчислення, так і комп’ютерне обладнання стало більш потужним [32].

Найголовнішою причиною вибору Python є існування пакетів, які дозволяють будувати сучасні штучні нейронні мережі, такі як TensorFlow, бібліотеки з вже готовими моделями “sklearn”, та пакетами, що стали вже класичними для первинної обробки даних: “numpy” і “pandas” [33].

Також існують зручні середовища для обробки даних і написання коду на мові Python – PyCharm і Jupyter Notebook.

PyCharm – інтегрована середа розробки. Надає засоби для аналізу коду, графічний відлагоджувач, інструменти для юніт-тестів та багато іншого. PyCharm розроблена компанією JetBrains. Перевагами середи розробки являються:

статичний аналіз коду;

швидка та зручна навігації серед проектів та їх структурою;

можливість задавати параметри до модулів;

вбудований відлагоджувач.

За цими перевагами зручно розробляти великі проекти та писати чистий код за PEP 8, але є два недоліки, з якими справляється Jupyter Notebook – будування графіків та покроковий запуск коду [34].

Jupyter Notebook – відкритий проект, який дозволяє запускати код Python та R на локальному хості. Хоч в ньому немає відлагоджувача та не так зручно керувати складними проектами, але його головною перевагою є можливість запускати код у різних клітинах. Обчислення в однієї клітині відбуваються незалежно від іншого, хоча вони й посилаються однаково до пам'яті. Це дуже зручно, коли треба побудувати декілька наглядних графіків, та становиться необхідним, коли йде мова про штучні нейронні мережі. Навчання мережі в найкращому випадку займає від півгодини до декількох годин. Але вона може навчатися від дня до неділь. Дуже неприємно, коли наприкінці навчання відбувається помилка і ваги не зберігаються, тоді весь процес приходиться робити спочатку та чекати знов тій час [35].

Тому доцільно вважати, що штучну нейронну мережу краще навчати в середовищі Jupyter Notebook, а готове рішення розробляти в PyCharm.

Google Colab – це продукт від Google Research, який дозволяє будь-кому писати та виконувати довільний код Python через браузер, і особливо добре підходить для машинного навчання, аналізу даних та освіти. Більш технічно, Colab – це розміщена служба ноутбуків Jupyter, яка не вимагає налаштування для використання, забезпечуючи безкоштовний доступ до обчислювальних ресурсів, включаючи графічні процесори.

За допомогою Colab можно безкоштовно прямо із свого браузера користуватися такими можливостями:

писати та виконувати код на Python;

коментувати свій код, який підтримує математичні рівняння;

створити/завантажувати/надавати доступ до блокнотів;

імпортуйте блокноти з Google Диску;

імпортувати блокноти із GitHub;

імпортувати зовнішні набори даних, наприклад від Kaggle;

застосовувати бібліотеки, наприклад pandas, numpy, sklearn.

SQL (Structured Query Language) – це стандартизована мова програмування, яка використовується для керування реляційними базами даних і виконання різних операцій над даними в них. SQL регулярно використовують не лише адміністратори баз даних, а й розробники, які пишуть сценарії інтеграції даних, і аналітики даних, які прагнуть налаштувати та запустити аналітичні запити.

Застосування SQL включають модифікацію структури таблиць та індексів бази даних; додавання, оновлення та видалення рядків даних; і отримання підмножин інформації з бази даних для обробки транзакцій та аналітичних додатків. Запити та інші операції SQL мають форму команд, записаних у вигляді операторів - зазвичай використовувані інструкції SQL включають вибір, додавання, вставку, оновлення, видалення, створення, зміну та скорочення.

SQL став фактично стандартною мовою програмування для реляційних баз даних після того, як вони з'явилися. Також відомі як бази даних SQL, реляційні системи містять набір таблиць, що містять дані в рядках і стовпцях. Кожен стовпець таблиці відповідає категорії даних, наприклад, імені або адреси клієнта, тоді як кожен рядок містить значення даних для стовпця, що перетинається [36].

В роботі буде застосована мова SQL задля того, щоб отримати всю необхідну інформацію з хмарного хранилища BigQuery, в якому в сотнях таблиць зберігаються дані о подіях користувачів, а також інформація про їх девайси, місця знаходження тощо.

BigQuery – це повністю кероване безсерверне сховище даних, яке забезпечує масштабований аналіз петабайтів даних. Це платформа як послуга (PaaS), яка підтримує запити за допомогою ANSI SQL. Він також має

вбудовані можливості для машинного навчання.

Серед функцій, що надає BigQuery:

управління даними – можливість створювати та видаляти такі об’єкти, як таблиці, запити та функції, визначені користувачем. Також імпортувати дані з Google Storage у таких форматах, як CSV, Parquet, Avro або JSON;

запити – вони виражаються на стандартному діалекті SQL, а результати повертаються у форматі JSON з максимальною довжиною відповіді приблизно 128 МБ або необмеженим розміром, якщо ввімкнено великі результати запитів;

інтеграція – BigQuery можна використовувати зі скрипту Google Apps (наприклад, як зв’язаний сценарій в Google Docs) або будь-яку мову, яка може працювати з його REST API або клієнтськими бібліотеками [37].

контроль доступу – є можливість ділитися наборами даних із довільними особами, групами чи світом.

машинне навчання – можливість створювати та виконувати моделі машинного навчання за допомогою запитів SQL [38].

Тож завдяки усьому набору цих інструментів: Python, PyCharm, Jupyter Notebook, BigQuery та SQL ми можемо зібрати і обробити (очистити) дані, побудувати декілька моделей класифікації за відфільтрованими даними, обрати найкращу і підібрати до неї певні параметри.

2.2. Бібліотеки для обробки даних у Python і моделювання

Розглянемо особливості використання бібліотек для обробки даних у Python і моделювання на їхній основі.

Numpy – це бібліотека в відкритому доступі, яка використовується для математичних обчислень завдяки швидким функціям, що можуть зрівнятись з функціями MatLab, та представленні даних у вигляді векторів та матриць. SciPy розширює функціонал numpy колекцією алгоритмів, таких як

мінімізація, перетворення Фур'є, регресія і інших.

Математичні алгоритми, реалізовані мовами, що інтерпретуються (наприклад, Python), часто працюють набагато повільніше тих же алгоритмів, реалізованих компілюваними мовами (наприклад, C, Java). Бібліотека NumPy надає реалізації обчислювальних алгоритмів (у вигляді функцій та операторів), оптимізованих для роботи з багатовимірними масивами. В результаті будь-який алгоритм, який може бути виражений у вигляді послідовності операцій над масивами (матрицями) та реалізований з використанням NumPy, працює так само швидко, як еквівалентний код, що виконується у MATLAB [39].

Масиви NumPy мають фіксований розмір під час створення, на відміну від списків Python (які можуть динамічно зростати). Зміна розміру ndarray створить новий масив і видалить оригінал.

Усі елементи в масиві NumPy повинні мати один тип даних і, таким чином, мати однаковий розмір пам'яті. Виняток: можна мати масиви об'єктів (Python, включаючи NumPy), що дозволяє використовувати масиви елементів різного розміру.

Зростаюча кількість науково-математичних пакетів на основі Python використовує масиви NumPy; хоча вони зазвичай підтримують введення послідовності Python, вони перетворюють такі вхідні дані в масиви NumPy перед обробкою, і вони часто виводять масиви NumPy. Іншими словами, для того, щоб ефективно використовувати більшу частину (можливо, навіть більшість) сучасного науково-математичного програмного забезпечення на основі Python, просто знати, як використовувати вбудовані типи послідовностей Python, недостатньо – потрібно також знати, як використовувати масиви NumPy [40].

Pandas – бібліотека, що надає змогу обробляти дані у вигляді таблиць із заголовками. Основними структурами зберігання даних являються Series і DataFrame. Series – аналог вектору, якщо розглядати його по значенням, але разом із іменованими індексами він більш походить на словник. DataFrame –

багатомірний масив, де кожен стовпець є структурою Series. Pandas дозволяє робити такі операції, як merge, join, concatenate, створювати Pivot таблиці і робити агрегатні обчислення. Дуже часто використовується для завантаження і вивантаження даних з csv, xlsx форматами (рис. 2.1). Основна сфера застосування - забезпечення роботи в рамках середовища Python не тільки для збору та очищення даних, але для завдань аналізу та моделювання даних, без перемикання на більш специфічні для статистичної обробки мови (такі, як R та Octave) [41].

```
anime.groupby(["type"]).agg({
  "rating": "sum",
  "episodes": "count",
  "name": "last"
}).reset_index()
```

	type	rating	episodes	name
0	Movie	14512.58	2348	Yasuji no Pornorama: Yacchimae!!
1	Music	2727.43	488	Yuu no Mahou
2	ONA	3679.43	659	Docchi mo Maid
3	OVA	20942.60	3311	Violence Gekiga Shin David no Hoshi: Inma Dens...
4	Special	10900.77	1676	Junjou Shoujo Et Cetera Specials
5	TV	25338.34	3787	Yuuki Yuuna wa Yuusha de Aru: Yuusha no Shou

Рис. 2.1. Ілюстрація можливостей Pandas

Пакет насамперед призначений для очищення та первинної оцінки даних за загальним показниками, наприклад, середнього значення, квантил і так далі; статистичним пакетом він у повному розумінні не є, проте набори даних типів DataFrame і Series застосовуються як вхідні в більшості модулів аналізу даних та машинного навчання (SciPy, sklearn та інших) [42].

Найкращий спосіб уявити структури даних pandas – це гнучкі контейнери для даних менших розмірів. Наприклад, DataFrame – це контейнер для серії, а Series – контейнер для скалярів. Ми хотіли б мати можливість вставляти та видаляти об’єкти з цих контейнерів у вигляді

словника [43].

Крім того, ми хотіли б розумної поведінки за замовчуванням для загальних функцій API, які враховують типову орієнтацію часових рядів і наборів даних перерізу. При використанні N-вимірного масиву (ndarrays) для зберігання 2- і 3-вимірних даних на користувача покладається тягар враховувати орієнтацію набору даних при написанні функцій; осі вважаються більш-менш еквівалентними (за винятком випадків, коли суміжність C або Fortran має значення для продуктивності). У пандах осі призначені для надання більшого семантичного значення даним; тобто для певного набору даних, ймовірно, існує «правильний» спосіб орієнтувати дані. Таким чином, мета полягає в тому, щоб зменшити кількість розумових зусиль, необхідних для кодування перетворень даних у нижчезазначених функціях [44].

Наприклад, з табличними даними (DataFrame) семантично корисніше думати про індекс (рядки) і стовпці, а не про вісь 0 і вісь 1. Таким чином, ітерація по стовпцях DataFrame призводить до більш читабельного коду [45].

Scikit-learn (також відомий як sklearn) – це безкоштовна бібліотека машинного навчання для мови програмування Python. Він містить різні алгоритми класифікації, регресії та кластеризації, включаючи машини опорних векторів, випадкові ліси, підвищення градієнта, k-середніх і DBSCAN, і розроблений для взаємодії з числовими та науковими бібліотеками Python NumPy і SciPy.

Однією з головних причин використання інструментів з відкритим кодом є величезна спільнота, яку вона має. Те ж саме стосується і sklearn. На сьогодні в scikit-learn є близько 35 учасників, найпомітнішим є Андреас Мюллер (шпаргалка Енді, зображена на рис. 2.2 з машинного навчання – одна з найкращих візуалізацій для розуміння спектру алгоритмів машинного навчання).

