ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ТЕЛЕКОМУНІКАЦІЙ
КАФЕДРА СИСТЕМНОГО АНАЛІЗУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «РОЗРОБКА ДОДАТКУ ДЛЯ АНАЛІТИКИ КОРИСТУВАЦЬКИХ ДАНИХ ТА ПОБУДОВИ СКОРИНГОВИХ МОДЕЛЕЙ»

на здобуття освітнього ступеня магістра

зі спеціальності 124 Системний аналіз

освітньо-професійної програми Інтелектуальні системи управління

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ МАКСИМ НЕМЧЕНКО
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач(ка) вищої освіт
групи САДМ-61
Максим НЕМЧЕНКО

Керівник: Ровіл НАФЄЄВ

Рецензент: _____

Київ 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ТЕЛЕКОМУНІКАЦІЙ

Кафедра Системного аналізу

Ступінь вищої освіти Магістр

Спеціальність 124 Системний аналіз

Освітньо-професійна програма Інтелектуальні системи управління

ЗАТВЕРДЖУЮ
Завідувач кафедрою СА
Ровіл НАФЄЄВ
«____» _____20___р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Немченку Максиму Ігоровчу

1. Тема кваліфікаційної роботи: Розробка додатку для аналітики користувацьких даних та побудови скорингових моделей.
керівник кваліфікаційної роботи Нафєєв Р.К. ,
затверджені наказом Державного університету інформаційно-комунікаційних технологій
від «___» _____20___р. № ____

2. Строк подання кваліфікаційної роботи «___» _____20___р.

3. Вихідні дані до кваліфікаційної роботи.
Об'єкт досліджень – побудова скорингових моделей з розрахуванням математичних показників churned, f-score, precision і recall
Предмет досліджень – методи машинного навчання.
Мета НДР – створити додаток з опрацювання користувацьких даних.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   1. Проєктування інформаційної системи.
   2. Аналіз методів машинного навчання.
   3. Створення додатку для аналітики.

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання «___» _____20___р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз теми та постановка задачі	12.09.2023-30.09.2023	виконано
2	Побудова моделі та алгоритмів навчання на даних датасету	01.10.2023-31.10.2023	виконано
3	Створення функціональності машинного навчання	01.11.2023-16.12.2023	виконано
4	Попередній захист роботи	17.01.2023	виконано

Здобувач(ка) вищої освіти _____ Максим НЕМЧЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Ровіл НАФЄЄВ

РЕФЕРАТ

Пояснювальна записка: 96 с., 20 рис., 3 дод., 54 джерела.

Об'єкт розробки: додаток для аналітики користувацьких даних.

Мета кваліфікаційної роботи: створення додатку що допоможе розвинути навички машинного навчання, та дати розуміння як опрацьовувати користувацькі дані.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій, а також програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформи для розробки, виконано проектування і розробка програми, описана робота програми, алгоритм і структура її функціювання, а також виклик та завантаження програми, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

У третьому розділі наведені приклади роботи з інтерфейсом програми, проаналізовані інші методи машинного навчання не задіяні у кваліфікаційній роботі.

Практичне значення полягає у створенні програмного додатка, що надає можливість покращити знання, та збільшити знання.

Актуальність даного програмного продукту визначається великим попитом на подібні розробки, що допомагають оптимізувати процеси обробки великих об’ємів даних.

Список ключових слів: КОМП’ЮТЕР, ІНФОРМАЦІЙНА СИСТЕМА, АЛГОРИТМ, ПРОЕКТУВАННЯ БАЗИ ДАНИХ, ІНТЕРФЕЙС, ДОДАТОК, МАШИННЕ НАВЧАННЯ.

ABSTRACT

Explanatory note: 96 p., 20 figures, 3 appendices, 54 sources.

Object of development: application for analytics of user data.

The goal of the qualification work: creating an application that will help develop machine learning skills and provide an understanding of how to process user data.

In the introduction, the analysis and current state of the problem is considered, the purpose of the qualification work and the field of its application are specified, the justification of the relevance of the topic is given, and the statement of the task is clarified.

In the first section, the subject area is analyzed, the relevance of the task and the purpose of the development is determined, the task statement is formulated, and the requirements for software implementation, technologies and software tools are specified.

In the second section, available solutions are analyzed, platforms for development are selected, program design and development is performed, program operation, algorithm and structure of its operation are described, as well as program calling and loading, input and output data are determined, and the composition of technical means parameters is characterized.

In the third section, examples of working with the program interface are given, other methods of machine learning are analyzed and are not involved in the qualification work.

The practical meaning is to create a software application that provides an opportunity to improve knowledge and increase knowledge.

The relevance of this software product is determined by the high demand for similar developments that help optimize the processes of processing large volumes of data.

List of keywords: COMPUTER, INFORMATION SYSTEM, ALGORITHM, DATABASE DESIGN, INTERFACE, APPLICATION, MACHINE LEARNING.

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ТЕЛЕКОМУНІКАЦІЙ

ПОДАННЯ
ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ
на здобуття освітнього ступеня магістра

Направляється здобувач Немченко М.І. до захисту кваліфікаційної роботи
(прізвище та ініціали)
за спеціальністю 124 Системний аналіз
освітньо-професійної програми Інтелектуальні системи правління
на тему: «Розробка додатку для аналітики користувацьких даних та побудови скорингових моделей».

Кваліфікаційна робота і рецензія додаються.

Директор ННІТ _____ Владислав КРАВЧЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Висновок керівника кваліфікаційної роботи

Здобувач Немченко Максим Ігорович показав свою інженерну підготовку разом із володінням теоретичного матеріалу із поглибленим розумінням методів машинного навчання, створення бази даних, аналізу та обробки даних, проєктуванню додатку, адже якісно вміє володіти новими ком'ютерними технологіями та вміє детально користуватися навчальною, довідниковою та науково-технічною літературою. Всі поставлені задачі винонувались сумлінно та якісно згідно технічного завдання.

Все це дозволяє оцінити виконану кваліфікаційну роботу здобувача Немченко Максима Ігоровича на оцінку «відмінно» та присвоїти йому(їй) кваліфікацію «Магістр з системного аналізу».

Керівник кваліфікаційної роботи _____ Ровіл НАФЄЄВ
(підпис) *(Ім'я,ПРІЗВИЩЕ)*

«___» _____20___року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута. Здобувач(ка) *Немченко М.І.* допускається до захисту даної роботи в Екзаменаційній комісії.

Завідувач кафедрою Системного аналізу _____ Ровіл НАФЄЄВ
(назва) *(підпис)* *(Ім'я, ПРІЗВИЩЕ)*

«___» _____20___року

ВІДГУК РЕЦЕНЗЕНТА
НА КВАЛІФІКАЦІЙНУ МАГІСТЕРВСЬКУ РОБОТУ

здобувача(ки) вищої освіти Немченко Максима Ігоровича
на тему «Розробка додатку для аналітики користувацьких даних та побудови скорингових моделей»

В першому розділі загальний опис наукової галузі, в якій ведеться дослідження та наведені теоретичні дані. Другий розділ містить короткий огляд проблеми та потенційні шляхи її вирішення. Третій розділ містить в собі огляд інтерфейсу користувача, а також результати виконання розрахунків.

В даній кваліфікаційній роботі студентом надано декілька пропозицій щодо вирішення поставлених задач. Кожна з пропозицій була обґрунтована та підкріплена науковими даними.

Результати роботи можуть бути застосовані для подальших наукових досліджень в даній сфері, а також вони можуть бути корисними для практичного використання.

Список літератури, наведений в роботі, налічує більше 20 джерел, що свідчить про вміння автора працювати з літературою та іншими джерелами інформації. Якість оформлення кваліфікаційної роботи можна визнати задовільною, він супроводжується достатньою кількістю малюнків . В роботі присутні заключні висновки.

Магістерська кваліфікаційна робота виконана у відповідності з завданням із дотриманням всіх вимог.

Робота заслуговує оцінки «відмінно» , а студент Немченко Максим Ігорович – присвоєння кваліфікації «магістра з системного аналізу» .

Рецензент:
науковий ступінь, вчене звання _____ *підпис* _____ *Ім'я, ПРІЗВИЩЕ*

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

СКБД - система керування базами даних;

БД - база даних;

ПЗ - програмне забезпечення

IDE - Integrated development environment;

ОС - операційна система;

TP - true positive

TN - true negative.

GCP – Google Cloud Platform

GBM - Gradient Boosting algorithms

KNN - Nearest Neighbors

SVM - Support Vector Machine

ID – Identification

GKE - Google Kubernetes Engine

ML – Machine Learning

ШІ – Штучний інтелект

SQL - Structured Query Language

ВСТУП

Тематика завдань та видів діяльності даної кваліфікаційної роботи безпосередньо пов'язана та відповідає загальній тематиці кваліфікаційному рівню та переліку типових завдань, умінь, знань та компетенцій конкретних виробничих функцій та видів діяльності, якими повинен володіти магістр зі спеціальності 124 "Системний аналіз".

На сьогоднішній день інтерес викликають різноманітні онлайн-сервіси, інтернет-магазини, онлайн-школи, курси дистанційного навчання та курси підвищення кваліфікації. Сучасні підприємства стикаються з великою потребою у зборі, зберіганні та обробці даних користувачів. Зберігання даних у базах даних недостатньо, сучасний світ вимагає вміння правильно їх використовувати. Дані використовуються для створення стратегій та планів розвитку бізнесу, аналізу роботи персоналу, створення портретів користувачів та їхніх звичок. Враховуючи вищезазначені фактори, кваліфікаційне завдання полягає в розробці додатку для аналізу даних користувачів в СУБД та створенні скорингової моделі, яка дозволить оцінити вірогідність конверсії кожного окремого користувача.

Метою кваліфікаційної роботи є вивчення методів створення баз даних та роботи з ними. Такі навички принесуть велику користь у сфері розробки програмного забезпечення, наприклад, для управління навчанням студентів, моніторингу ефективності та якості викладання, побудови скорингових моделей для прогнозування ефективності продажів та рекламних кампаній. Вона також забезпечить краще розуміння того, як працюють з об'єктами баз даних, як з ними взаємодіяти і як ефективно та оптимально використовувати машинне навчання.

У цьому документі розповідається про те, як обробляються, зберігаються, обробляються дані користувачів та інші взаємодії. Наведено приклади різних методів і підходів до машинного навчання, а також обґрунтування вибору СУБД та алгоритмів побудови скорингових моделей і графіків.

Основними вимогами до системи є простий спосіб обробки даних, надійна та стабільна робота системи, точність розрахунків.

Новизна даної розробки полягає в математичному підході до підвищення ефективності роботи менеджера з продажу. Користувачі поділяються на сегментні групи на основі даних про користувача, його вподобання та поведінкові особливості. На основі цього можна використовувати різні стратегії продажів та інші методи взаємодії бізнесу з клієнтом.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ПРОБЛЕМИ

1.1. Загальна інформація про предметну область

Система керування базами даних (СКБД) - це комп'ютерна програма, яка дозволяє користувачам маніпулювати базою даних; СКБД дозволяє користувачам контролювати доступ до бази даних, зберігати дані, виконувати запити та робити інші завдання, пов'язані з керуванням базами даних. Наприклад, оновлювати дані, видаляти їх чи зберігати.

Однак, щоб мати можливість виконувати ці завдання, СКБД повинна базуватися на моделі, яка описує організацію даних. Реляційна модель - це підхід до організації даних, який широко використовується в програмному забезпеченні баз даних з моменту їх появи в кінці 1960-х років. Цей підхід настільки поширений, що на момент написання цієї роботи чотири з п'яти найпопулярніших систем управління базами даних є реляційними [1].

База даних - це логічно організований набір інформації або даних. Кожен набір даних є базою даних, незалежно від того, де і як він зберігається. Шафа з платіжними відомостями, полиця в приймальні з історіями хвороб, файли клієнтів компанії, що зберігаються в окремих офісах - все це бази даних. До того, як комп'ютеризоване зберігання та управління даними набуло широкого розповсюдження, державні установи та комерційні компанії могли використовувати лише такі фізичні бази даних для зберігання інформації.

У середині XX століття розвиток комп'ютерних наук призвів до появи машин з вищою обчислювальною потужністю та більшим об'ємом внутрішньої і зовнішньої пам'яті [2]. Ці розробки змусили комп'ютерних вчених усвідомити потенціал таких пристроїв, як ЕОМ, для зберігання та управління великими масивами даних.

Однак не існувало теорії про те, як комп'ютери можуть організовувати дані в осмислений і логічний спосіб. Одна справа - зберігати на комп'ютері

несортовані дані, але набагато складніше створити систему, яка може послідовно додавати, видаляти, сортувати та іншим чином керувати цими даними. Потреба в логічній структурі для зберігання та організації даних призвела до того, що було запропоновано низку методів, які дозволяють полегшити процес використання комп'ютерів для управління даними.

Однією з перших моделей баз даних була ієрархічна модель, яка організовувала дані у вигляді деревоподібної структури, подібної до сучасних файлових систем [3].

Ієрархічна модель широко використовувалася в ранніх системах управління базами даних, але з часом їй не вистачало гнучкості. У цій моделі кожен запис може мати лише одного "предка", хоча кожен запис може мати декілька "нащадків" [4]. Тому ранні ієрархічні бази даних могли представляти лише зв'язки "один-до-одного" або "один-до-багатьох". Відсутність зв'язків "багато-до-багатьох" могла спричинити проблеми при роботі з точками даних, які потрібно було пов'язати з кількома предками.

Наприкінці 60-х років XX століття Едгар Ф. Кодд, програміст з IBM, розробив реляційну модель для управління базами даних. Реляційна модель Кодда дозволила пов'язувати окремі записи з декількома таблицями і встановлювати між точками даних зв'язки не лише "один до багатьох", а й "багато до багатьох". Це забезпечило більшу гнучкість у проектуванні структур баз даних, ніж інші існуючі моделі, а також дозволило реляційним системам управління базами даних (СКБД) задовольняти більш широкий спектр бізнес-потреб.

Кодд запропонував мову управління реляційними даними, відому як Alpha, яка вплинула на розвиток пізніших мов баз даних. Колеги Кодда з IBM, Дональд Чемберлен та Реймонд Бойс, створили мову під впливом Alpha. Вони назвали свою мову SEQUEL (скорочення від Structured English Query Language), але скоротили її до SQL (більш офіційна назва Structured Query Language) через існуючі торгові марки.

Через обмежені апаратні можливості перші реляційні бази даних все ще

працювали дуже повільно, і знадобився певний час, щоб ця технологія набула широкого розповсюдження. Однак до середини 1980-х років реляційна модель Кодда була реалізована в багатьох комерційних продуктах управління базами даних від IBM та її конкурентів. Слідом за IBM, ці постачальники почали розробляти та впроваджувати власні діалекти SQL. 1987 року Американський національний інститут стандартів та Міжнародна організація зі стандартизації затвердили та опублікували стандарт SQL, який став визнаною мовою для управління СУБД.

Реляційна модель стала широко використовуватися в багатьох галузях і зараз визнана стандартною моделлю для управління даними. Незважаючи на нещодавню появу різноманітних баз даних NoSQL, реляційні бази даних залишаються домінуючим інструментом для зберігання та організації даних.

Структура даних реляційних баз даних

Тепер, коли є загальне уявлення про історію реляційної моделі, постала проблема дослідження як ця модель структурує дані.

Найважливішим елементом реляційної моделі є відношення, відоме користувачам і в сучасних СКБД як таблиця. Відношення - це набір кортежів (рядків у таблиці), а кожен кортеж має набір атрибутів (стовпців).

Стовпці є найменшою організаційною структурою в реляційній базі даних і представляють собою різні комірки, які визначають записи в таблиці. Звідси походить більш формальна назва "атрибут". Кожен кортеж можна розглядати як унікальний екземпляр чогось, що може існувати в таблиці: категорія особи, об'єкта, події або відношення. Такими екземплярами можуть бути працівники компанії, показники продажів онлайн-бізнесу, результати аудиту, кількість реєстрацій, показники трафіку тощо. Наприклад, у таблиці, що містить записи про роботу шкільних вчителів, кортеж може мати такі атрибути, як ім'я, предмет, дата початку роботи, вік, стать, освітня ступінь тощо.

Коли створюється стовпець, вказується тип даних, який визначає, які записи можуть бути заповнені в цьому стовпці; СУБД часто використовують власні типи даних, які можуть бути не сумісні безпосередньо з аналогічними

типами даних в інших системах. До поширених типів даних належать дати, рядки, цілі числа та логічні значення.

У реляційній моделі кожна таблиця містить принаймні один стовпець, який однозначно ідентифікує кожен рядок. Це називається первинним ключем; СУБД відстежує кожен запис і може повернути його з певною метою. Це означає, що для записів не існує певного логічного порядку, і користувач може повертати дані в довільному порядку або з використанням будь-якого фільтра.

Переваги та обмеження реляційних баз даних

Враховуючи організаційну структуру, що лежить в основі реляційних баз даних, розглянемо їхні переваги та недоліки.

Сьогодні і мова SQL, і бази даних, які її використовують, дещо відрізняються від реляційної моделі Кодда. Наприклад, модель Кодда передбачає, що кожен рядок у таблиці повинен бути унікальним, але з практичних міркувань більшість сучасних реляційних баз даних допускають дублювання рядків. Інші вважають, що база даних на основі SQL, яка не відповідає всім критеріям реляційної моделі Кодда, не є справжньою реляційною базою даних. На практиці, будь-яка СУБД, яка використовує за основу програмну мову SQL певною мірою відповідає реляційній моделі, тому вона може бути класифікована як система управління реляційними базами даних.

Популярність реляційних баз даних швидко зростала, але зі збільшенням цінності та обсягу даних, що зберігаються, почали проявлятися деякі недоліки реляційної моделі. Наприклад, реляційні бази даних важко розширювати в горизонтальному напрямку. Горизонтальне масштабування передбачає додавання машин до існуючого стеку для розподілу навантаження, збільшення трафіку і прискорення процесу обробки. Це протилежність вертикальному масштабуванню, яке зазвичай використовує оновлення апаратного забезпечення існуючого сервера шляхом додавання оперативної пам'яті,центрального процесора або інших комплектуючих.

Горизонтальне масштабування є складним, оскільки реляційні бази даних розроблені таким чином, щоб бути узгодженими між собою. Горизонтальне

масштабування реляційної бази даних на всіх серверах ускладнює забезпечення узгодженості, оскільки користувачі можуть надавати дані лише одному вузлу, а не всім вузлам. Існує ймовірність затримки між початковим записом і оновленням даних на інших вузлах для відображення змін, що призводить до неузгодженості даних на різних вузлах одночасно.

Ще одне обмеження, яке існує в СКБД, полягає в тому, що реляційна модель була розроблена для управління структурованими даними, тобто даними, які відповідають заздалегідь визначеному типу даних, або, принаймні, даними, які попередньо організовані за певною структурою. Однак з поширенням персональних комп'ютерів і розвитком Інтернету на початку 90-х років з'явилися неструктуровані дані, такі як електронні листи, фотографії та відео.

Однак це не означає, що реляційні бази даних перестали бути корисними. Навпаки, понад 40 років потому реляційна модель все ще залишається домінуючою основою для управління даними. Повсюдність і довговічність реляційних баз даних свідчить про те, що це зріла технологія, що саме по собі є великою перевагою. Існує багато додатків, призначених для роботи з реляційною моделлю, і багато адміністраторів баз даних, які є експертами в реляційних базах даних. Існує також багато друкованих та онлайн ресурсів для тих, хто хоче почати працювати з реляційними базами даних.

Ще однією перевагою реляційних баз даних є те, що майже всі СКБД підтримують транзакції. Транзакція складається з одного або декількох операторів SQL, які виконуються послідовно як єдиний робочий блок. Транзакції - це підхід "все або нічого", і всі оператори SQL в транзакції повинні бути дійсними. В іншому випадку вся транзакція завершиться невдачею. Це дуже корисно для забезпечення цілісності даних при внесенні змін до декількох рядків або таблиць.

Нарешті, реляційні бази даних дуже гнучкі. Вони використовуються для створення найрізноманітніших додатків і продовжують ефективно працювати навіть з великими обсягами даних [5]. Мова SQL також має великий потенціал, дозволяючи додавати або змінювати дані в режимі онлайн, змінювати схему бази

даних або структуру таблиць, паралельно не впливаючи на існуючі дані.

Завдяки своїй гнучкості та цілісності даних, реляційні бази даних все ще залишаються основним способом управління та зберігання даних вже довгі 50 років після того, як вони були вперше запропоновані. Розуміння того, як реляційна модель працює з СКБД, має вирішальне значення для всіх, хто хоче створювати додатки, що використовують дані, хоча різні NoSQL бази даних стали популярними в останні часи, [6].

Інтерфейси користувача баз даних

У сучасних системах програмування більшість програм постачається з інтерфейсом користувача, який є невід'ємною частиною СКБД.

Інтерфейс користувача - це набір програм, які регулюють взаємодію між користувачем і базою даних і надають користувачеві доступ до інформації, що зберігається в базі даних (невидимо для користувача).

Наприклад, користувач може отримати інформацію зі списку, що відображається на екрані, здійснити швидкий пошук необхідних даних або відредагувати вміст таблиці. Інтерфейс користувача є невід'ємною частиною програми, а його дизайн і спосіб взаємодії з користувачем впливають на ефективність використання програми. Зручний, красивий та ефективний інтерфейс впливає на популярність того чи іншого програмного забезпечення.

