Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

магістра
(назва освітньо-кваліфікаційного рівня)

студента	*Немченка Максим Ігоровича*
	(ПІБ)
академічної групи	*122М-21-2*
	(шифр)
спеціальності	*122 Комп'ютерні науки*
	(код і назва спеціальності)
освітньої програми	*«122 Комп'ютерні науки»*
	(назва освітньої програми)
на тему:	*Дослідження методів машинного навчання по підвищенню ефективності оптимізації скорингових моделей*

М.І. Немченко

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
розділів кваліфікаційної роботи				
спеціальний	*проф. Мещеряков Л.І.*			

Рецензент				

Нормоконтролер	*проф. Лактіонов І.С*			

Дніпро
2022

л

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)

_____ М. О. Алєксеєв
(підпис) (прізвище, ініціали)

« » __ 20 22 року

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності _____ *122 Комп'ютерні науки*
(код і назва спеціальності)

студенту *122м-21-2* *Немченку Максиму Ігоровичу*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Дослідження методів машинного навчання по підвищенню ефективності оптимізації скорингових моделей*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 31.10.2022 р. № 1200 -с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – побудова скорингових моделей з розрахуванням математичних показників churned, f-score, precision і recall. Відслідковування залежності точності прогнозу від вагових коефіцієнтів характеристик користувачів.

Предмет досліджень – методи машинного навчання, перевірка їх переваг та недоліків, відслідковування ефективності їх роботи у предикативних моделях скорингу.

Мета НДР – створити додаток з опрацювання користувацьких даних, який дозволить швидко будувати математичні моделі з максимальною точністю та мінімальними затратами в апаратному забезпеченні користувача додатку.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Новизна запропонованих рішень полягає у знаходженні ефективних методів машинного навчання, а також оптимізація процесу побудови скорингових моделей за рахунок предиктивної моделі на основі використання алгоритмів лінійної та логістичної регресій та алгоритму випадкового лісі. Наведені алгоритми дозволяють економно використовувати ресурс оперативної пам′яті та швидко отримувати необхідний результат розрахунків.

л

Практична цінність полягає у тому, що побудований додаток дозволить зручніше аналізувати та будувати математичні моделі, створить умови для обробки користувацьких даних та оптимізації людської роботи.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результатом дослідження буде виявити найоптимальніші алгоритми машинного навчання для завдання кваліфікаційної роботи. З практичної точки зору, створений додаток має бути з простим, зрозумілим інтерфейсом та швидким часом виконання розрахунків та побудови моделей.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз теми та постановка задачі	12.09.2022-30.09.2022
Побудова моделі та алгоритмів навчання на даних датасету	01.10.2022-31.10.2022
Створення функціональності машинного навчання	01.11.2022-16.12.2022

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект - дозволить збільшити конверсії з продажів.

Соціальний ефект – дозволить оптимізувати роботу персоналу.

7 ДОДАТКОВІ ВИМОГИ

Завдання видав _____ *Мещеряков Л.І.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Немченко М.І.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *12.09.2022 р.*

Термін подання кваліфікаційної роботи до ЕК *12.12.2022*

л

РЕФЕРАТ

Пояснювальна записка: 83 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 уточняється 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.

л

ABSTRACT

Explanatory note: 83 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.

л

ЗМІСТ

л

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

л

ВСТУП

Завдання даної кваліфікаційної роботи та об’єкт його діяльності безпосередньо пов’язані та відповідає узагальненій тематиці кваліфікаційних робіт і переліку зазначених виробничих функцій, типових задач діяльності, умінню, знанням та компетенціям, якими повинні володіти магістри спеціальності 122 «Комп’ютерні науки».

На сьогоднішній день широка увага приділяється різним онлайн-сервісам, магазинам, онлайн школам навчання, дистанційним курсам освіти або курсам підвищенню кваліфікації. Перед сучасними бізнесами виникає велика необхідність у зборі, зберіганні та обробці користувацьких даних. Не вистачає лише зберегти дані у базі, сучасність вимагає вміння їх правильно використовувати. На основі даних будуються стратегії та плани на розвиток бізнесу, досліджується ефективність роботи персоналу, складається портрет та звички користувачів. Враховуючи всі вище перераховані фактори, тематикою кваліфікаційної роботи є розробка додатку для аналітики користувацьких даних та побудови скорингових моделей у СКБД.

Метою кваліфікаційної роботи є вивчення засобів створення та роботи з базою даних та засобів роботи з ними. Такі навички будуть доволі актуальними в сфері розробки програмного забезпечення для контролю за навчанням студентів, моніторингу продуктивності та якості занять, а також створення скорингових моделей для прогнозування продажів чи ефективності рекламних кампаній. Також вони нададуть кращого розуміння механізмів роботи з сутностями бази даних, взаємодії з ними, як ефективно та оптимально використовувати машинне навчання.

Дана робота знайомить з методами опрацювання користувацьких даних, їх зберігання, обробка та подальша взаємодії з ними. Будуть наведені приклади різних методів та підходів до машинного навчання та обґрунтування вибору СУБД, алгоритмів побудови скорингових моделей і графіків.

Основними вимогами до системи будуть: доступні для розуміння методи

л

роботи з даними, надійна та стабільна робота системи, точність розрахунків.

Новизна розробки, полягає у тому, що розроблений додаток збільшить ефективність роботи менеджерів з продажів, завдяки математичному підходу. Користувачі будуть поділені у сегментні групи, на основі їх користувацьких даних, побажань та патернів поведінки. Виходячи з цього, можна використовувати різні стратегії продажів та інших методів взаємодії.

л

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ПРОБЛЕМИ

1.1. Загальна інформація про предметну область

Машинне навчання (ML) - це галузь дослідження, присвячена розумінню та розробці методів, які «навчаються», тобто методів, які використовують дані для підвищення ефективності певного набору завдань. Його розглядають як частину штучного інтелекту. Алгоритми машинного навчання створюють модель на основі вибіркових даних, відомих як навчальні дані, щоб робити прогнози чи приймати рішення без явного програмування для цього. Алгоритми машинного навчання використовуються в широкому спектрі додатків, таких як медицина, фільтрація електронної пошти, розпізнавання мовлення, сільське господарство та комп'ютерне зір, де важко або неможливо розробити звичайні алгоритми для виконання необхідних завдань.

Підмножина машинного навчання тісно пов'язана з обчислювальною статистикою, яка зосереджена на створенні прогнозів за допомогою комп'ютерів, але не все машинне навчання є статистичним навчанням. Вивчення математичної оптимізації надає методи, теорію та області застосування в галузі машинного навчання. Інтелектуальний аналіз даних є спорідненою галуззю дослідження, яка зосереджена на дослідницькому аналізі даних за допомогою неконтрольованого навчання. Деякі реалізації машинного навчання використовують дані та нейронні мережі таким чином, щоб імітувати роботу біологічного мозку. Застосовуючи в бізнес-проблемах, машинне навчання також називають прогнозною аналітикою.

Алгоритми навчання працюють на основі того, що стратегії, алгоритми та умовиводи, які добре працювали в минулому, ймовірно, продовжуватимуть добре працювати в майбутньому. Ці висновки можуть бути очевидними, наприклад, «оскільки сонце сходило щоранку протягом останніх 10 000 днів,

л

воно, ймовірно, зійде і завтра вранці». Вони можуть бути відтінками, наприклад «X% сімейств мають географічно окремі види з варіантами забарвлення, тому існує Y% шансів, що існують невідомі чорні лебеді».

Програми машинного навчання можуть виконувати завдання без явного програмування на це. Він передбачає, що комп'ютери навчаються на основі наданих даних для виконання певних завдань. Для простих завдань, призначених комп'ютерам, можна запрограмувати алгоритми, які вказують машині, як виконати всі кроки, необхідні для вирішення поточної проблеми; з боку комп'ютера навчання не потрібне. Для більш складних завдань людині може бути важко вручну створити необхідні алгоритми. На практиці може виявитися ефективнішим допомогти машині розробити власний алгоритм замість того, щоб люди-програмісти вказували кожен необхідний крок.

Дисципліна машинного навчання використовує різні підходи, щоб навчити комп'ютери виконувати завдання, де немає повністю задовільного алгоритму. У випадках, коли існує величезна кількість потенційних відповідей, один підхід полягає в тому, щоб позначити деякі з правильних відповідей як дійсні. Потім це можна використовувати як навчальні дані для комп'ютера, щоб покращити алгоритм(и), які він використовує для визначення правильних відповідей. Наприклад, для навчання системи задачі розпізнавання цифрових символів часто використовувався набір рукописних цифр MNIST.

Термін «машинне навчання» був введений у 1959 році Артуром Семюелом, співробітником IBM і піонером у галузі комп'ютерних ігор і штучного інтелекту. У цей період також використовувався синонім «комп'ютери з самонавчанням».

На початку 1960-х років компанія Raytheon розробила експериментальну «навчальну машину» з пам'яттю на перфострічці під назвою «Кібертрон» для аналізу сигналів гідролокатора, електрокардіограм і мовленнєвих моделей за допомогою елементарного навчання з підкріпленням. Людина-оператор/вчитель неодноразово «навчав» розпізнавати закономірності та оснащував кнопкою «дурень», щоб змушувати його повторно оцінювати неправильні рішення. Репрезентативною книгою про дослідження машинного навчання в 1960-х роках

л

була книга Нільссона про Learning Machines, присвячена переважно машинному навчанню для класифікації шаблонів. Інтерес до розпізнавання образів тривав у 1970-х роках, як описали Дуда та Харт у 1973 році. У 1981 році було надано звіт про використання стратегій навчання, завдяки яким нейронна мережа навчиться розпізнавати 40 символів (26 букв, 10 цифр і 4 спеціальні символи) з комп'ютерного терміналу.

Том М. Мітчелл надав широко цитоване, більш формальне визначення алгоритмів, які вивчаються в області машинного навчання: «Кажуть, що комп'ютерна програма вчиться на досвіді E щодо деякого класу завдань T і мірою продуктивності P, якщо її продуктивність у завданнях у T, як вимірюється P, покращується з досвідом E.» Це визначення завдань, у яких стосується машинне навчання, пропонує принципово операційне визначення, а не визначення поля в когнітивних термінах. Це слідує за пропозицією Алана Тюрінга в його статті «Обчислювальна техніка та інтелект», в якій питання «Чи можуть машини мислити?» замінюється запитанням «Чи можуть машини робити те, що ми (як мислячі істоти) можемо робити?».

Сучасне машинне навчання має дві мети: одна - класифікувати дані на основі розроблених моделей, інша - спрогнозувати майбутні результати на основі цих моделей. Гіпотетичний алгоритм, призначений для класифікації даних, може використовувати комп'ютерний зір родимок у поєднанні з навчанням під наглядом, щоб навчити його класифікувати ракові родимки. Алгоритм машинного навчання для біржової торгівлі може інформувати трейдера про майбутні потенційні прогнози.

Як науковий напрям, машинне навчання виросло з пошуків штучного інтелекту. На початку ШІ як академічної дисципліни деякі дослідники були зацікавлені в тому, щоб машини навчалися на даних. Вони намагалися підійти до проблеми різними символьними методами, а також тим, що тоді називали «нейронними мережами»; це були здебільшого персептрони та інші моделі, які згодом були визнані повторними винаходами узагальнених лінійних моделей статистики. Імовірнісні міркування також використовувалися, особливо в

л

автоматизованій медичній діагностиці.

Однак посилення акценту на логічному, заснованому на знаннях підході спричинило розрив між ШІ та машинним навчанням (рис.1.1). Імовірнісні системи страждали від теоретичних і практичних проблем збору та представлення даних. До 1980 року експертні системи стали домінувати над штучним інтелектом, і статистика була в нелюбовності. Робота над символічним навчанням/навчанням, заснованим на знаннях, продовжувалася в рамках штучного інтелекту, що призвело до індуктивного логічного програмування, але більш статистична лінія досліджень тепер була поза областю власне штучного інтелекту, розпізнавання образів і пошуку інформації.

Машинне навчання (ML), реорганізоване в окрему галузь, почало процвітати в 1990-х роках. Сфера змінила свою мету з досягнення штучного інтелекту на вирішення вирішуваних проблем практичного характеру. Він змістив фокус із символічних підходів, успадкованих від ШІ, на методи та моделі, запозичені зі статистики, нечіткої логіки та теорії ймовірностей.