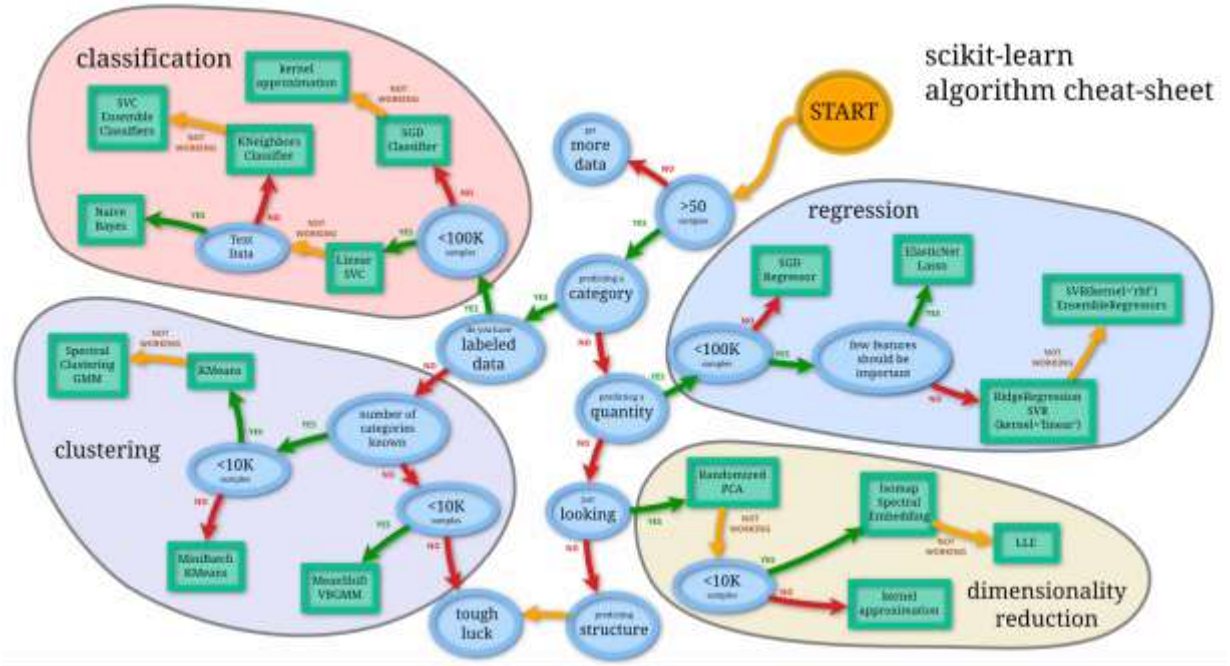

Рис. 2.2. Шпаргалка з машинного навчання Енді

Існують різні організації, такі як Evernote, Inria та AWeber, які відображаються на домашній сторінці scikit learn як користувачі. Але я справді вірю, що реальне використання набагато більше.

Крім цих спільнот, по всьому світу проводяться різноманітні зустрічі. Був також конкурс знань Kaggle, який нещодавно завершився, але все ще може стати одним із найкращих місць, щоб почати грати з бібліотекою.

В роботі будуть використані такі методи, імпортовані з бібліотеки sklearn, як GradientBoostingClassifier, LogisticRegression, LinearSVC, DecisionTreeClassifier, RandomForestClassifier, GaussianNB, KNeighborsClassifier для побудови відповідних моделей, а також GridSearchCV для подбору найкращих параметрів, train_test_split для розподілу групи на тестові і тренувальні вибірки, preprocessing для нормалізації даних, а також confusion_matrix для розрахунку ключових критеріїв моделі [46].

2.3. Класифікація, регресія та кластеризація у Python

Алгоритми машинного навчання, як правило, відрізняються за типом вихідної змінної та типом проблеми, яку необхідно вирішити. Ці алгоритми в цілому поділяються на три типи, тобто регресія, кластеризація та класифікація. Регресія та класифікація – це типи алгоритмів навчання з наглядом, тоді як кластеризація – це тип неконтрольованого алгоритму.

Якщо вихідна змінна є безперервною, то це проблема регресії, а якщо вона містить бінарні значення, це проблема класифікації. Алгоритми кластеризації зазвичай використовуються, коли нам потрібно створити кластери на основі характеристик точок даних.

Класифікація – це тип контрольованого алгоритму машинного навчання. Для будь-якого заданого входу алгоритми класифікації допомагають передбачити клас вихідної змінної. Може існувати кілька типів класифікацій, як бінарна класифікація, багатокласова класифікація тощо. Це залежить від кількості класів у вихідній змінній. Моделі класифікації бувають наступні:

логістична регресія – це одна з лінійних моделей, які можна використовувати для класифікації. Він використовує сигмовидну функцію для обчислення ймовірності настання певної події. Це ідеальний метод для класифікації двійкових змінних;

k-найближчі сусіди – цей метод використовує показники відстані, такі як евклідова відстань, відстань Манхеттена тощо, щоб обчислити відстань однієї точки даних від кожної іншої точки даних. Щоб класифікувати вихід, потрібна більшість голосів від k найближчих сусідів кожної точки даних;

дерева рішень – це нелінійна модель, яка долає деякі недоліки лінійних алгоритмів, як-от логістична регресія. Він будує модель класифікації у вигляді деревоподібної структури, яка включає вузли та листки. Цей алгоритм включає кілька операторів if-else, які допомагають розбити структуру на менші структури і в кінцевому підсумку забезпечити кінцевий

результат. Його можна використовувати для регресії, а також для задач класифікації;

випадковий ліс – це метод ансамблевого навчання, який включає кілька дерев рішень для прогнозування результату цільової змінної. Кожне дерево рішень дає свій власний результат. У випадку з проблемою класифікації для класифікації кінцевого результату потрібна більшість голосів із цих кількох дерев рішень. У випадку задачі регресії вона приймає середнє значення, передбачене деревами рішень;

наївний Байєс – це алгоритм, який базується на теоремі Байєса. Він передбачає, що будь-яка конкретна ознака не залежить від включення інших ознак. тобто вони не співвідносяться один з одним. Зазвичай він погано працює зі складними даними через це припущення, оскільки в більшості наборів даних існує певний зв’язок між ознаками;

машини опорних векторів – ця модель представляє точки даних у багатовимірному просторі. Ці точки даних потім поділяються на класи за допомогою гіперплощин. Він малює n-вимірний простір для n кількості об’єктів у наборі даних, а потім намагається створити гіперплощини так, щоб вони розділяли точки даних з максимальним запасом [47].

Більш детально методи класифікації будуть розглянуті із зображеннями в підрозділі 2.4. Розглянемо наступний метод вирішення задач машинного навчання – кластеризацію.

Кластеризація – це тип неконтрольованого алгоритму машинного навчання. Він використовується для групування точок даних, які мають подібні характеристики, як кластери. В ідеалі, точки даних в одному кластері повинні мати подібні властивості, а точки в різних кластерах повинні бути максимально різними.

Кластеризація поділяється на дві групи – жорстка кластеризація і м’яка кластеризація. У жорсткій кластеризації точка даних призначається лише одному з кластерів, тоді як у м’якій кластеризації вона забезпечує ймовірність того, що точка даних буде в кожному з кластерів. Кластеризація

може бути таких типів:

кластеризація k-середніх – ініціалізує попередньо визначену кількість k кластерів і використовує метрику відстані для обчислення відстані кожної точки даних від центроїда кожного кластера. Він призначає точки даних одному з k кластерів на основі його відстані [48];

ієрархічна кластеризація (підхід знизу вгору) розглядає кожну точку даних як кластер і об'єднує ці точки даних на основі метрики відстані та критерію, який використовується для зв'язування цих кластерів;

ієрархічна кластеризація розділення (підхід зверху вниз) ініціалізується з усіма точками даних як одним кластером і розбиває ці точки даних на основі метрики відстані та критерію. Агломеративна та роздільна кластеризація може бути представлена у вигляді дендрограми та кількості кластерів, які потрібно вибрати, посилаючись на те саме;

DBSCAN (просторова кластеризація додатків із шумом на основі щільності) – це метод кластеризації на основі щільності. Такі алгоритми, як k-середнє, добре працюють з кластерами, які досить розділені, і створюють кластери сферичної форми. DBSCAN використовується, коли дані мають довільну форму, і він також менш чутливий до викидів. Він групує точки даних, які мають багато сусідніх точок даних у певному радіусі;

OPTICS (впорядкування точок для визначення структури кластеризації) – це інший тип методу кластеризації на основі щільності, за процесом він подібний до DBSCAN, за винятком того, що враховує ще кілька параметрів. Але він більш складний у обчислювальному відношенні, ніж DBSCAN. Крім того, він не розділяє точки даних на кластери, але створює графік досяжності, який може допомогти в інтерпретації створення кластерів;

BIRCH (збалансоване ітераційне скорочення та кластеризація з використанням ієрархій) – ця модель створює кластери, генеруючи підсумок даних. Вона добре працює з величезними наборами даних, оскільки спочатку підсумовує дані, а потім використовує їх для створення кластерів. Однак модель може мати справу лише з числовими атрибутами, які можуть бути

представлені в просторі.

Різниця між моделями кластеризації та класифікації:

кластеризація – це метод навчання без нагляду, тоді як класифікація – це метод навчання під керівництвом;

у кластеризації точки даних групуються як кластери на основі їх схожості. Класифікація передбачає класифікацію вхідних даних як одну з міток класу вихідної змінної;

класифікація передбачає передбачення вхідної змінної на основі побудови моделі. Кластеризація зазвичай використовується для аналізу даних і отримання висновків з них для кращого прийняття рішень;

алгоритмам класифікації необхідно, щоб дані були розділені як навчальні та тестові дані для прогнозування та оцінки моделі. Алгоритми кластеризації не потребують розбиття даних для його використання;

алгоритми класифікації мають справу з міченими даними, тоді як алгоритми кластеризації мають справу з нерозміченими даними;

процес класифікації включає два етапи – навчання та тестування. Процес кластеризації передбачає лише групування даних;

оскільки класифікація має справу з більшою кількістю етапів, складність алгоритмів класифікації вища, ніж алгоритмів кластеризації, метою яких є лише групування даних.

Регресійний аналіз – це процес оцінки зв'язку між залежною змінною та незалежною змінною. Простішими словами, це означає підгонку функції з вибраного сімейства функцій до вибіркових даних за певною функцією помилки. Регресійний аналіз є одним з основних інструментів у сфері машинного навчання, що використовується для прогнозування. Використовуючи регресію, ви підходите до функції з доступними даних і намагаєтеся передбачити результат на майбутнє. Така відповідність функції виконує дві цілі:

інтерполяція – ви можете оцінити відсутні дані у вашому діапазоні даних;

екстраполяція – ви можете оцінити майбутні дані за межами діапазону даних.

Регресія також буває різних типів. У лінійній регресії мета полягає в тому, щоб вмістити гіперплощину (лінію для точок 2D даних), мінімізуючи суму середньоквадратичної помилки для кожної точки даних [49].

Лінійна регресія передбачає, що зв'язок між залежною (y) і незалежною (x) змінними є лінійною. Вона не зможе знайти точки даних, якщо зв'язок між ними не є лінійною. Поліноміальна регресія розширює можливості підгонки лінійної регресії шляхом підгонки полінома ступеня m до точок даних. Чим багатше розглянута функція, тим краще (загалом) її можливості підгонки

"Хребетна" регресія вирішує проблему переобладнання в регресійному аналізі. Вона намагається мінімізувати помилку узагальнення, порушуючи підгонку в навчальних точках.

Регресія LASSO подібна до "хребетної" регресії, оскільки обидва вони використовуються як регулювачі проти переобладнання в точках тренувальних даних. Але LASSO має додаткову перевагу. Це забезпечує розрідженість вивчених ваг.

"Хребетна" регресія змушує норму ваг бути малою, що дає набір ваг, де загальна норма зменшується. Більшість ваг (якщо не всі) будуть відмінними від нуля. LASSO, з іншого боку, намагається знайти набір ваг, роблячи більшість з них дійсно близькими до нуля. Це дає розріджену вагову матрицю, реалізація якої може бути набагато більш енергоефективною, ніж нерозріджена вагова матриця, зберігаючи подібну точність з точки зору підгонки до точок даних.