Модель оцінки СУБД. Назва скорингової системи походить від англійського слова "score", що перекладається як оцінка або бал. У моїй контрольній роботі скоринг використовується для оцінки платоспроможності користувача. Коли клієнт реєструється на навчання, він повинен заповнити обов'язкову анкету на сайті. Певні бали присвоюються його професійним, демографічним та соціальним характеристикам. Програма автоматично "оцінює" анкету і робить висновки про потенціал користувача. На основі отриманих балів менеджер команди вирішує, кому з представленого списку кандидатів зателефонувати в першу чергу.

Система оцінювання базується на припущенні, що люди зі схожими соціальними характеристиками поводяться однаково. Виходячи з цього

припущення, можна побудувати різні статистичні моделі, які є дуже корисними для ведення будь-якого бізнесу.

Якщо певним соціальним характеристикам клієнтів (стать, вік, місце проживання, посада, тривалість перебування на одному місці тощо) присвоїти певні вагові коефіцієнти, то кожного нового клієнта можна класифікувати як сильного чи слабкого партнера для бізнесу на основі його опитування. Іншими словами, клієнтам автоматично присвоюється певний бал, який вказує на ступінь довіри та важливості, яку даний бізнес повинен надавати клієнту.

Етапи скорингу можна розділити наступним чином:

- ідентифікація специфічної особливості, що цікавить; - Збір вторинної інформації про цінність специфічної особливості для клієнта
- збір вторинної інформації про клієнтів та цінність специфічних особливостей;
- розробка скорингової моделі на основі наявних даних (зважування вторинних даних);
- автоматичне ранжування нових клієнтів за пріоритетними групами з використанням скорингової моделі.

Якщо здатність клієнта здійснити певну покупку є характеристикою, що цікавить, можна виділити дві групи клієнтів: тих, кому потрібно зателефонувати першими, і тих, хто "чекає".

Машинне навчання (ML) - це галузь досліджень, присвячена розумінню та розробці способів "навчання", тобто використання даних для підвищення ефективності виконання певного набору завдань. Вважається частиною штучного інтелекту. Алгоритми машинного навчання будують моделі на основі вибіркових даних, відомих як навчальні дані, щоб робити прогнози і приймати рішення без явного програмування. Алгоритми машинного навчання використовуються в широкому спектрі застосувань, де важко або неможливо розробити традиційні алгоритми для виконання необхідних завдань, таких як

медицина, фільтрація електронної пошти, розпізнавання мови, сільське господарство та комп'ютерний зір.

Підрозділ машинного навчання тісно пов'язаний з обчислювальною статистикою, яка фокусується на використанні комп'ютерів для прогнозування, але не все машинне навчання є статистичним навчанням. Вивчення математичної оптимізації забезпечує методи, теорію і застосування в галузі машинного навчання. Інтелектуальний аналіз даних - це суміжна галузь досліджень, яка фокусується на дослідницькому аналізі даних за допомогою неконтрольованого навчання. Деякі програми машинного навчання використовують дані та нейронні мережі у спосіб, що імітує роботу біологічного мозку. У застосуванні до бізнес-проблем машинне навчання також називають предиктивною аналітикою.

Алгоритми навчання працюють на припущенні, що стратегії, алгоритми і міркування, які добре працювали в минулому, будуть добре працювати і в майбутньому. Ці результати можуть бути настільки ж очевидними, як "сонце сходить щоранку протягом останніх 10 000 днів, тому воно зійде знову завтра вранці". Нюанс, наприклад, "X% сімейства географічно різноманітні за видами і забарвленням, тому ймовірність присутності невідомого чорного лебедя становить Y%".

Програми машинного навчання можуть виконувати завдання, не будучи явно запрограмованими. Це означає, що комп'ютер навчається на основі наданих даних для виконання певного завдання. Для простих завдань, які ставляться перед комп'ютером, можна запрограмувати алгоритм, який вказує машині, як виконати всі кроки, необхідні для вирішення поставленої задачі. Для більш складних завдань людині може бути важко вручну створити необхідні алгоритми. На практиці для людини-програміста може бути ефективніше допомогти машині розробити власний алгоритм, а не вказувати на кожен необхідний крок.

У сфері машинного навчання використовуються різні підходи, щоб навчити комп'ютери виконувати завдання, для яких не існує абсолютно задовільного алгоритму. У випадках, коли є багато потенційних відповідей, один

з підходів полягає в тому, щоб позначити деякі з правильних відповідей як дійсні. Це може бути використано як навчальні дані для розробки алгоритму, що використовується комп'ютером для визначення правильної відповіді. Наприклад, набір рукописних чисел MNIST часто використовується для навчання систем розпізнавання чисел.

Термін "машинне навчання" був введений в 1959 році Артуром Семюелом, співробітником IBM і піонером в області комп'ютерних ігор і штучного інтелекту. У цей період також використовувався синонім "самонавчальний комп'ютер".

На початку 1960-х років компанія Raytheon розробила експериментальну "машину, що навчається" під назвою Cybertron з перфострічкою, яка аналізувала сонарні сигнали, ЕКГ і мовні патерни, використовуючи базове навчання з підкріпленням. Люди-оператори/викладачі неодноразово "тренувалися" розпізнавати шаблони і були оснащені "фіктивною" кнопкою, яка дозволяла їм переоцінювати помилкові рішення. Провідною книгою з досліджень машинного навчання в 1960-х роках була книга Нільссона "Машина, що навчається" (The Learning Machine), в якій основна увага приділялася машинному навчанню для класифікації образів. Інтерес до розпізнавання образів не згас і в 1970-х роках, коли Дуда і Харт представили в 1973 році книгу "Стратегії навчання", а в 1981 році - "Навчальні стратегії", яка дозволила нейронним мережам навчитися розпізнавати 40 символів (26 літер, 10 цифр і 4 спеціальних символи) з комп'ютерних терміналів. Було представлено доповідь про використання системи.

Том М. Мітчелл дав широко цитоване і більш формальне визначення алгоритмів, що вивчаються в галузі машинного навчання: "Кажуть, що комп'ютерна програма навчається на досвіді E завдання з класом T і мірою ефективності P". Це визначення задачі, якою займається машинне навчання, є, по суті, операційним визначенням, а не визначенням галузі в когнітивних термінах. Воно відповідає пропозиції Алана Тюрінга в його роботі "Комп'ютери та інтелект" про те, що питання "Чи можуть машини мислити?" слід замінити на

"Чи можуть машини (як мислячі істоти) робити те, що можемо робити ми?".

Сучасне машинне навчання має дві мети: Перше - класифікувати дані на основі розроблених моделей, а друге - передбачати майбутні результати на основі цих моделей. Віртуальні алгоритми, розроблені для класифікації даних, можна навчити класифікувати ракові родимки, поєднуючи комп'ютерну візуалізацію родимок з керованим навчанням. Алгоритми машинного навчання для біржової торгівлі можуть інформувати інвесторів про можливі майбутні прогнози.

Машинне навчання як наукова дисципліна розвинулося з пошуків штучного інтелекту. Коли штучний інтелект вперше з'явився як дисципліна, деякі дослідники були зацікавлені в тому, щоб дати машинам можливість навчатися на основі даних. Вони намагалися вирішити цю проблему за допомогою різних символічних методів, а також того, що тоді називали "нейронними мережами". Здебільшого це були персептрони та інші моделі, які пізніше вважалися переосмисленням узагальненої лінійної статистичної моделі. Імовірнісні міркування також використовувалися, особливо в автоматизованій медичній діагностиці.

Однак зростаючий акцент на логічних підходах, заснованих на знаннях, створив розрив між ШІ та машинним навчанням (рис. 1.1). Імовірнісні системи стикалися з теоретичними і практичними проблемами збору і представлення даних. До 1980 року експертні системи почали домінувати в ШІ, а статистика перестала користуватися популярністю. В рамках ШІ продовжувалися дослідження символічного і заснованого на знаннях навчання, що призвело до індуктивного логічного програмування, але подальші статистичні дослідження відійшли від власне сфери ШІ - розпізнавання образів і пошуку інформації.

У 1990-х роках машинне навчання (МН) почало перегруповуватися і розвиватися як незалежна галузь. Мета галузі змістилася від отримання штучного інтелекту до вирішення вирішуваних завдань практичного характеру, змістивши фокус з символічного підходу, успадкованого від ШІ, на методи і моделі, запозичені зі статистики, нечіткої логіки і теорії ймовірностей.

Різницю між машинним навчанням і ШІ часто не розуміють: Машинне навчання навчається і робить прогнози на основі пасивних спостережень, тоді як ШІ передбачає взаємодію агента з навколишнім середовищем, навчання і вжиття заходів для максимізації ймовірності досягнення своїх цілей.

Станом на 2022 рік багато джерел продовжують стверджувати, що машинне навчання є піддисципліною ШІ. Інші стверджують, що не все машинне навчання є частиною ШІ і що лише "інтелектуальну підмножину" машинного навчання слід вважати ШІ.

Машинне навчання та інтелектуальний аналіз даних часто використовують однакові методи, і вони багато в чому перетинаються, але машинне навчання фокусується на прогнозах, заснованих на відомих особливостях навчальних даних, тоді як інтелектуальний аналіз даних фокусується на виявленні (раніше) невідомих особливостей у даних (це фаза виявлення знань в процесі інтелектуального аналізу баз даних, майнінгу). З іншого боку, машинне навчання також використовує методи інтелектуального аналізу даних як "неконтрольоване навчання" або попередню обробку для підвищення точності навчання. У машинному навчанні ефективність зазвичай вимірюється здатністю відтворювати відомі знання, тоді як у виявленні знань та інтелектуальному аналізі даних (KDD) основною метою є виявлення невідомих знань. Неконтрольовані методи, оцінені на основі відомих знань, легко перевершують інші контрольовані методи. З іншого боку, контрольовані методи не можуть бути використані в типових завданнях KDD через брак навчальних даних.

Машинне навчання також тісно пов'язане з оптимізацією. Багато задач навчання формулюються як мінімізація деякої функції втрат на множині навчених прикладів. Функція втрат - це розбіжність між передбаченнями навчальної моделі та реальними прикладами задачі (наприклад, класифікація вимагає присвоєння міток прикладам, і модель навчається правильно передбачати попередньо присвоєні мітки набору).

У той час як алгоритми оптимізації можуть мінімізувати втрату навчальної вибірки, машинне навчання має на меті мінімізувати втрату невідомих зразків.

Визначення ефективності узагальнення різних алгоритмів навчання наразі є активною темою досліджень, особливо в алгоритмах глибокого машинного навчання, що базуються на статистичних даних.

Машинне навчання і статистика - це близькі галузі з точки зору методів, але їхні основні цілі відрізняються. У той час як статистика робить висновки про популяцію на основі вибірки, машинне навчання знаходить прогностичні закономірності, які можна узагальнити. За словами Майкла І. Джордана, ідеї машинного навчання, від методологічних принципів до теоретичних інструментів, мають довгу передісторію в статистиці. Він також запропонував термін "наука про дані" для позначення галузі в цілому.

Лео Брейман розрізняв дві парадигми статистичного моделювання: моделі даних та алгоритмічні моделі.

Деякі статистики взяли на озброєння методи машинного навчання, які розвинулися в комплексну галузь, що отримала назву статистичне навчання.

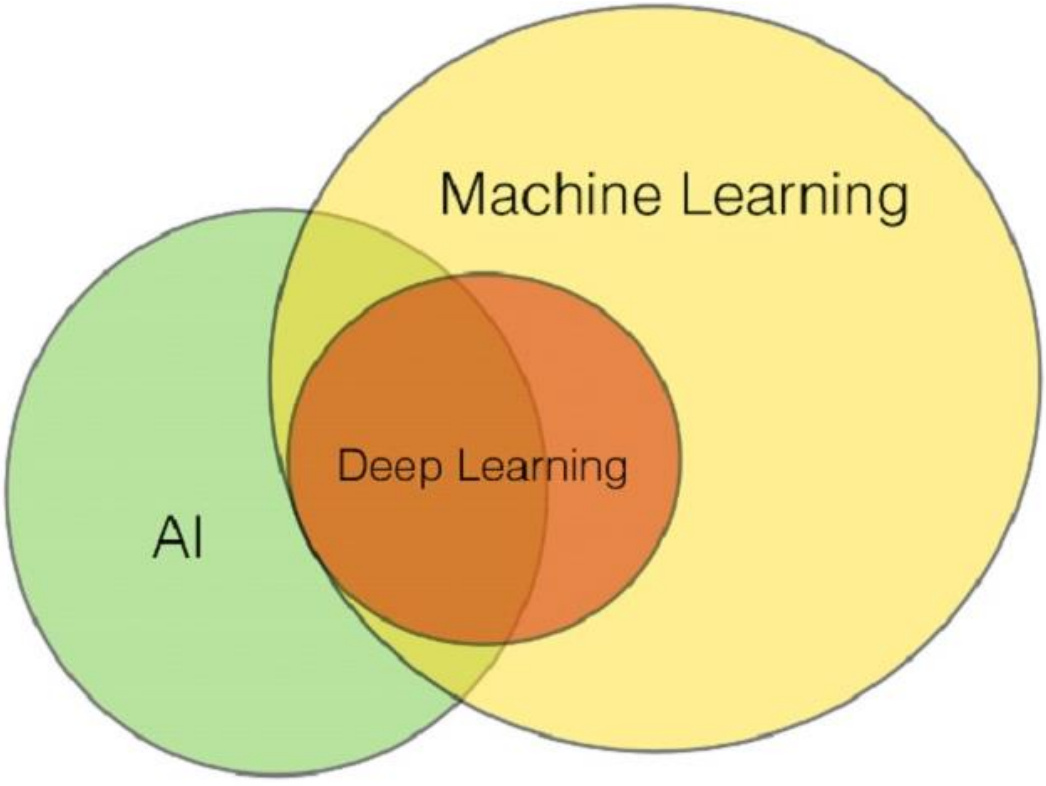


Рис. 1.1. Машинне навчання як підмножина ШІ

1.1.1. Огляд та аналіз існуючих аналогів

Лідоскоринг не є універсальною системою. Кожна компанія має власну стратегію підрахунку, яка відповідає її цілям і завданням. Навіть зважаючи на те, що оцінка лідів є індивідуальною за своєю природою, корисно навести кілька прикладів, які допоможуть вам побудувати маркетингову стратегію і зрозуміти, як найкраще розраховувати лідо-скоринг.

Лідо-скоринг - це процес присвоєння числового значення кожному отриманому ліду, щоб передбачити, як з ним слід поводитися. Важливо знати, як розраховується оцінка лідів, оскільки не всі ліди однакові. Наприклад, ви не можете поводитися з людиною, яка знаходиться вгорі конверсії, так само, як з людиною, яка знаходиться внизу конверсії, з тим же змістом або інтересом, як і з людиною внизу конверсії.

Деякі з найкращих способів підрахунку лідів включають оцінку того, як часто лід взаємодіє з компанією, запис того, скільки разів він відвідує веб-сайт, і оцінку того, наскільки він готовий отримувати рекламні матеріали.

Наведена нижче модель є гарною відправною точкою для створення власної стратегії підрахунку лідів.

Lead Pilot - це інструмент вхідного маркетингу для фінансових консультантів з власним інтегрованим інструментом оцінки потенційних клієнтів.

Принцип роботи дуже простий: кожному потенційному клієнту присвоюється оцінка від 1 до 100 балів, яка змінюється і оновлюється в режимі реального часу, щоб врахувати кожну поведінку. Чим вищий бал, тим більш кваліфікованим є потенційний клієнт.

Алгоритм враховує, як лід взаємодіє з контентом, які дії він виконує на сайті, скільки часу проводить на ньому та інші фактори. Lead Pilot (рис. 1.2) використовує штучний інтелект для оцінки потенційних клієнтів, але

користувачі також можуть коригувати свої оцінки.

Наприклад, уявімо, що є веб-сайт, який просуває фінансові продукти (рис. 1.3), і користувач виконує такі дії:

- завантажив електроний товар та залишив свої особисті дані: +10 балів;
- кліки певних кнопок на сайні: +2 бали;
- переглянув онлайн-уроків: +10 балів;
- підписався на електронні розсилки: +3 бали;
- переглянува сторінку з цінами на сайті: +20 балів;

У цьому випадку лід матиме загальну оцінку 45 балів.

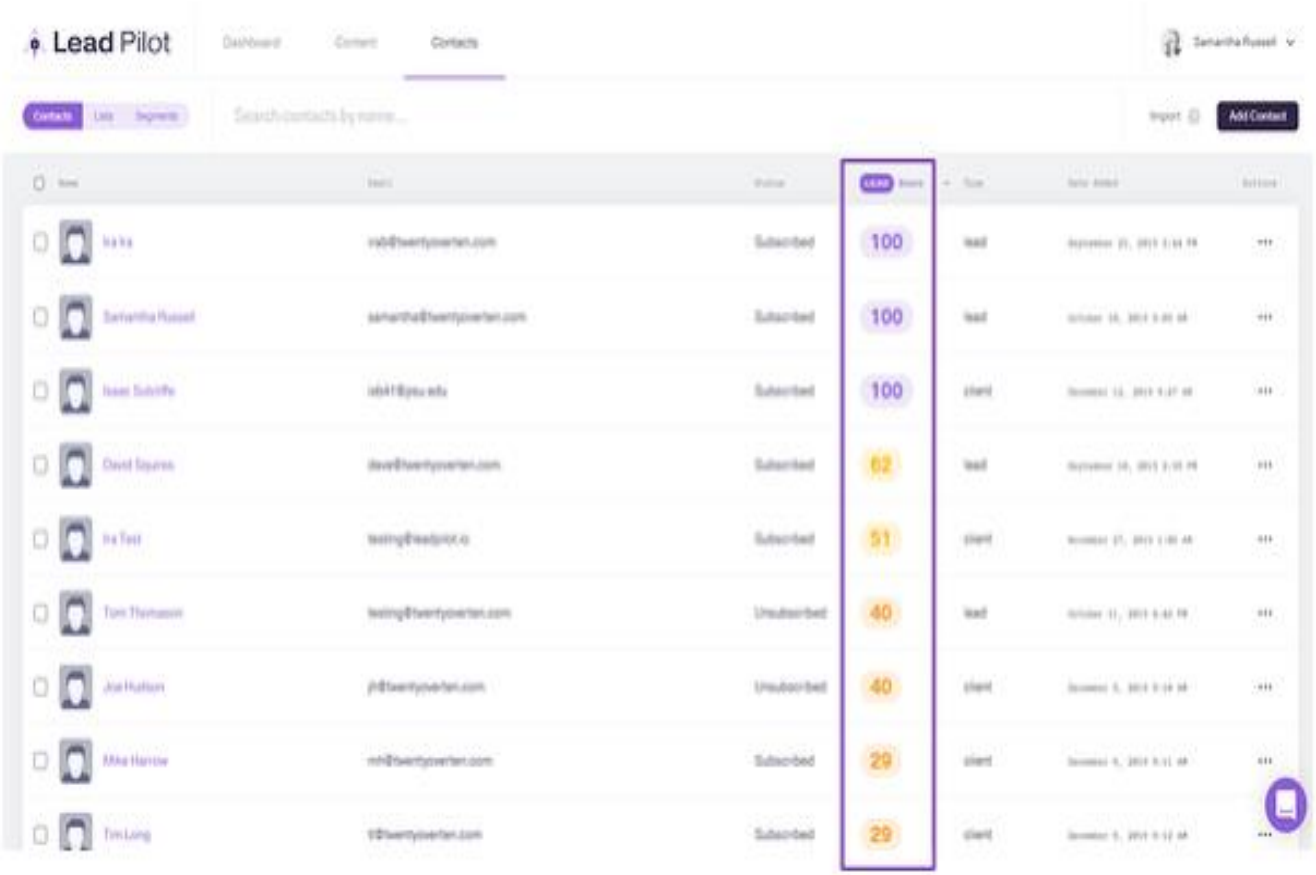


Рис. 1.2. Інтерфейс платформи Lead Pilot

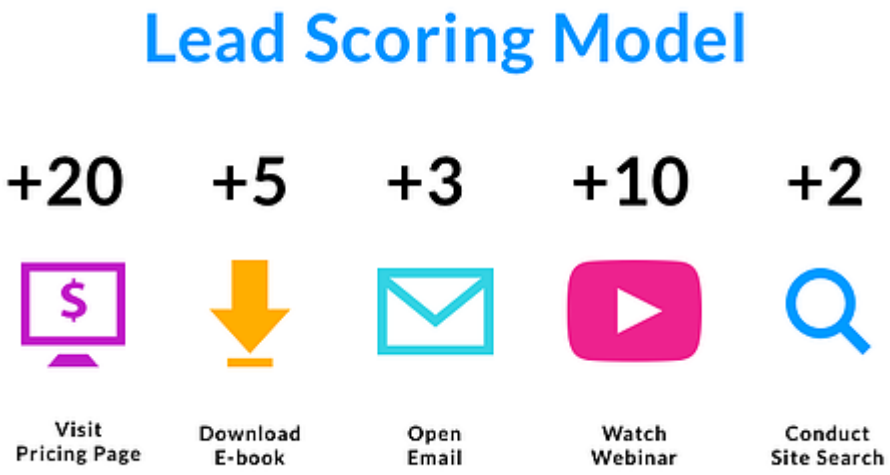


Рис.1.3 Візуалізація дій на вебсайті

Матриця оцінки потенційних клієнтів Merodio базується на двох різних показниках: PAIN і FIT.

