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

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Немченка Максима Ігоровича*
(ПІБ)

академічної групи *122-17-3*
(шифр)

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

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка додадтку для анілітики користувацьких даних та побудови скорингових моделей у СУБД*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи				
розділів:				
спеціальний	*доц.Приходченко С.Д.*			
економічний	*доц.Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-17-3* *Немченко М.І.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка додатку для аналітики користувацьких даних та побудови скорингових моделей у СУБД*

затверджена наказом ректора НТУ «ДП» від *07.06.2021 р.* № *317-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2021 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2021 р.***

Завдання видав _____ *доц.Приходченко С.Д.*
(підпис) (посада, прізвище, ініціали)

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

Дата видачі завдання: *14.01.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *11.06.2020 р.*

РЕФЕРАТ

Пояснювальна записка: 76 с., 12 рис., 3 дод., 22 джерела.

Об'єкт розробки: додаток для аналітики користувацьких даних.

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

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

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

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

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

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

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

ABSTRACT

Explanatory note: 76 p., 12 figs., 3 apps., 22 sources.

Object of development: application for user data analytics.

The purpose of the qualification work: to create an application that will help develop machine learning skills, and to give an understanding of how to process user data.

The introduction examines the current problem statement, specifies the purpose of the qualification work and the area of its application, justifies the relevance of the topic and specifies the problem statement.

In the first section carries out the analysis of the subject area, determines the relevance of the task and the dedication of the development, creates task statement, the software and hardware requirements of the product, specifies technologies and tools for development.

In the second section analyzes the existing solutions, chose the platform for development, creates design and finish development of the product, describes the algorithm, structure and architecture solutions in the system, defines the input and output data, describes characteristics of the technical resources used, describes how to run a program, features of user interaction, differences between serve and client parts of product.

The economic section defines the complexity of the information system, calculates the cost of work on creating the application and defines the time for its creation.

The practical significance is creation a product which provides an opportunity to improve knowledge and increase knowledge.

The relevance of this software product is determined by the high demand for such developments, which help to optimize the processing of large amounts of data.

Keywords: COMPUTER, INFORMATION SYSTEM, ALGORITHM, DATABASE DESIGN, INTERFACE, APPENDIX, MACHINE LEARNING.

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

СУБД - Система управління базами даних;

БД - база даних;

ПЗ - програмне забезпечення

IDE - Integrated development environment;

ОС - операційна система;

TP - true positive

TN - true negative.

ЗМІСТ

ВСТУП

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

У наш час широкого розповсюдження набувають різні онлайн-сервіси та онлайн-платформи. Перед нами постає велике різноманіття інтернет-магазинів, онлайн-кінотеатрів, засобів доставки їжі, платформ для навчання у багатьох освітніх напрямках, сервісів відеозв’язку. Пандемія коронавірусу дала потужний поштовх розвитку всього онлайн-сектору. Все більше людей користується онлайн-сервісами, а користувацька інформація, яку надають люди все більше зростає в об’ємі. Найбільше мене зацікавило питання як ці дані обробити? Враховуючи всі вище перераховані фактори, тематикою кваліфікаційної роботи є розробка додатку для аналітики користувацьких даних та побудови скорингових моделей у СУБД.

Метою кваліфікаційної роботи є вивчення засобів створення та роботи з базою даних та засобів роботи з ними. На мою думку, ці 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. Також вони нададуть кращого розуміння механізмів роботи з сутностями бази даних та взаємодії з ними.

Дана робота знайомить з методами опрацювання користувацьких даних та подальшої взаємодії з ними.

Основними вимогами до системи будуть: доступні для розуміння методи роботи з даними, надійна та стабільна робота системи, точність розрахунків

РОЗДІЛ І АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Системи управління базами даних (СУБД) - це комп'ютерні програми, які дозволяють користувачам взаємодіяти з базою даних. СУБД дозволяє користувачам контролювати доступ до бази даних, записувати дані, запускати запити і виконувати будь-які інші завдання, пов'язані з управлінням базами даних.

Однак для виконання будь-якої з цих завдань СУБД повинна мати в основі модель, що визначає організацію даних. Реляційна модель - це один з підходів до організації даних, який широко використовується в програмному забезпеченні баз даних з моменту своєї появи в кінці 60-х років. Цей підхід настільки поширений, що на момент написання даної кваліфікаційної роботи чотири з п'яти найпопулярніших систем управління базами даних є реляційними [1].