Логістична регресія корисна в задачах класифікації, де результатом має бути умовна ймовірність результату з урахуванням вхідних даних. Її було розглянуто вище, і більш детально вона буде описана у наступному підрозділі.

2.4. Моделі бінарної класифікації в машинному навчанні

Класифікація – завдання машинного навчання, що дуже часто зустрічається, і полягає в побудові моделей, що виконують віднесення цікавить нас об'єкта до одного з декількох відомих класів. Існують сотні методів класифікації, які можна використовувати для передбачення значення відгуку з двома та більше класами.

Бінарний класифікатор формує деяке діагностичне правило і оцінює, до якого з двох можливих класів слід віднести об'єкт, що вивчається (згідно з медичною термінологією умовно назвемо ці класи "норма" або "патологія"). Групи точок “патологія/норма” у заданому просторі предикторів, як правило, статистично нероздільні: наприклад, підвищення температури тіла до 37.5 °C часто свідчить про захворювання, хоча не завжди хвороба може супроводжуватися високою температурою. Тому під час тестування моделі можливі помилкові ситуації, такі як пропуск позитивного (патологічного) укладання FN чи його “гіпердіагностика” FP, тобто. віднесення нормального стану до патологічного [50].

Результати тесту на деякій контрольній вибірці можна уявити звичайною таблицею сполученості, яку часто називають матрицею неточностей (confusion matrix), яка наведена у табл. 2.1.

Таблиця 2.1

Матриця неточностей

Справжній стан	Передбачена патологія	Передбачена норма
1	2	3
Патологія	True Positive (TP)	False Negative (FP)
Норма	False Positive (FP)	True Negatives (TN)

Розроблена поведінкова модель буде прикладом бінарної класифікації, а залежною змінною (класом) буде той факт, здійснив користувач покупку, чи ні (1 або 0). А для того, щоб порівняти їх між собою та вибрати найкращий варіант, було побудовано одразу 7 моделей бінарної класифікації: логістичну регресію, машин опорних векторів, дерева рішень, Байєсівський класифікатор, метод найближчих сусідів, рандом-форест та градієнтний бустінг.

Наївні байєсівські класифікатори засновані на теоремі Байєса. Одне з прийнятих припущень – це тверді припущення про незалежність між ознаками. Ці класифікатори припускають, що значення певної ознаки не залежить від значення будь-якої іншої ознаки. У ситуації навчання під наглядом наївні байєсівські класифікатори навчаються дуже ефективно. Наївним класифікаторам Байєса потрібні невеликі навчальні дані, щоб оцінити параметри, необхідні для класифікації. Наївні класифікатори Байєса (рис. 2.3) мають просту конструкцію та реалізацію, і їх можна застосувати до багатьох ситуацій реального життя [51].

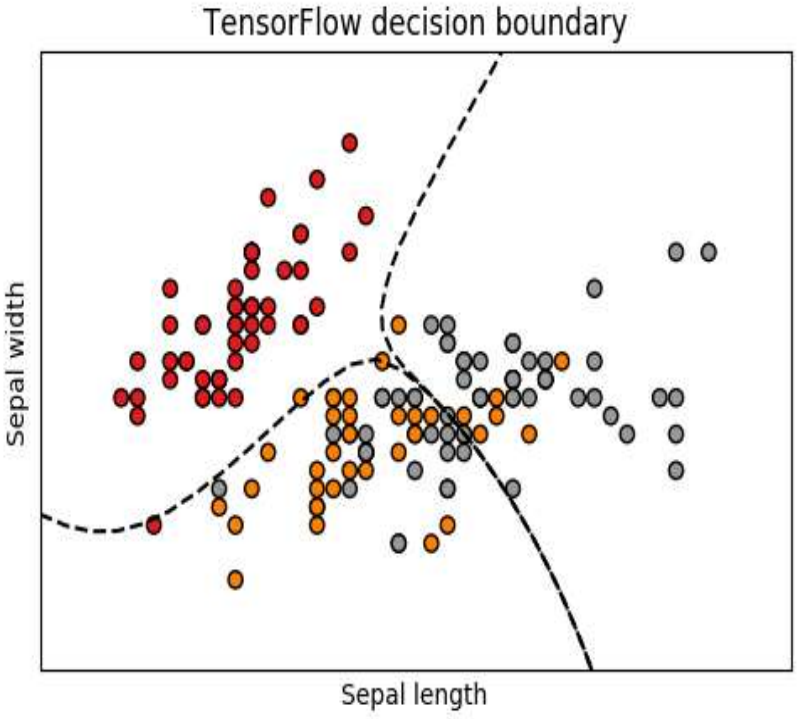

Рис. 2.3. Ілюстрація роботи наївного класифікатора Байєса

Під час роботи з безперервними даними часто прийнято вважати, що безперервні значення, пов’язані з кожним класом, розподіляються відповідно до нормального (або гаусового) розподілу. Вважається, що ймовірність ознак:

Логістична регресія, незважаючи на свою назву, є лінійною моделлю для класифікації, а не регресією. Логістична регресія також відома в літературі як логіт-регресія, максимальна ентропійна класифікація (MaxEnt) або логарифмічний лінійний класифікатор (рис. 2.4) [52].

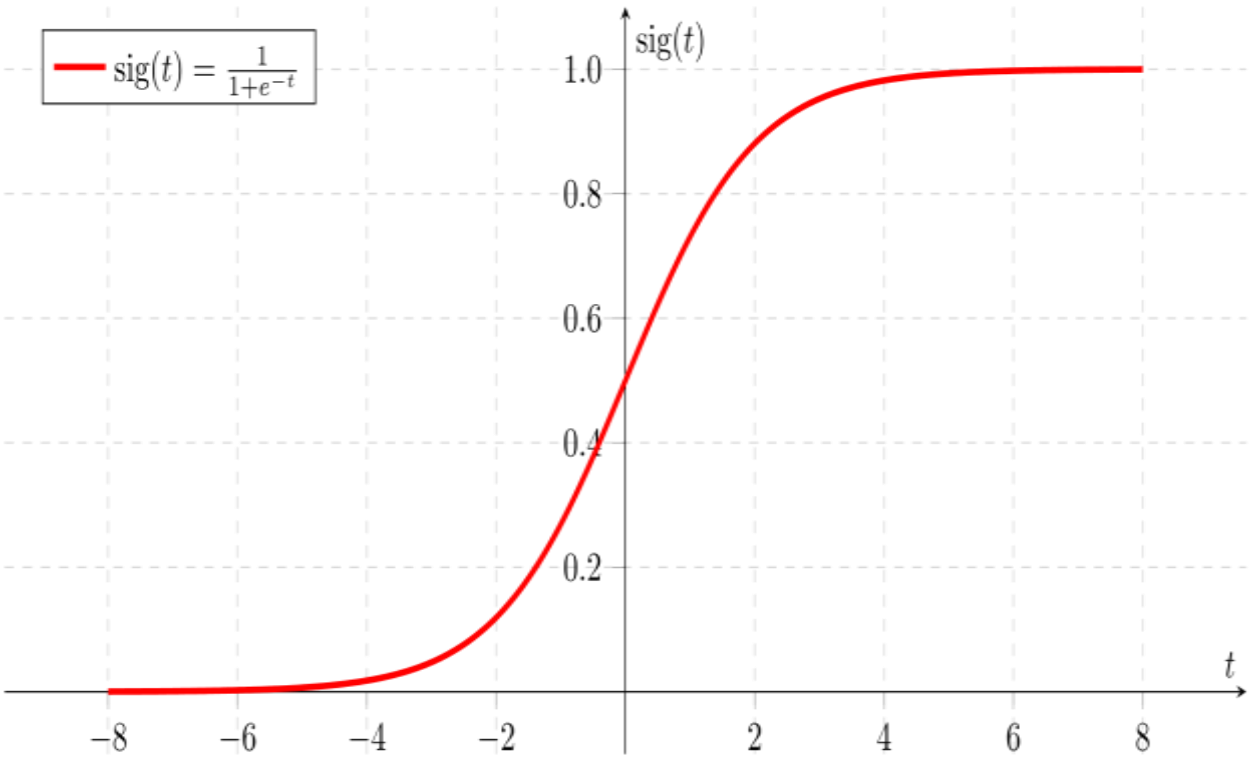

Рис. 2.4. Логістична регресія

Класифікація на основі сусідів – це тип навчання на основі екземплярів або навчання без узагальнення: вона не намагається побудувати загальну внутрішню модель, а просто зберігає екземпляри навчальних даних.

Класифікація обчислюється простою більшістю голосів найближчих сусідів кожної точки: точці запиту призначається клас даних, який має найбільшу кількість представників у найближчих сусідах точки.

Класифікація k-найближчих сусідів є найбільш часто використовуваною технікою. Оптимальний вибір значення сильно залежить від даних: загалом, більша величина пригнічує вплив шуму, але робить межі класифікації менш чіткими (рис. 2.5) [53].

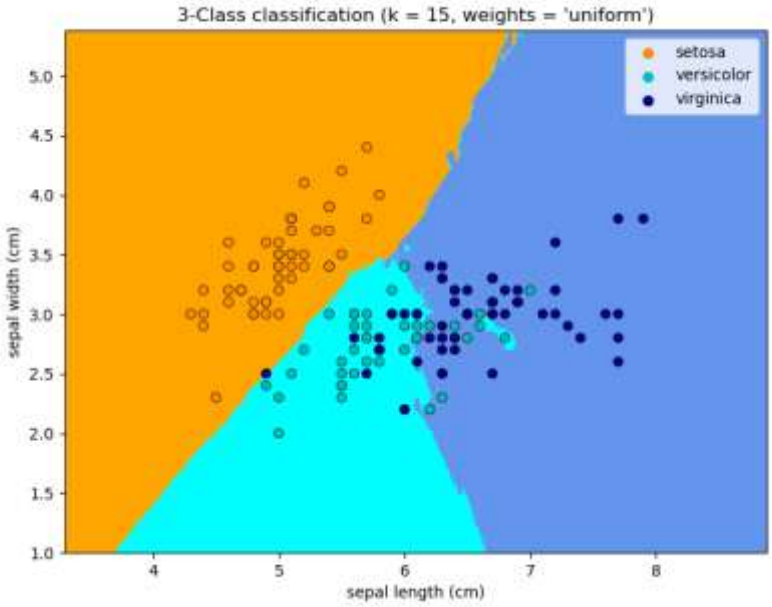

Рис. 2.5. Класифікація k-найближчих сусідів на 3 класи

У випадкових лісах (random-forest) кожне дерево в ансамблі будується із вибірки, намальованої із заміною (тобто за допомогою bootstrap) з навчального набору (рис. 2.6) [54].