Оцінка PAIN представляє інтенсивність проблеми або точки болю, з якою стикається клієнт. Можна поставити оцінку від 0 (у ліда немає проблем) до 10 (проблема дуже актуальна для ліда й її потрібно швидко вирішити).

Оцінка FIT показує, наскільки користувач близький до покупця або ідеального клієнта компанії. Наприклад, якщо лідер має економічні ресурси для придбання продукту оцінка скорингу потребує ефективного її застосування.

Остаточна оцінка відведення має бути сумою балів PAIN та FIT.

Використовуючи оцінку, Merodio (рис.1.4.) класифікує потенційного клієнта на різні групи (холодний, теплий, гарячий) або чи готовий він до більшої маркетингової діяльності (MQL) або продажів (SQL)[1-3].

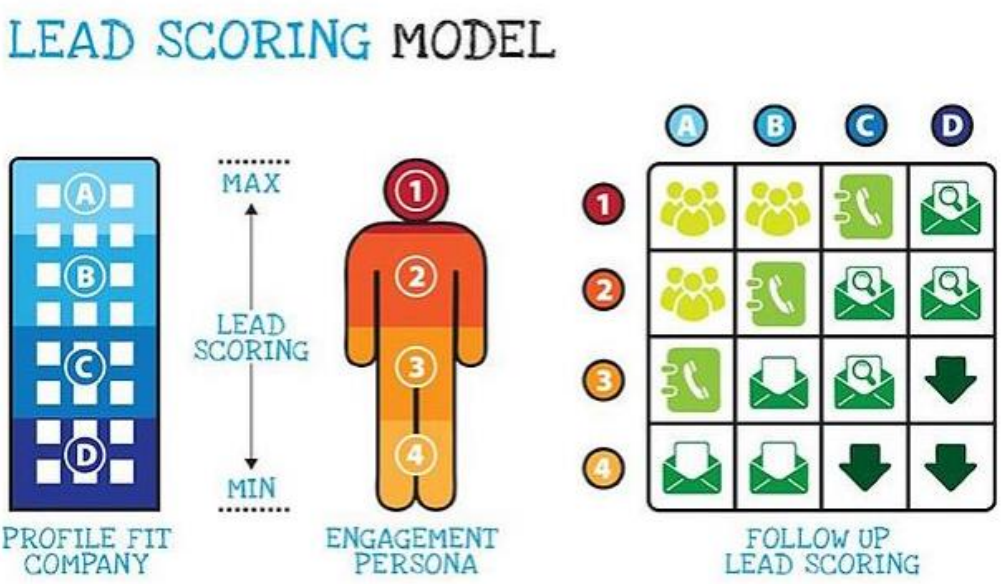


Рис.1.4. Візуалізація оцінок Merodio

CyberClick розробив унікальну матрицю оцінювання лідів на основі двох параметрів.

Наприклад, демографічний профіль потенційного клієнта, отриманий на основі даних, залишених ним під час заповнення форми. Потім ліди класифікуються за різними категоріями, такими як незнайомі (для аналізу яких недостатньо даних), непридатні, придатні та дуже придатні.

Залежно від поведінки потенційного клієнта, включаючи взаємодію з сайтом, контентом і банерами, ліди оцінюються як неактивні, не дуже активні, активні або дуже активні.

Згідно з таблицею (рис. 1.5), можливі до 16 різних комбінацій лідів. Для наочності, присвоєння різної "температури" кожній категорії полегшує розуміння того, як діяти в кожній ситуації.

Холодний контакт: недостатня інформація або відсутність активності в

джерелі.

Теплий контакт: більш активний, ніж холодний, але ще не досяг оптимального поєднання обох параметрів.

Теплі ліди: дуже активні ліди. Це найцікавіші та найдопитливіші люди для роботи.

Рис.1.5. Таблиця категоризації клієнтів

Система Sliverpop схожа на ту, що зображена на (рис.1.5.). Вона містить шаблон із 16 різними можливими комбінаціями інтересу та придатності. Різниця полягає в тому, що їх остаточна система класифікації ділиться на чотири наступні категорії клієнтів:

- неякісні потенційні клієнти (ті, хто мало цікавить і з ними не потребна ніяка комунікація чи увага);
- маркетингові кваліфіковані потенційні клієнти, які вважають цільовими;
- потенційні клієнти, кваліфіковані для дій із створення попиту;
- кваліфіковані потенційні клієнти з продажу.

У Business2Community є дуже простий приклад підрахунку потенційних клієнтів для бізнесу B2B. Матриця (рис.1.6.) базується на відповідях, які користувачі дають на 4 поставлені запитання: посада, відділ, розмір компанії та тип компанії. Кожній відповіді можна поставити 4 можливі бали.

- Найкраща відповідь: максимум балів
- Друга найкраща відповідь: 2-й рівень балів
- 3-я найкраща відповідь: 3-й рівень балів
- Негативна відповідь: Негативні бали

Demographic	Lead Input	Score
Job Level/ Seniority	VP	+ The most points (10)
	Director	+ 2nd most points (8)
	Manager	+ Few points (6)
Most Important Factor for Fictional Company	HR	+ The most points (15)
	Finance	+ 2nd most points (10)
	Legal	+ Few points (2)
Employee Size	500-999	+ The most points (5)
	999-5,000	+ 2nd most points (4)
	250-499	+ Few points (3)
Company Type	B2B	+ The most points (10)
	B2C	+ 2nd most points (7)
	Non-Profit	- Negative points (-5)

Рис.1.6. Матриця підрахунку корпоративних клієнтів

Після визначення балів різні фактори додаються разом, щоб вирахувати остаточний бал для кожного кандидата. Наприклад, згідно з таблицею вище, HR-менеджер НУО з 600 працівниками отримає 23 бали (простий розрахунок: 8 + 15 + 5 - 5).

Hubspot (рис. 1.7) має прогностичну модель оцінки потенційних клієнтів, яка використовує машинне навчання для сортування тисяч даних, щоб знайти найкращих кандидатів. Оскільки цей процес автоматизований, його можна оновлювати та оптимізувати з часом, заощаджуючи час і зусилля.

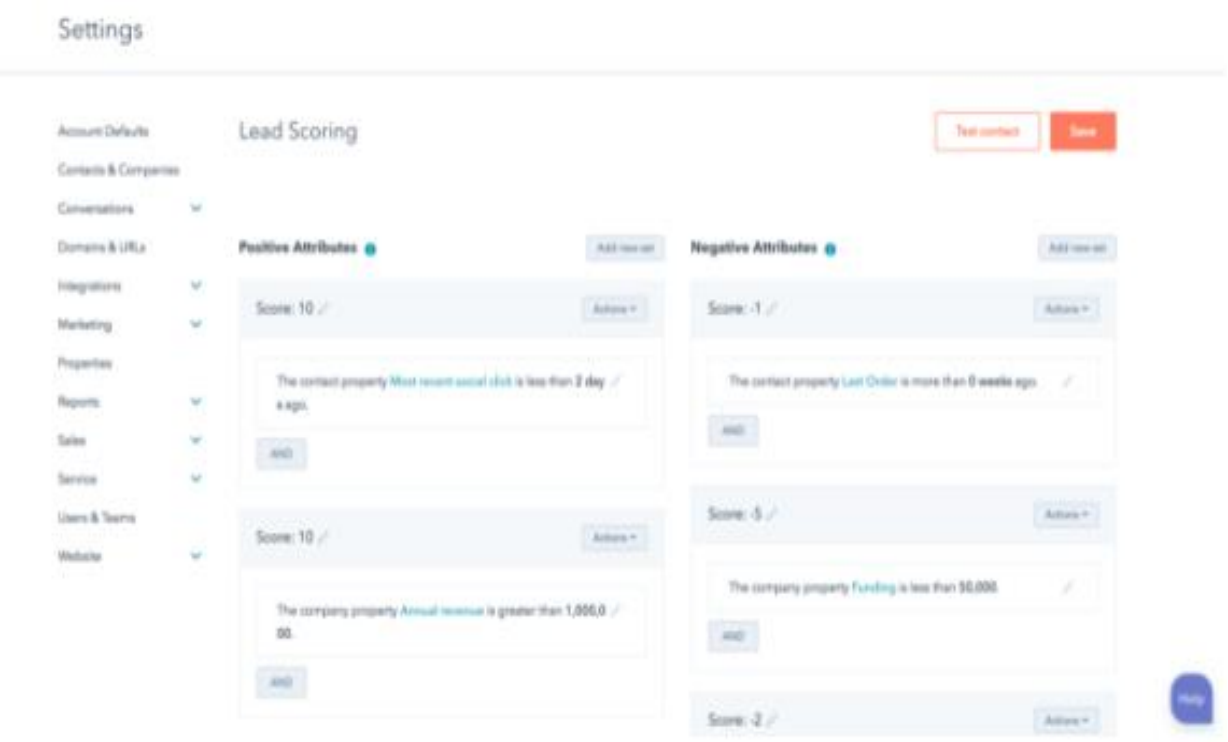


Рис. 1.7. Інтерфейс сервісу Hubspot

Інструменти Hubspot також дозволяють кастомізувати оцінку лідів, щоб зробити цей процес більш ефективним для бізнесу.

Ще одна корисна функція Hubspot - можливість створювати різні таблиці для різних аудиторій та лідів. Це особливо корисно, коли бізнес зростає, розширюється на різні галузі, країни тощо і потребує залучення потенційних клієнтів різними способами. З Hubspot ви можете мати до 25 унікальних моделей оцінки лідів, які адаптуються до будь-яких потреб бізнесу.

1.1.2. Вибір операційної системи

Для розробки було обрано Windows 10. Windows 10 - операційна система

для персональних комп'ютерів і робочих станцій, розроблена компанією Microsoft як частина сімейства Windows NT. Після Windows 8.1 система отримала номер 10 мінус 9. Windows 10 для серверів включає в себе Windows Server 2016, Windows Server 2019 і Windows Server 2022.

Система розроблена як єдина платформа для цілого ряду пристроїв, включаючи ПК, планшети, смартфони і консолі Xbox One. Доступна єдина платформа розробки та єдиний універсальний магазин додатків, сумісний з усіма підтримуваними пристроями; Windows 10 надається як послуга, а оновлення випускаються протягом усього циклу підтримки. Протягом першого року після запуску системи користувачі могли безкоштовно оновитися до Windows 10 на пристроях з ліцензійними версіями Windows 7, Windows 8.1 і Windows Phone 8.1. Серед ключових нових функцій - голосовий помічник Cortana та можливість створювати та перемикатися між кількома робочими столами.

Угода користувача Windows 10 дозволяє корпорації Майкрософт збирати велику кількість інформації про користувача, включаючи історію інтернет-перегляду, паролі точок доступу та дані про вибір клавіатури.

Згідно зі статистикою сайту W3Schools, у квітні 2017 року Windows 10 посіла перше місце серед операційних систем, що використовуються для доступу до Інтернету, випередивши попереднього лідера, Windows 7 .[4-7]

1.2. Мета розробки та сфера застосування

Основні терміни та ключові слова:

Google Cloud Platform - це набір загальнодоступних хмарних обчислювальних сервісів, що надаються компанією Google. Платформа включає в себе набір хостингових сервісів для обчислень, зберігання даних і розробки додатків, що працюють на обладнанні Google. Розробники програмного забезпечення, хмарні адміністратори та інші корпоративні IT-спеціалісти можуть отримати доступ до сервісів Google Cloud через загальнодоступний Інтернет або приватне мережеве з'єднання.

Python - високорівнева мова програмування загального призначення з динамічною строгою типізацією та автоматичним керуванням пам'яттю, орієнтована на підвищення продуктивності розробників, читабельності та якості коду, а також забезпечення переносимості програм, написаних на Python. Мова є повністю об'єктно-орієнтованою і все в ній є об'єктом [8].

Розроблений продукт може бути використаний в компаніях, де виконується спеціалізований збір та аналіз даних, наприклад, в банківському секторі. На мою думку, системи збору та аналізу інформації стають все більш поширеними.

Метою розробки є надання можливості користувачам СУБД експортувати дані користувачів для обробки та аналізу. Ця комп'ютерна система допомагає оптимізувати процес пошуку клієнтів онлайн-шкіл з вивчення іноземних мов. Менеджер колл-центру отримує середньозважену оцінку, яка вказує на ймовірність того, що клієнт є студентом онлайн-школи.

1.3. Постановка проблеми

Метою проекту є розробити додаток для аналітики користувацьких даних та побудови скорингових моделей у СКБД, а також збереження та обробки користувацької інформації з метою покращення та оптимізації процессу комунікації з клієнтами.

Дане програмне забезпечення дозволить зберігати великий об’єм інформацію у зручному для користувача вигляді та прогнозувати вірогідність оплати послуг користувачем.

На етапі проєктування необхідно дослідити методи машинного навчання, оптимізувати скорингові моделі.

1.4. Вимоги до програмного забезпечення

Основна вимога – надати можливість користувачу самостійно будувати

скорингові моделі, бачити графічні моделі, відслідковувати математичні показники у консолі, наприклад f-score або churned.

1.4.1. Функціональні вимоги

Кінцевий продукт повинен дотримуватися наступних функціональних вимог:

- інтуїтивно зрозумілий інтерфейс користувача;
- дані повинні виводитися користувачу на екран;
- розрахунок скоригу користувачів;
- побудова графічних моделей.

1.4.2. Вимоги інформаційної безпеки

Організація безпечної і актуальною настройки СКБД. Дана вимога включає в себе загальні завдання забезпечення безпеки, такі як своєчасна установка оновлень, відключення невикористовуваних функцій або застосування ефективної політики паролів.

Безпека призначеного для користувача ПЗ. Сюди можна віднести завдання побудови безпечних інтерфейсів і механізмів доступу до даних.

Безпечна організація і робота з даними. Питання організації даних і управління ними є ключовим в системах зберігання інформації. У цю галузь входять завдання організації даних з контролем цілісності та інші, специфічні для СКБД проблеми безпеки. Фактично це завдання включає в себе основний обсяг залежать від даних вразливостей і захисту від них.

1.4.3. Вимоги до апаратного середовища

Для підтримки стабільної роботи СКБД, слід дотримуватися таким технічним вимогам:

– операційна система: Windows 10, Windows Server 2016 або Windows Server 2019;

– оперативної пам'яті 1 гігабайт;

– процесор x64 з тактовою частотою 1,4 ГГц;

– 100 мегабайт вільного місця на диску.

1.4.4. Вимоги до сумісності

Основною мовою програмування був Python. Також ця програмна мова була використана для побудови скорингових моделей та розрахунків скорингу.

База данних була розроблема у середовищі MySQL Workbench, для редагування коду Python був використаний додаток Geany.

РОЗДІЛ 2
ДИЗАЙН ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональні цілі програми

Результатом цього кваліфікаційного дослідження має стати додаток, який розраховує платоспроможність користувача за певний період часу на основі характеристик, отриманих з бази даних MySQL.

Кінцевим результатом роботи програми будуть дві графічні моделі, засновані на середньозважених коефіцієнтах рейтингу користувачів і присвоєння балів кожному користувачеві. Результати зберігаються у файлі з розширенням ".csv".

Основне призначення додатку:

- спрощення роботи з великими об’ємами даних;
- оцінювання кожного користувача окремо;
- оптимізація роботи call-центру.

Для виконання цього завдання додаток повинен вміти обробляти великі обсяги даних користувача і відображати результати в графічному і текстовому форматах.

2.2. Застосовані математичні методи

В даній кваліфікаційній роботі для оцінки якості моделей та порівняння різних алгоритмів використовуються метрики, підбір та аналіз яких є невід'ємною частиною роботи data scientist.

Точність, достовірність і пригадування

Перш ніж перейти до самих метрик, необхідно ввести важливе поняття для пояснення цих метрик з точки зору помилки класифікації: матриця плутанини [6].

Якщо припустити, що є два класи і алгоритм, який визначає, до якого класу належить кожен об'єкт, то матриця помилок класифікації (рис. 2.8) має такий вигляд:

	$y = 1$	$y = 0$
$\hat{y} = 1$	True Positive (TP)	False Positive (FP)
$\hat{y} = 0$	False Negative (FN)	True Negative (TN)

Рис. 2.8. Матриця помилок класифікації

Тут $\hat{y}$ - це відповідь алгоритму на об'єкті, а y - справжня мітка класу на цьому об'єкті. Таким чином, помилки класифікації бувають двох видів: False Negative (FN) і False Positive (FP).

Accuracy

Інтуїтивно зрозумілою, очевидною і майже невикористаної метрикою є accuracy - частка правильних відповідей алгоритму:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

Ця метрика марна в задачах з нерівними класами, і це легко показати на прикладі.

Припустимо, необхідно оцінити роботу спам-фільтра пошти. Існує 100 НЕ-спам листів, 90 з яких класифікатор визначив вірно (True Negative = 90, False Positive = 10), і 10 спам-листів, 5 з яких класифікатор також визначив вірно (True Positive = 5, False Negative = 5).

Тоді accuracy:

$$accuracy = \frac{5 + 90}{5 + 90 + 10 + 5} = 86,4 \tag{2.2}$$

Однак якщо просто буде передбачати всі листи що не-спам, то в результаті буде отримано більш високу accuracy:

$$accuracy = \frac{0 + 100}{0 + 100 + 0 + 10} = 90,9 \tag{2.3}$$

При цьому, модель абсолютно не володіє ніякою прогностичної сили, так як спочатку необхідно визначати листи зі спамом. Подолати це допоможе перехід із загальною для всіх класів метрики до окремими показниками якості класів.

Precision, recall і F-міра (рис.2.9.)

Для оцінки якості роботи алгоритму на кожному з класів окремо необхідно ввести метрики precision (точність) і recall (повнота).

$$precision = \frac{TP}{TP + FP} \tag{2.4}$$

$$recall = \frac{TP}{TP + FN} \tag{2.5}$$

Precision можна інтерпретувати як частку об'єктів, названих класифікатором позитивними і при цьому дійсно є позитивними, а recall показує, яку частку об'єктів позитивного класу з усіх об'єктів позитивного класу знайшов алгоритм машинного навчання.

Саме введення precision не дозволяє записувати всі об'єкти в один клас, так як в цьому випадку зростає рівень False Positive. Recall демонструє здатність алгоритму виявляти даний клас взагалі, а precision - здатність відрізняти цей клас від інших класів.

Як зазначено раніше, помилки класифікації бувають двох видів: False Positive і False Negative. У статистиці перший вид помилок називають помилкою I-го роду, а другий - помилкою II-го роду.

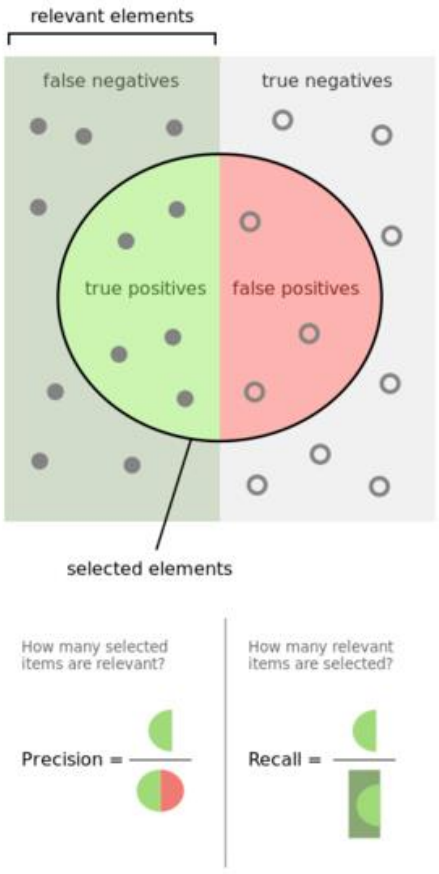


Рис. 2.9. Графічне зображення метрик precision і recall

Precision і recall не залежить, на відміну від accuracy і, не залежать від пропорцій класу і тому можуть бути застосовані до незбалансованих зразків.

Часто в реальних застосуваннях завдання полягає в тому, щоб знайти оптимальний (для замовника) баланс між цими двома метриками. Типовим прикладом є задача визначення рейтингу споживача.

Очевидно, що неможливо знайти всіх клієнтів, які хотіли б купити певний продукт. Однак, визначивши стратегії та ресурси для виявлення платоспроможних клієнтів, можна обрати відповідні пороги точності та пригадування. Наприклад, колл-центри та менеджери мають обмежені ресурси і можуть зосередитися лише на пошуку клієнтів з найвищими показниками. Стратегії збору необхідних показників мають бути продумані та розставлені за пріоритетами. Наприклад, створення цікавого та зручного користувацького інтерфейсу, обмеження кількості полів для реєстрації тощо, адже інтерес

потенційних клієнтів може швидко згаснути і вони можуть передумати реєструватися.

2.3. Опис використовуваних технологій та мов програмування

Дана комп'ютерна система розроблена за допомогою наступних технологій:

- мова запитів SQL;
- мова програмування Python.

Structured Query Language (SQL) - мова структурованих запитів. Мова запитів SQL - універсальна мова для роботи з даними бази. Мова SQL була розроблений фірмою IBM в кінці 70-х років. Перший міжнародний стандарт мови був прийнятий міжнародної стандартизуючою організацією ISO в 1989 р. В даний час всі виробники реляційних СКБД підтримують з різним ступенем відповідності стандарт SQL92[10-11].

Мова запитів SQL використовується для управління масивами даних в базах даних, множинами. Мова SQL надає можливість для виведення чи введення структурованої заданої інформації з бази та в неї. SQL також застосовується для зміни даних та їх оновлення.