Різницю між ML та AI часто неправильно розуміють. ML навчається та прогнозує на основі пасивних спостережень, тоді як AI передбачає взаємодію агента з середовищем, щоб навчатися та вживати дій, які максимізують його шанси на успішне досягнення своїх цілей.

Станом на 2022 рік багато джерел продовжують стверджувати, що ML залишається підгалуззю AI. Інші вважають, що не все ML є частиною AI, а лише «інтелектуальну підмножину» ML слід вважати AI.

Машинне навчання та інтелектуальний аналіз даних часто використовують однакові методи та значною мірою збігаються, але в той час як машинне навчання зосереджується на прогнозуванні на основі відомих властивостей, отриманих із навчальних даних, інтелектуальний аналіз даних фокусується на виявленні (раніше) невідомих властивостей у даних (це етап аналізу виявлення знань у базах даних). Інтелектуальний аналіз даних використовує багато методів машинного навчання, але з різними цілями; з іншого боку, машинне навчання також використовує методи інтелектуального аналізу даних як «навчання без

л

нагляду» або як етап попередньої обробки для підвищення точності навчання. Значна частина плутанини між цими двома дослідницькими спільнотами (які часто мають окремі конференції та окремі журнали, головним винятком є ECML PKDD) походить від основних припущень, з якими вони працюють: у машинному навчанні ефективність зазвичай оцінюється з огляду на здатність відтворювати відомі знання, тоді як у виявленні знань та аналізі даних (KDD) ключовим завданням є відкриття раніше невідомих знань. Оцінений з огляду на відомі знання, неінформований (неконтрольований) метод буде легко перевершувати інші контрольовані методи, тоді як у типовому завданні KDD контрольовані методи не можуть бути використані через відсутність навчальних даних.

Машинне навчання також має тісний зв’язок з оптимізацією: багато проблем навчання сформулюються як мінімізація деякої функції втрат на навчальному наборі прикладів. Функції втрат виражають розбіжність між прогнозами моделі, що навчається і фактичними екземплярами проблеми (наприклад, у класифікації потрібно призначити мітку екземплярам, а моделі навчаються правильно прогнозувати, попередньо призначені мітки набору).

Різниця між оптимізацією та машинним навчанням виникає через мету узагальнення: хоча алгоритми оптимізації можуть мінімізувати втрати на навчальному наборі, машинне навчання займається мінімізацією втрат на невидимих зразках. Характеристика узагальнення різних алгоритмів навчання є активною темою поточних досліджень, особливо для алгоритмів глибокого машинного навчання на основі статистичних даних.

Машинне навчання та статистика - тісно пов’язані галузі з точки зору методів, але відрізняються своєю головною метою: статистика робить висновки про сукупність із вибірки, тоді як машинне навчання знаходить узагальнювані прогностичні шаблони. 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 як заповнювач для позначення загальної галузі.

л

Лео Брейман розрізняв дві парадигми статистичного моделювання: модель даних і алгоритмічну модель, де «алгоритмічна модель» означає більш-менш алгоритми машинного навчання, такі як випадковий ліс.

Деякі статистики перейняли методи машинного навчання, що призвело до комбінованої галузі, яку вони називають статистичним навчанням.

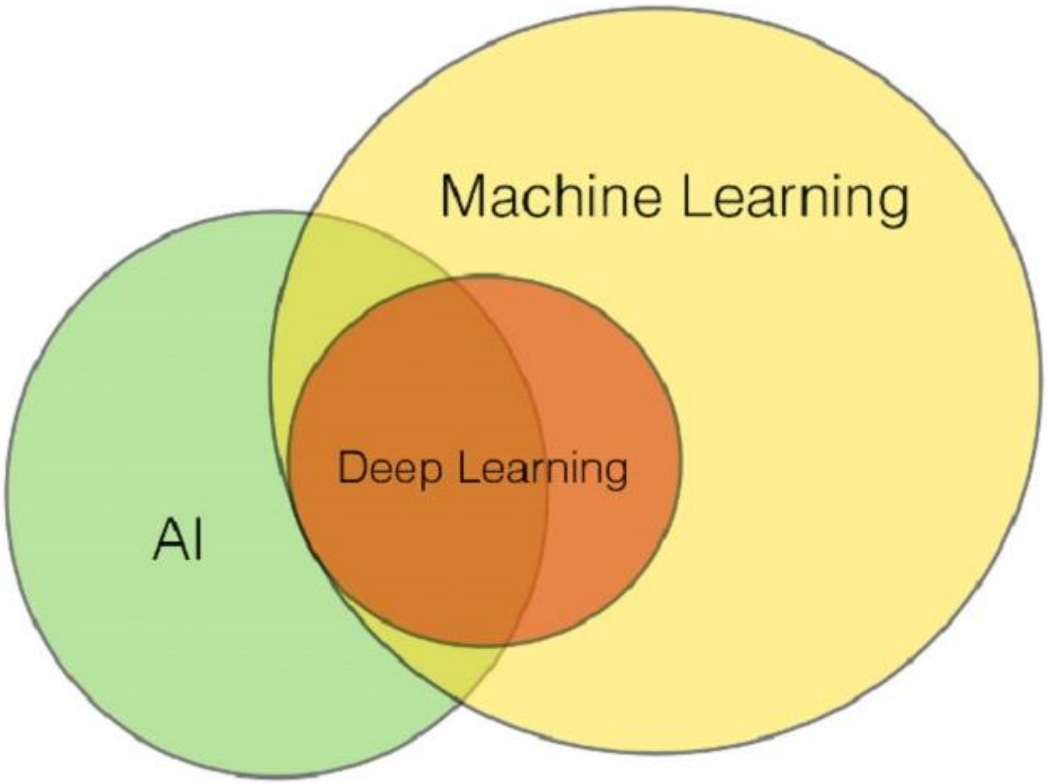


Рис. 1.1. Машинне навчання як підмножина ШІ

1.1.1. Огляд та аналіз існуючих аналогів

Оцінка потенційних клієнтів не є універсальною системою. Кожна компанія має власну унікальну стратегію оцінки, яка відповідає її конкретним цілям і цілям. Навіть зважаючи на невід’ємно індивідуальну природу прогнозного оцінювання потенційних клієнтів, корисно мати кілька прикладів, щоб побудувати свою маркетингову стратегію та з’ясувати, як найкраще розрахувати оцінку потенційних клієнтів.

Підрахунок потенційних клієнтів – це процес присвоєння числового

л

значення кожному потенційному клієнту, який ви отримуєте, щоб оцінити, як ви повинні з ним поводитися. Важливо знати, як розрахувати потенційний бал, оскільки не всі потенційні клієнти однакові. Наприклад, не можна звернутися до когось у верхній частині конверсії з таким же вмістом і увагою, як до когось у нижній частині конверсії.

Кілька найкращих методів підрахунку потенційних клієнтів включають оцінку того, як часто потенційний клієнт взаємодіяв із компанією, відзначаючи, скільки разів він відвідував веб-сайт, і оцінку того, наскільки він готовий отримати рекламні матеріали.

Наведені нижче моделі служать хорошою точкою відліку, щоб почати будувати власну стратегію підрахунку потенційних клієнтів.

Lead Pilot — це інструмент вхідного маркетингу для фінансових радників, який має власний інтегрований інструмент підрахунку потенційних клієнтів.

Принцип роботи досить простий: кожному потенційному клієнту призначається оцінка від 1 до 100, і оцінка змінюється й оновлюється в режимі реального часу, щоб врахувати кожну з їхніх дій. Що вищий бал, то кваліфікованіший лідер.

Алгоритм враховує те, як потенційний клієнт взаємодіяв із вмістом, дії, які він виконував на веб-сайті, кількість часу, який він витратив, та інші фактори. Lead Pilot (рис.1.2.) використовує штучний інтелект для кількісного визначення потенційних клієнтів, але користувачі також можуть коригувати бали.

Наприклад, є веб-сайт (рис.1.3.) із фінансовими продуктами, і користувач виконує такі дії:

- завантажив електронну книгу та залишив свої дані: +5 балів;
- пошук на сайті: +2 бали;
- переглянув вебінар: +10 балів;
- відкритий електронний лист: +3 бали;
- перейшов на сторінку з цінами на сайті: +20 балів;

У цьому випадку лідерство матиме загальну оцінку 40 балів.

л

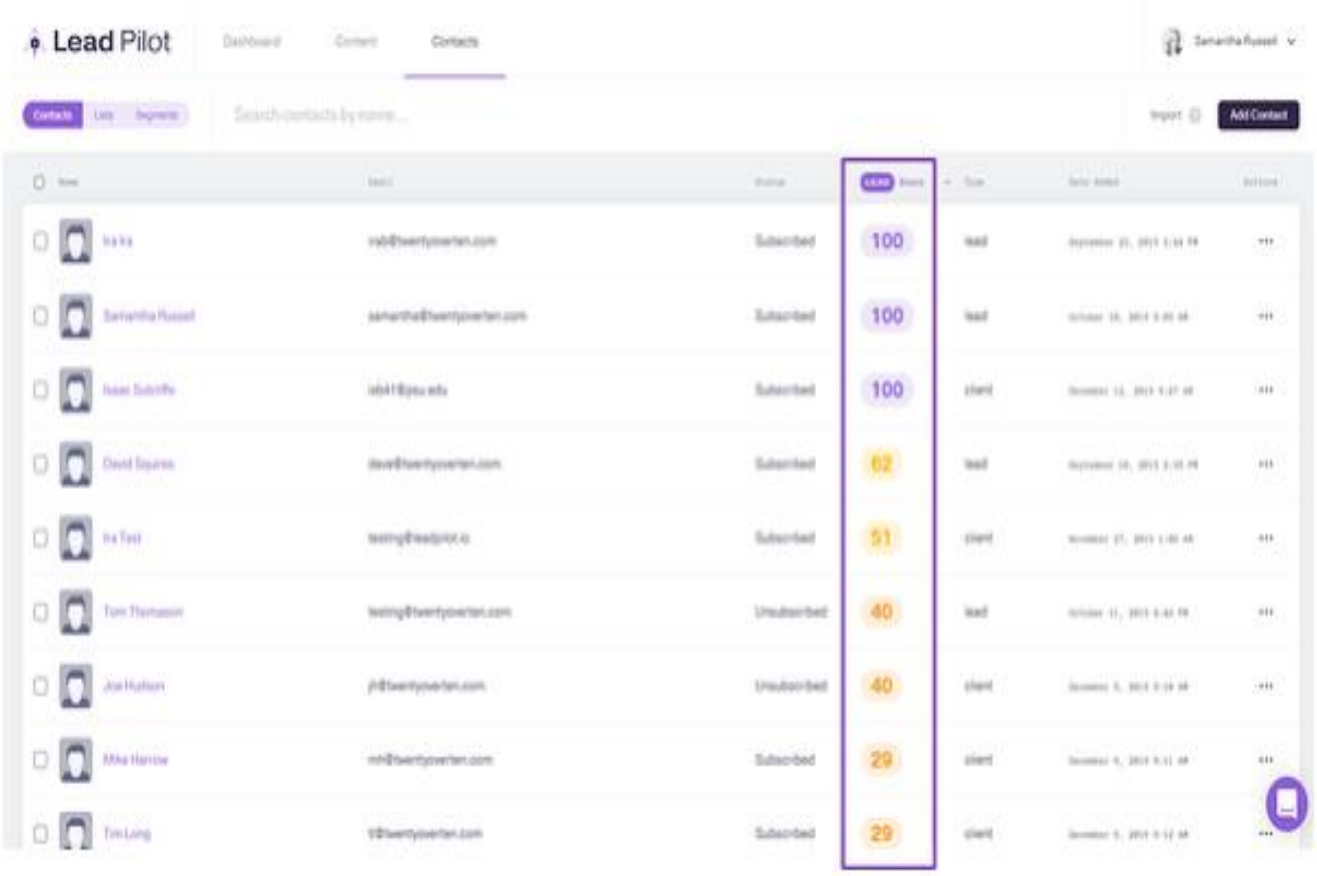


Рис. 1.2. Інтерфейс платформи Lead Pilot

Рис.1.3 Візуалізація дій на вебсайті

л

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].