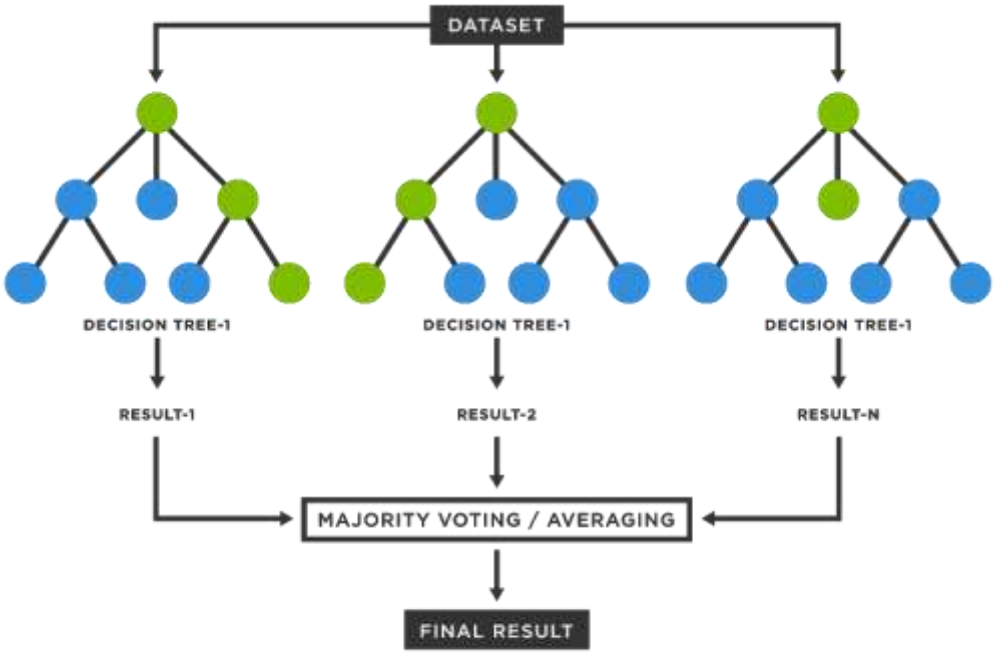

Рис. 2.6. Схема роботи моделі Random Forest

Дерево рішень – це інструмент підтримки прийняття рішень, який використовує деревоподібну модель рішень та їх можливих наслідків, включаючи випадкові результати подій, витрати ресурсів та корисність. Це один із способів відображення алгоритму, який містить лише умови умовного керування. Типовим прикладом для дерева рішень є модель з вижившими пасажирами на Титаніку (рис. 2.7), яка ілюструє від яких критеріїв залежала ймовірність вибратися з борту корабля живим, на якій описані ймовірності та без зусиль зрозуміло, що в хлопців, наприклад, шансів було значно менше, ніж у жінок [55].

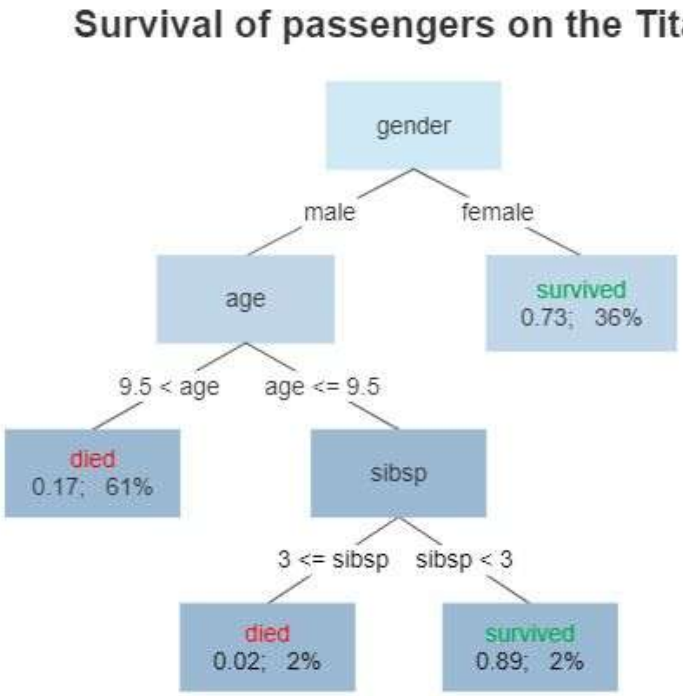

Рис. 2.7. Дерево рішень моделі вижившихна Титаніці

Метою лінійного SVC (класифікатора опорних векторів) є відповідність наданим вами даним, повертаючи «найкраще відповідну» гіперплощину, яка розділяє або категоризує ваші дані (рис. 2.8).

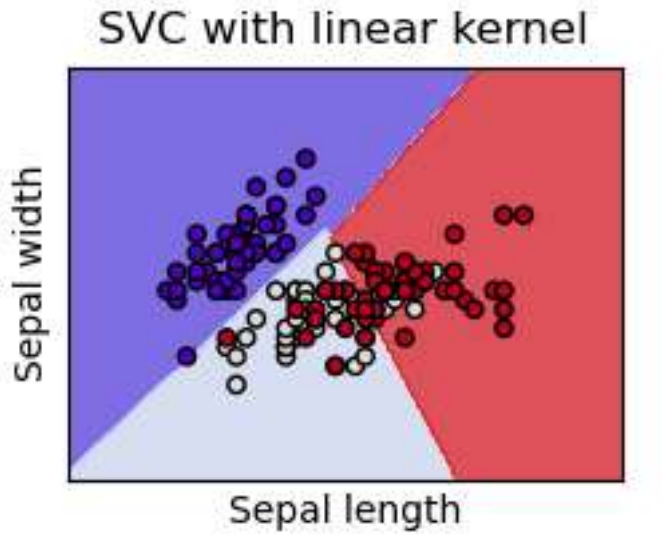

Рис. 2.8. Розподіл гіперплощини за класами на прикладі рослини

Звідти, після отримання гіперплощини, ви можете передати деякі функції своєму класифікатору, щоб побачити, що таке «передбачуваний» клас. Це робить цей конкретний алгоритм досить придатним нашого використання (рис. 2.8)

Ідея «градієнтного бустінгу» полягає в тому, щоб взяти слабку гіпотезу або слабкий алгоритм навчання та внести до неї ряд налаштувань, які підвищать міцність гіпотези.

Під час підвищення гіпотези переглядаються всі спостереження, на яких тренується алгоритм машинного навчання, і залишаються позаду лише спостереження, які метод машинного навчання успішно класифікував, видаляючи інші спостереження (рис. 2.9).

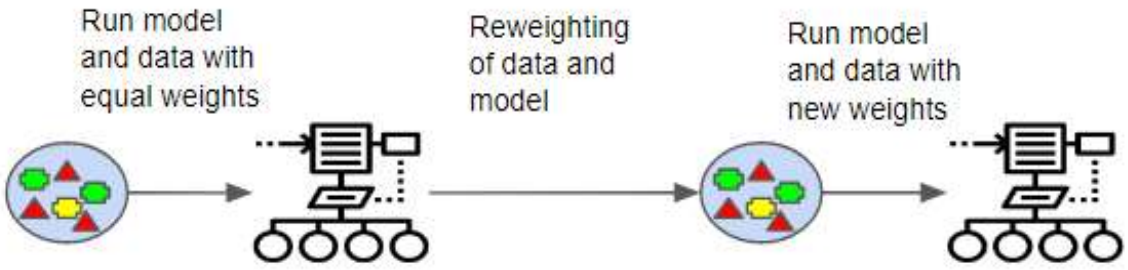

Рис. 2.9. Ілюстрація принципу роботи градієнтного бустінгу

Новий слабкий учень створюється і тестується на наборі даних, які були погано класифіковані, а потім зберігаються лише ті приклади, які були успішно класифіковані [56].

В даному розділі було розглянуто інструменти, які будуть використані в практичній частині роботи, а також була наведена інформація за моделями класифікацій, порівняно їх та розглянуто на прикладах.

РОЗДІЛ 3. ПОБУДОВА МОДЕЛІ ПОВЕДІНКИ СПОЖИВАЧА ЗА ДОПОМОГОЮ МЕТОДІВ КЛАСИФІКАЦІЇ МАШИННОГО НАВЧАННЯ

3.1. Збір вихідних даних з баз даних за допомогою SQL

Збір вихідних даних користувачів мобільної гри для побудови моделей машинного навчання в рамках даної роботи було виконано за допомогою SQL-запитів до десятків таблиць, що лежать на хмарному сховищі BigQuery. Крім того, щоб побудувати модель, було відокремлено 13 незалежних факторів, що можуть впливати на залежну метрику, якою є здійснення покупки споживачем. Загальний список факторів виглядає наступним чином:

Y - paid (1 чи 0) - позначає наявність покупки у користувача;

X_1 - тип девайсу (планшет чи телефон);

X_2 - назва девайсу;

X_3- джерело трафіку (звідки прийшов в ігру);

X_4- довжина першої сесії;

X_5 - кількість сесій в перший день;

X_6 - возраст;

X_7 - рід;

X_8 - кількість намальованих персонажів;

X_9 - кількість намальованих унікальних персонажів;

X_{10} - час, за який був намальований перший персонаж;

X_{11} - кількість часу, що було проведено в меню покупки;

X_{12} - кількість натискань на кнопку “купити”;

X_{13} - кількість відкриттів меню покупки.

Оскільки майже всі дані знаходяться в окремих таблицях, мені довелося писати майже з десяток окремих запитів, а також самому вираховувати окремі метрики, що не були надані в таблицях. Приклади таких запитів будуть надані в цьому розділі, а фінальний запит можна буде побачити в Додатку А.

Наприклад, запит для пошуку довжини першої сесії та кількості сесій за перші 24 години виглядає наступним чином (рис. 3.1).

```
SELECT user_pseudo_id,
  ROUND(((event_timestamp - user_first_touch_timestamp)/1000000/60)) as first_session_duration_min,
  (SELECT
    count(user_pseudo_id)
   FROM
   WHERE user_pseudo_id = minm.user_pseudo_id
    AND event_timestamp BETWEEN user_first_touch_timestamp
    AND (user_first_touch_timestamp + 86400 * 1000000) #проміжок між першим івентом та через 24 години
    AND event_date BETWEEN "2021-09-10" AND DATE_ADD(DATE "2021-09-10", INTERVAL 1 DAY)) as sessions_in_first_day
FROM
                                                    as minm
WHERE event_name = 'max_event'
  AND event_date = "2021-09-10"
  AND (SELECT value.int_value FROM UNNEST(user_properties) WHERE key = 'ga_session_number') = 1
```

Рис. 3.1. SQL запит на довжину першої сесії та кількості сесій за 24 години

А кількість часу, що було проведено в меню покупки, розраховувалась як різниця між івентами відкриття та закриття меню покупки (рис. 3.2).

```
SELECT
    DISTINCT user_pseudo_id,
    ((
        (
        SELECT MIN(event_timestamp)
        FROM
        WHERE event_date = gp.event_date AND user_pseudo_id = gp.user_pseudo_id
        ) - MIN(gp.event_timestamp)) / 1000000) as delta
FROM
                                                    as gp
WHERE
    event_date = "2021-09-10" AND
    (SELECT value.int_value FROM UNNEST(user_properties) WHERE key = 'ga_session_number') = 1 AND
    user_pseudo_id IN (
                        SELECT
                            user_pseudo_id
                        FROM
                        WHERE
                            event_date = '2021-09-10'
                        )
group by 1, gp.event_date
HAVING delta IS NOT NULL
```

Рис. 3.2. SQL запит на кількість часу, проведеного в меню покупки

А найпростішим запитом був запит на довжину першої сесії користувача (рис. 3.3), бо він розраховувався не інакше як різниця між часом першого входу в гру та часом останнього івенту за першу сесію [57].

```
SELECT user_pseudo_id,
  ROUND(((event_timestamp - user_first_touch_timestamp)/1000000/60)) as session_duration_in_min
FROM
                                                                    as minm
WHERE event_name = 'max_event'
  AND event_date = "2021-09-10"
  AND (SELECT value.int_value FROM UNNEST(user_properties) WHERE key = 'ga_session_number') = 1
```

Рис. 3.3. Довжина першої сесії користувача

Наступним кроком був розрахунок кількості відкриттів меню покупки, що дозволило б нам зробити припущення про те, що користувач зацікавлений у покупці (рис. 3.4)

```
SELECT user_pseudo_id,
    (
      SELECT
        COUNT(user_pseudo_id) FROM                                                    , UNNEST(event_params)
      WHERE
        user_pseudo_id = gp.user_pseudo_id
        AND event_date = gp.event_date
        AND key = 'source'
        AND value.string_value IN ('lock', 'gameMenu', 'instrumentLock')
    ) as lock_clicks
FROM
                                                                    as gp
WHERE
    event_date = "2021-09-10" AND
    platform = 'IOS' AND
    (SELECT value.int_value FROM UNNEST(user_properties) WHERE key = 'ga_session_number') = 1 AND
    user_pseudo_id IN (
                        SELECT
                            user_pseudo_id
                        FROM
                        WHERE
                            event_date = '2021-09-10'
                        )
GROUP BY 1, 2
```