Бази даних - це логічно сформовані кластери інформації, або даних. Будь-яка колекція даних є базою даних, незалежно від того, як і де вона зберігається. Шафа з платіжними відомостями, полку в реєстратурі з картками пацієнтів або зберігається в різних офісах клієнтська картотека компанії - все це бази даних. Перш ніж зберігання даних і управління ними за допомогою комп'ютерів стало загальною практикою, урядовим організаціям та комерційним компаніям для зберігання інформації були доступні тільки фізичні бази даних такого роду.

Приблизно в середині XX століття розвиток комп'ютерної науки привело до створення машин з більшою обчислювальною потужністю, а також зі збільшеними можливостями вбудованої і зовнішньої пам'яті [2]. Ці досягнення дозволили фахівцям в області обчислювальної техніки усвідомити потенціал таких пристроїв в області зберігання і управління великими масивами даних.

Однак не існувало ніяких теорій про те, як комп'ютери можуть організовувати дані осмисленим, логічним чином. Одна справа зберігати

несортовані дані на комп'ютері, але набагато складніше створити системи, які дозволяють послідовно додавати, вилучати, сортувати і іншим чином управляти цими даними на практиці. Необхідність в логічній конструкції для зберігання і організації даних привела до появи ряду пропозицій щодо використання комп'ютерів для управління даними.

Однією з ранніх моделей бази даних була ієрархічна модель, в якій дані були організовані у вигляді дерева, подібної сучасним файловим системам [3]. Наступний приклад показує, як може виглядати частина ієрархічної бази даних учнів шкіл певної області.