SQL - це функціональна мова програмування. Вона відрізняється від алгоритмічних мов. Основою мови є не алгоритм, а набір команд, які визначають відношення між множинами та підмножинами інформації.

Слід зазначити, що системи управління базами даних (СУБД), такі як ORACLE, MS SQL і MYSQL, мають різні реалізації. Мови SQL різних СУБД мають незначні відмінності, наприклад, у детальному синтаксисі, що визначає оператори. Такі відмінності існують для спеціалізованих функцій, пов'язаних з конкретною СУБД, але в основному мова має майже однаковий загальний синтаксис у будь-якій СУБД [11-13].

Реляційна база даних (рис. 2.10) - це таблиця, в якій інформація організована у стовпцях (полях або властивостях) і рядках (записах або

кортежах). Щоб змінити або видалити дані в стовпцях і рядках, а також дані в певних комірках (розриви стовпців або рядків), можна скористатися прикладним інструментом (наприклад, phpmyadmin) або зробити SQL-запит до бази даних і виконати необхідну дію.

У реляційної моделі даних таблиця має такі основні властивості:

- ідентифікується унікальним ім'ям;
- має кінцеву (як правило, постійне) не нульову кількість стовпців;
- має кінцеве (можливо, нульовий) число рядків;
- стовпчики таблиці ідентифікуються своїми унікальними іменами і номерами;
- вміст всіх комірок стовпчика належить одному типу даних (тобто стовпці однорідні), вмістом комірки стовпчика не може бути таблиця;
- рядки таблиці не мають будь-якої впорядкованості та ідентифікуються тільки своїм вмістом;
- в загальному випадку елементи таблиці можуть залишатися порожніми, тобто не містити жодного значень, в такому випадку їх стан позначається як NULL.

За допомогою запитів SQL можна:

- створювати таблиці БД;
- змінювати таблиці БД;
- видаляти таблиці БД;
- вставляти записи (рядки) в таблиці БД;
- редагувати записи в таблицях БД;
- витягувати вибіркову інформацію з таблиць БД;
- видаляти вибіркову інформацію з БД.

Це не повний перелік можливостей SQL запитів, але і він дає уявлення, що за допомогою SQL запитів можна зробити з базою даних все що необхідно.

Основні оператори sql запитів:

- CREATE TABLE - оператор sql для створення таблиці бази даних;
- ALTER TABLE - оператор sql для зміни таблиці БД;
- INSERT INTRO - вставка інформації (рядків) в таблиці БД;
- UPDATE - оператор для редагування інформації в таблицях БД;
- SELECT - вилучення інформації з таблиць БД;
- DELETE - видалення інформації з таблиць БД.

Python - високорівнева мова програмування загального призначення з динамічною строгою типізацією та автоматичним керуванням пам'яттю, з акцентом на підвищення продуктивності розробника, читабельності та якості коду, а також забезпечення переносимості програм, написаних на Python. Мова повністю об'єктно-орієнтована, і все в ній є об'єктом. Незвичайною особливістю мови є те, що блоки коду відокремлюються пробілами. Синтаксис ядра мови мінімальний, тому звертатися до документації практично не потрібно. Сама мова називається інтерпретованим типом і використовується, серед іншого, для написання скриптів. Недоліком цієї мови є те, що програми, написані нею, працюють повільніше та займають більше пам'яті, ніж аналогічний код, написаний на компільованих мовах, таких як C та C++ [14].

Python - мультипарадигмальна мова програмування, яка підтримує імперативне, процедурне, структурне, об'єктно-орієнтоване, метапрограмування та функціональне програмування. Узагальнені задачі програмування вирішуються за допомогою динамічної типізації. Аспектно-орієнтоване програмування частково підтримується декораторами, тоді як більш повну підтримку надають додаткові фреймворки. За допомогою бібліотек та розширень можна реалізувати такі техніки, як контрактне та логічне програмування. Ключові архітектурні особливості включають динамічну типізацію, автоматичне керування пам'яттю, повну інтроспекцію, механізми обробки винятків, підтримку багатопотокових обчислень з глобальним блокуванням інтерпретатора (GIL) та високорівневі структури даних. Підтримується розбиття програм на модулі, які можна об'єднувати в пакети [17] Мова програмування Python була створена в 1991 році голландцем Гвідо ван Россумом.

Python має простий для розуміння синтаксис: Допоміжні синтаксичні елементи, такі як дужки та крапка з комою, рідко використовуються в Python, що робить код легшим для читання, ніж в інших мовах програмування. З іншого боку, правила мови вимагають від програмістів робити відступи у вкладених структурах. Зрозуміло, що добре оформлений текст з невеликою кількістю відволікаючих елементів легше читати і розуміти [15-19].

Python є повноцінною, майже універсальною мовою програмування, яка використовується в різних галузях; основною парадигмою, яку підтримує Python, є виключно об'єктно-орієнтоване програмування. Однак у цьому курсі згадуються лише об'єкти та викладаються основи структурного програмування. Немає сенсу вивчати більш складні парадигми без знання базових типів даних, гілок, циклів та функцій.

Інтерпретатор Python вільно розповсюджується за ліцензією, подібною до GNU General Public License.

У цій кваліфікаційній роботі використовувалася бібліотека Python pandas. pandas - це високорівнева бібліотека Python для аналізу даних. Вона побудована на основі низькорівневої бібліотеки NumPy (написаної на C) і має значні переваги в продуктивності. pandas є найдосконалішою та найбільш швидкозростаючою бібліотекою для обробки та аналізу даних в екосистемі Python [21].

Ми також використовували бібліотеку matplotlib - бібліотеку 2D-графіки для мови програмування python, здатну створювати високоякісні графіки в різних форматах. matplotlib - це модульний пакет для мови програмування Python.

Ми обрали Scikit-learn - найпопулярнішу бібліотеку для розв'язання класичних задач машинного навчання. Вона пропонує широкий спектр алгоритмів контрольованого та неконтрольованого навчання. Навчання під наглядом вимагає наявності маркованого набору даних з відомими значеннями цільового показника. З іншого боку, неконтрольоване навчання не потребує маркованого набору даних і вимагає навчитися витягувати корисну інформацію

з випадкових даних. Однією з головних переваг цієї бібліотеки є те, що вона працює на основі багатьох популярних математичних бібліотек і може бути легко інтегрована з ними. Ще однією перевагою є велика спільнота та детальна документація; Scikit-learn широко використовується промисловими системами, що використовують класичні алгоритми машинного навчання, в наукових дослідженнях і навіть новачками, які роблять перші кроки в машинному навчанні [22].

2.4. Структура системи та алгоритмів її функціювання

Для проектування та розробки структури бази даних необхідно як мінімум виконати наступні дві вимоги:

- зберегти всю інформацію після поділу її на таблиці;
- мінімізувати надмірність того, як ця інформація зберігається.

Другий пункт важливий не тільки з-за того, що надмірність впливає на розмір БД. Найчастіше при оновленні даних потрібно обробити багато рядків. В такому випадку ви ризикуєте просто забути оновити деякі з них, що призведе до колізій всередині БД.

Нижче перераховані деякі рекомендації, які допоможуть домогтися ефективної структури:

- використовуйте хоча б третю нормальну форму;
- створюйте обмеження для вхідних даних;
- не зберігайте ПІБ в одному полі, так само як і повна адреса;
- встановіть для себе правила іменування таблиць і полів.

Нормальні форми - це вимоги, яких слід дотримуватися при правильній проектуванні бази даних. Нормальних форм існує цілих 6 штук, проте зазвичай дотримуються всього лише 3 і для початку цього більш ніж достатньо.

Перша нормальна форма відповідає наступним вимогам:

- в відношенні немає однакових кортежів;

– кортежі не впорядковані;
– атрибути не впорядковані і розрізняються по найменуванню;
– всі значення атрибутів атомарний.

Асоціативна змінна перебуває у другій нормальній формі тоді і тільки тоді, коли вона перебуває у першій нормальній формі і всі неключові атрибути залежать (відповідно) від її потенційного ключа.

Якщо потенційний ключ простий, тобто складається з одного атрибута, то функціональна залежність від цього ключа є наперед зв'язаною (повною). Якщо потенційний ключ складений, то, згідно з визначенням другої нормальної форми, у відношенні не повинно бути неключових атрибутів, які залежать від деяких складених потенційних ключів.

Визначення другої нормальної форми забороняє існування неключових атрибутів, які взагалі не пов'язані з потенційним ключем. Тому друга нормальна форма також забороняє побудову відношення як незв'язаного (нерегулярного, випадкового) набору атрибутів.

Якщо відношення знаходиться у другій нормальній формі і всі неключові атрибути є взаємно незалежними, то відношення знаходиться у третій нормальній формі.

Щоб усунути залежності неключових атрибутів, відношення необхідно декомпонувати на декілька інших відношень. При цьому залежні неключові атрибути виносяться в окреме відношення [23-25].

У цьому якісному дослідженні розглядаються такі сутності та їхні атрибути:

В даній кваліфікаційній роботі фігурують наступні сутності та їх атрибути:

1. Таблиця користувачі (users). Атрибути:
   – ID користувача;
   – електронна адреса;
   – ім’я;
   – прізвище;

- дата реєстрації;
- стать;
- місто;
- вік
- країна;
- мета навчання;
- дата народження;
- джерело реєстрації;
- номер мобільного телефону;
- ім’я в додатку Skype.

2. Таблиця анкети (applications). Атрибути:
   - ID анкети;
   - ім’я;
   - прізвище;
   - номер мобільного телефону;
   - електронна адреса;
   - ім’я в додатку Skype;
   - часовий пояс;
   - дата подачі анкети;
   - ID користувача;
   - джерело подачі анкети.
3. Таблиця з розкладом онлайн-уроків (skype_lessons). Атрибути:
   - ID онлайн-уроку;
   - ID студента;
   - ID вчителя;
   - дата проведення заняття;
   - тип вчителя.
4. Таблиця вчителі (teachers). Атрибути:
   - ID вчителя;

- національність;
- стать;
- список мов, якими володіє вчитель;
- вік;
- ім’я
- бінарне поле, яке показує вчитель носій мови чи ні;
- ID користувача;
- рівень знань вчителя.

5. Таблиця ввідних занять (skype_lesson_interview). Атрибути:
   - ID ввідного заняття;
   - ID студента;
   - ID вчителя;
   - дата проходження заняття;
   - статус проходження заняття;
   - граматичні навички;
   - навички спілкування;
   - навички аудіювання;
   - рекомендований курс.
6. Таблиця оплат (payment_transaction). Атрибути:
   - ID транзакції;
   - кількість придбаних уроків;
   - тип вчителя;
   - ціна у валюті оплати;
   - валюта оплати;
   - дата транзакції;
   - ID користувача.

Далі наведено ключі, за рахунок яких пов’язані сутності БД.

1. Користувачі (ID користувача).

- ID користувача - первинний ключ.

2. Анкети (ID анкети, ID користувача).
   - ID анкети - первинний ключ;
   - ID користувача - зовнішній ключ.
3. Розклад онлайн-уроків (ID онлайн-уроку, ID студента, ID вчителя).
   - ID онлайн-уроку - первинний ключ;
   - ID користувача-студента - зовнішній ключ;
   - ID користувача-вчителя - зовнішній ключ.
4. Вчителі (ID вчителя, ID користувача).
   - ID вчителя - первинний ключ;
   - ID користувача - зовнішній ключ.
5. Ввідні заняття (ID ввідного заняття, ID користувача-студента, ID користувача-вчителя).
   - ID ввідного заняття - первинний ключ;
   - ID користувача-студента - зовнішній ключ;
   - ID користувача-вчителя - зовнішній ключ.
6. Оплати (ID транзакції, ID користувача).
   - ID транзакції - первинний ключ;
   - ID користувача - зовнішній ключ.

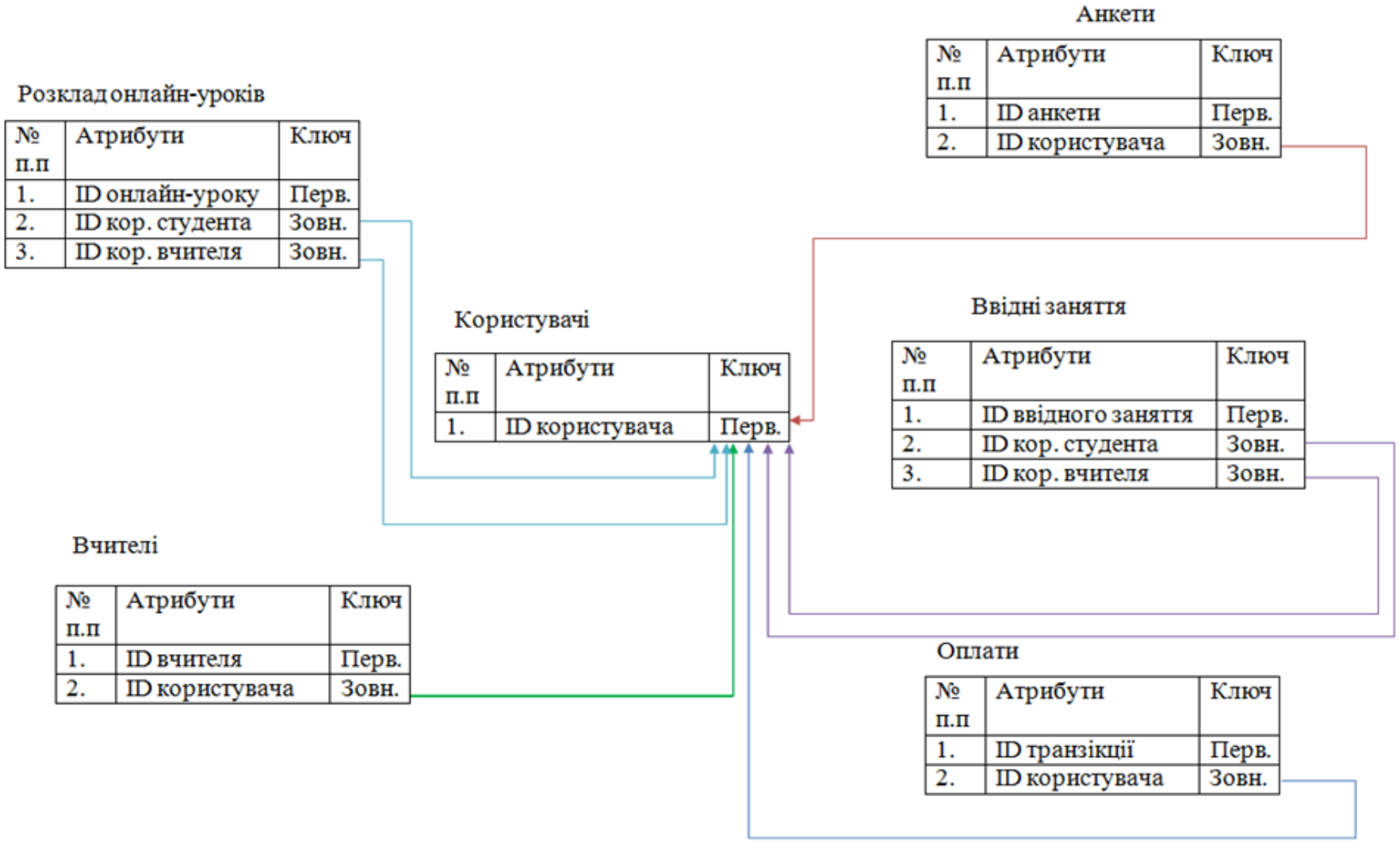


рис.2.10. Схема реляційної бази даних

На реляційній моделі(рис.2.10.) показано зв’язки 2-6 таблиць з таблицею користувачів. Усі таблиці поєднані між собою зовнішнім ключем (ID користувача), на рисинку показана лише певна частка усіх зв’язків заради більшої наглядності реляційної моделі.

У даній кваліфікаційній роботі було використано кілька методів машинного навчання: RandomForestClassifier, RandomForestRegressor, LogisticRegression та ROC-аналіз. Перші два методи безпосередньо перекладаються з англійської як "випадковий ліс", тоді як третій метод перекладається як "логістична регресія".

Випадкові ліси - це алгоритм керованого навчання, який використовується як для класифікації, так і для регресії. Однак, в основному він використовується для задач класифікації [27]. Це тому, що ми знаємо, що ліс складається з дерев, і чим більше дерев, тим стабільнішим є ліс. Аналогічно, алгоритм випадкового лісу будує дерева рішень для вибірок даних, отримує прогнози для кожного з них і, нарешті, обирає оптимальне рішення шляхом голосування. Це ансамблева методика, і вона є кращою за окреме дерево рішень, оскільки зменшує надмірне пристосування шляхом усереднення результатів.

Дерево рішень є евристичною базовою одиницею алгоритму випадкового лісу. Його можна уявити як серію запитань з відповідями "так" чи "ні" для введення даних. В кінцевому підсумку, питання призводять до передбачення певного класу (або значення у випадку регресії). Це інтерпретується моделлю так, що рішення приймаються так само, як і у випадку з людиною. Питання задаються щодо наявних даних, поки (в ідеальному світі) не буде досягнуто рішення.

Основна ідея дерева рішень полягає в тому, що алгоритм формулює запит для доступу до даних [32]; при використанні алгоритму CART питання (також звані розбиттям вузлів) визначаються таким чином, щоб відповіді призводили до зменшення домішки Джині. Це означає, що дерево рішень генерує вузли, які містять велику кількість екземплярів (з вихідного набору даних), що належать до одного класу. Алгоритм намагається ідентифікувати параметри зі схожими

значеннями.

Забруднення Джині - це ймовірність того, що випадково вибрана вибірка у вузлі буде неправильно позначена.

У кожному вузлі дерево рішень шукає значення параметра, яке дає максимальне зменшення забруднення Джині. Крім того, вузли можна відокремити, використовуючи концепцію зберігання інформації.

Потім цей процес повторюється за допомогою "жадібної" рекурсивної процедури, поки дерево не досягне максимальної глибини або поки кожен вузол не буде містити тільки приклади з одного класу. Середньозважене забруднення за індексом Джині має зменшуватися на кожному рівні.

Коефіцієнт забруднення Джині для кожного вузла дорівнює відношенню кількості прикладів, оброблених цим вузлом, до кількості прикладів, оброблених батьківським вузлом. Використовуючи дані візуалізації, можна розрахувати забруднення Джині наступних рівнів дерева та окремих вузлів незалежно. Таким чином, ефективні моделі базуються на базових математичних операціях [28-30].

Однак слід зазначити, що алгоритм лише повністю реорганізує навчальні дані. Мета машинного навчання - узагальнити отримані знання і навчити алгоритм правильно обробляти нові дані, яких він раніше не бачив [31].

Перенавчання відбувається при використанні дуже гнучких моделей (великої ємності), які лише зберігають навчальний набір даних і підлаштовують вузли під нього [16]. Проблема полягає в тому, що такі моделі виявляють не тільки закономірності в даних, але й шум, присутній в даних. Оскільки параметри, що генеруються під час навчання (наприклад, структура дерева рішень), можуть суттєво змінюватися залежно від навчального набору даних, такі гнучкі моделі часто називають моделями з високою варіабельністю.

З іншого боку, недостатньо гнучкі моделі роблять припущення про навчальні дані (тобто модель має тенденцію робити упереджені припущення про дані), що призводить до високого рівня помилок. Наприклад, лінійні класифікатори припускають, що дані розподілені лінійно. Як наслідок, вони не мають достатньої гнучкості, щоб пристосуватися до нелінійних структур.

Грядові моделі можуть навіть не мати достатньої потужності, щоб підігнати навчальні дані для алгоритмів машинного навчання [33-35]. В обох випадках - висока дисперсія і висока помилка - модель не може ефективно обробляти нові дані.

Пошук балансу між надмірною та недостатньою гнучкістю моделі є важливою концепцією машинного навчання, яка називається компромісом між похибкою та дисперсією.

Алгоритм дерева рішень перенавчається, якщо його максимальна глибина не обмежена. Дерева рішень мають необмежену гнучкість і можуть рости до тих пір, поки не досягнуть стану ідеальної класифікації, коли кожному зразку в наборі даних відповідає один лист. Повертаючись до побудови дерева, якщо його глибина обмежена двома шарами (лише одним розділом), класифікація вже не є 100% точною. Варіабельність зменшується за рахунок збільшення помилки [36].

Як альтернатива обмеженню глибини, багато дерев можна об'єднати в одну модель. Це може призвести до створення класифікатора на основі комітету дерев рішень або просто "випадкового лісу".

Логістична регресія є корисним класичним інструментом для вирішення проблем регресії та класифікації, тоді як ROC-аналіз є інструментом для аналізу якості моделі. Обидва алгоритми активно використовуються в медицині для побудови моделей і проведення клінічних випробувань [37].

Логістична регресія широко використовується в скорингу для розрахунку рейтингів позичальників та управління кредитним ризиком. Таким чином, незважаючи на своє статистичне "походження", логістичну регресію та ROC-аналіз майже завжди можна знайти в ряді алгоритмів інтелектуального аналізу даних.

Логістична регресія - це тип множинної регресії, загальною метою якої є аналіз зв'язку між кількома незалежними змінними (також званими регресорами або предикторами) і залежною змінною. Бінарна логістична регресія використовується, коли залежна змінна є бінарною (тобто може приймати тільки два значення). Логістичну регресію можна використовувати для прогнозування

ймовірності настання певної події для певного суб'єкта (наприклад, хвороба/здоров'я, погашення кредиту/дефолт).