Рис. 3.4. SQL запит на кількість відкриттів меню покупки

Далі треба було порахувати кількість натискань конкретно на кнопку “купити” в меню покупки, що було не так складно зробити завдяки тому, що цей івент мав окрему таблицю і всю інформацію за кликами певного юзера, якого ми шукаємо (рис. 3.5).

```
SELECT
    user_pseudo_id,
    COUNT(user_pseudo_id) as clicks_in_menu
FROM
    [redacted] as gp
WHERE
    event_date = "2021-09-10" AND
    (SELECT value.int_value FROM UNNEST(user_properties) WHERE key = 'ga_session_number') = 1 AND
    user_pseudo_id IN (
                        SELECT
                            user_pseudo_id
                        FROM
                            [redacted]
                        WHERE
                            event_date = '2021-09-10'
                        )
group by 1
ORDER BY 2 DESC
```

Рис. 3.5. SQL запит на кількість натискань на кнопку “купити”

Всю інформацію про девайс, країну реєстрації користувача тощо було витягнуто разом з запитом на унікальних та загальну кількість відрисованих персонажів (рис. 3.6).

```
SELECT
    user_pseudo_id,
    [redacted] device_type,
    [redacted]_name as device_name,
    [redacted] as traf_source,
    COUNT(user_pseudo_id) as all_painted,
    COUNT(DISTINCT value.string_value) as unique_painted
FROM
    [redacted] as gp, UNNEST(event_params) as ep
WHERE
    event_date = "2021-09-10" AND
    (SELECT value.int_value FROM UNNEST(user_properties) WHERE key = 'ga_session_number') = 1
    AND user_pseudo_id IN
                        (
                        SELECT
                            user_pseudo_id
                        FROM
                            `acad-bini-drawing-2019-06.general_data.fb_first_open`
                        WHERE
                            event_date = '2021-09-10'
                        )
    AND key = 'name'
    AND platform = 'IOS'
GROUP BY
    user_pseudo_id,
    [redacted]
ORDER BY 2 DESC
```

Рис. 3.6. SQL запит на кількість отрисованих персонажів

Важливим елементом усіх SQL запросів було використання різноманітних конструкцій обмежень задля того, щоб уникнути дублювання

інформації чи для того, щоб не витягнути два різних проміжку часу однієї і тієї ж людини, але з різних івентів, бо це цілком може посприяти на якість нашої моделі.

Тому, задля того, щоб дані були чистими і без помилковими, за відточенням і перевіркою SQL запросів було проведено не менш з десяток годин.

Не менш важливим пунктом було переведення категоріальних даних, наприклад “age”, “gender”, “device_type” та “device_name” до числових показників, наприклад в колонці “gender” хлопці були позначені як “1”, дівчини як “2”, а ті, хто не вказав свій род були позначені як “0”. А дані за колонкою “device_type” приймали два значення - “1” для мобільного телефона чи “2” для планшету.

Після того, як усі окремі запити були написані, вони були згруповані в один за допомогою функції SQL WITH(), яка дозволяє використовувати окремі запити як проміжні таблиці: групувати, сортувати, поєднувати їх тощо. Фінальний запит вийшов на 245 строки, а знайти його можна у додатку А.

3.2. Імпорт та обробка даних у Python за допомогою бібліотек

Для початку роботи з даними, треба обрати, в якому середовищі буде відбуватися робота з даними. Переваги окремих середовищ розглядались у підрозділі 2.1. За тією причиною, що було необхідно поетапно запускати окремі куски коду, та працювати з даними, що вже були оброблені, а не обробляти їх кожен раз окремо, вибір пав на Jupyter Notebook.

Але задля того, щоб випадково процес дипломної роботи не зник через неполадки з комп’ютером тощо, використався аналогічний сервіс, який працює на хмарному хранилищі - Google Colab, реалізован за тим же

принципом, що і Jupyter Notebook і також дозволяє дивитися проміжні результати, ілюстрації тощо.

Першим ділом треба імпортувати всі необхідні бібліотеки (пакети) Python, функції та методи яких ми будемо використовувати, вони розглядалися в підрозділі 2.2. Список імпортованих бібліотек виглядає наступним чином:

```
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
```

Кожен з імпортованих методів буде виконувати свою роль у моделі, якщо перші два модулі ми використовуємо для роботи з даними у табличному форматі, то інші задля того, щоб очистити, нормалізувати та обробити дані.

Наступним кроком дипломної роботи, та першим кроком, який ми робимо вже в середовищі, є імпортування даних в наш “ноутбук”, в якому ми будемо проводити подальші маніпуляції. Робиться це за допомогою, напевно, найбільш популярного метода бібліотеки pandas - pd.read_csv():

```
raw_data = pd.read_csv('/content/data-set.csv', sep=',',header=0)
```

Задаємо методу окремі параметри, зазначив, що перша строка - це назви колонок, а роздільником в нашому файлі є кома.

Далі перевіряємо, що наші дані були коректно завантажені в наш проект за допомогою метода head() і отримуємо наступні результати (рис. 3.7).

```
raw_data.head()
```

	user_pseudo_id	device_type	device_name	first_session_duration_sec	sessions_in_first_day	age	gender
0	[redacted]	1	5	2421	4	3.0	NaN
1	[redacted]	1	5	221	1	5.0	2.0
2	[redacted]	1	8	1985	2	NaN	NaN
3	[redacted]	1	1	60	1	1.0	NaN
4	[redacted]	1	1	157	1	1.0	2.0

Рис. 3.7. Результат виводу методу head()

Оскільки таблиця має 15 колонок, всі вони не вміщуються до рисунку, тож дані на рис. 3.7 предоставлені в скороченому форматі для перших п’яти строк.

Також за допомогою функції .describe() можна подивитися на числові якості даних за колонками: квартилі, мінімуми та максимуми, стандартні відхилення та середні значення. Результати наведені на рисунках 3.8 та 3.9:

	device_type	device_name	first_session_duration_sec	sessions_in_first_day	age	gender	all_painted
count	52536.000000	52536.000000	52536.000000	52536.000000	52536.00000	52536.000000	52536.000000
mean	1.359944	12.157930	911.023831	1.831259	1.94775	1.061710	4.676622
std	0.479988	9.974931	1374.207064	1.291383	1.73857	0.998104	8.213348
min	1.000000	1.000000	5.000000	0.000000	0.00000	0.000000	0.000000
25%	1.000000	1.000000	124.000000	1.000000	1.00000	0.000000	0.000000
50%	1.000000	12.000000	418.000000	1.000000	2.00000	2.000000	2.000000
75%	2.000000	17.000000	1163.000000	2.000000	3.00000	2.000000	6.000000
max	2.000000	37.000000	29899.000000	13.000000	6.00000	2.000000	385.000000

Рис. 3.8. Перша половина числової якостей даних

	unique_painted	sec_to_first_finish	sec_in_ia_menu	clicks_to_sub	lock_clicks	paid
count	52536.000000	5.253600e+04	52536.000000	52536.000000	52536.000000	52536.000000
mean	3.632576	4.724415e+02	283.947465	2.188537	1.345554	0.070847
std	5.503458	1.872160e+04	3648.526053	4.262098	4.188689	0.256571
min	0.000000	0.000000e+00	-6305.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000e+00	2.000000	0.000000	0.000000	0.000000
50%	2.000000	1.600000e+02	5.000000	1.000000	0.000000	0.000000
75%	5.000000	2.560000e+02	14.000000	3.000000	1.000000	0.000000
max	115.000000	2.046806e+06	84983.000000	157.000000	147.000000	1.000000

Рис. 3.9. Друга половина числової якостей даних

За цими результатами можна побачити, що дані в цілком в нормальному стані: кількість записів за колонками збігаються, а самі дані (квантилі, мінімуми, максимуми) виглядають цілком реальними. Але в усіх даних, які ми збираємо по користувачам завжди є виброси, так називаємі outliers, бо хтось може заспамити кнопку покупки, хтось може бросити включеним свій телефон під час першої сесії та інше. Тому з даними ще доведеться попрацювати в python для того, щоб вони набули належного виду.

Для початку заполнимо всі дані, яких в нас немає (NaN) нулями, бо інакше ми не зможемо адекватно використовувати їх в нашій моделі. Такі дані можна побачити на рисунку 3.3. За допомогою функції fillna() ми знаходимо значення NaN по колонках і замінюємо їх на нулі:

```
raw_data['age'] = raw_data['age'].fillna(0)
raw_data['gender'] = raw_data['gender'].fillna(0)
```

Далі обираємо колонки, в яких нам треба очистити виброси - це кількісні дані користувачів і записуємо їх у список для того, щоб потім швидко пробігтись по ним в циклі і відформатувати наші дані:

```
list_to_filter = ['sessions_in_first_day', 'sec_to_first_finish',
                  'sec_in_ia_menu', 'clicks_to_sub', 'lock_clicks']
```

Та створюємо нову таблицю, копіюючи наші “сирі” дані, над якою ми далі і будемо працювати:

```
filtered = raw_data.copy()
```

Для початку проводимо чистку даних лише з колонкою “first_session_duration”, яку ми не долучили до списку “list_to_filter” бо її треба очистити від викидів з обох сторін, відкинувши 2% нижніх і 2% верхніх даних:

```
low = raw_data['first_session_duration_sec'].quantile(0.02)
high  = raw_data['first_session_duration_sec'].quantile(0.98)
filtered = filtered[(filtered['first_session_duration_sec'] < high) &
                    (filtered['first_session_duration_sec'] > low)]
```

А в інших колонках, що наведені у списку, за якими ми будемо проходити у циклі, треба почистити лише зверху, бо там багато нулів серед даних, а видалення нижчих 2% убире ці дані з таблиці, тож:

```
for col in list_to_filter:
    high  = raw_data[col].quantile(0.98)
    filtered = filtered[(filtered[col] < high)]
```

І поглянемо на скільки змінилась картина в нашій таблиці за допомогою використання методу describe():

```
filtered.describe()
```

	device_type	device_name	first_session_duration_sec	sessions_in_first_day	age	gender	all_painted
count	46090.000000	46090.000000	46090.000000	46090.000000	46090.000000	46090.000000	46090.000000
mean	1.367151	12.319223	717.341484	1.655739	1.971425	1.076025	3.810523
std	0.482033	9.995002	859.734860	0.992919	1.738755	0.997117	5.946010
min	1.000000	1.000000	16.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	1.000000	121.000000	1.000000	1.000000	0.000000	0.000000
50%	1.000000	12.000000	378.500000	1.000000	2.000000	2.000000	2.000000
75%	2.000000	17.000000	973.000000	2.000000	3.000000	2.000000	5.000000
max	2.000000	37.000000	4790.000000	5.000000	6.000000	2.000000	90.000000

Рис. 3.10. Числова якість даних після чистки

Але на і на цьому чистка не була завершена, бо в таблицях з немалою кількістю даних, тим більш в іграх, завжди можуть зустрітися дублікати: коли на одного і того же юзера приходиться одразу дві перших сесії, чи його результати записуються двічі. Тут на допомогу приходить наступна функція drop_dublicates(). Ми вказуємо, що у фільтрованій таблиці бажаємо видалити

дублікати за колонкою “user_pseudo_id” - що є унікальним ідентифікатором користувача, та отримуємо результати на рис. 3.11.

```
filtered = filtered.drop_duplicates(subset=['user_pseudo_id'])
```

```
filtered.describe()
```

	device_type	device_name	first_session_duration_sec	sessions_in_first_day	age	gender	all_painted
count	43456.000000	43456.000000	43456.000000	43456.000000	43456.000000	43456.000000	43456.000000
mean	1.369017	12.329667	681.800534	1.634389	1.959200	1.066458	3.368119
std	0.482544	9.996756	827.966597	0.981208	1.740861	0.997801	5.172145
min	1.000000	1.000000	16.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	1.000000	112.000000	1.000000	1.000000	0.000000	0.000000
50%	1.000000	12.000000	359.000000	1.000000	2.000000	2.000000	2.000000
75%	2.000000	17.000000	914.000000	2.000000	3.000000	2.000000	4.000000
max	2.000000	37.000000	4790.000000	5.000000	6.000000	2.000000	90.000000

Рис. 3.11. Числові якості даних після видалення дублікатів

За цими результатами можна побачити, що в таблиці існували близько 2.5 тисяч дублікатів, які були видалені, а наші дані стали більш схожими на адекватні: різниця між середнім і максимальним а також стандартна помилка знизились.