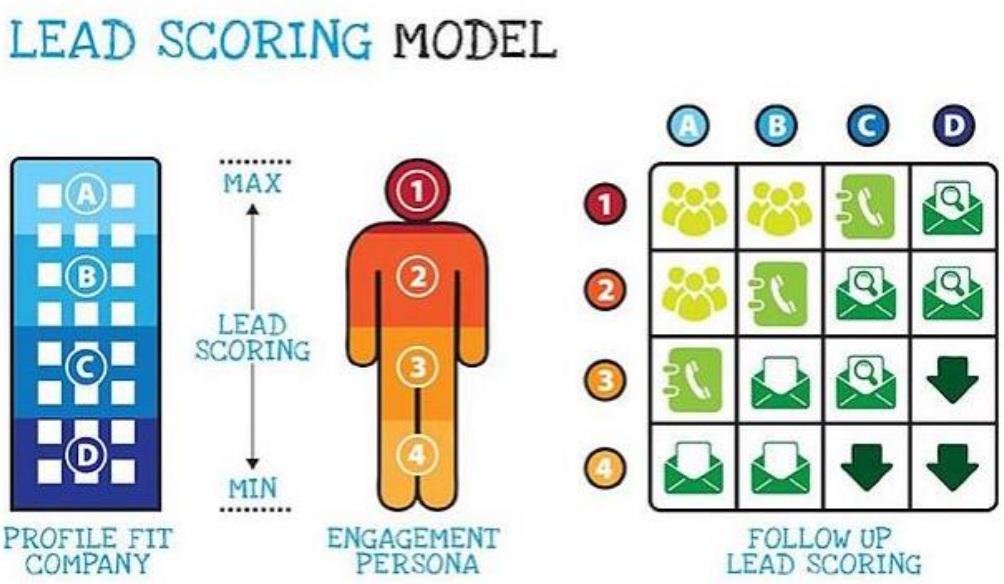


Рис.1.4. Візуалізація оцінок Merodio

У Cyberclick розроблена власну матрицю оцінки потенційних клієнтів на основі двох параметрів.

Демографічний профіль потенційного клієнта, отриманий, наприклад, з даних, які потенційний клієнт залишив під час завантаження форми. Потім його

л

класифікують в ролі потенційного клієнта на різні категорії, наприклад незнайомець (той, про кого у немає необдідної кількості даних для аналізу), непридатний, придатний і дуже придатний.

На основі поведінки потенційного клієнта, включаючи його взаємодію з веб-сайтом, його вмістом, банерами, клієнта можна вважати неактивним, менш активним, активним або дуже активним.

На основі таблиці (рис.1.5.) можна мати до 16 різних можливих комбінацій потенційних клієнтів. Для наглядності краще призначити різні «температури» для кожної категорії, щоб краще зрозуміти, як поводитися з кожним випадком.

Холодні потенційні клієнти: мають недостатню кількість інформації або вони неактивні, тобто не продляють жодної активності на ресурсі

Теплі контакти: вони більш активні, ніж холодні, але ще не досягли оптимального поєднання обох параметрів.

Гарячі потенційні клієнти: ці потенційні клієнти дуже активні. Це ті, хто найбільше зацікавлені та цікаві для бізнесу.

Рис.1.5. Таблиця категоризації клієнтів

Система Sliverpop схожа на ту, що зображена на (рис.1.5.). Вона містить

л

шаблон із 16 різними можливими комбінаціями інтересу та придатності. Різниця полягає в тому, що їх остаточна система класифікації ділиться на чотири наступні категорії клієнтів:

- мертві потенційні клієнти (ті, хто мало цікавить і не потребує уваги);
- маркетингові кваліфіковані потенційні клієнти;
- 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

л

Призначивши бали, комбінуються різні фактори, щоб отримати остаточну оцінку для кожного потенційного клієнта. Наприклад, згідно з наведеною вище таблицею, директор з кадрів НУО з 600 співробітниками отримає 23 бали (короткий підрахунок: 8 + 15 + 5 - 5).

Hubspot (рис.1.7.) має прогнозну модель оцінки потенційних клієнтів, яка використовує машинне навчання для сортування тисяч фрагментів даних, щоб знайти найкращих потенційних клієнтів. Оскільки ця система є автоматизованою, вона з часом оновлюватиме й оптимізуватиме процес підрахунку потенційних клієнтів, заощаджуючи час і енергію.

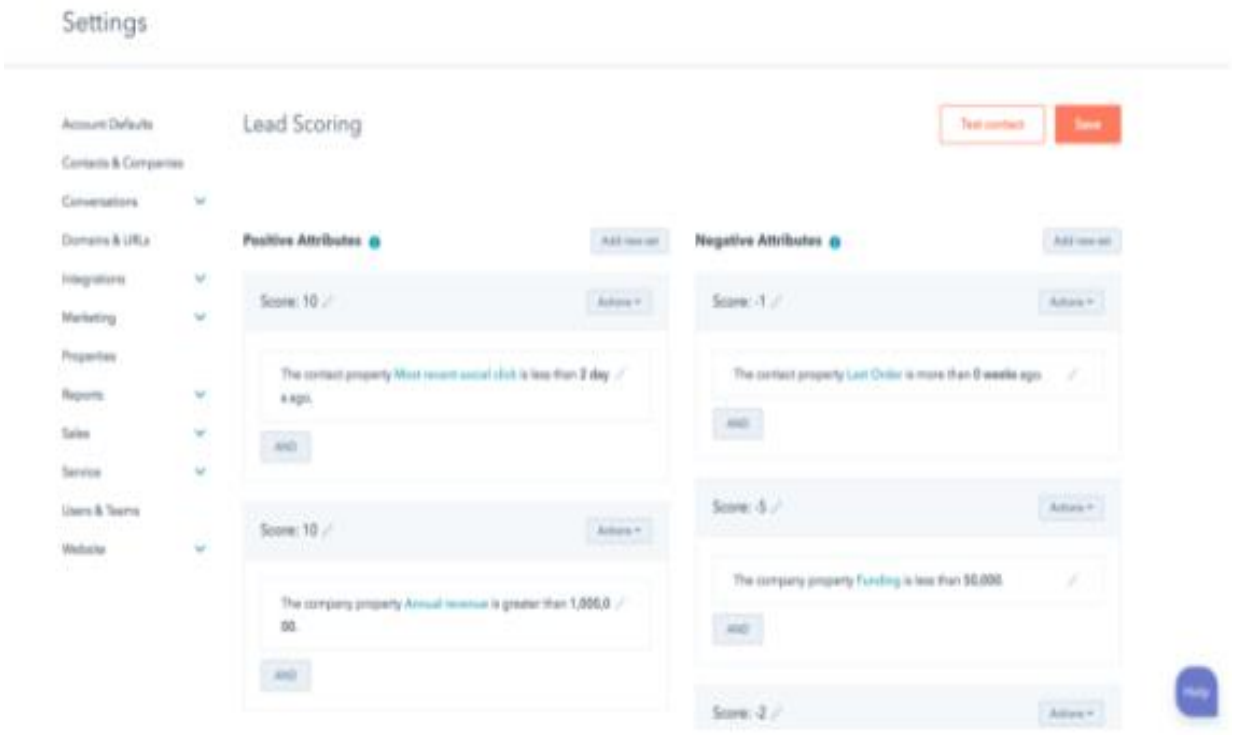


Рис. 1.7. Інтерфейс сервісу Hubspot

Інструмент Hubspot також дозволяє налаштовувати підрахунок потенційних клієнтів, щоб гарантувати ефективність процесу для бізнесу.

Ще одна корисна функція, яку пропонує Hubspot, - це створення різних

л

таблиць для різних аудиторій або потенційних клієнтів. Це особливо корисно, коли бізнес розвивається, оскільки можна розширюватися в різних галузях, країнах тощо, і доведеться залучати ці потенційні клієнти різними способами. Завдяки Hubspot можна мати до 25 унікальних моделей оцінки потенційних клієнтів, які адаптуються до всіх потреб бізнесу.

1.1.2. Вибір операційної системи

Для виконання розробки була вибрана ОС Windows 10. Windows 10 - операційна система для персональних комп'ютерів і робочих станцій, розроблена корпорацією Microsoft в рамках сімейства Windows NT. Після Windows 8.1 система отримала номер 10, мінус 9. Серверні аналоги Windows 10 - Windows Server 2016, Windows Server 2019 і Windows Server 2022.

Система призвана стати єдиною для різних пристроїв, таких як персональні комп'ютери, планшети, смартфони, консолі Xbox One тощо. Доступна єдина платформа розробки та єдиний магазин універсальних додатків, сумісних зі всіма підтримуваними пристроями. Windows 10 постачається як послуга з випуском оновлень протягом усього циклу підтримки. У перший рік після виходу системи користувачі могли безкоштовно оновитися до Windows 10 на пристроях під керуванням ліцензійних копій Windows 7, Windows 8.1 і Windows Phone 8.1. Серед значущих нововведень - голосова помічниця Кортана, можливість створення та переключення кількох робочих столів та інші.

Користувацька угода Windows 10 дозволяє компанії Microsoft зібрати численні відомості про користувачі, історію його інтернет-діяльності, паролі до точок доступу, дані, що вибираються на клавіатурі, і багато іншого.

Згідно зі статистичними даними сайту W3Schools, Windows 10 займає перше місце в середовищі операційних систем, які використовуються для доступу до мережі Інтернет, випереджаючи в квітні 2017 року попереднього лідера – операційну систему Windows 7.[4-7]

л

1.2. Мета розробки та сфера застосування

Основні терміни та ключові слова:

Google Cloud Platform – це набір загальнодоступних хмарних обчислювальних служб, які пропонує Google. Платформа включає низку розміщених служб для обчислення, зберігання та розробки додатків, які 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 через спеціальне мережеве підключення..

Python - високорівнева мова програмування загального призначення з динамічної строгою типізацією і автоматичним управлінням пам'яттю, орієнтований на підвищення продуктивності розробника, читання коду і його якості, а також на забезпечення переносимості написаних на ньому програм. Мова є повністю об'єктно-орієнтованим - все є об'єктами[8].

Розроблений продукт може використовуватися на підприємствах, де відбувається певний збір та аналіз даних, наприклад, у банківській сфері. На мою думку, системи збору та аналізу інформації стають все більш розповсюдженими.

Мета розробки - надати можливість користувачам СКБД експортувати користувацькі дані для їх обробки та аналізу. Комп’ютерна система дозволить оптимізувати процес обзвону клієнтів онлайн-школи, де вивчають іноземні мови. Менеджер call-центру отримає середньозважену оцінку клієнта, яка покаже можливу вірогідність його конверсії в студента 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.

Даний продукт дозволить зберігати великий об’єм інформацію у зручному

л

для користувача вигляді та прогнозувати вірогідність оплати послуг користувачем.

На етапі проектування 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

л

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 даних була розроблена у середовищі MySQL Workbench, для редагування коду Python був використаний додаток Geany.

л

РОЗДІЛ 2
ДИЗАЙН ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональні цілі програми

Результатом даної кваліфікаційної роботи має бути застосунок, який за певний проміжок часа виконує розрахунки платоспрожності користувача на основі характеристик, які беруться з СКБД MySQL .

Кінцевим результатом роботи програми повинні бути дві графічні моделі, які будовані основі середньозважених коефіцієнтів оцінки користувача, а також кожному користувачу буде присвоєна оцінка. Результати будуть зберігатися в файлі з розширенням «.csv».

Основне призначення додатку:

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

Для досягнення поставленої задачі додаток повинен уміти оброблювати великі об’єми користувацьких даних та видавати результат у графічному та текстовому вигляді.

2.2. Застосовані математичні методи

В даній кваліфікаційній роботі для оцінки якості моделей і порівняння різних алгоритмів використовуються метрики, а їх вибір і аналіз - неодмінна частина роботи датасайнтиста.

Accuracy, precision и recall

Перед переходом до самих метрик необхідно ввести важливу концепцію для опису цих метрик в термінах помилок класифікації - confusion matrix (матриця помилок)[6].

л

Припустимо, що є два класи і алгоритм, який пророкує приналежність кожного об'єкта одному з класів, тоді матриця помилок класифікації (рис. 2.8.) буде виглядати наступним чином:

	$y = 1$	$y = 0$
$\hat{y} = 1$	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:

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

л

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.