Всі регресійні моделі можуть бути записані у вигляді формули:

$$y = F(x_1, x_2, \dots, x_n) \quad (2.6)$$

ROC-крива (рис. 2.11) є найбільш поширеною кривою в машинному навчанні для опису результатів бінарної класифікації. Назва походить від систем обробки сигналів [21], оскільки існує два класи, один з позитивними і один з негативними результатами; ROC-крива показує залежність між кількістю правильно класифікованих позитивних прикладів і кількістю неправильно класифікованих негативних прикладів.

У термінології ROC-аналізу перший клас називається істинно-позитивною множиною, а другий - хибно-негативною множиною. Вважається, що класифікатор має певний параметр, який, змінюючись, забезпечує певний поділ на два класи. Цей параметр часто називають порогом або значенням відсікання. Таким чином, ми отримуємо різні значення для помилок першого та другого класу.

У логістичній регресії значення відсікання коливається від 0 до 1 - це оцінка рівняння регресії. Ми називаємо це оцінкою ...

Щоб зрозуміти природу помилок першого і другого роду, розглянемо чотирипольну матрицю помилок, побудовану на основі результатів класифікації моделі і фактичної (об'єктивної) класової приналежності вибірки.

У машинному навчанні, особливо в задачах статистичної класифікації, матриця помилок, також відома як матриця помилок, є спеціальною табличною схемою (у некерованому навчанні її часто називають матрицею відповідності) для візуалізації роботи алгоритму (зазвичай керованого навчання). Кожен рядок матриці представляє екземпляр істинного класу, а кожен стовпець - екземпляр передбачуваного класу. Ця назва походить від того, що легко перевірити, чи не переплутала система два класи (тобто, часто навмисно неправильно класифікуючи один як інший, тобто, змінюючи логічне значення безпосередньо

на хибне).

Це спеціальна таблиця непередбачених обставин з двома вимірами ("фактичний" і "прогнозований") і однаковим набором "класів" для обох вимірів (кожна комбінація виміру і класу є змінною в таблиці непередбачених обставин).

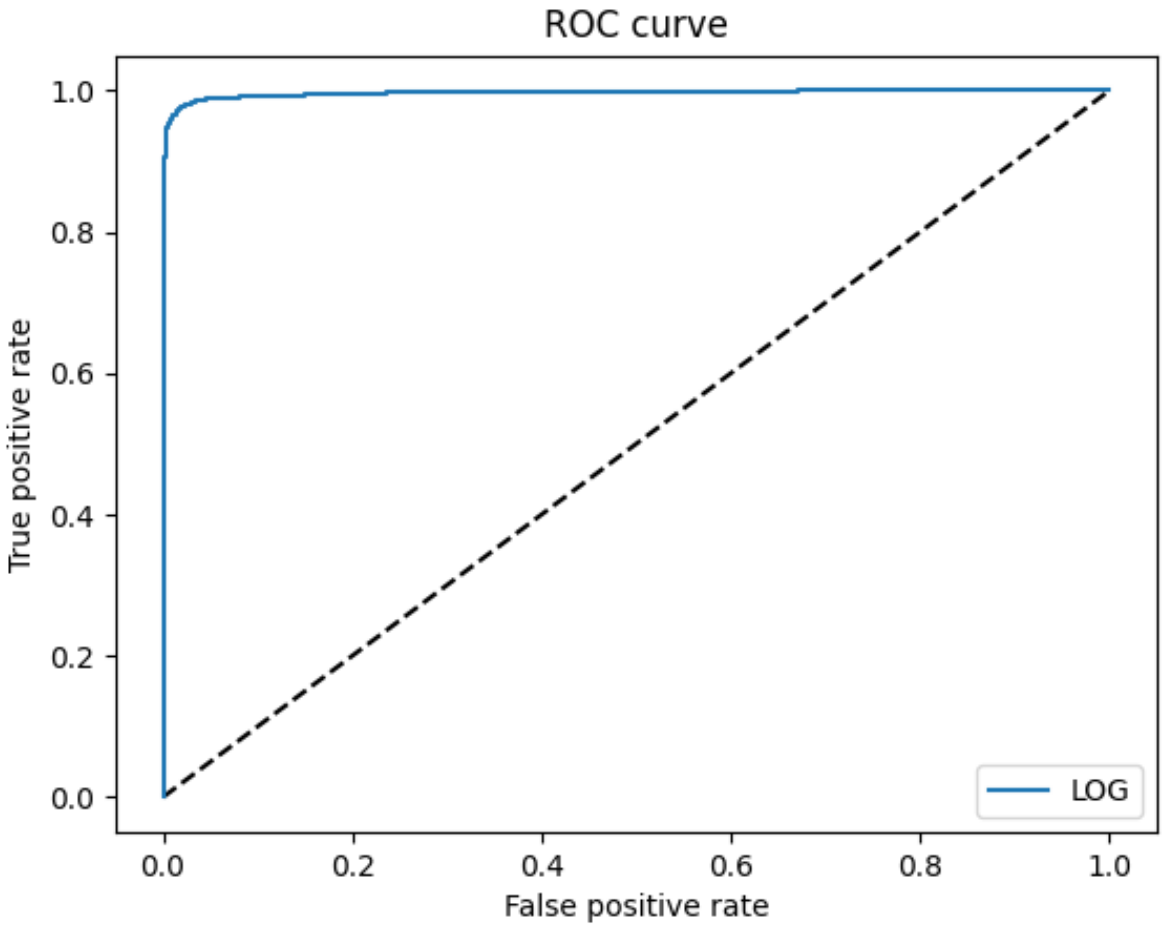


Рис. 2.11. Крива ROC

В ідеальному класифікаторі ROC-крива проходить через лівий верхній кут, а частка правильних спрацьовувань дорівнює 100% або 1,0 (повна чутливість), а частка помилкових спрацьовувань дорівнює 0. Отже, чим ближче крива до верхнього лівого кута, тим вища прогностична здатність моделі. І навпаки, чим менший вигин кривої і чим ближче до діагоналі, тим менш ефективною є модель. Діагональ відповідає повній нерозрізненності класифікаторів, тобто двох класів.

При візуальній оцінці ROC-кривих їхнє взаємне розташування вказує на їхню порівняльну ефективність. Криві, розташовані вище і лівіше, вказують на

те, що модель має вищу прогностичну здатність [38-44].

2.5. Раціоналізація та організація вхідних та вихідних даних

У цьому кваліфікаційному завданні обробляються дані клієнта, частина з яких надається користувачем під час реєстрації (наприклад, ім'я, вік, стать), а частина автоматично зберігається в системі (наприклад, дані операційної системи, дані браузера). Для підрахунку балів потрібно якомога більше навчальних даних для навчання системи машинного навчання та тестових даних для виставлення балів.

Існує дві лінії продажів: спочатку студент надсилає навчальну анкету, і протягом п'яти хвилин менеджер зв'язується з клієнтом і намагається продати конкретний продукт. Перед дзвінком менеджер закріплює цього клієнта за собою на кілька тижнів, за цей час він успішно конвертує клієнта в студента. Якщо за цей час користувач відмовляється від придбання послуги, його переводять на другу лінію продажів. Тепер перейдемо до основної мети цього розрахунку, а саме оптимізації роботи менеджера другої лінії продажів.

Для формування навчальних даних формується вибірка клієнтів за останній річний період роботи компанії. Наприклад, нам потрібно розрахувати оцінки опитувань за березень 2021 року. Для цього за допомогою SQL-запиту створюється вибірка з березня 2020 року по лютий 2021 року. Таким чином створюється навчальна база. За допомогою аналогічного SQL-запиту створюються місяці, які потрібно обчислити, і період вибірки змінюється на потрібний період. Таким чином створюється тестова база даних. Результати експортуються у форматі ".csv".

Після виконання машинних розрахунків будується графік ROC-кривих. Результати оцінювання створюються в окремому файлі у форматі ".csv".

2.6. Розроблений опис програмного продукту

Щоб скористатися програмою, просто запустіть додаток. Після цього програма вже буде запущена. Якщо мінімальні технічні характеристики пристрою, на якому запускається програма, не збігаються, користувач не зможе коректно використовувати цю програму. Якщо ви хочете створити зразок даних і використовувати його в подальшому, вам потрібно дотримуватися інструкцій, наведених у Додатку А.

2.6.1. Задіяні апаратні ресурси

При розробці та тестуванні системи була використана клієнтська персональна ЕОМ з наступними мінімальними характеристиками:

- процесор AMD Athlon(tm) II X3 2.90 GHz;
- монітор;
- не менше 4000Мб ОЗУ;
- 100Мб вільного місяця для тренувальних та тестових даних та самого додатку;
- клавіатура;
- комп’ютерна миша.

2.6.2. Використані програмні ресурси

Для своєї інформаційної системи я обрав середовище Google Cloud.

Google Cloud - це набір загальнодоступних хмарних обчислювальних сервісів, що надаються компанією Google. Платформа включає в себе набір хостингових сервісів для обчислень, зберігання даних і розробки додатків, що працюють на обладнанні Google. Розробники програмного забезпечення, хмарні адміністратори та інші корпоративні IT-спеціалісти можуть отримати доступ до

сервісів Google Cloud через загальнодоступний Інтернет або приватне мережеве з'єднання.

Google Cloud пропонує такі послуги, як обчислення, зберігання, мережеві технології, великі дані, машинне навчання та Інтернет речей, а також управління хмарою, безпеку та інструменти для розробників (рис. 2.12):

Google Compute Engine - це інфраструктура як послуга (IaaS), яка надає користувачам екземпляри віртуальних машин для розміщення робочих навантажень.

Google App Engine - це PaaS (платформа як послуга), яка надає розробникам програмного забезпечення доступ до масштабованих послуг хостингу Google. Розробники також можуть використовувати SDK для розробки програмних продуктів, які працюють на App Engine.

Google Cloud Storage - хмарна платформа для зберігання великих неструктурованих даних; Google пропонує Cloud Datastore для нереляційного NoSQL-сховища, Cloud Datastore для повністю реляційного Cloud SQL-сховища та власну базу даних Google Cloud Bigtable, серед інших варіантів зберігання баз даних.

Google Kubernetes Engine (GKE) - це система управління та оркестрування контейнерів Docker і кластерів контейнерів, що працюють на публічних хмарних сервісах Google Kubernetes Engine, заснована на системі управління контейнерами Google з відкритим вихідним кодом Kubernetes.

Google Cloud Operations Suite (раніше Stackdriver) - набір інтегрованих інструментів для моніторингу, ведення журналів і звітів про керовані сервіси, що працюють з додатками та системами в Google Cloud.

Cloud Functions для створення функцій для обробки хмарних подій, Cloud Run для управління та запуску контейнерних додатків, а також Workflows для організації безсерверних продуктів і функцій API для запуску робочих навантажень, керованих подіями. Безсерверні обчислення надають інструменти та послуги.

Бази даних - набір продуктів для роботи з базами даних, що надаються як

повністю керовані сервіси, включаючи Cloud Bigtable для великих робочих навантажень з низькою затримкою, Camera для документів, CloudSpanner - масштабовану та надійну реляційну базу даних, а також CloudSQL - повністю керовану базу даних для MySQL, PostgreSQL і SQL Server.

Google Cloud пропонує послуги з розробки та інтеграції додатків. Наприклад, Google Cloud Pub/Sub - це керована служба обміну повідомленнями в режимі реального часу, яка дозволяє обмінюватися повідомленнями між додатками. Google Cloud Endpoints також дозволяє розробникам створювати сервіси на основі RESTful API і робити ці сервіси доступними для клієнтів Apple iOS, Android і JavaScript. Інші пропозиції включають DNS-сервери Anycast, пряме підключення до мережі, балансування навантаження, послуги моніторингу та реєстрацію подій на веб-сайті [45].

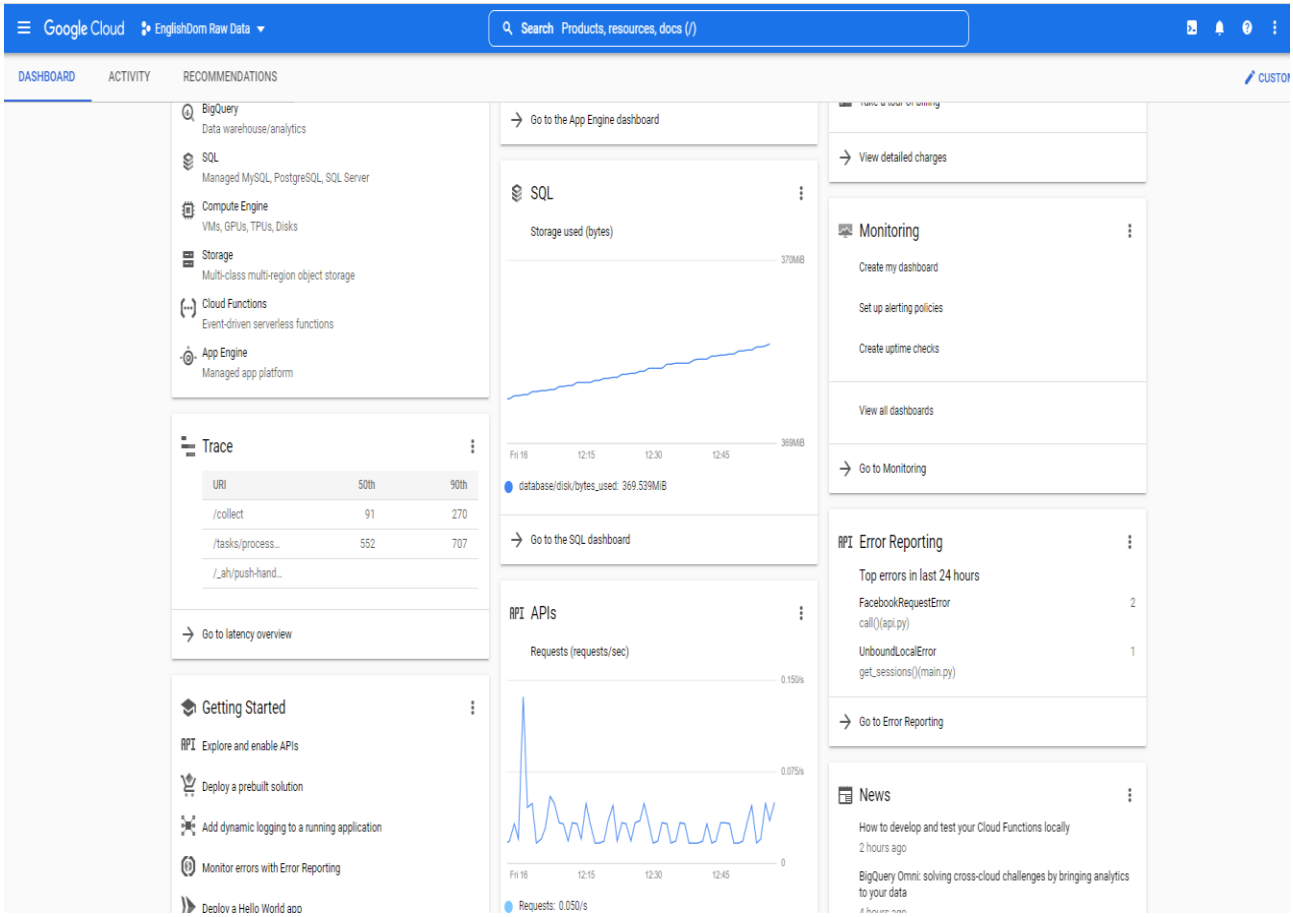


Рис. 2.12. Головна сторінка застосунку Google Cloud Platform

У якості середовища програмування я обрав ПЗ Geany. Geany (рис. 2.13.) - середовище розробки програмного забезпечення, написана з використанням бібліотеки GTK +. Доступна для наступних операційних систем: BSD, Linux, Mac OS X, Solaris і Windows. Geany поширюється згідно GNU General Public License.

Geany не включає до свого складу компілятор. Для створення виконуваного коду використовується GNU Compiler Collection або, при необхідності, будь-який інший компілятор. Сучасні IDE дуже важкі і зовсім незручні для розробки простих консольних додатків, скриптів, верстки, саме тому я обрав це середовище.

Основні функції програмного середовища Geany:

- підсвічування вихідного коду з урахуванням синтаксису мови програмування (мова визначається автоматично по розширенню файлу);
- автозавершення слів;
- автоматична підстановка закривають тегів HTML / XML. Автопідстановка стандартних і існуючих у відкритих файлах функцій;
- простий менеджер проектів;
- підтримка плагінів;
- вбудований емулятор терміналу;
- підтримка великої кількості кодувань;
- гнучкий інтерфейс;
- можливість використання і створення фрагментів. Для цього використовується спеціальний файл snippets.conf в каталозі /home/user/.config/geany дозволяє створювати свої сніппети;
- можливість використання і створення шаблонів файлів. Шаблони повинні бути розташовані в каталозі /home/user/.config/geany/templates/files;

– налагодження коду за допомогою модуля (плагіну) GeanyGDB (використовує відладчик GDB).

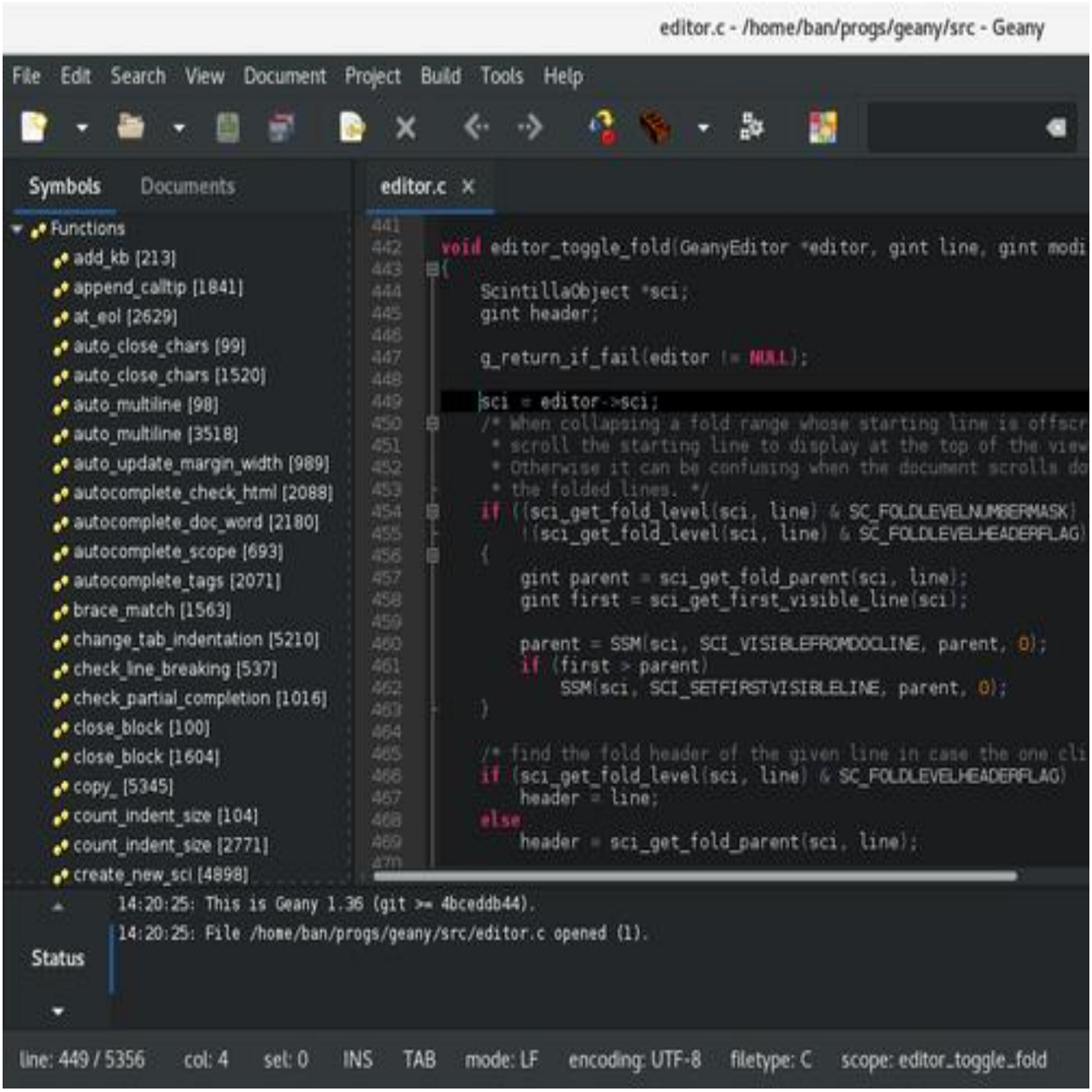


Рис. 2.13. Інтерфейс Geany

2.6.3. Відкриття та завантаження програми

Для роботи з програмою, досить запустити застосунок. Обрати файли з тренувальними й тестовими даними та натиснути на кнопку Submit.

РОЗДІЛ 3
ОПИС РОЗРОБЛЕНОЇ СИСТЕМИ

3.1. Опис інтерфейсу користувача

Після запуску додатку з’явиться віконце інтерфейсу (рис. 2.14.). Необхідно обрати файли для тестової та тренувальної бази розрахунків, увести значення коефіцієнтів там показник ітерації. Після чого натискаємо на кнопку Submit, або Cancel якщо необхідно завершити роботу програми.

Рис. 2.14. Інтерфейс скорингу

Після натиску кнопки Submit починаються розрахунки (рис. 2.15.).