Залишилось лише нормалізувати дані для того, щоб моделі їх обробили без якихось проблем. Для цього нам потрібно створити об’єкт нормалізатора, за допомогою функції MinMaxScaler(), та список колонок, які треба нормалізувати:

```
min_max_scaler = preprocessing.MinMaxScaler()
normalize_list = ['first_session_duration_sec', 'sessions_in_first_day',
                  'all_painted', 'unique_painted', 'sec_to_first_finish',
                  'sec_in_ia_menu', 'clicks_to_sub', 'lock_clicks']
```

І запустити цикл, що колонка за колонкою нормалізує усі дані в нашому дата-фреймі (рис. 3.12)

```
for col in normalize_list:
  filtered[col] = min_max_scaler.fit_transform(np.array(filtered[col]).reshape(-1,1))
```

```
filtered.head()
```

	user_pseudo_id	device_type	device_name	first_session_duration_sec	sessions_in_first_day	age	gender	all_painted	unique_painted
0	[redacted]	1	5	0.503770	0.8	3.0	0.0	0.111111	0.204082
3	[redacted]	1	1	0.009217	0.2	1.0	0.0	0.000000	0.000000
4	[redacted]	1	1	0.029535	0.2	1.0	2.0	0.000000	0.000000
6	[redacted]	2	28	0.062631	0.2	3.0	2.0	0.033333	0.020408
7	[redacted]	1	1	0.178676	0.2	0.0	0.0	0.011111	0.020408

Рис. 3.12. Нормалізовані дані у дата-фреймі

Отримавши адекватні дані, що були нормалізовані тепер є змога працювати з цими даними в моделях і отримувати чіткі результати.

3.3. Побудова та оцінка моделей класифікації

Коли дані повністю готові і очищені можна створювати модель. Для початку обозначимо в нашій моделі залежну і незалежні змінні:

```
y = filtered['paid']
X = filtered.iloc[:,1:13]
```

Зверніть увагу, що як незалежні змінні я беру з першої по тринадцяту колонку, хоча лічення в Python починається з нуля. Це за тією причиною, що ідентифікатор користувача “user_pseudo_id” нам був потрібен лише задля того, щоб ідентифікувати юзерів в процесі обробки даних та видаленні дублікатів, а далі ми будемо працювати з даними без нього.

Далі, за допомогою функції train_test_split дата-фрейм поділяється на тренувальну та тестову групи: одну для того, щоб навчити нашу модель ідентифікувати покупця від не покупця, а іншу - щоб перевірити на вже обученій моделі наскільки добре вона справляється зі своєю задачею:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

30% для тестової групи - це стандартний розмір розподілу, коли працюєш з моделями бінарної класифікації.

Далі був створений словник, в який спочатку були записані назви моделей, а далі будуть записані отриманні результати по тій чи іншій моделі:

```
models = {}
models['Logistic Regression'] = LogisticRegression(max_iter=5000)
models['Support Vector Machines'] = LinearSVC(max_iter=5000)
models['Decision Trees'] = DecisionTreeClassifier()
models['Random Forest'] = RandomForestClassifier()
models['Naive Bayes'] = GaussianNB()
models['K-Nearest Neighbor'] = KNeighborsClassifier()
models['Gradient Boosting'] = GradientBoostingClassifier()
```

Для кожного елементу словника було створено об’єкт з однойменною функцією для розрахунку моделі, а для моделей логістичної регресії та машин опорних векторів ми задаємо одразу параметр максимальної кількості ітерацій в 5000, бо при стандартній настройці вони не зможуть обробити таку кількість даних.

Далі було створено ще п’ять словників з критеріями оцінювання моделі, які в наступному циклі буду записувати в словник з назвою моделі, це буде словник, в якому значенням виступає також словник з результатами за критеріями оцінювання моделі, з якого ми зробимо дата-фрейм для більш приємного сприйняття результатів зором (рис. 3.13)

```
true_positive, false_positive, true_negative,
false_negative, accuracy = {}, {}, {}, {}, {}
for key in models.keys():
    models[key].fit(X_train, y_train)
    predictions = models[key].predict(X_test)

    true_negative[key], false_positive[key], false_negative[key],
    true_positive[key] = confusion_matrix(y_test, predictions).ravel()
```

accuracy[key] = (true_positive[key]+true_negative[key]) /
(true_positive[key]+false_positive[key]
+true_negative[key]+false_negative[key])

```
df_model = pd.DataFrame(index=models.keys(), columns=['True Positive', 'False Positive',
                                                      'True Negative', 'False Negative', 'Accuracy'])
df_model['True Positive'] = true_positive.values()
df_model['False Positive'] = false_positive.values()
df_model['True Negative'] = true_negative.values()
df_model['False Negative'] = false_negative.values()
df_model['Accuracy'] = accuracy.values()

df_model
```

	True Positive	False Positive	True Negative	False Negative	Accuracy
Logistic Regression	27	24	12159	827	0.934724
Support Vector Machines	12	5	12178	842	0.935031
Decision Trees	366	648	11535	488	0.912863
Random Forest	323	103	12080	531	0.951369
Naive Bayes	257	874	11309	597	0.887167
K-Nearest Neighbor	77	114	12069	777	0.931656
Gradient Boosting	326	89	12094	528	0.952673

Рис. 3.13. Результати класифікації всіх моделей

Тлумачення перших чотирьох колонок в дата-фреймі описані в підрозділі 2.3 разом з поясненням що таке confusion matrix, яку ми використовували для підрахунку результатів.

А з accuracy все простіше, це відсоток результатів, які правильно розподілила наша модель. І з однієї сторони все доволі ясно: беремо модель з найвищим показником точності та застосовуємо її до нашої моделі, але бувають випадки, коли не загальна точність результатів стоїть на першому місці, а точність правильно класифікованих true positive результатів, тобто в нашому випадку це ті люди, що купили, і модель так і сказала, що вони купили. В нашій ситуації відокремлюються три моделі: Random Forest, Decision Trees та Gradient Boosting, за True Positive результатами, які нам важливі, але дивлячись на загальну точність було обрано дві моделі, з якими

буде відбуватися подальша праця: градієнтний бустінг та метод випадкових лісів.

Підбір параметрів для обраних моделей можна здійснювати завдяки функції GridSearchCV, який створений для того, щоб перебрати всі комбінації параметрів, але в цього метода є значний мінус. Перебор декількох параметрів с десятками або сотнями можливих значень на виробничих потужностях може затягнутися до тижнів, було вибрано більш зручний варіант - ручний перебір параметрів.

За допомогою методу .get_params() ми дивимось які параметри може приймати наша модель та створюємо словник з декількома можливими значеннями для цих параметрів, згідно до рекомендацій документації sklearn

Попробувавши майже всі можливі варіанти параметрів, які є в документації, найкращий варіант було отримано з наступними параметрами (рис. 3.14).

```
mod = GradientBoostingClassifier(loss='deviance', max_depth=7, criterion='friedman_mse',
                                 learning_rate=0.1, validation_fraction=0.2, random_state=27)
mod.fit(X_train, y_train)
pr1 = mod.predict(X_test)
cm = confusion_matrix(y_test, pr1)

TN, FP, FN, TP = confusion_matrix(y_test, pr1).ravel()

print('True Positive(TP)  = ', TP)
print('False Positive(FP) = ', FP)
print('True Negative(TN)  = ', TN)
print('False Negative(FN) = ', FN)

accuracy1 =  (TP+TN) /(TP+FP+TN+FN)

print('Accuracy of the binary classification = {:0.3f}'.format(accuracy1))
```

```
True Positive(TP)  =  358
False Positive(FP) =  146
True Negative(TN)  =  12074
False Negative(FN) =  459
Accuracy of the binary classification = 0.954
```

Рис. 3.14. Оптимальні параметри моделі Gradient Boosting

Результати налаштування параметрів не дали сильного впливу на модель, але все одно вона змогла покращити результат пошуку True Positive результатів, які зросли на 10% у новій моделі, не втрачаючи загальну точність.

Наступним етапом було виконання тих самих інструкцій та маніпуляцій з моделлю Random Forest (рис. 3.15).

```
mod = RandomForestClassifier(criterion='entropy', n_estimators=100, max_depth=12,
                             bootstrap=False, class_weight='balanced')
mod.fit(X_train, y_train)
pr1 = mod.predict(X_test)
cm = confusion_matrix(y_test, pr1)

TN, FP, FN, TP = confusion_matrix(y_test, pr1).ravel()

print('True Positive(TP)  = ', TP)
print('False Positive(FP) = ', FP)
print('True Negative(TN)  = ', TN)
print('False Negative(FN) = ', FN)

accuracy1 =  (TP+TN) /(TP+FP+TN+FN)

print('Accuracy of the binary classification = {:0.3f}'.format(accuracy1))
```

```
True Positive(TP)  =  483
False Positive(FP) =  682
True Negative(TN)  =  11538
False Negative(FN) =  334
Accuracy of the binary classification = 0.922
```

Рис. 3.15. Оптимальні параметри моделі Random Forest

Цей випадок вже виглядає набагато краще, ніж попередній. І не дивлячись на те, що кількість False Positive результатів також дуже сильно виросла, якщо порівнювати з варіантом моделі без налаштованих параметрів, головною метрикою, яку треба було піднімати - це кількість True Positive результатів, що збільшилась на 50%, а зменшення точності самої моделі склало лише 3 процентних пунктів.

Якщо говорити про ефективність цієї моделі, то після введення її в розробку, буде змога надавати користувачам з більш низькою ймовірністю покупки (покупці, яким модель спрогнозувала “0”) знижку в 50%, і, якщо хоча б 1 із 50 таких покупців придбає наш товар зі знижкою, то прибуток компанії може зрости на 12.5%, що при поточних масштабах – дуже великі цифри. Але зробити остаточні висновки та розрахувати результати імплементації моделі можна буде згодом, коли модель запрацює в застосунку. А повну версію коду моделі можна знайти в додатку Б.

ВИСНОВКИ

В ході роботи було розкрито феномен машинного навчання в людському житті і в моделюванні поведінки користувачів зокрема. Також було розглянуто такі методи обробки даних як класифікація, регресія і кластеризація.

Окремо було розглянута тема бінарної класифікації користувачів за окремими признаками поведінки та проведене зрівняння різних методів.

Проведено витягання даних з хмарного сховища для подальшої роботи, а також очищення отриманих даних, видалення дублів, та ігнорування аномально великих та маленьких значень задля кращої роботи моделі.

Був написаний програмний скрипт для проведень маніпуляцій з даними за допомогою мови Python, інструменту Google Colab та підключених бібліотек для роботи з даними: pandas, numpy, та sklearn для побудування сьоми різних моделей машинного навчання для аналізу поведінки користувача та присвоєння йому статусу платника або не платника.