л

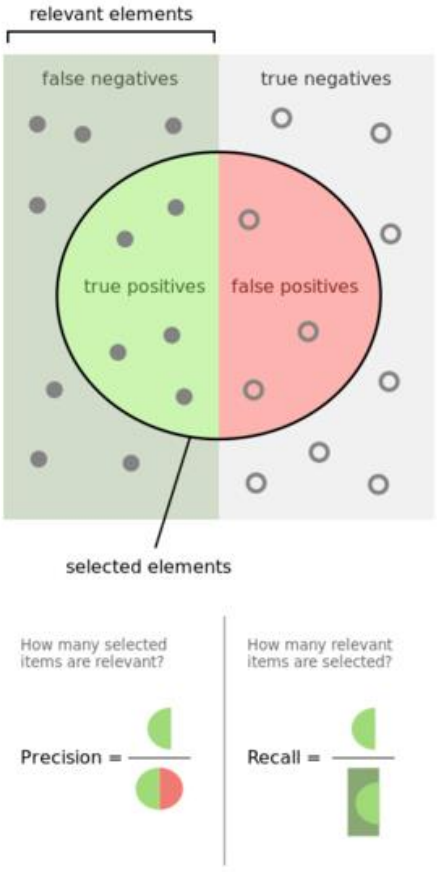


Рис. 2.9. Графічне зображення метрик precision і recall

Precision і recall не залежить, на відміну від accuracy, від співвідношення класів і тому застосовні в умовах незбалансованих вибірок.

Часто в реальній практиці стоїть завдання знайти оптимальний (для замовника) баланс між цими двома метриками. Класичним прикладом є задача визначення скорингу клієнтів.

Очевидно, що неможливо знаходити всіх клієнтів, які хочуть купити продукт. Але, визначивши стратегію і ресурс для виявлення платоспроможних клієнтів, можна підібрати потрібні пороги по precision і recall. Наприклад, можна зосередитися обдзвони клієнтів лише с високим скорингом, так як у ресурсах колл-центру чи менеджев обмежений. Необхідно продумати стратегію збору необхідних для нас метрик, визначити їх пріоритет. Наприклад, зробити цікавий та приємний інтерфейс у користувацькій формі, обмежити кількість полів для реєстрації, адже увага потенційного клієнта може швидко згаснути і він взагалі

л

передумає робити реєстрацію.

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 БД, множинами. Мова SQL надає можливість для виведення структурованої заданої інформації з бази. SQL також застосовується для зміни даних, додавання даних до бази.

Мова SQL відноситься до функціональних мов програмування. Вона відрізняється від алгоритмічних мов. Основу мови становить не алгоритм як такої, а сукупність команд, що визначають взаємини інформаційних множин і підмножин.

Слід зазначити, що системи управління базами даних - СКБД - мають різні реалізації, такі як ORACLE, MS SQL, MYSQL. Мова SQL в різних СКБД має невеликі відмінності, наприклад, в детальному синтаксисі опису операторів. Такі відмінності присутні в спеціальних функціях, що відносяться до тієї чи іншої СКБД, але все ж в основному мова - це загальний синтаксис, практично ідентичний для будь-якої СКБД[11-13].

Реляційна база даних (рис. 2.10.) - це таблиця з інформацією, рознесеною за стовпцями (поля або атрибути) і рядкам (записи або кортежі) таблиці. Щоб змінити або видалити дані в стовпцях і рядках, а також дані в певних осередках

л

(припинення стовпця і рядка) можна скористатися прикладними інструментами (наприклад, phpmyadmin) або зробити SQL запит до бази даних, за яким виконається потрібну 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;

л

- 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 динамічної строгою типізацією і автоматичним управлінням пам'яттю, орієнтований на підвищення продуктивності розробника, читання коду і його якості, а також на забезпечення переносимості написаних на ньому програм. Мова є повністю об'єктно-орієнтованим - все є об'єктами. Незвичайною особливістю мови є виділення блоків коду пробільними відступами. Синтаксис ядра мови мінімалістичний, за рахунок чого на практиці рідко виникає необхідність звертатися до документації. Сам же мова відома як інтерпретується і використовується в тому числі для написання скриптів. Недоліками мови є часто більш низька швидкість роботи і більш високе споживання пам'яті написаних на ньому програм в порівнянні з аналогічним кодом, написаним на компільованих мовах, таких як Сі або C ++[14].

Python є мультипарадигменою мовою програмування, що підтримує імперативне, процедурне, структурний, об'єктно-орієнтоване програмування, метапрограмування і функціональне програмування. Завдання узагальненого програмування вирішуються за рахунок динамічної типізації. Аспектно-орієнтоване програмування частково підтримується через декоратори, більш повноцінна підтримка забезпечується додатковими фреймворками. Такі методики як контрактне і логічне програмування можна реалізувати за допомогою бібліотек або розширень. Основні архітектурні риси - динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточних обчислень з глобальної блокуванням інтерпретатора (GIL), високорівневі структури даних. Підтримується розбиття програм на модулі, які, в свою чергу, можуть об'єднуватися в пакети[17]. Мова програмування Python була створена у 1991

л

році голландцем Гвідо ван Россумом.

Python характеризується простим для розуміння синтаксисом. Читати код на ньому легше, ніж на інших мовах програмування, так як в Пітоні мало використовуються такі допоміжні синтаксичні елементи як дужки, крапки з комою. З іншого боку, правила мови змушують програмістів робити відступи для позначення вкладених конструкцій. Зрозуміло, що добре оформлений текст з малою кількістю відволікаючих елементів читати і розуміти легше[15-19].

Python - це повноцінний багато в чому універсальна мова програмування, що використовується в різних сферах. Основна, але не єдина, підтримувана їм парадигма, - об'єктно-орієнтоване програмування. Однак в даному курсі ми тільки згадаємо про об'єкти, а будемо вивчати структурне програмування, так як воно є базою. Без знання основних типів даних, розгалужень, циклів, функцій немає сенсу вивчати більш складні парадигми, так як в них все це використовується[20].

Інтерпретатори Python поширюється вільно на підставі ліцензії подібної GNU General Public License.

У кваліфікаційній роботі використано Pytnon бібліотеку pandas. Pandas - це високорівнева Python бібліотека для аналізу даних. Вона побудована поверх більш низкоуровневой бібліотеки NumPy (написана на Ci), що є великим плюсом в продуктивності. В екосистемі Python, pandas є найбільш просунутою що швидко розвивалася бібліотекою для обробки і аналізу даних [21].

Також використана бібліотеку matplotlib. Matplotlib - це бібіліотека двовимірної графіки для мови програмування python за допомогою якої можна створювати високоякісні малюнки різних форматів. Matplotlib є модуль-пакет для мови програмування Python.

Для вирішення завдань класичного машинного навчання обрано найпоширенішу бібліотеку Scikit-learn. Вона надає широкий вибір алгоритмів навчання з учителем і без вчителя. Навчання з учителем передбачає наявність розміченого датасета, в якому відомо значення цільового показника. У той час як навчання без вчителі не передбачає наявності розмітки в датасета - потрібно

л

навчитися отримувати корисну інформацію з довільних даних. Одне з основних переваг бібліотеки полягає в тому, що вона працює на основі декількох поширених математичних бібліотек, і легко інтегрує їх один з одним. Ще однією перевагою є широка спільнота і докладна документація. Scikit-learn широко використовується для промислових систем, в яких застосовуються алгоритми класичного машинного навчання, для досліджень, а так само для новачків, які тільки робить перші кроки в області машинного навчання[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;

л

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

Змінна відношення знаходиться в другій нормальній формі тоді і тільки тоді, коли вона знаходиться в першій нормальній формі і кожен неключовий атрибут залежить від (кожного) її потенційного ключа.

Якщо потенційний ключ є простим, тобто складається з єдиного атрибута, то будь-яка функціональна залежність від нього є не зводима (повна). Якщо потенційний ключ є складовим, то, згідно з визначенням другий нормальної форми, у відношенні не повинно бути неключових атрибутів, залежних від частини складеного потенційного ключа.

Друга нормальна форма за визначенням забороняє наявність неключових атрибутів, які взагалі не залежать від потенційного ключа. Таким чином, друга нормальна форма в тому числі забороняє створювати відношення як незв'язані (хаотичні, випадкові) набори атрибутів.

Відношення знаходиться в третій нормальній формі, коли відношення знаходиться в другій нормальній формі й усі неключові атрибути взаємно незалежні.

Для того, щоб усунути залежність неключових атрибутів, потрібно зробити декомпозицію відносини ще на кілька відносин. При цьому ті неключові атрибути, які є залежними, виносяться в окреме відношення[23-25].

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;

л

- 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;

л

- 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).

л

- 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.

л

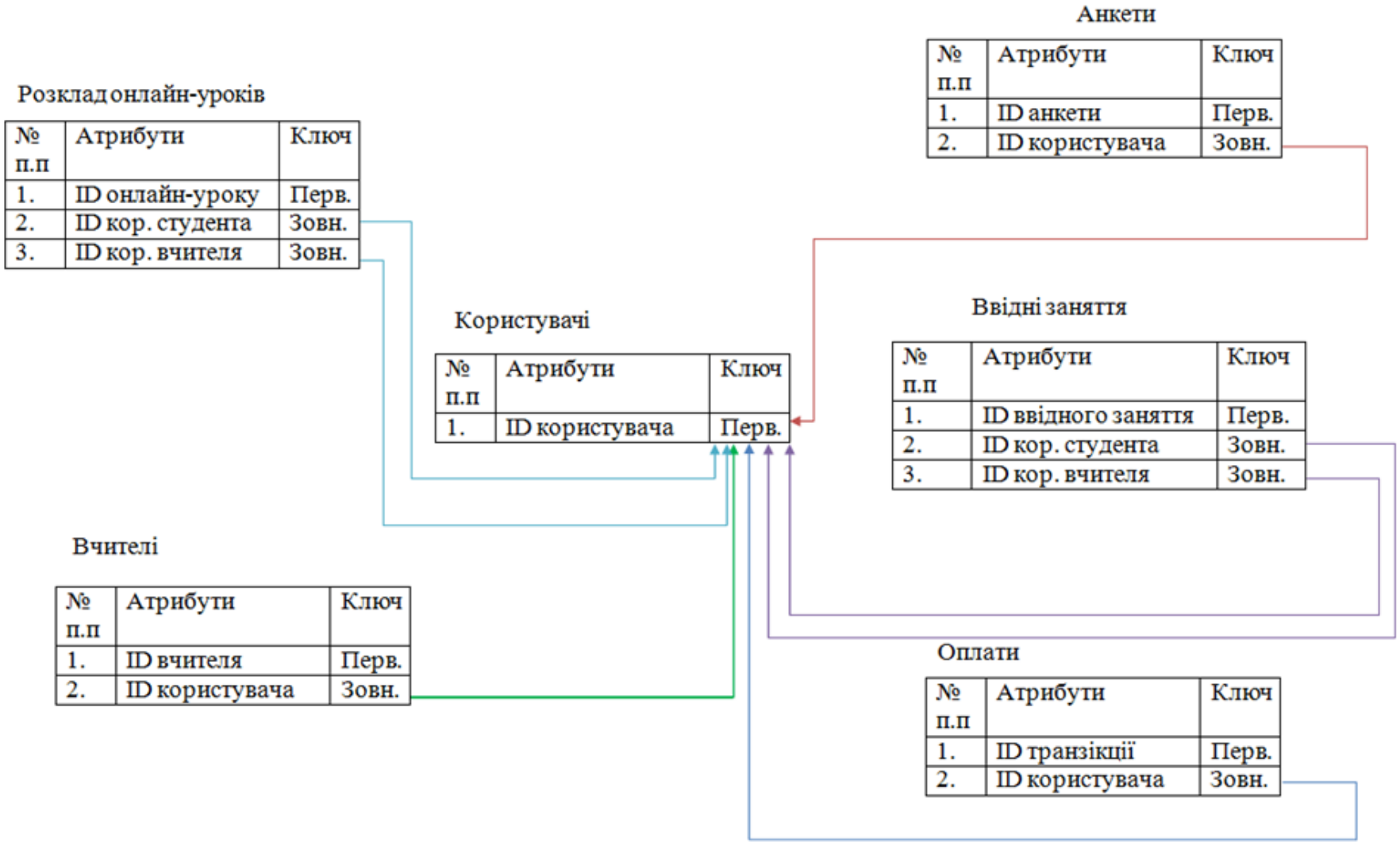


рис.2.10. Схема реляційної бази даних

л

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 застосовані декілька методів машинного навчання: RandomForestClassifier, RandomForestRegressor, LogisticRegression, а також ROC-аналіз. Перші два методи з англійської мови дослівно перекладаються як «випадковий ліс», а третій метод перекладається як «логістична регресія».