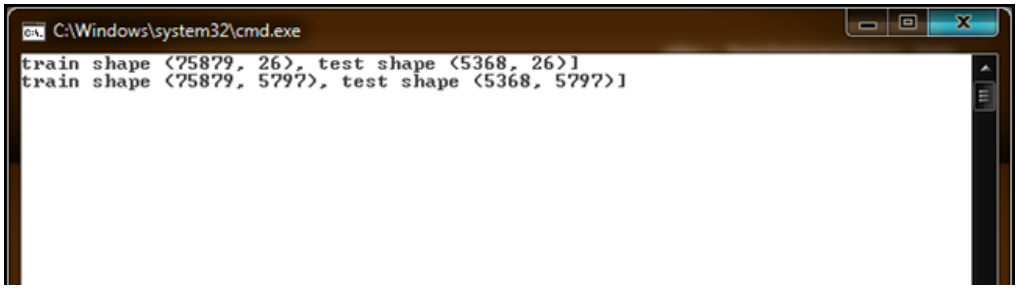


Рис. 2.15. Консольне вікно, яке повідомляє про початок розрахунків

Після виконання першої фази розрахунків з’являється два графіки: крива ROC (рис. 2.16.) та крива (рис. 2.17.), яка показує точку оптимального балансу для розділення бази.

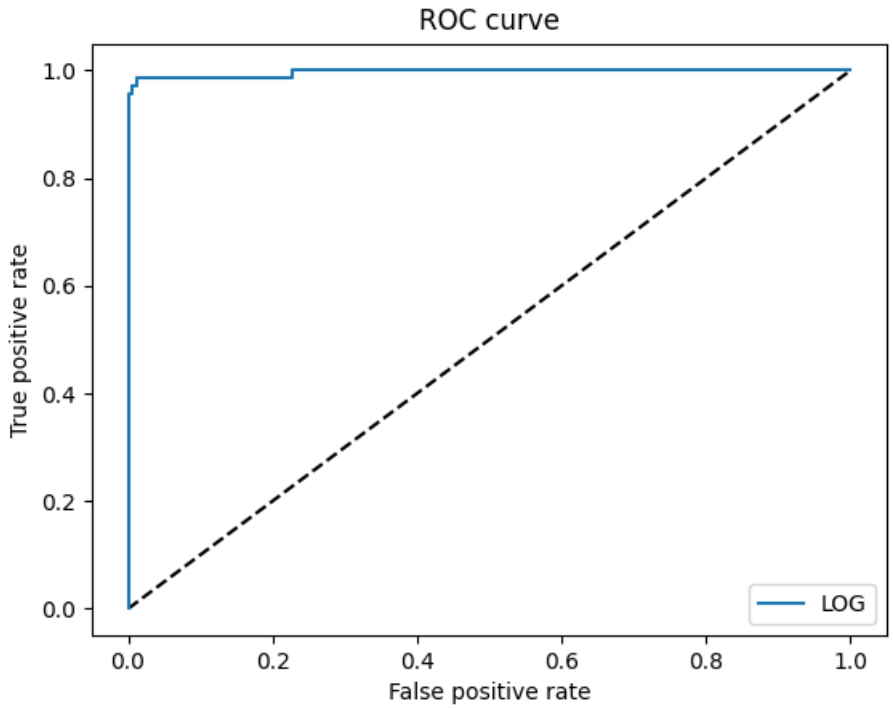


Рис. 2.16. Крива ROC

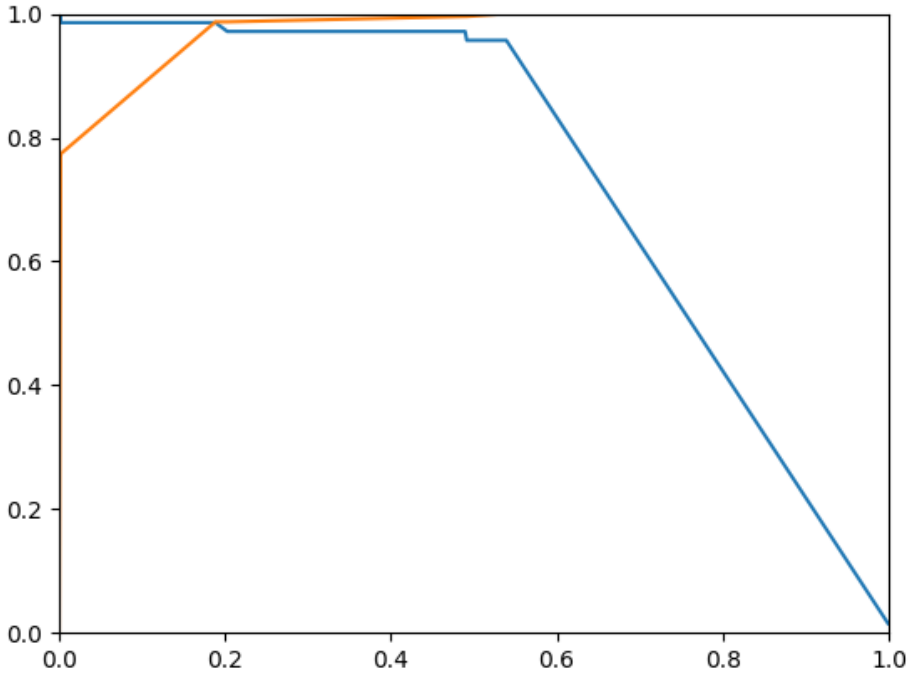


Рис. 2.17. Крива, яка точку оптимального балансу для розділення бази

3.2 Дослідження методів машинного навчання

Ймовірно, людство живе в, можливо, найбільш визначальну епоху в історії людства. Це час, коли обчислення переходять від величезних мейнфреймів до персональних комп'ютерів та хмарних технологій. Але вирішальне значення має не те, що відбулося, а те, що лежить за цим.

Що робить цю епоху такою захоплюючою, так це демократизація різних інструментів і технологій, яка послідувала за розвитком обчислювальної техніки.

Існує три типи алгоритмів машинного навчання: перший - це алгоритм керованого навчання. Цей алгоритм складається з цільової/результативної змінної, яку потрібно передбачити на основі заданого набору предикторів або незалежних змінних. Використовуючи цей набір змінних, створюється функція, яка зіставляє вхідні дані з бажаними результатами. Процес навчання триває доти, доки модель не досягне бажаного рівня точності на навчальних даних. Прикладами керованого навчання є регресія, дерева рішень, випадкові ліси, KNN та логістична регресія.

Другий тип - це алгоритми неконтрольованого навчання. У цьому алгоритмі немає цільової або результуючої змінної, яку потрібно передбачити/оцінити. Він використовується для кластеризації популяцій на різні групи і зазвичай застосовується для розподілу клієнтів на різні групи для певних видів діяльності. Прикладами неконтрольованого навчання є апріорні алгоритми, K-середні.

Третій - навчання з підкріпленням. У цьому алгоритмі машини навчаються приймати певні рішення. Машина постійно перебуває в середовищі, в якому вона навчається методом проб і помилок. Машина вчиться на своєму минулому досвіді і намагається отримати найкращі знання для прийняття правильних бізнес-рішень. Прикладом навчання з підкріпленням є марковське прийняття рішень [46].

Найпоширеніші алгоритми машинного начання:

- лінійна регресія;
- логістична регресія;
- дерево рішень;
- SVM;
- наївний Байєс;
- kNN;
- K-Means;
- випадковий ліс;
- алгоритми зменшення розмірності;
- алгоритми посилення градієнта;
- GBM;
- XGBoost;
- LightGBM;
- CatBoost.

Огляд алгоритмів випадкового лісу, дерева рішень лінійної та логістичної регресій наведено в розділі 2.5. Обґрунтування їх вибору буде наведено у висновках цього розділу.

SVM (Support Vector Machine) це метод класифікації. У цьому алгоритмі малюється кожен елемент даних як точку в n-вимірному просторі (де n – це кількість наявних у вас функцій), причому значення кожної функції є значенням конкретної координати (рис.3.18.)[47].

Наприклад, якщо взяти лише дві характеристики, такі як зріст і довжина волосся особи, спочатку необхідно побудували ці дві змінні у двовимірному просторі, де кожна точка має дві координати (ці координати відомі як опорні вектори).

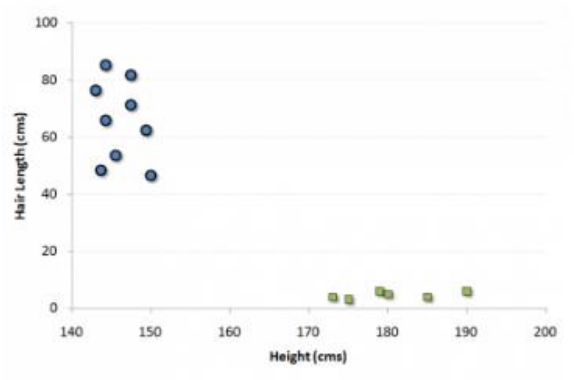


Рис.3.18. графік методу SVM

Тепер знаходяться рядки, які розділяють дані між двома різними класифікованими групами даних. Це буде така лінія, що відстані від найближчої точки в кожній із двох груп будуть найдальшими (рис.3.19.).

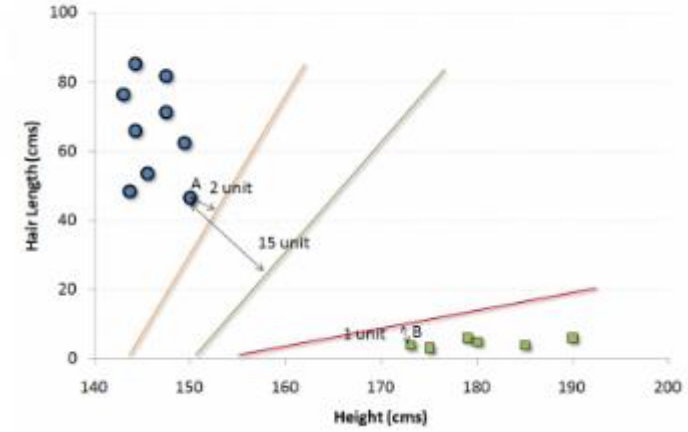


Рис.3.19. Поділ класифікованих груп

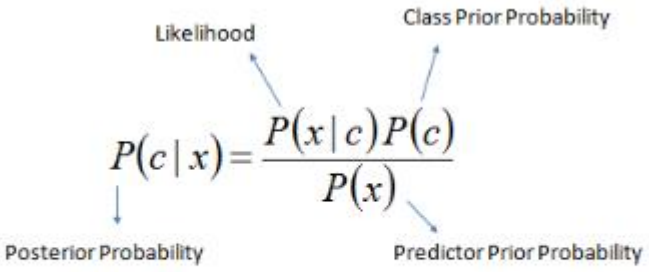
У наведеному вище прикладі лінія, яка розділяє дані на дві різні класифіковані групи, є чорною лінією, оскільки дві найближчі точки є найвіддаленіші від лінії. Цей рядок є класифікатором. Потім, залежно від того, куди по обидві сторони лінії потрапляють дані тестування, можна віднести до цього класу нові дані.

Наївний Байєс - це метод класифікації, заснований на теоремі Байєса, яка передбачає незалежність між предикторами. Простіше кажучи, наївний байєсівський класифікатор припускає, що наявність однієї ознаки в класі не залежить від наявності будь-якої іншої ознаки. Наприклад, фрукт можна вважати

яблуком, якщо він червоний, круглий і має діаметр близько 10 см. Навіть якщо ці ознаки залежать одна від одної або від наявності інших ознак, наївний класифікатор Байєса припускає, що всі ці ознаки незалежно один від одного роблять внесок у ймовірність того, що фрукт є яблуком [48].

Наївні моделі Байєса легко побудувати, і вони особливо корисні для дуже великих наборів даних. Окрім своєї простоти, наївний Байєс, як відомо, перевершує навіть дуже складні методи класифікації.

Теорема Байєса надає спосіб обчислення апостеріорної ймовірності P(c|x) з P(c), P(x) і P(x|c). Подивіться на рівняння нижче:



(3.7)

$$P(c \mid \mathbf{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

P(c|x) — апостеріорна ймовірність класу (цільового) заданого предиктора (атрибуту).

P(c) – апріорна ймовірність класу.

P(x|c) — ймовірність, яка є ймовірністю предиктора даного класу.

P(x) – це пріоритетна ймовірність предиктора.

kNN (k-найближчих сусідів) - це універсальний алгоритм, який можна використовувати як для класифікації, так і для регресії. Однак найчастіше його використовують у задачах промислової класифікації. k Найближчих сусідів - це простий алгоритм, який зберігає всі існуючі випадки і класифікує нові випадки на основі більшості голосів їхніх k найближчих сусідів. Випадки, віднесені до класу, є найпоширенішими серед його k найближчих сусідів, що вимірюється функціями відстані. Ці функції відстані включають Евклідову, Манхеттенську, Мінковського та Хеммінга. Перші три використовуються для неперервних

функцій, а четверта (Хеммінга) - для категоріальних змінних; коли K = 1, випадок просто віднесено до класу найближчого сусіда; вибір K може бути складним у kNN-моделюванні (Рисунок 3.19).

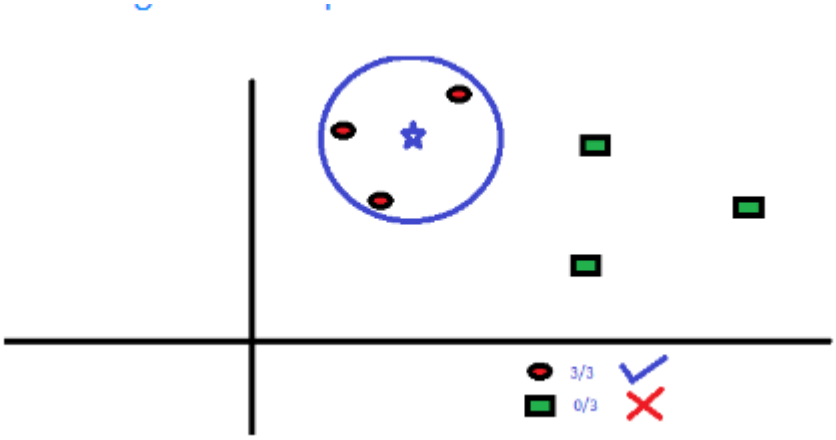


Рис.3.19. Моделювання методу KNN

KNN можна легко зіставити з реальним життям. Якщо необхідно дізнатися про людину, про яку немає інформації, можна дізнатися про її близьких друзів і кола, в яких вона перебуває, і отримати доступ до її/її інформації.

Речі, які слід враховувати перед вибором kNN:

- KNN обчислювально дорогий;
- змінні повинні бути нормалізовані, інакше змінні вищого діапазону можуть змінити його;
- більше працює на етапі попередньої обробки, перш ніж перейти до kNN, як викид, видалення шуму.

K-Means - це тип неконтрольованого алгоритму, який використовується для вирішення завдань кластеризації. Процедура являє собою простий і зрозумілий спосіб класифікації заданого набору даних на певну кількість кластерів (назвемо їх k-кластерами). Точки даних в межах кластера є однорідними і неоднорідними по відношенню до груп подібних типів.

«K» означає щось подібне до активності вилучення форм з чорнильних плям (Рис. 3.20). Дивлячись на форму і розподіл, можна визначити, скільки існує різних кластерів/популяцій.

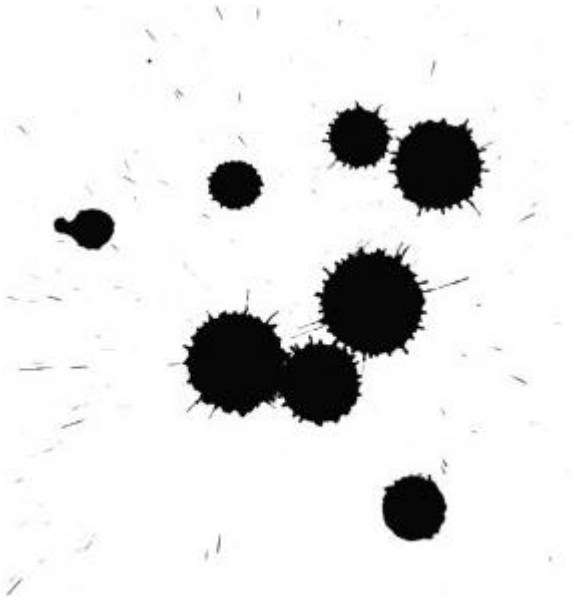


Рис.3.20. Метод неконтрольованого алгоритму

Як K-середнє формує кластер:

- K-середнє вибирає k кількість точок для кожного кластера, відомих як центроїди.
- Кожна точка даних утворює кластер із найближчими центроїдами, тобто k кластерів.
- Знаходить центроїд кожного кластера на основі існуючих членів кластера. Так виникають нові центроїди.
- Оскільки з′являються нові центроїди, необхідно повторити попередні кроки. Необхідно знайдіть найближчу відстань для кожної точки даних від нових центроїдів і зв’язати з новими k-кластерами та повторювати цей процес, доки не відбудеться конвергенція, тобто центроїди не зміняться.

K-середні мають кластери, і кожен кластер має центроїд. Сума квадратів різниць між центроїдом і точками даних у кластері є сумою квадратів цього кластера. Крім того, сума квадратів усіх кластерів підсумовується, щоб отримати

суму квадратів кластерних розв'язків.

Ця величина продовжує зменшуватися зі збільшенням кількості кластерів, але графік результатів показує, що сума квадратів відстаней швидко зменшується до певного значення k, а потім зменшується набагато повільніше (рис. 3.21).

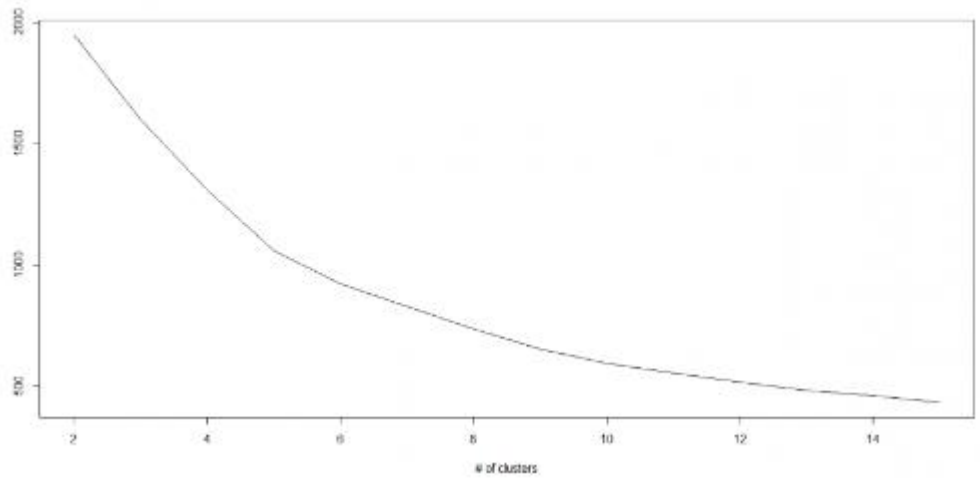


Рис.3.21. Кластерізація значень центроїдів

Алгоритми зменшення розмірності. За останні чотири-п'ять років збір даних на всіх можливих етапах значно збільшився. Компанії, державні установи та дослідницькі інститути не лише вигадують нові джерела інформації, але й збирають детальні дані.

Наприклад, компанії, що займаються електронною комерцією, збирають більш детальну інформацію про своїх клієнтів, таку як демографічні дані, історія веб-перегляду, вподобання, не вподобання, історія покупок та відгуки, щоб забезпечити більш персоналізовану увагу до своїх клієнтів, ніж у найближчому продуктовому магазині.

GBM - це алгоритм бустінгу, який використовується для аналізу великих обсягів даних з метою отримання точних прогнозів. Boosting - це набір алгоритмів навчання, які об'єднують прогнози декількох базових прогнозів, щоб зробити їх більш надійними, ніж один прогноз. Він об'єднує кілька слабких або

середніх прогнозів у сильний прогноз. Ці алгоритми підвищення стабільно демонструють хороші результати на таких змаганнях з науки про дані, як Kaggle, AV Hackathon та CrowdAnalytix.

Іншим класичним алгоритмом є Gradient Boost, відомий як вирішальний вибір для перемоги або поразки в деяких змаганнях Kaggle.

XGBoost має як лінійну модель, так і алгоритм деревовидного навчання і працює приблизно в 10 разів швидше, ніж існуючі методи градієнтного бустингу, що робить його відмінним вибором для дуже високої прогностичної здатності і точності прогнозування подій.

Підвищення включає в себе ряд цільових функцій, включаючи регресію, класифікацію та ранжування.

Найцікавішим аспектом XGBoost є те, що його також називають методом регуляризації. Він допомагає зменшити надмірне моделювання і має широку підтримку ряду мов, включаючи Scala, Java, R, Python, Julia та C++.

Він підтримує розподілене та масштабне багатомашинне навчання на кластерах GCE, AWS, Azure та Yarn. XGBoost також може інтегруватися з Spark, Flink та іншими хмарними системами потокової передачі даних і має вбудовану перехресну перевірку на кожній ітерації процесу бустінгу.

LightGBM - це фреймворк для градієнтного бустингу, який використовує алгоритм навчання на основі дерева. Він розроблений для розгортання та ефективності і пропонує наступні переваги:

- швидша швидкість навчання та вища ефективність;
- менше використання пам'яті;
- краща точність;
- підтримується паралельне навчання та навчання GPU;
- здатність обробляти великі дані.