Було проведено зрівняння отриманих моделей між собою та подальший відбір кращих моделей, для яких було зроблено операцію з підбору оптимальних параметрів для кращої та точнішої роботи з прогнозом.

Згідно з отриманими результатами з нашими даними по поведінці клієнтів за допомогою моделі Random Forest ми зможемо з ймовірністю в 92.2% правильно розподіляти користувачів в групу платників або не платників.

Після проведеного дослідження можна зробити висновок, що використання моделей та методів машинного навчання на ігровому ринці однозначно може принести користь у вигляді додатково зароблених грошей з користувачів, що мали значно нижчі шанси на покупку платного контенту, якщо правильно використовувати отриману інформацію і намагатися зробити таким користувачам пропозицію, що буде виглядати вигідною для нього, а також залишатися вигідною для розробника.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Русина К.А. Модели поведения потребителей / Русина К.А. // Электронный вестник Ростовского социально-экономического института. – Ростов, 2014. – № 2. – С. 2-4.

2. Поведение потребителей: что это такое и какие социальные и психологические факторы влияют на принятие решения о покупке товаров и услуг / [Електронний ресурс]. – Режим доступу: https://www.cleverence.ru/articles/biznes/povedenie-potrebiteley-chto-eto-takoe-i-kakie-sotsialnye-i-psikhologicheskie-faktory-vliyayut-na-pri

3. Туртин Д.В. Об одной экономико-математической модели поведения покупателя на потребительском рынке / Туртин Д.В., Аржаных Т.Ф., Смирнова А.Н. // Современные наукоемкие технологии. Региональное приложение. – Иваново, 2016. – № 1. – С. 6-9.

4. Логинова Ю.В. Моделирование поведения интернет-потребителей на основе сложной вероятностной модели / Логинова Ю.В. // Вестник Кемеровского государственного университета. – Кемерово, 2014. – № 4. – С. 14-15.

5. Внутренние факторы поведения потребителей / [Електронний ресурс]. – Режим доступу: https://warmedia.ru/vnutrennie-faktory-povedeniya-potrebitelej

6. Лопатина А.Б. Психо-социальные модели поведения / Лопатина А.Б. // Международный научно-исследовательский журнал. – Пермь, 2016. – № 2. – С. 5.

7. Словарь маркетинговых терминов / [Електронний ресурс]. – Режим доступу: https://www.infosystems.ru/library/glossary/slovar-marketingovykh-terminov

8. A. Y. Vladova Data preprocessing for machine analysis of sales representatives' key performance indicators / A. Y. Vladova, E.D. Shek // Business-Informatics. – Moskow, 2021. – № 3. – С. 33.

9. Machine learning / [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Machine_learning

10. П.Ю. Богданов Программные среды для изучения основ нейронных сетей / П.Ю. Богданов, Е.В. Краева, С.А. Веревкин // Программные продукты и системы. – Санкт-Петербург, 2021. – № 1. – С. 145-150.

11. E.A. Olayori The impact of machine learning/artificial intelligence technology on the functionalities of corporate consumers / E. A. Olayori // StudNet. – Pavlodar, 2021. – № 7. – С. 241-257.

12. 2016 Letter to Shareholders / [Електронний ресурс]. – Режим доступу: https://www.aboutamazon.com/news/company-news/2016-letter-to-shareholders

13. На что способны алгоритмы? Возможности машинного обучения / [Електронний ресурс]. – Режим доступу: https://te-st.ru/2019/01/29/what-can-robots-do/

14. A. Y. Vladova Data preprocessing for machine analysis of sales representatives' key performance indicators / A. Y. Vladova, E.D. Shek // Business-Informatics. – Moskow, 2021. – № 3. – С. 33.

15. What is machine learning? / [Електронний ресурс]. – Режим доступу: https://azure.microsoft.com/en-us/overview/what-is-machine-learning-platform/#uses

16. Что такое машинное обучение? / [Електронний ресурс]. – Режим доступу: https://www.oracle.com/ru/data-science/machine-learning/what-is-machine-learning/

17. Д.Ю. Черкасов Машинное обучение / Д.Ю. Черкасов, В.В. Иванов // Наука, техника и образование. – Москва, 2018. – № 7. – С. 311.

18. Титова А.Ю. Разработка модели анализа сложных данных на основе классификации machine learning / А.Ю. Титова // Вестник

Национального технического университета Харьковский политехнический институт. Серия: Информатика и моделирование. – Харьков, 2018. – № 3. – С. 113-119.

19. Лобин М.А. Машинное обучение в экономике / М.А. Лобин, И.А. Филиппова // Вестник Ульяновского государственного технического университета. – Ульянов, 2019. – № 3. – С. 68-71.

20. 7 Popular Applications of Machine Learning in Daily Life / [Електронний ресурс]. – Режим доступу: https://www.analyticssteps.com/blogs/7-popular-applications-machine-learning-daily-life

21. Флах П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных / пер. с англ. А.А. Слинкина. – М.: ДМК Пресс, 2015. – 400 с.

22. Top 10 real-life examples of Machine Learning / [Електронний ресурс]. – Режим доступу: https://bigdata-madesimple.com/top-10-real-life-examples-of-machine-learning/

23. How video games can help Artificial Intelligence deliver real-world impact / [Електронний ресурс]. – Режим доступу: https://news.itu.int/video-games-artificial-intelligence

24. Does Machine Learning have a Place in Game Development? / [Електронний ресурс]. – Режим доступу: https://medium.com/@TheGeekiestOne/does-machine-learning-have-a-place-in-game-development-b34b0352d78a

25. Breiman L. Bagging Predictors / L. Breiman // Machine Learning. – 1996. P. 123–140.

26. Чернявский И.И. Применение машинного обучения для создания управляющих автоматов на примере игры «Robocode» / И.И. Чернявский // Международный журнал исследований культуры. – Санкт-Петербург, 2018. – № 2(72). – С. 22-26.

27. Латыпова А.Р. Игровой искусственный интеллект как медиум социального мира / А.Р. Латыпова // Научно-технический вестник информационных технологий, механики и оптики. – Санкт-Петербург, 2018. – № 1(34). – С. 46-61.

28. Machine Learning for Game Developers / [Електронний ресурс]. – Режим доступу: https://www.youtube.com/watch?v=2h-Wg5FDbtU

29. 6 Ways Machine Learning will be used in Game Development / [Електронний ресурс]. – Режим доступу: https://www.logikk.com/articles/machine-learning-in-game-development/

30. Буковшин В.А. Интеллектуальные системы в компьютерных играх. Перспективы развития искусственного интеллекта в игровой индустрии / В.А. Буковшин, С.Г. Воскобойников // Современные материалы, техника и технологии. – Ростов-на-Дону, 2017. – № 3(11). – С. 21-36.

31. Как геймдев использует машинное обучение. И на что он может вдохновить другой бизнес / [Електронний ресурс]. – Режим доступу: https://vc.ru/playgendary/51241-kak-geymdev-ispolzuet-mashinnoe-obuchenie-i-na-chto-on-mozhet-vdohnovit-drugoy-biznes

32. Python about / [Електронний ресурс]. – Режим доступу: https://python.org

33. Хайкин С. Нейронные сети: Полный курс / пер. с англ. Н.Н. Куссуль, А.Ю. Шелестова. – 2-е изд., испр. – М.: Издательский дом Вильямс, 2008. – 1103 с.

34. Вандерплас Д. Python для сложных задач: наука о данных и машинное обучение. – СПб.: Питер, 2018. – 576 с.

35. Yoruk U. Automatic Renal Segmentation for MR Urography Using 3D-GrabCut and Random Forests / U. Yoruk, B.A. Hargreaves, S.S. Vasanawala // International Society for Magnetic Resonance in Medicine. – 2017. – Vol. 79, No. 3. – P. 1696–1707.

36. Jupyter project about / [Електронний ресурс]. – Режим доступу: https://jupyter.org/

37. Khalil M.M. Mastering Google cloud: building the platform that serves your needs / M.M. Khalil, S.E. Adadurov, M.S. Mahmood // Models and Methods for Researching Information Systems in Transport 2020 (MMRIST 2020). – Moskow, 2020. – № 7. – С. 55-66.

38. What is SQL? / [Електронний ресурс]. – Режим доступу: https://searchdatamanagement.techtarget.com/definition/SQL

39. BigQuery / [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/BigQuery

40. NumPy / [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/NumPy

41. What is Numpy? / [Електронний ресурс]. – Режим доступу: https://numpy.org/devdocs/user/whatisnumpy.html

42. pandas - Python Data Analysis Library / [Електронний ресурс]. – Режим доступу: https://pandas.pydata.org

43. Pandas / [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Pandas

44. pandas - Package overview / [Електронний ресурс]. – Режим доступу: https://pandas.pydata.org/pandas-docs/stable/getting_started/overview.html

45. Иванов А.А. Кластеризация данных на основе марковской цепи с помощью алгоритма DBSCAN / А.А. Иванов // Новые информационные технологии в автоматизированных системах – Белгород, 2018. – № 3. – С. 11-15.

46. Альбовский А.В. Реализация нейронной сети с помощью языка программирования Python / А.В. Альбовский, Н.А. Егоров, А.Г. Романюк // Colloquium-journal – Москва, 2020. – № 9(61). – С. 9-11.

47. Lambodar J. Distributed Data Mining Classification Algorithms for Prediction of Chronic Kidney Disease / J. Lambodar, K. Narendra // International Journal of Emerging Research in Management and Technology. – 2015. – Vol. 4, No. 11. – P. 110–180.

48. Gopika S. Machine learning Approach of Chronic Kidney Disease Prediction using Clustering Technique / S. Gopika, Dr.M. Vanitha // International Journal of Innovative Research in Science, Engineering and Technology. – 2017. – Vol. 6, No. 7. – P. 14488–14496.

49. Кобзарь А. И. Прикладная математическая статистика. – М.: Физматлит, 2006. – 626–628 с

50. Модели для предсказания класса объектов / [Електронний ресурс]. – Режим доступу: https://ranalytics.github.io/data-mining/023-Models-for-Class-Prediction.html

51. Шелухин О.И. Классификация зашифрованного трафика мобильных приложений методом машинного обучения / О.И. Шелухин, В.В. Барков, М.В. Полковников // Труды Международной конференции «АПВПМ» – Белгород, 2018. – № 4. – С. 21-28.

52. Лукинов В.Л. Численно устойчивый вероятностный классификатор логистической регрессии / В.Л. Лукинов // Вопросы кибербезопасности – Новосибирск, 2018. – № 4. – С. 42-48.

53. Клячкин В.Н. Выбор метода бинарной классификации при технической диагностике с применением машинного обучения / В.Н. Клячкин, Ю.Е. Кувайскова, Д.А. Жуков // Известия Самарского научного центра Российской академии наук. – Самара, 2018. – № 4(3). – С. 494-497.

54. Реализация регрессионных и классификационных задач с помощью метода Random Forest / [Електронний ресурс]. – Режим доступу: https://scikit-learn.org/stable/modules/ensemble.html#ensemble

55. Classes - skikit-learn / [Електронний ресурс]. – Режим доступу: https://scikit-learn.org/stable/modules/classes.html

56. Konstantinov A.V. Deep gradient boosting for regression problems / A.V. Konstantinov // Информатика, телекоммуникации и управление. – Санкт-Петербург, 2021. – Vol. 14, No. 3. – P. 7-18.

57. Моисеенко С.И. Упражнения по sql. Модификация данных / A.V. Konstantinov // Образовательные технологии и общество. – Дон, 2005. –№ 8(1). – С. 165-170.