Випадковий ліс - це контрольований алгоритм навчання, який використовується як для класифікації, так і для регресії. Але, тим не менш, він в основному використовується для задач класифікації [27]. Оскільки ми знаємо, що ліс складений з дерев, і більше дерев означає більш стійкий ліс. Точно так же алгоритм випадкового лісу створює дерева рішень для вибірок даних, а потім отримує прогноз по кожній з них і, нарешті, вибирає краще рішення за допомогою голосування. Це метод ансамблю, який краще, ніж єдине дерево рішень, тому що він зменшує перенавчання шляхом усереднення результату.

Дерево рішень - інтуїтивно зрозуміла базова одиниця алгоритму випадковий ліс. Ми можемо розглядати його як серію питань так / ні про вхідних даних. В кінцевому підсумку питання призводять до передбачення певного класу (або величини в разі регресії). Це інтерпретується модель, так як рішення приймаються так само, як і людиною: ми задаємо питання про доступних даних до тих пір, поки не дійдемо до певного рішення (в ідеальному світі).

Базова ідея дерева рішень полягає у формуванні запитів, з якими алгоритм звертається до даних [32]. При використанні алгоритму CART питання (також звані поділом вузлів) визначаються таким чином, щоб відповіді вели до зменшення забруднення Джині (Gini Impurity). Це означає, що дерево рішень формує вузли, що містять велику кількість зразків (з набору вихідних даних), що належать до одного класу. Алгоритм намагається виявити параметри з подібними значеннями.

л

Забруднення Джині - ймовірність невірного маркування в вузлі випадково обраного зразка.

У кожному вузлі дерево рішень шукає таке значення певного параметра, яке призведе до максимального зменшення забруднення Джині. В якості альтернативи для поділу вузлів також можна використовувати концепцію накопичення інформації.

Потім процес поділу повторюється з використанням «жадібної», рекурсивної процедури, поки дерево не досягне максимальної глибини або в кожному вузлі не залишаться тільки зразки одного класу. Середньозважене забруднення Джині має зменшуватися з кожним рівнем.

Питома вага забруднення Джині для кожного вузла дорівнює відношенню кількості зразків, оброблених цим вузлом, до кількості оброблених батьківським вузлом. Ви можете самостійно розрахувати забруднення Джині для наступних рівнів дерева і окремих вузлів, використовуючи дані візуалізації. Таким чином, ефективна модель будується на базових математичних операціях[28-30].

Однак важливо пам'ятати, що алгоритм безпомилково відсортувавує тільки тренувальні дані. Мета машинного навчання полягає в тому, щоб навчити алгоритм узагальнювати отриману інформацію і правильно обробляти нові, раніше не зустрічалися дані [31].

Перенавчання відбувається, коли ми використовуємо дуже гнучку модель (з високою місткістю), яка просто запам'ятовує навчальний набір даних, підганяючи вузли під нього [16]. Проблема в тому, що така модель виявляє не тільки закономірності в даних, але і будь-який присутній в них шум. Таку гнучку модель часто називають високоваріативною, оскільки параметри, що формуються в процесі навчання (такі як структура дерева рішень) будуть значно варіюватися в залежності від навчального набору даних.

З іншого боку, у недостатньо гнучкою моделі буде високий рівень похибки, оскільки вона робить припущення щодо тренувальних даних (модель зміщується в бік упереджених припущень про дані). Наприклад, лінійний класифікатор передбачає, що дані розподілені лінійно. Через це він не володіє

л

достатньою гнучкістю для відповідності нелінійним структурам. Ригідна модель може виявитися недостатньо ємною навіть для відповідності тренувальним даними для алгоритму машинного навчання[33-35].

В обох випадках - і при високій варіативності, і при високій похибки - модель не зможе ефективно обробляти нові дані.

Пошук балансу між зайвою і недостатньою гнучкістю моделі є ключовою концепцією машинного навчання і називається компромісом між варіативністю і похибкою (з англ. "bias-variance tradeoff").

Алгоритм дерева рішень перенавчати, якщо не обмежити його максимальну глибину. Він має необмежену гнучкістю і може розростатися, поки не досягне стану ідеальної класифікації, в якій кожному зразку з набору даних буде відповідати один лист. Якщо повернутися назад до створення дерева і обмежити його глибину двома шарами (зробивши лише один поділ), класифікація більше не буде на 100% вірною. Ми зменшуємо варіативність за рахунок збільшення похибки[36].

В якості альтернативи обмеження глибини, яке веде до зменшення варіативності (добре) і збільшення похибки (погано), ми можемо зібрати безліч дерев в єдину модель. Це і буде класифікатор на основі комітету дерев прийняття рішень або просто «випадковий ліс».

Логістична регресія - корисний класичний інструмент для вирішення завдання регресії і класифікації. ROC-аналіз - апарат для аналізу якості моделей. Обидва алгоритми активно використовуються для побудови моделей в медицині і проведення клінічних досліджень[37].

Логістична регресія набула поширення в скорингу для розрахунку рейтингу позичальників і управління кредитними ризиками. Тому, незважаючи на своє «походження» з статистики, логістичну регресію і ROC-аналіз майже завжди можна побачити в наборі Data Mining алгоритмів.

Логістична регресія - це різновид множинної регресії, загальне призначення якої полягає в аналізі зв'язку між декількома незалежними змінними (званими також регресорів або предикторами) і залежною змінною.

л

Бінарна логістична регресія застосовується в разі, коли залежна змінна є бінарною (тобто може приймати тільки два значення). За допомогою логістичної регресії можна оцінювати вірогідність того, що подія настане для конкретного випробуваного (хворий / здоровий, повернення кредиту / дефолт і т.д.).

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.) - крива, яка найбільш часто використовується для представлення результатів бінарної класифікації в машинному навчанні. Назва прийшла з систем обробки сигналів [21]. Оскільки класів два, один з них називається класом з позитивними наслідками, другий - з негативними наслідками. ROC-крива показує залежність кількості вірно класифікованих позитивних прикладів від кількості невірно класифікованих або негативних.

У термінології ROC-аналізу перші називаються істинно позитивним, другі - хибно негативним безліччю. При цьому передбачається, що у класифікатора є деякий параметр, варіюючи який, ми будемо отримувати ту чи іншу розбиття на два класи. Цей параметр часто називають порогом, або точкою відсікання (cut-off value). Залежно від нього будуть виходити різні величини помилок I і II роду.

У логістичної регресії поріг відсікання змінюється від 0 до 1 - це і є розрахункове значення рівняння регресії. Будемо називати його рейтингом.

Для розуміння суті помилок I і II роду розглянемо чотирьохполу таблицю спряженості (confusion matrix), яка будується на основі результатів класифікації моделлю і фактичної (об'єктивної) приналежністю прикладів до класів.

У сфері машинного навчання та, зокрема, у проблемі статистичної класифікації, матриця плутанини, також відома як матриця помилок, - це спеціальний макет таблиці, який дозволяє візуалізувати продуктивність алгоритму, як правило, навчання з наглядом (у неконтрольованому навчанні це зазвичай називають узгоджувальною матрицею). Кожен рядок матриці представляє екземпляри в фактичному класі, а кожен стовпець представляє екземпляри в прогнозованому класі, або навпаки - обидва варіанти зустрічаються в літературі. Назва походить від того факту, що вона дозволяє легко побачити,

л

чи система плутає два класи (тобто зазвичай навмисне неправильно позначає один як інший, тобто міняє логічне значення правильний на неправильний).

Це особливий вид таблиці непередбачуваності з двома вимірами («фактичним» і «прогнозованим») і ідентичними наборами «класів» в обох вимірах (кожна комбінація виміру та класу є змінною в таблиці непередбачуваності).

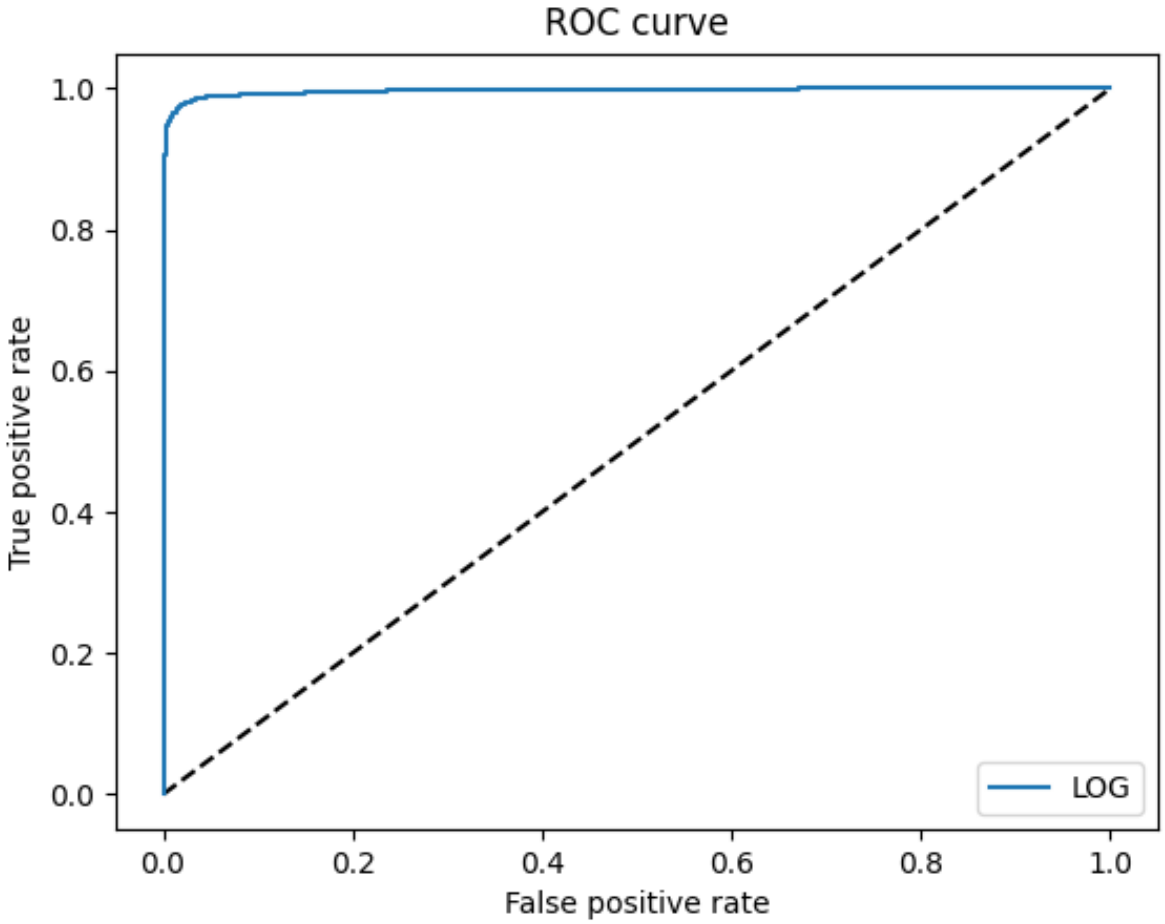


Рис. 2.11. Крива ROC

Для ідеального класифікатора графік ROC-кривої проходить через верхній лівий кут, де частка істинно позитивних випадків становить 100% або 1,0 (ідеальна чутливість), а частка хибно позитивних прикладів дорівнює нулю. Тому чим ближче крива до верхнього лівого кута, тим вище передбачувальна здатність моделі. Навпаки, чим менше вигин кривої і чим ближче вона розташована до діагональної прямої, тим менш ефективна модель. Діагональна

л

лінія відповідає класифікатором, тобто повної нерозрізненості двох класів.

При візуальній оцінці ROC-кривих розташування їх відносно один одного вказує на їх порівняльну ефективність. Крива, розташована вище і лівіше, свідчить про більшу передбачувану здатність моделі [38-44].

2.5. Раціоналізація та організація вхідних та вихідних даних

В даній кваліфікаційній роботі оброблюються клієнтські дані, деякі користувач вказує при реєстрації (ім’я, вік, стать тощо), деякі записуються в систему автоматично (наприклад, дані про операційну систему, дані про браузер). Для розрахунку скорингу необхідна як умова більша повнота тренувальних даних для навчання системи машинному обчисленню та тестові дані, яким згодом буде присвоєна оцінка скорингу.