Фреймворк є швидкою, високопродуктивною системою градієнтного бустингу, заснованою на алгоритмах дерев рішень, що використовуються для сортування, класифікації та багатьох інших задач машинного навчання. Він був розроблений в рамках проекту Microsoft Distributed Machine Learning Toolkit як

частина проекту Microsoft Distributed Machine Learning Toolkit.

Оскільки LightGBM базується на алгоритмі дерева рішень, він розбиває дерево на найкращі листки, тоді як інші алгоритми бустінгу розбивають дерево за глибиною або рівнем, а не за листям. Таким чином, при вирощуванні на одному листі в LightGBM алгоритм листків може зменшити втрати більше, ніж алгоритм рівнів, що призводить до набагато кращої точності, яка рідко досягається існуючими алгоритмами бустінгу [49-54].

Для виконання даної кваліфікаційної роботи було обрано алгоритм випадкового лісу, лінійні та логістичні регресійні дерева рішень. Для виконання завдання необхідно використовувати цільові/результативні змінні, які потрібно передбачити на основі заданого набору предикторів (незалежних змінних). Основною характеристикою, що генерується, є дані користувача, яким присвоюються певні коефіцієнти, і цей набір змінних використовується для створення функції, яка відображає вхідні дані на бажаний результат. Процес навчання триває до тих пір, поки модель не досягне бажаного рівня точності навчальних даних.

ВИСНОВКИ

У даній кваліфікаційній роботі було розроблено додаток для аналізу даних користувача та побудови скорингових моделей у СУБД.

Програмне забезпечення розроблено для аналізу великих обсягів даних користувачів і генерації оцінок окремо для кожного користувача вибірки за певний періоду часу. Додаток надає можливість створювати скорингові моделі для оцінки якості аналізів. На основі цієї оцінки можна зрозуміти та оцінити точність введених даних, успішність вибору вагового коефіцієнта або частоту повторення.

На практиці такий аналіз даних може оптимізувати роботу колл-центрів. Це тому, що людські ресурси дуже обмежені. Ця програма дозволяє призначати оцінки балів кожному користувачеві окремо. Таке рішення дозволяє менеджерам працювати ефективніше та охоплювати аудиторію онлайн-школи.

Під час виконання даної сертифікаційної роботи виконувалися такі завдання:

- аналізується предметна область розв'язуваної проблеми.
- Уточнено вимоги та обґрунтування розробки цієї інформаційної системи.
- Підібрано розумну структуру та параметри програми.
- Створено реляційну базу даних.
- Створено інтерфейс для аналізу даних користувачів.
- Дано рекомендації щодо використання програми.
- Аналізуються існуючі методи машинного навчання та їх переваги.

Базу даних створено мовою SQL в середовищі Google Cloud Platrofm. Механізм аналізу даних реалізовано за допомогою мови програмування Python, а графічний інтерфейс програми створено за допомогою бібліотеки PySimpleGUI Python.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jeff Proise. Applied Machine Learning and AI for Engineers: Solve Business Problems That Can't Be Solved Algorithmically, 1st Edition. - O'Reilly Media, 2022. – 422c.
2. Laurence Moroney. AI and Machine Learning for Coders: A Programmer's Guide to Artificial Intelligence, 1st Edition. - O'Reilly Media, 2020. - 392 c.
3. Chip Huyen. Designing Machine Learning Systems: An Iterative Process for Production-Ready Applications, 1st Edition. - O'Reilly Media, 2022. - 386 c.
4. Stuart J. Russel, Peter Norving. Artificial Intelligence: A Modern Approach, 1st Edition. - Pearson Education India, 2015. – 1164c.
5. Perry Xiao. Artificial Intelligence Programming with Python: From Zero to Hero, 1st Edition. - Wiley, 2022. – 720c.
6. Joshua Gans, Avi Goldfarb. Prediction Machines, Updated and Expanded: The Simple Economics of Artificial Intelligence, 1st Edition. - Harvard Business Review Press, 2022. – 304c.
7. Eric Matthes. Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to Programming, 2nd Edition. - No Starch Press, 2019. – 544c.
8. Mark Lutz. Learning Python, 5th Edition. - O'Reilly Media, 2013. – 1643c.
9. Wes McKinney. Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter, 3rd Edition. - O'Reilly Media, 2022. – 579c.
10. John Canning. Data Structures & Algorithms in Python (Developer's Library), 1st Edition. - Addison-Wesley Professional, 2022. – 928c.
11. Gordon S. Linoff. Data Analysis Using SQL and Excel, 2nd Edition. - Wiley, 2015. – 800c.
12. Walter Shields. SQL QuickStart Guide: The Simplified Beginner's Guide to Managing, Analyzing, and Manipulating Data With SQL, 1st Edition. - ClydeBank Media LLC, 2019. – 249c.
13. Cathy Tanimura. SQL for Data Analysis: Advanced Techniques for Transforming Data into Insights, 1st Edition.- O'Reilly Media, 2021. – 360c.

14. Joe Reis, Matt Housley. Fundamentals of Data Engineering: Plan and Build Robust Data Systems, 1st Edition. - O'Reilly Media, 2022. – 446c.
15. George Mount. Advancing into Analytics: From Excel to Python and R, 1st Edition. - O'Reilly Media, 2021. – 250c.
16. Paul McFedries. Excel Data Analysis For Dummies (For Dummies (Computer/Tech)), 5th Edition. - For Dummies, 2022. – 368c.
17. Galit Shmueli. Machine Learning for Business Analytics: Concepts, Techniques, and Applications with Analytic Solver Data Mining, 4th Edition.- Wiley, 2019. – 608c.
18. Christina Inge. Marketing Metrics: Leverage Analytics and Data to Optimize Marketing Strategies, 1st Edition. - Kogan Page, 2022. – 336c.
19. Alan Beaulieu. Learning SQL: Generate, Manipulate, and Retrieve Data, 3rd Edition. - O'Reilly Media, 2020. – 377c.
20. Chad Knowels. SQL for Data Analytics: Perform efficient and fast data analysis with the power of SQL, Kindle Edition - O'Reilly Media, 2022. - 95p.
21. Alice Zhao. SQL Pocket Guide: A Guide to SQL Usage, 4th Edition. - O'Reilly Media, 2021. – 356c.
22. Robert de Graaf. SQL Cookbook: Query Solutions and Techniques for All SQL Users, 2nd Edition. - O'Reilly Media, 2020. – 570c.
23. Renee M. Teate. SQL for Data Scientists: A Beginner's Guide for Building Datasets for Analysis, 1st Edition. - Wiley, 2021. – 288c.
24. Valliappa Lakshmanan. Data Science on the Google Cloud Platform: Implementing End-to-End Real-Time Data Pipelines: From Ingest to Machine Learning, 2nd Edition. - O'Reilly Media, 2022. – 458c.
25. Adney Ainsley. Google Cloud: GCP: Google Cloud Platform: Learn Google Cloud Platform from the Scratch: The Ultimate Guide for Beginners, 1st Edition. - Independently published, 2020. -83c.
26. Micheal Lahman. Practical AI on the Google Cloud Platform: Utilizing Google's State-of-the-Art AI Cloud Services, 1st Edition. - O'Reilly Media, 2022. – 473c.

27. Vitthal Srinivasan, Janani Ravi. Google Cloud Platform for Architects: Design and manage powerful cloud solutions, 1st Edition. - Packt Publishing, 2018. – 374c.
28. JJ Geewax. Google Cloud Platform in Action, 1st Edition. - Manning, 2018. – 632c.
29. Mark Mucchetti. BigQuery for Data Warehousing: Managed Data Analysis in the Google Cloud, 1st Edition. - Apress, 2020. – 696c.
30. Ekaba Bisong. Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners, 1st Edition. - Apress, 2019. – 855c.
31. Valliappa Lakshmanan, Jordan Tigani. Google BigQuery: The Definitive Guide: Data Warehousing, Analytics, and Machine Learning at Scale, 1st Edition. - O'Reilly Media, 2019. – 522c.
32. Thirukkumaran Haridass, Eric Brown. Learning Google BigQuery: A beginner's guide to mining massive datasets through interactive analysis, 1st Edition. - Packt Publishing, 2017. – 264c.
33. Jordan Tigani, Siddartha Naidu. Google BigQuery Analytics, 1st Edition. - Wiley, 2014. – 492c.
34. Gerardus Blokdyk. Google BigQuery Standard Requirements, Kindle Edition. - 5STARCooks, 2018. – 304c.
35. Mark Mucchetti. BigQuery for Data Warehousing: Managed Data Analysis in the Google Cloud, 1st Edition. - Apress, 2020. – 696c.
36. Daniel Y. Chen. Pandas for Everyone: Python Data Analysis (Addison-Wesley Data & Analytics Series), 1st Edition. - Addison-Wesley Professional, 2017. – 406c.
37. Matt Harrison. Learning the Pandas Library: Python Tools for Data Munging, Analysis, and Visual, 1st Edition. - CreateSpace Independent Publishing Platform, 2016. – 212c.
38. Matt Harrison. Effective Pandas: Patterns for Data Manipulation, 1st Edition. - Independently published, 2021. – 497c.

39. Stefanie Molin. Hands-On Data Analysis with Pandas: A Python data science handbook for data collection, wrangling, analysis, and visualization, 2nd Edition. - Packt Publishing, 2021. – 788c.
40. Matt Harrison, Ted Petrou. Pandas 1.x Cookbook: Practical recipes for scientific computing, time series analysis, and exploratory data analysis using Python, 2nd Edition. - Packt Publishing, 2020. – 626c.
41. Aurelien Geron. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 1st Edition. - O'Reilly Media, 2017. – 574c.
42. Noah Gift. Practical MLOps: Operationalizing Machine Learning Models, 1st Edition. - O'Reilly Media, 2021. – 548c.
43. Francois Chollet. Deep Learning with Python, Second Edition, 2nd Edition. - Manning, 2021. – 504c.
44. Nicolas P Rougier. Scientific Visualization: Python + Matplotlib, 1st Edition. - AFNIL, 2021. – 247c.
45. Kevin P. Murphy. Probabilistic Machine Learning: An Introduction (Adaptive Computation and Machine Learning series), 1st Edition. - The MIT Press, 2022. – 864c.
46. Saurav Singla. Machine Learning for Finance: Beginner's guide to explore machine learning in banking and finance (English Edition), 1st Edition. - BPB Publications, 2020. – 256c.
47. Perez. Machine learning with MATLAB. kNN, CLUSTER analysis and pattern recognition with neural networks, 1st Edition. - Scientific Books, 2022. - 209p.
48. Cesar Perez Lopez. Data mining and machine learning: cluster analysis and kNN classifiers. Examples with MATLAB, 1st Edition. - Lulu.com, 2021. – 251c.
49. Ken Richards. Your Comprehensive Guide for Markov Models, Reinforced Learning, Model Evaluation, SVM, Naives Bayes Classifier: Machine Learning: For Beginners, Book 3. - Ken Richards, 2018. – 231c.

50. Steven Deleon. Machine Learning Naive Bayes Base Belong To Us, 1st Edition. - Independently published, 2020. – 114c.
51. David Forsyth. Applied Machine Learning, 1st Edition. - Springer, 2020. – 515c.
52. Mehryar Mohri. Foundations of Machine Learning, 2nd Edition. - The MIT Press, 2018. – 504c.
53. Mark Stamp. Introduction to Machine Learning with Applications in Information Security, 1st Edition. - Chapman and Hall/CRC, 2017. – 346c.
54. Henrik Brink. Real-World Machine Learning. - Audiobook, 2018. - 7 годин.

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

```
CREATE TABLE `mydb`.`teachers` (
  `teacher_id` INT(10) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  `list_of_languages` VARCHAR(255) NOT NULL,
  `nationality` VARCHAR(100) NOT NULL,
  `is_native` TINYINT(1) NOT NULL,
  `user_id` INT(10) NOT NULL,
  `teacher_level` ENUM('junior', 'middle', 'senior') NOT NULL,
  `gender` ENUM('male', 'female') NOT NULL,
  `age` TINYINT(1) NOT NULL,

  PRIMARY KEY (`teacher_id`));

ALTER TABLE `mydb`.`users`
ADD INDEX `to_users` (`teacher_id` ASC);

ALTER TABLE `mydb`.`users`
ADD CONSTRAINT `to_users`
  FOREIGN KEY (`teacher_id`)
  REFERENCES `mydb`.`teachers` (`teacher_id`)
  ON DELETE CASCADE
  ON UPDATE CASCADE;

  ALTER TABLE `mydb`.`skype_lesson_interview`
ADD INDEX `to_skype_lessons` (`teacher_id` ASC);
ALTER TABLE `mydb`.`skype_lesson_interview`
ADD CONSTRAINT `to_skype_lessons_interview`
  FOREIGN KEY (`teacher_id`)
  REFERENCES `mydb`.`teachers` (`teacher_user_id`)
  ON DELETE CASCADE
  ON UPDATE CASCADE;
```

```
 ALTER TABLE `mydb`.`skype_lessons`
ADD INDEX `to_skype_lessons` (`teacher_id` ASC);
ALTER TABLE `mydb`.`skype_lessons`
ADD CONSTRAINT `to_skype_lessons`
  FOREIGN KEY (`teacher_id`)
  REFERENCES `mydb`.`teachers` (`teacher_user_id`)
  ON DELETE CASCADE
  ON UPDATE CASCADE;


CREATE TABLE `mydb`.`users` (
  `user_id` INT(10) NOT NULL AUTO_INCREMENT,
  `email` VARCHAR(255) NOT NULL,
  `first_name` VARCHAR(255) NOT NULL,
   `last_name` VARCHAR(255) NOT NULL,
  `dt_created` DATE NOT NULL,
  `city` VARCHAR(100) NOT NULL,
  `country` VARCHAR(100) NOT NULL,
  `goal` VARCHAR(100) NOT NULL,
 `birthday` date NOT NULL,
 `source_medium` VARCHAR(255) NOT NULL,
 `phone` VARCHAR(100) NOT NULL,
 `skype` VARCHAR(100) NOT NULL,
  `gender` ENUM('male', 'female') NOT NULL,
  `age` TINYINT(1) NOT NULL,

  PRIMARY KEY (`user_id`));

ALTER TABLE `mydb`.`ind_lessons`
ADD INDEX `to_teachers_idx` (`teacher_id` ASC);
;
ALTER TABLE `mydb`.`ind_lessons`
ADD CONSTRAINT `to_teachers`
  FOREIGN KEY (`teacher_id`)
```

```
  REFERENCES `mydb`.`teachers` (`teacher_id`)
  ON DELETE CASCADE
  ON UPDATE CASCADE;

  CREATE TABLE `mydb`.`applications` (
  `application_id` INT(10) NOT NULL AUTO_INCREMENT,

  `first_name` VARCHAR(255) NOT NULL,
   `last_name` VARCHAR(255) NOT NULL,
  `dt_created` DATE NOT NULL,
  `gmt` VARCHAR(10) NOT NULL,
  `phone` VARCHAR(100) NOT NULL,
  `email` VARCHAR(100) NOT NULL,

 `from_page` VARCHAR(255) NOT NULL,
`user_id` INT(10),
 `skype` VARCHAR(100) NOT NULL,


  PRIMARY KEY (`application_id`));

ALTER TABLE `mydb`.`users`
ADD INDEX `to_users_idx` (`user_id` ASC);
;
ALTER TABLE `mydb`.`users`
ADD CONSTRAINT `to_applications`
  FOREIGN KEY (`user_id`)
  REFERENCES `mydb`.`applications` (`user_id`)
  ON DELETE CASCADE
  ON UPDATE CASCADE;

  ALTER TABLE `mydb`.`skype_lessons`
ADD INDEX `to_skype_lessons_idx` (`user_id` ASC);
;
ALTER TABLE `mydb`.`skype_lessons`
```

```
ADD CONSTRAINT `to_applications`
  FOREIGN KEY (`user_id`)
  REFERENCES `mydb`.`applications` (`student_user_id`)
  ON DELETE CASCADE
  ON UPDATE CASCADE;

  ALTER TABLE `mydb`.`skype_lessons_interview`
ADD INDEX `to_skype_lessons_interview` (`user_id` ASC);
;

ALTER TABLE `mydb`.`skype_lessons_interview`
ADD CONSTRAINT `to_applications`
  FOREIGN KEY (`user_id`)
  REFERENCES `mydb`.`applications` (`student_users_id`)
  ON DELETE CASCADE
  ON UPDATE CASCADE;

  CREATE TABLE `mydb`.`skype_lessons` (
  `lesson_id` INT(10) NOT NULL AUTO_INCREMENT,

  `student_user_id` int(10) NOT NULL,
`teacher_user_id` int(10) NOT NULL,
  `dt_created` DATE NOT NULL,
  `teacher_type` ENUM('native', 'ukrainian') NOT NULL,



  PRIMARY KEY (`lesson_id`));

ALTER TABLE `mydb`.`ind_lessons`
ADD INDEX `to_teachers_idx` (`teacher_id` ASC);
;
ALTER TABLE `mydb`.`ind_lessons`
ADD CONSTRAINT `to_teachers`
  FOREIGN KEY (`teacher_id`)
  REFERENCES `mydb`.`teachers` (`teacher_id`)
```

```
  ON DELETE CASCADE
  ON UPDATE CASCADE;

   CREATE TABLE `mydb`.`skype_lesson_interview` (
  `skype_lesson_interview_id` INT(10) NOT NULL AUTO_INCREMENT,

  `student_user_id` int(10) NOT NULL,
`teacher_user_id` int(10) NOT NULL,
  `dt_created` DATE NOT NULL,
  `status` ENUM('cancel', 'complete') NOT NULL,
  `grammar_skills` VARCHAR(100) NOT NULL,
  `speaking_skills` VARCHAR(100) NOT NULL,
  `listening_skills` VARCHAR(100) NOT NULL,
  `recommended_cource` VARCHAR(100) NOT NULL,

  PRIMARY KEY (`skype_lesson_interview_id`));

ALTER TABLE `mydb`.`skype_lesson_interview`
ADD INDEX `to_teachers_idx` (`teacher_user_id` ASC);
;
ALTER TABLE `mydb`.`skype_lesson_interview`
ADD CONSTRAINT `to_teachers_idx`
  FOREIGN KEY (`teacher_users_id`)
  REFERENCES `mydb`.`teachers` (`teacher_id`)
  ON DELETE CASCADE
  ON UPDATE CASCADE;

  ALTER TABLE `mydb`.`skype_lesson_interview`
ADD INDEX `to_users_idx` (`student_user_id` ASC);
;
ALTER TABLE `mydb`.`skype_lesson_interview`
ADD CONSTRAINT `to_users_idx`
  FOREIGN KEY (`student_user_id`)
  REFERENCES `mydb`.`users` (`user_id`)
  ON DELETE CASCADE
```

```
  ON UPDATE CASCADE;

   CREATE TABLE `mydb`.`payment_transaction` (
  `transaction_id` INT(10) NOT NULL AUTO_INCREMENT,

  `amount` int(10) NOT NULL,
`teacher_type` ENUM('native', 'ukrainian') NOT NULL,
 `cost` int(10) NOT NULL,
 `currency` int(10) NOT NULL,
 `user_id` int(10) NOT NULL,
  PRIMARY KEY (`transaction_id`));

ALTER TABLE `mydb`.`users`
ADD INDEX `to_users` (`user_id` ASC);
;
ALTER TABLE `mydb`.`users`
ADD CONSTRAINT `to_pt_idx`
  FOREIGN KEY (`users_id`)
  REFERENCES `mydb`.`payment_transaction` (`user_id`)
  ON DELETE CASCADE
  ON UPDATE CASCADE;


ALTER TABLE `mydb`.`applications `
ADD INDEX `to_applications_idx` (`user_id` ASC);
;
ALTER TABLE `mydb`.`applications`
ADD CONSTRAINT `to_payment_transaction`
  FOREIGN KEY (`user_id`)
  REFERENCES `mydb`.`payment_transaction` (`user_id`)
  ON DELETE CASCADE
  ON UPDATE CASCADE;
```