ДОДАТКИ

Додаток А.
Запит до баз даних BigQuery

```
WITH session_info AS
(
SELECT user_pseudo_id,
    device.category as device_type,
    device.mobile_model_name as device_name,
    traffic_source.source  as traf_source,
  ROUND(((event_timestamp - user_first_touch_timestamp)/1000000)) as
first_session_duration_sec,
  (SELECT
    count(user_pseudo_id)
   FROM
    session_start
   WHERE user_pseudo_id = minm.user_pseudo_id
    AND event_timestamp BETWEEN user_first_touch_timestamp AND
(user_first_touch_timestamp + 86400 * 1000000) #промежуток между первым
ивентом и первым ивентом + 24 часа
    AND event_date BETWEEN --date1-- AND DATE_ADD(DATE --date2--,
INTERVAL 1 DAY)) as sessions_in_first_day
FROM
max_events as minm
WHERE event_name = 'max_event'
  AND event_date BETWEEN --date1-- AND --date2--
  AND platform = "IOS"
  AND geo.country = "United States"
  AND (SELECT value.int_value FROM UNNEST(user_properties) WHERE key
= 'ga_session_number') = 1
),
```

```
lock_clicks AS
(
SELECT user_pseudo_id,
   (
    SELECT
     COUNT(user_pseudo_id) FROM menu_show, UNNEST(event_params)
    WHERE
     user_pseudo_id = gp.user_pseudo_id
     AND event_date = gp.event_date
     AND key = 'source'
     AND value.string_value IN ('lock', 'gameMenu', 'instrumentLock')
   ) as lock_clicks
FROM
   menu_show as gp
WHERE
   event_date BETWEEN --date1-- AND --date2-- AND
   platform = 'IOS' AND
   geo.country = "United States" AND
   (SELECT value.int_value FROM UNNEST(user_properties) WHERE key =
'ga_session_number') = 1 AND
   user_pseudo_id IN (
                  SELECT
                     user_pseudo_id
                  FROM
                     first_open
                  WHERE
                     event_date BETWEEN --date1-- AND --date2--
                  )
GROUP BY 1, 2
```

```
),

ia_clicks AS
(
SELECT
   user_pseudo_id,
   COUNT(user_pseudo_id) as clicks_to_sub
FROM
   menu_clicks as gp
WHERE
   event_date BETWEEN --date1-- AND --date2--
   AND platform = "IOS"
   AND geo.country = "United States"
   AND
   (SELECT value.int_value FROM UNNEST(user_properties) WHERE key =
'ga_session_number') = 1 AND
   user_pseudo_id IN (
                  SELECT
                     user_pseudo_id
                  FROM
                     first_open
                  WHERE
                     event_date BETWEEN --date1-- AND --date2--
                  )
group by 1
ORDER BY 2 DESC
),

ia_menu_watch AS
(
```

```
SELECT
   DISTINCT user_pseudo_id,
   ROUND((((select min(event_timestamp) from menu_close
   where event_date = gp.event_date and user_pseudo_id = gp.user_pseudo_id)-
min(gp.event_timestamp))/1000000)) as sec_in_ia_menu
FROM
   menu_show as gp
WHERE
   event_date BETWEEN --date1-- AND --date2--
   AND platform = "IOS"
   AND geo.country = "United States"
   AND
   (SELECT value.int_value FROM UNNEST(user_properties) WHERE key =
'ga_session_number') = 1 AND
   user_pseudo_id IN (
                  SELECT
                     user_pseudo_id
                  FROM
                     first_open
                  WHERE
                     event_date BETWEEN --date1-- AND --date2--
                  )
group by 1, gp.event_date
HAVING sec_in_ia_menu IS NOT NULL
),

gender_age AS
(
SELECT
  DISTINCT user_pseudo_id,
```

```
  MAX((SELECT value.string_value FROM UNNEST(event_params) WHERE
key = 'answer' AND event_name = 'app_app_pu_age')) as age,
  MAX((SELECT value.string_value FROM UNNEST(event_params) WHERE
key = 'answer' AND event_name = 'app_app_pu_gender')) as gender
FROM
  info as surv
WHERE
  event_date BETWEEN --date1-- AND --date2--
  AND platform = 'IOS'
  AND geo.country = "United States"
  AND (SELECT value.int_value FROM UNNEST(event_params) WHERE key =
'ga_session_number') = 1
GROUP BY user_pseudo_id
),

user_info AS
(
SELECT
    user_pseudo_id,
    COUNT(user_pseudo_id) as all_painted,
    COUNT(DISTINCT value.string_value) as unique_painted
FROM
    game_finish as gp, UNNEST(event_params) as ep
WHERE
    event_date BETWEEN --date1-- AND --date2--
    AND geo.country = "United States"
    AND
    (SELECT value.int_value FROM UNNEST(user_properties) WHERE key =
'ga_session_number') = 1
    AND user_pseudo_id IN
```

```
                (
                SELECT
                   user_pseudo_id
                FROM
                   first_open
                WHERE
                   event_date BETWEEN --date1-- AND --date2--
                )
   AND key = 'name'
   AND platform = 'IOS'
GROUP BY
   user_pseudo_id,
   device.category,
   device.mobile_model_name,
   traffic_source.name
ORDER BY 2 DESC
),

first_character AS
(
SELECT
   DISTINCT user_pseudo_id,
   ROUND((((SELECT MIN(event_timestamp) FROM game_finish
   WHERE
      user_pseudo_id = gp.user_pseudo_id AND
      event_date = gp.event_date) - user_first_touch_timestamp)/1000000)) as
sec_to_first_finish
FROM
   game_finish as gp
WHERE
```

```
    event_date BETWEEN --date1-- AND --date2--
    AND platform = "IOS"
    AND geo.country = "United States"
    AND
    (SELECT value.int_value FROM UNNEST(user_properties) WHERE key =
'ga_session_number') = 1 AND
    user_pseudo_id IN (
                    SELECT
                      user_pseudo_id
                    FROM
                      first_open
                    WHERE
                      event_date BETWEEN --date1-- AND --date2--
                    )
ORDER BY sec_to_first_finish DESC
)

SELECT
  si.user_pseudo_id,
  si.device_type,
  si.device_name,
  si.traf_source,
  si.first_session_duration_sec,
  si.sessions_in_first_day,
  CASE ga.age
    WHEN "0-2" THEN 1
    WHEN "3-4" THEN 2
    WHEN "5-6" THEN 3
    WHEN "7-8" THEN 4
    WHEN "9-11" THEN 5
```

```
  WHEN "12+" THEN 6
END AS age,
CASE ga.gender
  WHEN "buy" THEN 1
  WHEN "girl" THEN 2
END AS gender,
CASE
  WHEN ui.all_painted IS null THEN 0
  ELSE ui.all_painted
END AS all_painted,
CASE
  WHEN ui.unique_painted IS null THEN 0
  ELSE ui.unique_painted
END AS unique_painted,
CASE
  WHEN fc.sec_to_first_finish IS null THEN 0
  ELSE fc.sec_to_first_finish
END AS sec_to_first_finish,
CASE
  WHEN mw.sec_in_ia_menu IS null THEN 0
  ELSE mw.sec_in_ia_menu
END AS sec_in_ia_menu,
CASE
  WHEN cl.clicks_to_sub IS null THEN 0
  ELSE cl.clicks_to_sub
END AS clicks_to_sub,
CASE
  WHEN lc.lock_clicks IS null THEN 0
  ELSE lc.lock_clicks
END AS lock_clicks,
```

```
 CASE
  WHEN si.user_pseudo_id IN (
                    SELECT user_pseudo_id
                    FROM purchase
                    WHERE platform = 'IOS'
                    AND activation = true
                    )
  THEN 1
  ELSE 0
 END AS paid
FROM session_info as si
FULL JOIN user_info as ui
ON ui.user_pseudo_id = si.user_pseudo_id
FULL JOIN first_character as fc
ON fc.user_pseudo_id = si.user_pseudo_id
FULL JOIN gender_age as ga
ON ga.user_pseudo_id = si.user_pseudo_id
FULL JOIN ia_menu_watch as mw
ON mw.user_pseudo_id = si.user_pseudo_id
FULL JOIN ia_clicks as cl
ON cl.user_pseudo_id = si.user_pseudo_id
FULL JOIN lock_clicks as lc
ON lc.user_pseudo_id = si.user_pseudo_id
```

Додаток Б.

Повна версія скрипту визначення моделі бінарної класифікації

```
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV

raw_data = pd.read_csv('/content/data-set.csv', sep=',',header=0)

raw_data.head()

raw_data.describe()

raw_data['age'] = raw_data['age'].fillna(0)
raw_data['gender'] = raw_data['gender'].fillna(0)
list_to_filter = ['sessions_in_first_day', 'sec_to_first_finish', 'sec_in_ia_menu',
'clicks_to_sub', 'lock_clicks']
filtered = raw_data.copy()

low = raw_data['first_session_duration_sec'].quantile(0.02)
```

```
high  = raw_data['first_session_duration_sec'].quantile(0.98)
filtered = filtered[(filtered['first_session_duration_sec'] < high) &
(filtered['first_session_duration_sec'] > low)]

for col in list_to_filter:
  high  = raw_data[col].quantile(0.98)
  filtered = filtered[(filtered[col] < high)]

filtered = filtered.drop_duplicates(subset=['user_pseudo_id'])

filtered.describe()

min_max_scaler = preprocessing.MinMaxScaler()
normalize_list = ['first_session_duration_sec', 'sessions_in_first_day', 'all_painted',
'unique_painted', 'sec_to_first_finish', 'sec_in_ia_menu', 'clicks_to_sub',
'lock_clicks']

for col in normalize_list:
  filtered[col] = min_max_scaler.fit_transform(np.array(filtered[col]).reshape(-
1,1))

filtered.head()

filtered['paid'].value_counts()

y = filtered['paid']
X = filtered.iloc[:,1:13]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
models = {}

models['Logistic Regression'] = LogisticRegression(max_iter=5000)
models['Support Vector Machines'] = LinearSVC(max_iter=5000)
models['Decision Trees'] = DecisionTreeClassifier()
models['Random Forest'] = RandomForestClassifier()
models['Naive Bayes'] = GaussianNB()
models['K-Nearest Neighbor'] = KNeighborsClassifier()
models['Gradient Boosting'] = GradientBoostingClassifier()

true_positive, false_positive, true_negative, false_negative, accuracy = {}, {}, {}, {}, {}

for key in models.keys():

    models[key].fit(X_train, y_train)

    predictions = models[key].predict(X_test)

    true_negative[key], false_positive[key], false_negative[key], true_positive[key] = confusion_matrix(y_test, predictions).ravel()
    accuracy[key] = (true_positive[key]+true_negative[key]) / (true_positive[key]+false_positive[key]+true_negative[key]+false_negative[key])

df_model = pd.DataFrame(index=models.keys(), columns=['True Positive', 'False Positive',
                                                      'True Negative', 'False Negative', 'Accuracy'])
df_model['True Positive'] = true_positive.values()
df_model['False Positive'] = false_positive.values()
df_model['True Negative'] = true_negative.values()
```

```
df_model['False Negative'] = false_negative.values()
df_model['Accuracy'] = accuracy.values()

df_model

mod = RandomForestClassifier(criterion='entropy', n_estimators=100,
max_depth=12,
                             bootstrap=False, class_weight='balanced')
mod.fit(X_train, y_train)
pr1 = mod.predict(X_test)
cm = confusion_matrix(y_test, pr1)

TN, FP, FN, TP = confusion_matrix(y_test, pr1).ravel()

print('True Positive(TP)  = ', TP)
print('False Positive(FP) = ', FP)
print('True Negative(TN)  = ', TN)
print('False Negative(FN) = ', FN)

accuracy1 =  (TP+TN) /(TP+FP+TN+FN)

print('Accuracy of the binary classification = {:0.3f}'.format(accuracy1))
```