Існують дві лінії продажів. Перша - це коли учень залишає анкету на навчання, протягом п’яти хвилин менеджер зв’язується з клієнтом та намагається продати йому певний продукт. Перед дзвінком менеджер закріплює цього клієнта за собою на декілька тижнів, щоб за цей час зробити вдалу конверсію з клієнта в студента. Якщо за цей час користувач відмовляється купувати послуги, він потрапляє в другу лінію продажів. Тепер переходимо до основної цілі даних розрахунків, а саме оптимізація роботи менеджера другої лінії продажів.

Для формування тренувальних даних формується вибірка з клієнтів за останній рік роботи компанії. Наприклад, необхідно розрахувати скоринг березневих анкет 2021р. Для цього за допомогою SQL-запиту створюється вибірка за період з березня 2020р. по лютий 2021р. Так формується тренувальна база. Місяць, який треба розрахувати формується за допомогою аналогічного SQL-запиту, змінюючи в ньому період вибірки на потрібний. Так формується тестова база. Отримані результати експортуються у форматі «.csv».

Після виконання машинних обчислень будується графік кривої ROC Оцінка скорингу формується в окремому файлі у форматі «.csv».

л

2.6. Розроблений опис програмного продукту

Для роботи з програмою, досить запустити додаток. Після чого програма вже буде запущеною. Якщо мінімальні технічні характеристики обладнання на якому було запущено даний додаток не будуть відповідати, користувач не зможе коректно працювати з додатком. Під час формування вибірок даних та подальшої роботи з ними необхідно слідкувати інструкції, яка буде наведена 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 служб, які пропонує Google. Платформа включає низку розміщених служб для обчислення, зберігання та розробки додатків, які 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 через спеціальне мережеве з’єднання.

Google Cloud пропонує послуги для обчислення, зберігання, мереж,

л

великих даних, машинного навчання та Інтернету речей, а також керування хмарою, безпеки та інструменти розробника (рис.2.12.). Деякі з продуктів хмарних обчислень у Google Cloud включають:

Google Compute Engine, яка є пропозицією інфраструктури як послуги (IaaS), яка надає користувачам екземпляри віртуальних машин для розміщення робочого навантаження.

Google App Engine, яка є пропозицією платформи як послуги (PaaS), яка надає розробникам програмного забезпечення доступ до масштабованого хостингу Google. Розробники також можуть використовувати SDK для розробки програмних продуктів, які працюють на App Engine.

Хмарне сховище Google, яке є платформою хмарного сховища, призначеною для зберігання великих неструктурованих наборів даних. Google також пропонує варіанти зберігання баз даних, зокрема Cloud Datastore для нереляційного сховища NoSQL, повністю реляційного сховища Cloud SQL для MySQL і власну базу даних Google Cloud Bigtable.

Google Kubernetes Engine (GKE), яка є системою керування та оркестровки для контейнера Docker і контейнерних кластерів, які працюють у публічних хмарних службах Google. Google Kubernetes Engine базується на Kubernetes, системі керування контейнерами Google з відкритим кодом.

Набір операцій Google Cloud, раніше Stackdriver, який є набором інтегрованих інструментів для моніторингу, журналювання та звітування про керовані служби, що керують програмами та системами в Google Cloud.

Безсерверні обчислення, які надають інструменти та послуги для виконання робочого навантаження на основі подій, наприклад Cloud Functions для створення функцій, які обробляють хмарні події, Cloud Run для керування та запуску контейнерних програм і Workflows для оркестрування безсерверних продуктів і API функцій.

Бази даних, які є набором продуктів баз даних, що надаються як повністю керовані служби, включаючи Cloud Bigtable для великомасштабних робочих навантажень із низькою затримкою; Камера для документів; CloudSpanner як

л

масштабована, високонадійна реляційна база даних; і CloudSQL як повністю керована база даних для MySQL, PostgreSQL і SQL Server.

Google Cloud пропонує послуги з розробки та інтеграції програм. Наприклад, Google Cloud Pub/Sub - це керована служба обміну повідомленнями в реальному часі, яка дозволяє обмінюватися повідомленнями між програмами. Крім того, Google Cloud Endpoints дозволяє розробникам створювати служби на основі RESTful API, а потім робити ці служби доступними для клієнтів Apple iOS, Android і JavaScript. Інші пропозиції включають DNS-сервери Anycast, прямі мережеві з'єднання, балансування навантаження, послуги моніторингу та журналювання подій на вебсайті[45].

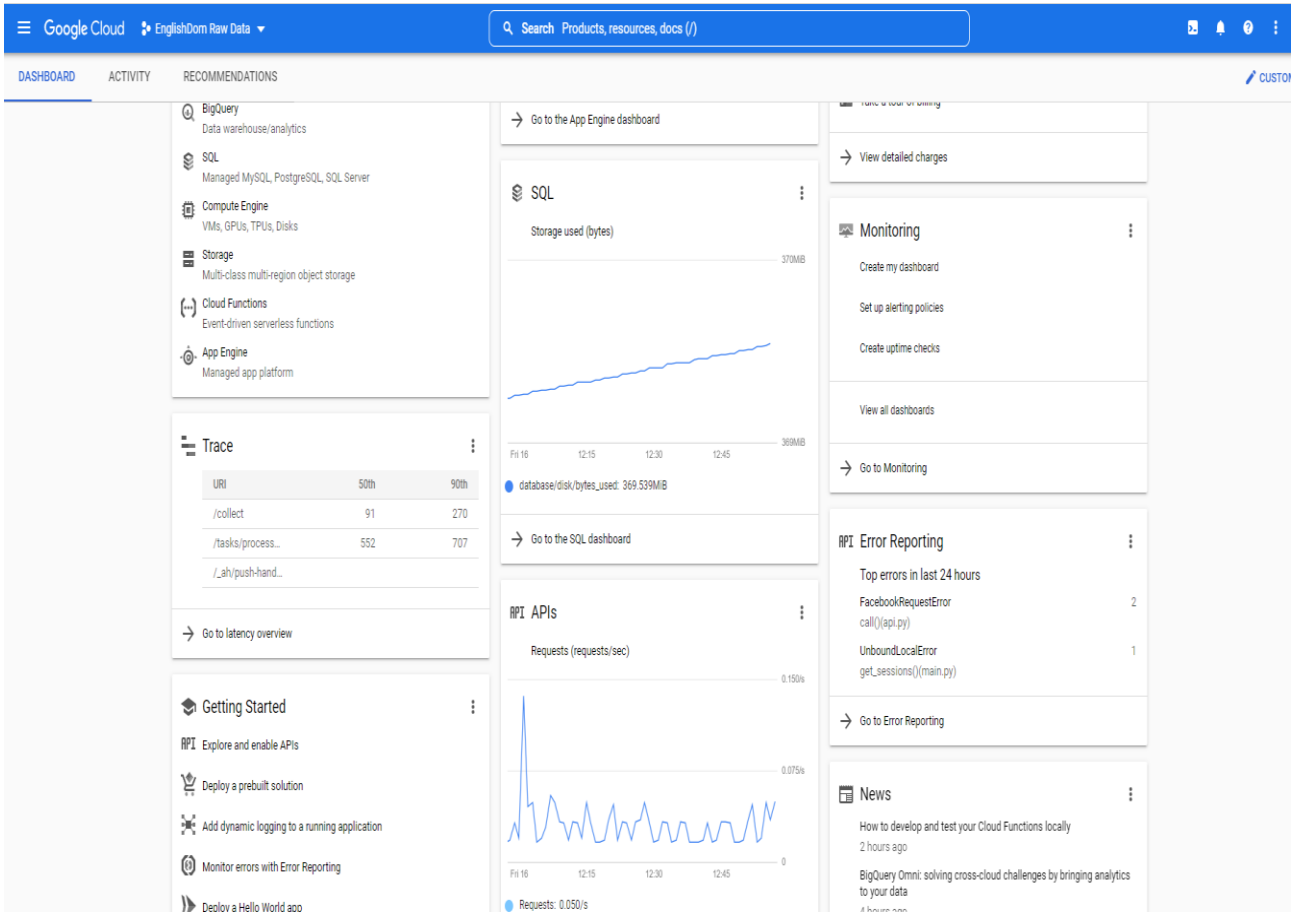


Рис. 2.12. Головна сторінка застосунку Google Cloud Platform

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

л

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).

л

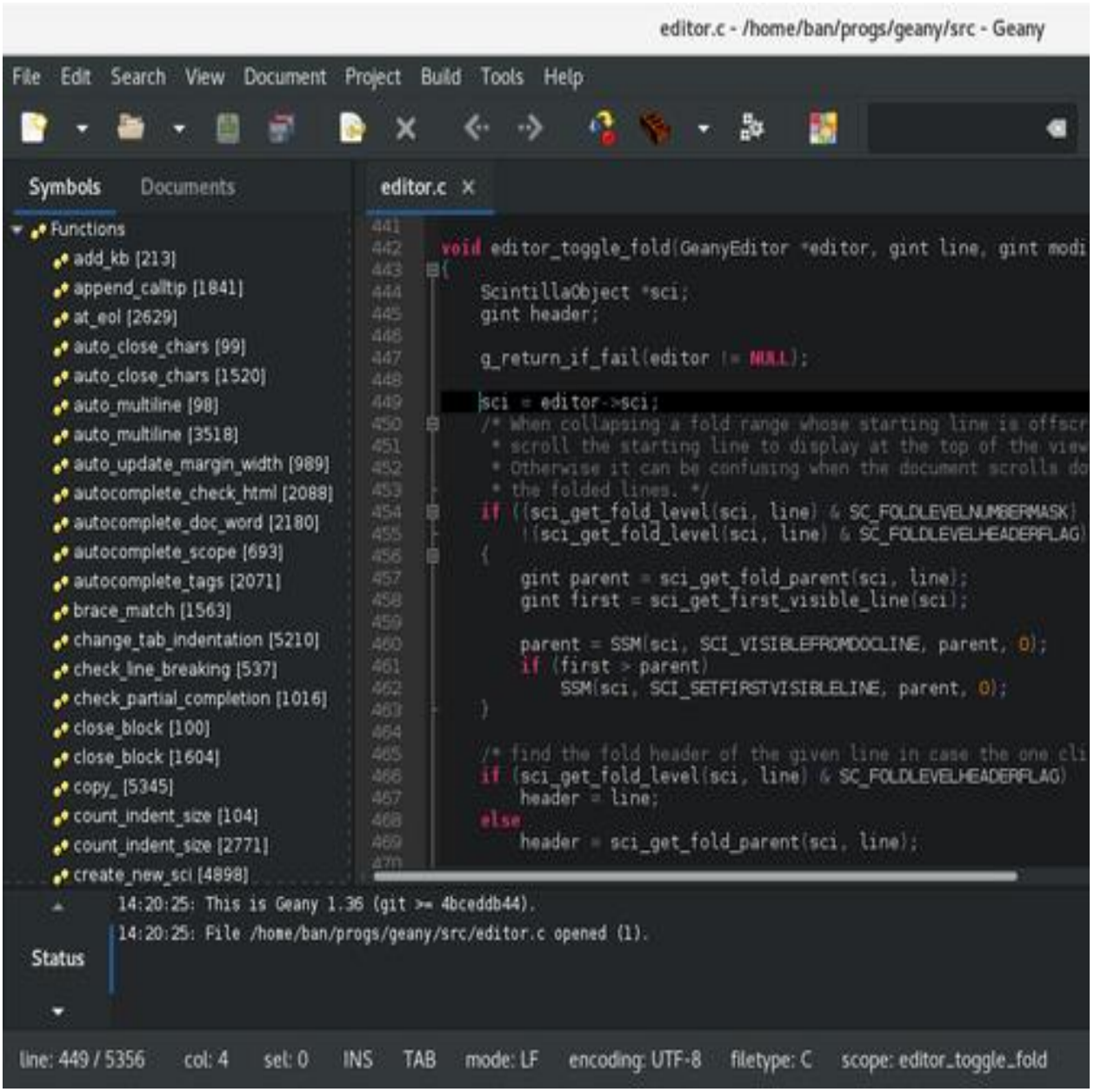


Рис. 2.13. Інтерфейс Geany

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.

л

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.).