Запит для виводу вхідних даних у БД:

```
select user_id,#clientid,
ARRAY_AGG(day order by timestamp asc)[offset(0)] as date_appl,
EXTRACT(DAYOFWEEK FROM ARRAY_AGG(day order by timestamp asc)[offset(0)]) as
day_of_week,
EXTRACT(MONTH FROM ARRAY_AGG(day order by timestamp asc)[offset(0)]) as month,
ifnull(ARRAY_AGG(source order by timestamp asc)[offset(0)],'unknown') as source_appl,
ifnull(ARRAY_AGG(medium order by timestamp asc)[offset(0)],'unknown') as medium_appl,
ifnull(if(ARRAY_AGG(campaign order by timestamp
asc)[offset(0)]='','unknown',ARRAY_AGG(campaign order by timestamp
asc)[offset(0)]),'unknown') as campaign_appl,
ifnull(if(ARRAY_AGG(content order by timestamp
asc)[offset(0)]='','unknown',ARRAY_AGG(content order by timestamp asc)[offset(0)]),'unknown')
as content_appl,
ifnull(ARRAY_AGG(if(Page='/','/',if(REGEXP_CONTAINS(Page,'/l/')=false,
REGEXP_EXTRACT(if(ENDS_WITH(Page, "/")=false,concat(Page,'/'),Page), '/[^/]+/' ),
REGEXP_EXTRACT(if(ENDS_WITH(Page, "/")=false,concat(Page,'/'),Page), '/[^/]+/[^/]+/' )))
order by timestamp asc)[offset(0)],'unknown') as Page_appl,

count(user_id) as numb_of_sessions,round(avg(pages_per_session1),2) as
avg_pages_per_session,round(avg(Session_time_seconds1),2) as
avg_session_time_seconds,sum(appl_dist) as total_numb_of_appl,
DATE_DIFF(max(date_s_n),max(day1),DAY) as days_from_last_session,if(max(date_sales) is
null,0,1) as is_student,
DATE_DIFF(max(date_s_n),max(date_created),DAY) as days_from_registration,

ifnull(ARRAY_AGG(source_reg order by date_created asc)[offset(0)],'unknown') as source_reg,
ifnull(ARRAY_AGG(medium_reg order by date_created asc)[offset(0)],'unknown') as
medium_reg,
ifnull(ARRAY_AGG(campaign_reg order by date_created asc)[offset(0)],'unknown') as
campaign_reg

from
(select v.*,ns.date_sales,if(ns.date_sales is null,'2021-02-26',ns.date_sales) as date_s_n,
us.country,SPLIT(us.source_medium," / ")[offset(0)] as source_reg, SPLIT(us.source_medium," /
")[offset(1)] as medium_reg,us.campaign as campaign_reg,us.landing_page as
landing_page_reg,us.date_created
from
(SELECT ap.* except(session_id,Registrations_dist,Applications),
ifnull(
  (
        SELECT
        distinct user_id
        FROM (
        SELECT
        user_id,
        statistic_id,
        COUNT(DISTINCT user_id) OVER (PARTITION BY statistic_id) cnt
        FROM
        `englishdom-raw-data.ga.user_statistics_rel`
        ORDER BY
```

```
        COUNT(DISTINCT user_id) OVER (PARTITION BY statistic_id) DESC)v
        WHERE
        cnt=1
        AND statistic_id=ap.clientid),
        (
        SELECT
        distinct userid
        FROM (
        SELECT
        userid,
        clientid,
        COUNT(DISTINCT userid) OVER (PARTITION BY clientid) cnt
        FROM
        `englishdom-raw-data.ga.all_userid_clientid_final`
        GROUP BY
        userid,
        clientid)v
        WHERE
        cnt=1
        and clientid=ap.clientid)
        ) as user_id,
  v.day as day1,v.channel as channel1,v.source as source1,v.medium as medium1,v.page as
page1,v.page_last as page_last1,v.pages_per_session as pages_per_session1,
v.Session_time_seconds as Session_time_seconds1, v.timestamp as timestamp1, v.applications_dist
as appl_dist
FROM `englishdom-raw-data.ga.ds_users_visits_appl_reg_final_*` ap
left join (SELECT * FROM `englishdom-raw-data.ga.ds_users_visits_appl_reg_final_*` where
_table_suffix>='20190801' and _table_suffix<'20210201')v on ap.clientid=v.clientid

#where regexp_contains(lower(content),'cpa-ua')=true
where ap.Applications_dist=1
and _table_suffix>='20190801'
and _table_suffix<'20210201'
and v.timestamp>=ap.timestamp and v.day>=ap.day)v
left join `englishdom-raw-data.ga.New_students_by_channel_new` ns on v.user_id=ns.user_id
left join `englishdom-raw-data.ga.users_*` us on v.user_id=us.user_id
where v.day1<if(ns.date_sales is null,current_date(),ns.date_sales)
#and v.user_id is not null
order by v.user_id,v.timestamp,v.timestamp1)v
#where user_id is not null
group by user_id,clientid
order by user_id,clientid
```

Interface.py

```
#import libraries
from pandas import read_csv, DataFrame, get_dummies, concat
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
```

```
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.metrics import roc_curve
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC, SVC, SVR
from sklearn.metrics import accuracy_score,f1_score,classification_report,roc_auc_score
#чтение датасетов
train_df=read_csv('train_2019-09-01 - 2021-03-14.csv',index_col=None) #тренувальний датасет
#train_2019-09-01 - 2021-03-14
#train_2019-08-01 - 2021-01-31 BQ_7
#train_df=read_csv('train_2019-09-01 - 2021-03-14',index_col=None) # тренувальний датасет
test_df=read_csv('test_2020-09-01 - 2020-10-01.csv',index_col=None) #тестовий датасет
#test_df=read_csv('test_2020-09-01 - 2020-10-01.csv',index_col=None) #тестовий датасет
result = DataFrame(test_df.user_id) #для запису результата у файл
print('train shape {0}, test shape {1}]'. format(train_df.shape, test_df.shape)) #необхідно для
подальшої контрольної точки
# прибираємо із датасету непотрібні данні(столбці)
#если есть clientid - убрать!!!
trainy = train_df['is_student'] #обираємо бінарне,яке є ключовою ознакою(у даному випадку
stud=1 означає що дана заявка вже сконвертована )
testy = test_df['is_student'] # обираємо бінарне,яке є ключовою ознакою(у даному випадку
stud=1 означає що дана заявка вже сконвертована ),       'campaign_reg'],axis=1)
test_df = test_df.drop(['user_id','is_student','date_appl','month',       'source_reg',    'medium_reg',
        'campaign_reg'],axis=1)
#наступні строки для коректного кодування якісних полів(треин+тест)
mergedata = train_df.append(test_df) #об’єднуємо  тренувальний и тестовий датасети
testcount = len(test_df)
count = len(mergedata)-testcount
#обираємо та кодуємо якісні подя
mergedata_enc = get_dummies(mergedata, prefix_sep='*', columns=['day_of_week',
        'source_appl',  'medium_appl',          'campaign_appl',        'content_appl', 'Page_appl','gmt',
        'segment',       'country',         'gender',          'age_range'])
#mergedata =
mergedata.drop(['hour_req','dayofweek_req','month_req','gmt','from_page_pretty','country','Source','
Medium','from_page_hash'],axis=1)
'''
#нормалізуємо датасет по столбцям типу float
x = mergedata_enc.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
scaled_mergedata_enc = DataFrame(x_scaled, columns = mergedata_enc.columns)
#print(mergedata_enc)
#print(scaled_mergedata_enc)
'''
#розділяємо закодований назад на трейн и тест датасети
#train = scaled_mergedata_enc[:count]
#test = scaled_mergedata_enc[count:]
trainx = mergedata_enc[:count]
testx = mergedata_enc[count:]
print('train shape {0}, test shape {1}]'. format(trainx.shape, testx.shape)) #використовується в
якосі точки контроля для перевірки коректності попередніх кроків
#приводимд закодовані датасеті к стандартному виду
```

```
#trainy = train['is_student'] # обираємо бінарне,яке є ключовою ознакою(у даному випадку stud=1 означає що дана заявка вже сконвертована )
#trainx = train.drop(['is_student'], axis=1) #дропаємо із датасету ключеве поле із минулої строки
#testy = test['is_student'] # обираємо бінарне,яке є ключовою ознакою(у даному випадку stud=1 означає що дана заявка вже сконвертована )
#testx = test.drop(['is_student'], axis=1) # дропаємо із датасету ключеве поле із минулої строки
#разделяємо тренувальний датасет рандомно на два датасета (в соотношении test_size) для тестування алгоритмів і оцінки їх точності (й інших параметрів)
## для уже отлаженной задачи и выбранного алгоритма этот пункт не требуется и тогда, формально, X_train=trainx, y_train=trainy, X_test=testx (test_size=0)
X_train,X_test,y_train,y_test = train_test_split(trainx, trainy, test_size=0.2, stratify=trainy, random_state=43)
#model_log = RandomForestClassifier(n_estimators = 100)
#model_svc = SVC(kernel= 'rbf', random_state=42 , gamma=2, C=50)
model_log=LogisticRegression(solver='liblinear',max_iter=1000, class_weight={0:.1, 1:.9}, C=100) #задаем модель и ее параметры
#model_log=SVC(kernel= 'rbf', random_state=42 , gamma=2, C=50)
#model_log=SVR(C=50)
#model_log=RandomForestClassifier(random_state=43,class_weight='balanced_subsample',n_estimators=100,max_features=None)
#max_depth=100, n_estimators=10, max_features=10
#возможно max_features=None
model_log.fit(X_train, y_train) #навчаємо модель
pred=model_log.predict(X_test) #робимо передбачення на тестовому датасеті з параметрами, які отримали в результаті навчання
#print( scaled:  results (pred, real): ,list(zip(pred,y_test_part))) – якщо хочемо вивести на екран реальне значення шукаємої ключової ознаки і його передбачуване
#вивід на екран метрик: точність, ф-міра, precision, recall
print('scaled: accuracy = {}, f1-score= {}'. format( accuracy_score(y_test,pred), f1_score(y_test,pred, average='macro')))
print('scaled: f1-score_weighted = {}'. format(f1_score(y_test,pred, average='weighted')))
report = classification_report(y_test, pred, target_names=['Non-churned', 'Churned'])
print(report)
y_pred_rf = model_log.predict_proba(X_test)[:, 1]
fpr_log, tpr_log, d = roc_curve(y_test, y_pred_rf)
print(roc_auc_score(y_test, y_pred_rf))
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_log, tpr_log, label='LOG')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.figure(2)
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.plot(d, tpr_log, label='Se (TPR)')
plt.plot(d, 1-fpr_log, label='Sp (1-FPR)')
#plt.legend(loc='Se&Sp')
plt.show()
```

```
pred=model_log.predict(testx) #робимо передбачення на тестовому датасеті з параметрами, які отримали в результаті навчання моделі
#print( scaled:  results (pred, real): ,list(zip(pred,y_test_part))) - якщо хочемо вивести на екран реальне значення шукаємої ключової ознаки і його передбачуване
#вивід на екран метрик: точність, ф-міра, precision, recall
print('scaled: accuracy = {}, f1-score= {}'. format( accuracy_score(testy,pred), f1_score(testy,pred, average='macro')))
print('scaled: f1-score_weighted = {}'. format(f1_score(testy,pred, average='weighted')))
report = classification_report(testy, pred, target_names=['Non-churned', 'Churned'])
print(report)
#виводим вірогідність
pred_probability = model_log.predict_proba(testx)
#записуємо  у файл розраховані дані: передбачувані дані, вірогідність та скорінг
result.insert(1,'Real_status', DataFrame(testy))
result.insert(2,'Predicted_status', DataFrame(pred))
result.insert(3,'Probability_0', DataFrame(pred_probability,columns=['0','1']).iloc[:,0])
result.insert(4,'Probability_1', DataFrame(pred_probability,columns=['0','1']).iloc[:,1])
result.insert(5,'Score', DataFrame(pred_probability,columns=['0','1']).iloc[:,1]*100) #тут скоринг рахується як помноження вірогідності позитивного результату на
result.to_csv('test_result.csv', index=False)
#оголошення коефіцієнтів
#coefs = np.abs(model_log.coef_[0])
inter=model_log.intercept_
print(inter)
coefs = model_log.coef_[0]
indices = np.argsort(coefs)[::-1]
'''
plt.figure()
plt.title(Feature importances (Logistic Regression))
plt.bar(range(15), coefs[indices[:15]],
      color=b, align=center)
numerical_columns = trainx.columns
plt.xticks(range(15), trainx.columns[indices[:15]], rotation=45, ha='right')
plt.subplots_adjust(bottom=0.3)
#plt.tight_layout()
#plt.show()
#rev0=get_dummies(mergedata).idxmax(1)
'''
res_importances = concat([DataFrame(trainx.columns[indices[:]]), DataFrame(coefs[indices[:]])], axis=1)
res_importances.columns=['parameter','importance_value']
res_importances.to_csv('importances_range.csv', index=False)
```

ДОДАТОК Б

СПИСОК ФАЙЛІВ НА ДИСКУ

Ім'я файлу	**Опис**
Пояснювальні документи	
Немченко САДМ-61doc	Пояснювальна записка до проекту. Документ Word.
Немченко САДМ-61.pdf	Пояснювальна записка до проекту у форматі PDF.
Program	
Немченко САДМ-61.zip	Архів. Містить програмні коди.
Presentation	
Немченко САДМ-61.ppt	Презентація проекту.

ДОДАТОК В

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

ОБ'ЄКТ ДОСЛІДЖЕННЯ

Скоринг користувацьких даних – це власне збір, структурування інформації для подальшого аналізу та побудови скорингових моделей.

МЕТА КВАЛІФІКАЦІЙНОЇ РОБОТИ

Розробка програмного забезпечення, яке надасть змогу користувачам аналізувати дані, визначати потенційних клієнтів онлайн-сервісів та оптимізувати роботу менеджерів call-центру.

2

ПОСТАНОВКА ЗАВДАННЯ

Розроблена інформаційна система повинна відповідати наступним функціональним характеристикам та давати змогу її користувачу:

- дати оцінку скорингу кожному клієнту окремо;
- власноруч обирати вагові коефіцієнти та показник ітерації;
- відстежувати якість машинного навчання;
- отримувати результати розрахунків у зручному вигляді;
- інтуїтивно зрозумілий інтерфейс.

3

ЧАСТИНИ РОЗРОБЛЕНОЇ СИСТЕМИ

• База даних – частина системи, яка відповідає за збір та збереження даних.

• Скоринговий додаток – частина системи, яка буде керуватися користувачем, будувати скорингові моделі, графіки моделей та інтерфейс відстеження якості роботи алгоритмів машинного навчання.

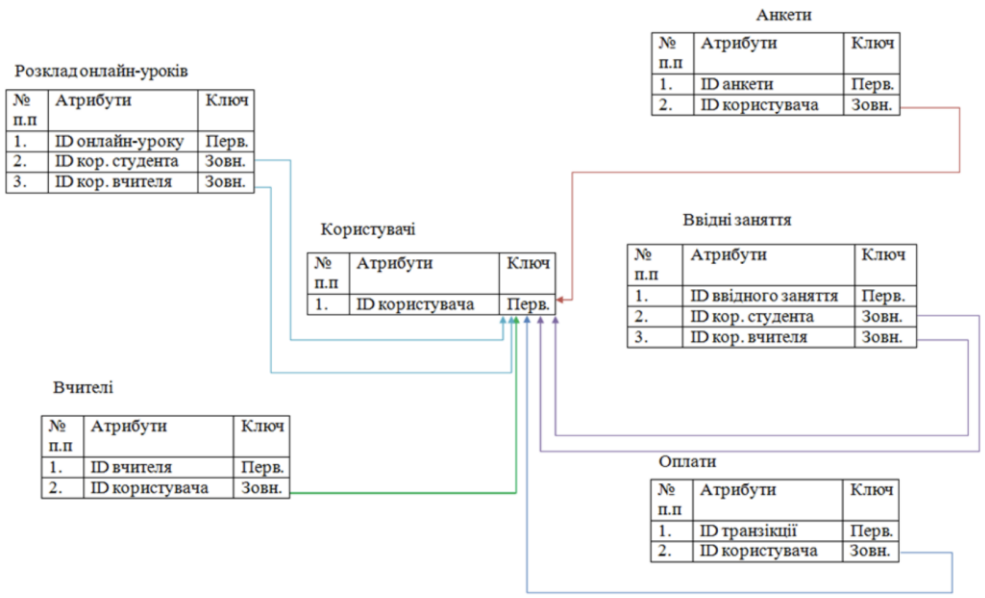
4

ВИКОРИСТАНІ ЗАСОБИ РОЗРОБКИ

• Мова програмування Python;
• мова запитів SQL;
• Python бібліотеки Pandas, Matplotlib, Scikit-learn.
• програмний засіб для реалізації бази даних – Google Cloud Platform;
• Geany – у якості середовища програмування.

5

Реляційна модель бази даних



6

ВИКОРИСТАНІ МЕТОДИ МАШИННОГО НАВЧАННЯ

- RandomForestClassifier – з англійського перекладається, як випадковий «ліс»;
- RandomForestRegressor – з англійського перекладається, як дерево рішень;
- LogisticRegression – логістична регресія.

7

ВИКОРИСТАНІ МЕТРИКИ ОЦІНЮВАННЯ ЯКОСТІ АНАЛІЗУ

- Accuracy;
- precision;
- recall.

Перед переходом до самих метрик необхідно ввести важливу концепцію для опису цих метрик в термінах помилок класифікації - confusion matrix (матриця

	$y = 1$	$y = 0$
$\hat{y} = 1$	True Positive (TP)	False Positive (FP)
$\hat{y} = 0$	False Negative (FN)	True Negative (TN)

8

ФОРМУЛИ РОЗРАХУНКУ МЕТРИК

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

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

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

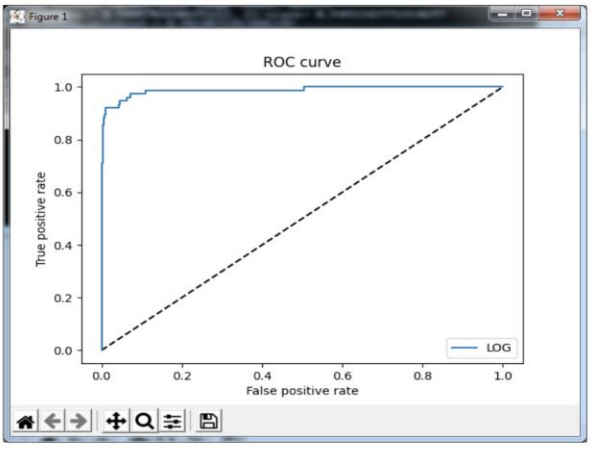
accuracy - частка правильних відповідей алгоритму
Для оцінки якості роботи алгоритму на кожному з класів окремо вводяться метрики precision (точність) і recall (повнота).

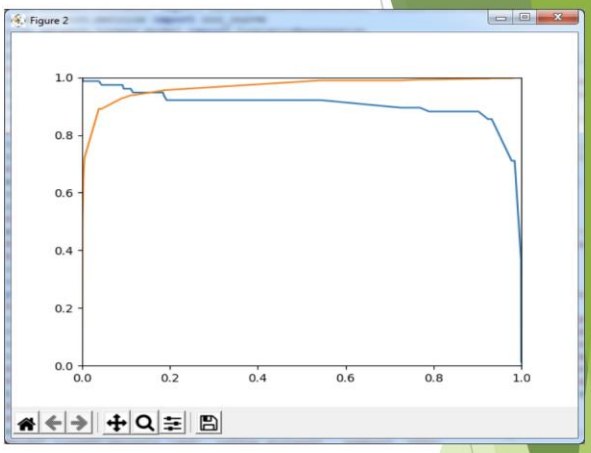
9

ІНТЕРФЕЙС КОРИСТУВАЧА

10

РЕЗУЛЬТАТ РОЗРУХУНКІВ





ROC - крива, яка найбільш часто використовується для представлення результатів бінарної класифікації в машинному навчанні. ROC-крива показує залежність кількості вірно класифікованих позитивних прикладів від кількості невірно класифікованих або негативних.

11

КОНСОЛЬНЕ ВІКНО З МЕТРИКАМИ

```
train shape (73543, 42), test shape (6362, 42)]
train shape (73543, 3899), test shape (6362, 3899)]
scaled: accuracy = 0.9809782608695652, f1-score= 0.9492343771554697
scaled: f1-score_weighted = 0.9810873463806102
              precision    recall  f1-score   support

 Non-churned       0.99      0.99      0.99       660
     Churned       0.90      0.92      0.91        76

    accuracy                           0.98       736
   macro avg       0.94      0.95      0.95       736
weighted avg       0.98      0.98      0.98       736

0.9875398724082933
scaled: accuracy = 0.911191449229802, f1-score= 0.8111013662756767
scaled: f1-score_weighted = 0.9208957133728385
              precision    recall  f1-score   support

 Non-churned       0.99      0.91      0.95      5721
     Churned       0.53      0.91      0.67       641

    accuracy                           0.91      6362
   macro avg       0.76      0.91      0.81      6362
weighted avg       0.94      0.91      0.92      6362

0.0


------------------
(program exited with code: 0)

Для продолжения нажмите любую клавишу . . .
```

12

ФРАГМЕНТ ВИБІРКИ З ПРОГНОЗОВАНИМ СТАТУСОМ ТА СКОРИНГОМ

user_id	Real_status	Predicted_status	Score
10588	0	0	0.07195677290336625
10681	1	1	88.55960214041153
11840	0	0	0.5118929618441792
13065	0	0	0.3019507514780663
13230	0	0	0.09450314665164486
16525	0	0	0.030407025643715585
16525	0	0	1.2049632368896432
43060	0	0	6.820476291762572
43889	0	0	0.6385425883410858
49727	0	0	0.33676827537571574
52875	1	1	99.06285263390345
60483	0	0	0.0049928228311410805
62077	0	0	0.0009505620332155751
63268	0	0	0.05345518008808002
66021	1	1	99.61097035739328
68571	1	1	72.07719772902826
71465	0	0	0.18598504986498893
72487	1	1	99.72976568507052
72628	0	0	0.0007798301878908192
77961	0	0	0.06187830122337555
82884	0	0	19.226937902586023
105545	0	0	0.04024157590915525
117894	0	0	0.0067434276158991
122568	0	0	0.006463013969089198
122766	0	0	2.5753513476870107
124313	0	0	0.022067900520153078
129476	1	1	99.30579179784101
131002	0	0	0.0055816064020417075
136697	0	0	0.07770134210122213

13

ВИСНОВКИ

У кваліфікаційній роботі виконані наступні основні задачі:

- Описано вибір платформи та архітектури для розробки.
- Виконано проектування і розробка інформаційної системи.
- Наведено опис структури функціонування додатку.
- Визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів.
- Наведені виклик та завантаження системи.
- Продемонстрована робота програми.
- Проведено порівняння систем-аналогів та інших методів машинного навчання

14