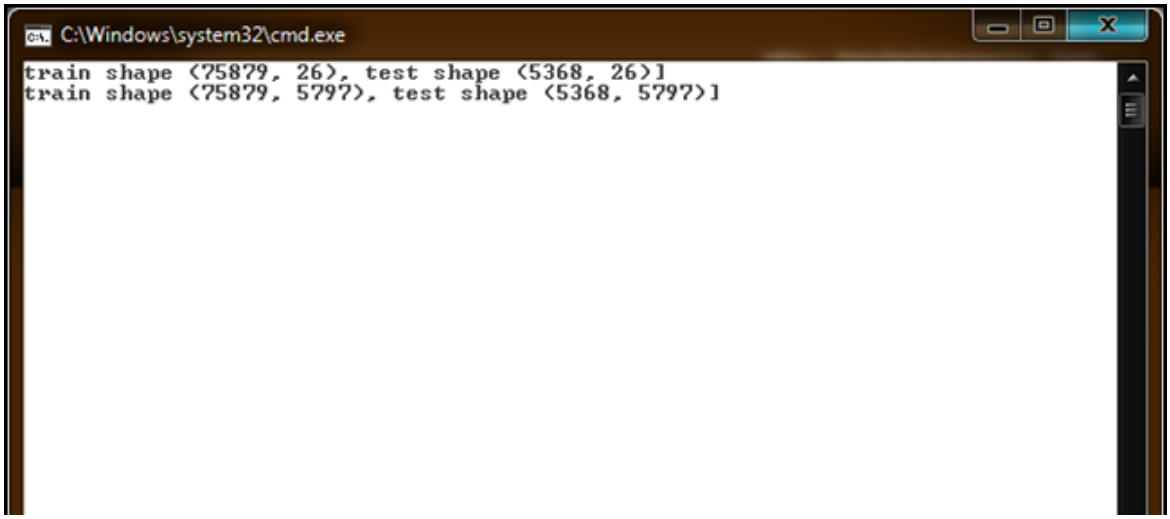


Рис. 2.15. Консольне вікно, яке повідомляє про початок розрахунків

л

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.

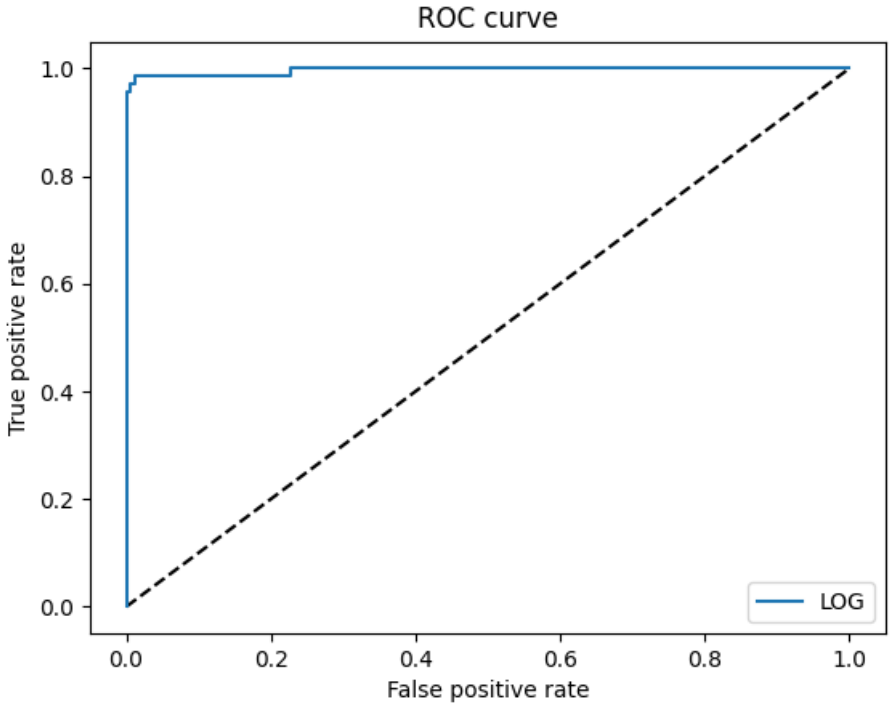


Рис. 2.16. Крива ROC

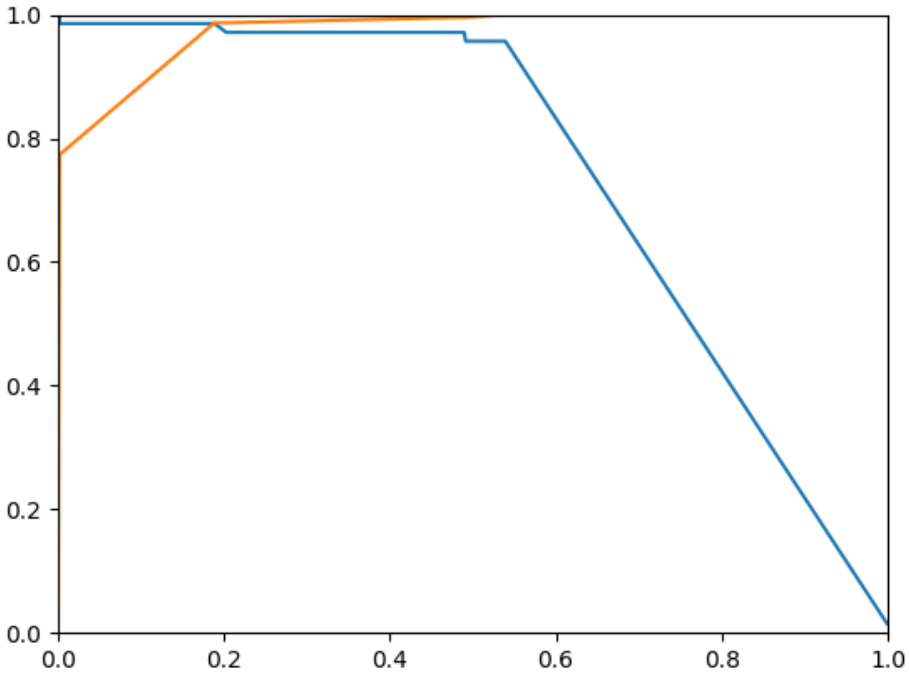


Рис. 2.17. Крива, яка точку оптимального балансу для розділення бази

л

3.2 Дослідження методів машинного навчання

Ймовірно, людство живемо в найбільш визначальний період історії людства. Період, коли обчислення перемістилися від великих мейнфреймів до ПК і хмари. Але визначальним є не те, що сталося, а те, що чекає через роки.

Те, що робить цей період захоплюючим - це демократизація різних інструментів і технік, що відбулася після зростання комп'ютерної техніки.

Існує три типи алгоритмів машинного навчання. Перший - це алгоритми навчання під наглядом. Цей алгоритм складається із цільової/результатної змінної яку потрібно передбачити на основі заданого набору предикторів або незалежних змінних. Використовуючи цей набір змінних, генерується функція, яка зіставляє вхідні дані з бажаними виходами. Процес навчання триває, доки модель не досягне бажаного рівня точності даних навчання. Приклади контрольованого навчання: регресія, дерево рішень, випадковий ліс, KNN, логістична регресія тощо.

Другий - це алгоритми неконтрольованого навчання. У цьому алгоритмі немає жодної цільової чи кінцевої змінної для прогнозування/оцінки. Він використовується для кластеризації популяцій у різні групи, що широко використовується для сегментації клієнтів у різні групи для конкретних заходів. Приклади неконтрольованого навчання: Апріорний алгоритм, K-середні.

Третій - це навчання з підкріпленням. За допомогою цього алгоритму машина навчена приймати конкретні рішення. Це працює таким чином: машина піддається впливу середовища, де вона постійно тренується методом проб і помилок. Ця машина вчиться на минулому досвіді та намагається отримати найкращі знання для прийняття точних бізнес-рішень. Приклад навчання з підкріпленням: Марківський процес прийняття рішень [46].

Найпоширеніші алгоритми машинного навчання:

- лінійна регресія;
- логістична регресія;
- дерево рішень;

л

- 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).

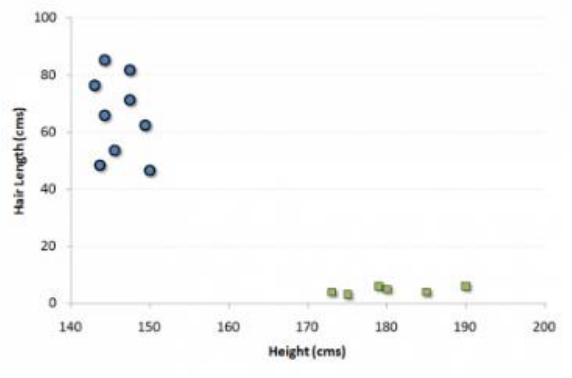


Рис.3.18. График методу SVM

л

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.).

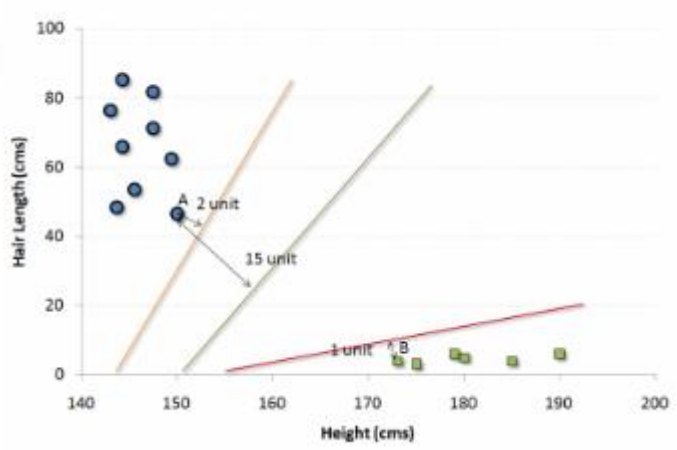


Рис.3.19. Поділ класифікованих груп

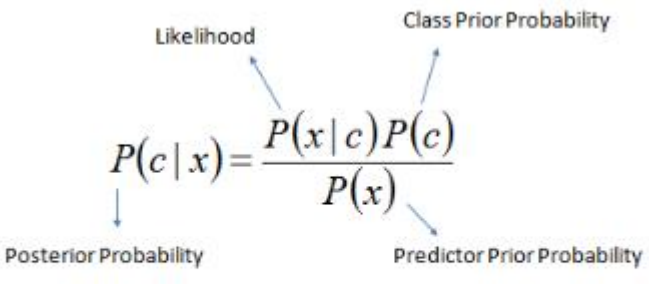
У наведеному вище прикладі лінія, яка розділяє дані на дві різні класифіковані групи, є чорною лінією, оскільки дві найближчі точки є найдальшими від лінії. Цей рядок є класифікатором. Потім, залежно від того, куди по обидві сторони лінії потрапляють дані тестування, можна віднести до цього класу нові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

Наївний Байєс - це метод класифікації, заснований на теоремі Байєса з припущенням незалежності між предикторами. Простіше кажучи, наївний класифікатор Байєса припускає, що наявність певної ознаки в класі не пов’язана з наявністю будь-якої іншої ознаки. Наприклад, фрукт можна вважати яблуком, якщо він червоний, круглий і діаметром близько 10 сантиметрів. Навіть якщо ці ознаки залежать одна від одної або від існування інших ознак, наївний класифікатор Байєса вважав би, що всі ці властивості незалежно сприяють імовірності того, що цей фрукт є яблуком[48].

Наївну байєсовську модель легко побудувати, і вона особливо корисна для дуже великих наборів даних. Відомо, що поряд із простотою Naive Bayes перевершує навіть дуже складні методи класифікації.

л

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- Nearest Neighbors) – це універсальний алгоритм, який можна використовувати як для задач класифікації, так і для регресії. Однак він більш широко використовується в проблемах класифікації в промисловості. K найближчих сусідів — це простий алгоритм, який зберігає всі доступні випадки та класифікує нові випадки більшістю голосів своїх k сусідів. Випадок, призначений класу, є найпоширенішим серед його K найближчих сусідів, виміряних функцією відстані.

Ці функції відстані можуть бути евклідовими, манхеттенськими, Мінковського та Хеммінга. Перші три функції використовуються для безперервної функції, а четверта (Хеммінга) для категоріальних змінних. Якщо K = 1, то випадок просто приписується класу його найближчого сусіда. Часом вибір K виявляється складним завданням під час моделювання kNN (рис.3.19.).

л

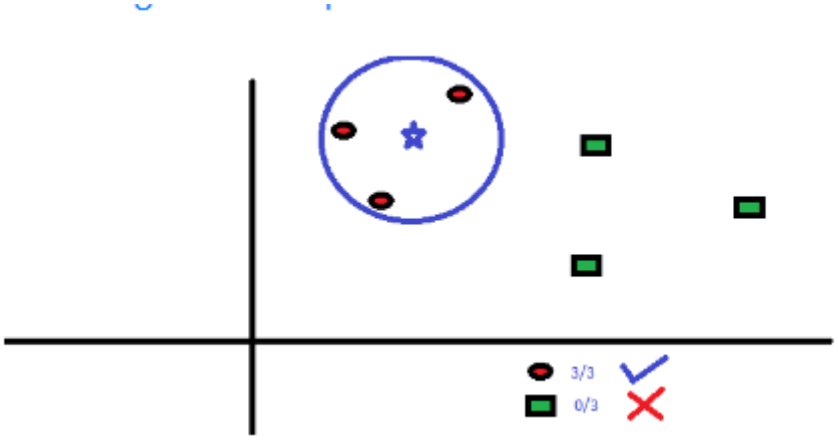


Рис.3.19. Моделювання методу KNN

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 вирішує проблему кластеризації. Його процедура полягає в простому та легкому способі класифікації даного набору даних через певну кількість кластерів (припустимо k кластерів). Точки даних усередині кластера є однорідними та неоднорідними щодо однорангових груп.

К означає дещо схоже на діяльність зі складання фігур з чорнильних плям (рис.3.20.). Дивлячись на форму та поширення, щоб розшифрувати, скільки різних кластерів/популяцій існує.

л

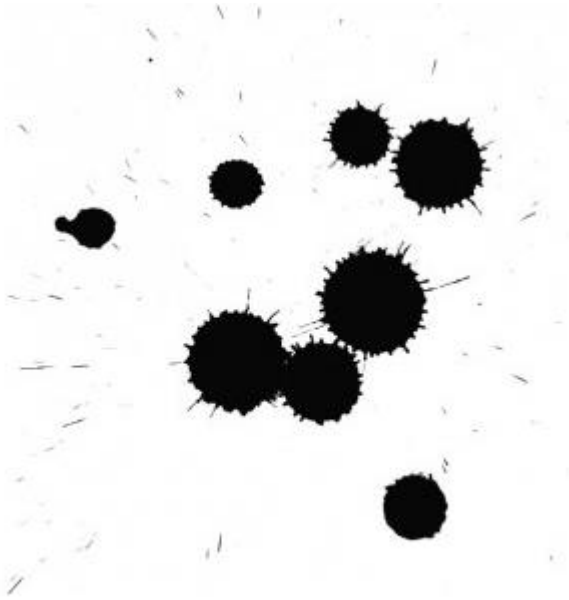


Рис.3.20. Метод неконтрольованого алгоритму

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.

У K-середніх є кластери, і кожен кластер має власний центроїд. Сума квадратів різниці між центроїдом і точками даних у межах кластера становить суму квадратів значення для цього кластера. Крім того, коли сума квадратних значень для всіх кластерів додається, вона стає загальною в межах суми квадратичних значень для кластерного рішення.

л

Зі збільшенням кількості кластерів це значення продовжує зменшуватися, але якщо побудувавши результат, можна побачити, що сума квадратів відстані різко зменшується до деякого значення k, а потім набагато повільніше (рис.3.21.).

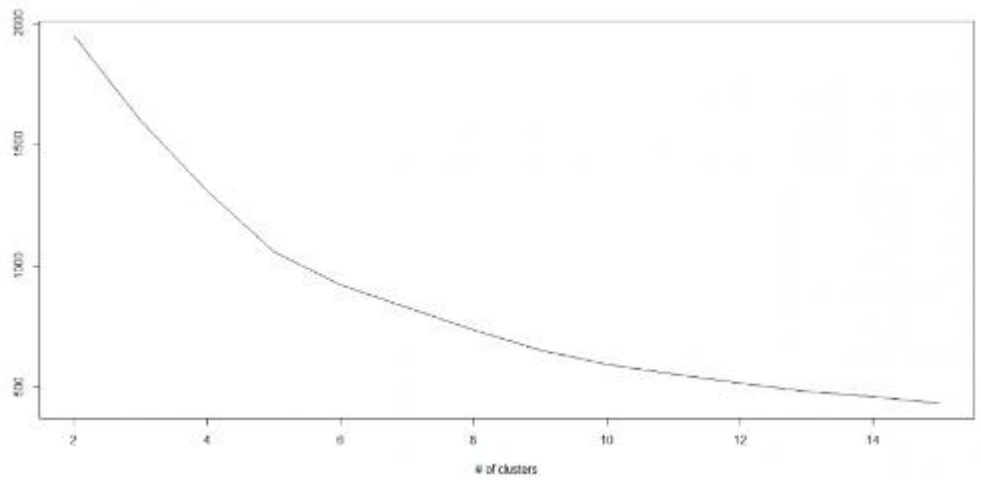


Рис.3.21. Кластеризація значень центроїдів

Алгоритми зменшення розмірності. За останні 4-5 років відбулося експоненціальне зростання збору даних на всіх можливих етапах. Корпорації, урядові агенції, дослідницькі організації не тільки винаходять нові джерела, але й збирають дані з великою кількістю деталей.

Наприклад, компанії електронної комерції збирають більше деталей про клієнтів, як-от їхні демографічні дані, історія веб-сканування, що їм подобається чи не подобається, історія покупок, відгуки та багато іншого, щоб приділяти їм персоналізовану увагу більше, ніж власнику найближчого продуктового магазину.

GBM - це алгоритм підвищення, який використовується для аналізу великої кількісті даних, щоб зробити прогноз із високою потужністю передбачення. Підвищення фактично є сукупністю алгоритмів навчання, які поєднують передбачення кількох базових оцінок, щоб підвищити стійкість порівняно з одним оцінювачем. Він поєднує кілька слабких або середніх предикторів у потужний предиктор. Ці алгоритми підвищення ефективності

л

завжди добре працюють на змаганнях з обробки даних, таких як Kaggle, AV Hackathon і CrowdAnalytix.

Ще один класичний алгоритм посилення градієнта, який, як відомо, є вирішальним вибором між перемогою чи поразкою в деяких змаганнях Kaggle.

XGBoost має надзвичайно високу передбачувану здатність, що робить його найкращим вибором для точності подій, оскільки він має як лінійну модель, так і алгоритм навчання дерева, що робить алгоритм майже в 10 разів швидшим за існуючі методи посилення градієнта.

Підтримка включає різні цільові функції, включаючи регресію, класифікацію та ранжування.

Однією з найцікавіших речей XGBoost є те, що її також називають регуляризованою технікою посилення. Це допомагає зменшити надмірне моделювання та має масову підтримку ряду мов, таких як Scala, Java, R, Python, Julia та C++.

Підтримує розподілене та широке навчання на багатьох машинах, які охоплюють кластери GCE, AWS, Azure та Yarn. XGBoost також можна інтегрувати з Spark, Flink та іншими хмарними системами потоку даних із вбудованою перехресною перевіркою на кожній ітерації процесу підвищення.

LightGBM — це структура підвищення градієнта, яка використовує алгоритми навчання на основі дерева. Він призначений для розподілу та ефективності з такими перевагами:

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

Фреймворк - це швидка та високопродуктивна система підвищення градієнта, заснована на алгоритмах дерева рішень, яка використовується для ранжування, класифікації та багатьох інших завдань машинного навчання. Його було розроблено в рамках проекту Microsoft Distributed Machine Learning Toolkit.

л

Оскільки LightGBM базується на алгоритмах дерева рішень, він розбиває дерево по листах із найкращим підходом, тоді як інші алгоритми підвищення розбивають дерево по глибині або рівням, а не по листах. Таким чином, коли росте на тому самому аркуші в Light GBM, листовий алгоритм може зменшити більше втрат, ніж алгоритм рівня, і, отже, призводить до набагато кращої точності, якої рідко можна досягти за допомогою будь-якого з існуючих алгоритмів посилення[49-54].

Висновки: в даній кваліфікаційній роботі були обрани методи алгоритмів випадкового лісу, дерева рішень лінійної та логістичної регресій. Для виконання завдання необхідно задіювати цільові/результатні змінні, які потрібно передбачувати на основі заданого набору предикторів (незалежних змінних). Основні характеристики для побудови – це користувацькі дані, яким присвоєні певні коефіцієнти, тому, використовуючи цей набір змінних, генерується функцію, яка зіставляє вхідні дані з бажаними виходами. Процес навчання триває, доки модель не досягне бажаного рівня точності даних навчання.

л

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 для кожного користувача протягом певного періоду часу. Додаток надає можливість створювати скорингові моделі для оцінки якості ваших аналізів. На основі цієї оцінки ви можете зрозуміти 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.

л

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.

л

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.

л

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.

л

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.

л

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.

л

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;
```

л

```
 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`)
```

л

```
  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`
```

л

```
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`)
```

л

```
  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
```

л

```
  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;
```

л

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
```

л

```
        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
```

л

```
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
```

л

```
#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()
```

л

```
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)
```

л

ДОДАТОК Б

СПИСОК ФАЙЛІВ НА ДИСКУ

Ім'я файлу	**Опис**
Пояснювальні документи	
Немченко 122-21-2м.doc	Пояснювальна записка до проекту. Документ Word.
Немченко 122-21-2м.pdf	Пояснювальна записка до проекту у форматі PDF.
Program	
Немченко 122-21-2м.zip	Архів. Містить програмні коди.
Presentation	
Немченко 122-21-2м.ppt	Презентація проекту.

л

ВІДГУК КЕРІВНИКА
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

Факультет інформаційних технологій
Кафедра програмного забезпечення комп'ютерних систем

ВІДГУК

Наукового керівника Мещерякова Леоніда Івановича, д.т.н., проф. каф. ПЗКС

на магістерську роботу

студента Немченка Максима Ігоровича

(прізвище, ім'я, по батькові)

курсу II групи 122м-21-2

спеціальності 122 Комп'ютерні науки

освітньої програми

на тему Дослідження методів машинного навчання по підвищенню ефективності оптимізації скорингових моделей

Представлена магістерська кваліфікаційна роботу присвячена пошуку та аналізу шляхів оптимізації роботи алгоритмів машинного навчання таких як, логістична та лінійна регресії, алгоритмів випадкового «лісу», дерева рішень.

Мета досліджень полягає в оптимізації роботи обчислювальних методів машинного навчання, швидкості роботи SQL запитів при вигрузці користувацьких даних, використовуючи існуючі програмні методи та бібліотеки мови програмування Python.

Перший розділ містить аналітичний огляд предметної області, формується мета розробки та постановка проблеми. Другий розділ включає в себе опис використаних програмних та математичних засобів та шляхи вирішення проблематики кваліфікаційної роботи. Третій розділ наводить опис інтерфейсу користувача, а також огляд найпоширеніших методів машинного

л

навчання та обґрунтування використаних у ході виконання кваліфікаційної роботи.

Отримані результати роботи є застосовними при розробці додатків, що використовують в своєму програмному коді методи машинного навчання.

Магістром було проведено аналіз та порівняння можливих методів розв’язання поставленої задачі та обрано оптимальний варіант. Під час виконання магістерської кваліфікаційної роботи студент Немченко М.І. проявив себе грамотним, кваліфікованим спеціалістом, здатним приймати самостійно складні технічні рішення. Вважаю, що магістерська кваліфікаційна робота заслуговує оцінку «добре» , а Немченко М.І. – присвоєння кваліфікації «магістра» з комп’ютерних наук.

Науковий керівник Мещеряков Леонід Іванович, д.т.н., проф. каф. ПЗКС

«___»_____2022 р. _____

(підпис)

л

РЕЦЕНЗІЯ

на магістерську роботу

студента Немченка Максима Ігоровича

(прізвище, ім’я, по батькові)

курсу II групи _122м-21-2___

кафедри програмного забезпечення комп’ютерних систем

спеціальності 122 Комп’ютерні науки

освітньої програми

Тема роботи Дослідження методів машинного навчання по Підвищенню ефективності оптимізації 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. Працездатність цієї інформаційної системи підтверджується експлуатаційними випробуваннями. Викладена основна суть проблеми, шо вирішується в ході виконання роботи.

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

л

Робота заслуговує оцінки «добре» , а студент Немченко Максим Ігорович – присвоєння кваліфікації «магістра» з комп’ютерних наук.

Рецензент _____

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання, посада, місце роботи)

«___»_____2022 р. _____

(підпис)