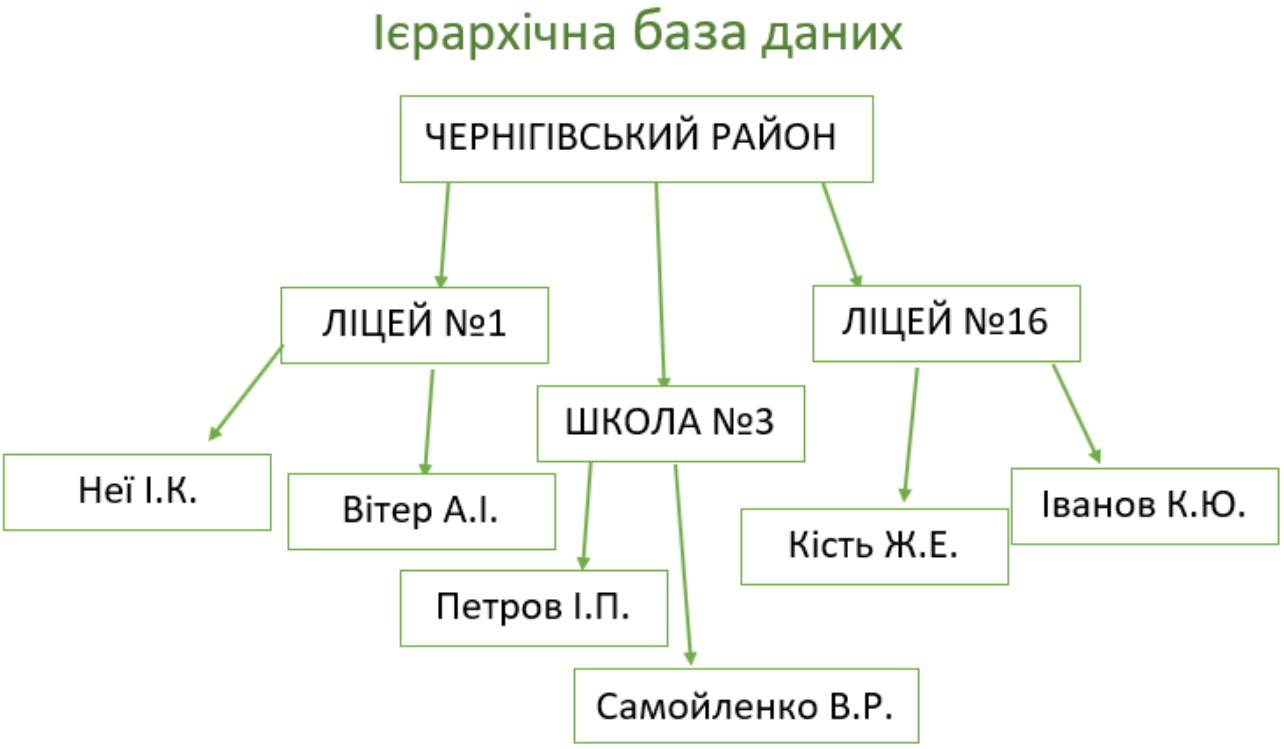

Рис. 1.1 - схема частини ієрархічної база даних

Ієрархічна модель була широко впроваджена в ранні системи управління базами даних, але вона відрізнялася відсутністю гнучкості. У цій моделі кожен запис може мати тільки одного «предка», навіть якщо окремі записи можуть мати кілька «нащадків» [4]. Через це ці ранні ієрархічні бази даних могли представляти тільки відносини «один до одного» або «один до багатьох». Відсутність відносин «багато до багатьох» могло призвести до виникнення

проблем при роботі з точками даних, які вимагають прив'язки до кількох предкам.

В кінці 60-х років Едгар Ф. Кодд (Edgar F. Codd), програміст з IBM, розробив реляційну модель управління базами даних. Реляційна модель Кодда дозволила зв'язати окремі записи з декількома таблицями, що дало можливість встановлювати між точками даних відносини «багато до багатьох» на додаток до «один до багатьох». Це забезпечило більшу гнучкість в порівнянні з іншими існуючими моделями, якщо говорити про розробку структур баз даних, а значить реляційні системи управління базами даних (РСУБД) могли задовольнити набагато ширший спектр бізнес-завдань.

Кодд запропонував мову для управління реляційними даними, відомий як Alpha, що зробив вплив на розробку більш пізніх мов баз даних. Колеги Кодда з IBM, Дональд Чемберлен (Donald Chamberlin) і Реймонд Бойс (Raymond Boyce), створили один з мов під впливом мови Alpha. Вони назвали свою мову SEQUEL, скорочена назва від Structured English Query Language (структурований англійська мова запитів), але через існуючий товарного знака скоротили назву до SQL (більш формальна назва - структурований мова запитів).

Через обмежені можливості апаратного забезпечення ранні реляційні бази даних були все ще недозволено повільними, і потрібно якийсь час, перш ніж технологія набула широкого поширення. Але до середини 80-х років реляційна модель Кодда була впроваджена в ряд комерційних продуктів з управління базами даних від компанії IBM і її конкурентів. Слідом за IBM, ці постачальники також стали розробляти і застосовувати свої власні діалекти SQL. До 1987 року Американський національний інститут стандартів і Міжнародна організація по стандартизації ратифікували і опублікували стандарти SQL, зміцнивши його статус визнаного мови для управління РСУБД.

Широке використання реляційної моделі у багатьох галузях призвело до того, що вона була визнана стандартною моделлю для управління даними. Навіть з появою останнім часом все більшого числа різних баз даних NoSQL

реляційні бази даних залишаються домінуючим інструментом зберігання та організації даних.

Як реляційні бази даних структурують дані

Тепер, коли у вас є загальне розуміння історії реляційної моделі, давайте більш детально розглянемо, як дана модель структурує дані.

Найбільш значущими елементами реляційної моделі є відносини, які відомі користувачам і сучасним РСУБД як таблиці. Відносини - це набір кортежів, або рядків в таблиці, де кожен кортеж має набір атрибутів, або стовпців.

Стовпець - це найменша організаційна структура реляційної бази даних, що представляє різні осередки, які визначають записи в таблиці. Звідси походить більш формальна назва - атрибути. Ви можете розглядати кожен кортеж як унікального екземпляра чого-небудь, що може перебувати в таблиці: категорії людей, предметів, подій або асоціацій. Такими екземплярами можуть бути співробітники компаній, продажу в онлайн-бізнесі або результати лабораторних тестів. Наприклад, в таблиці з трудовими записами вчителів в школі кортежі можуть мати такі атрибути, як name, subjects, start_date і т. Д.

При створенні стовпців ви вказуєте тип даних, що визначає, які записи можуть вноситися в даний стовпець. РСУБД часто використовують свої власні унікальні типи даних, які можуть не бути безпосередньо взаємозамінні з аналогічними типами даних з інших систем. Деякі поширені типи даних включають дати, рядки, цілі числа і логічні значення.

У реляційної моделі кожна таблиця містить принаймні один стовпець, який можна використовувати для унікальної ідентифікації кожного рядка. Він називається первинним ключем. Це важливо, оскільки це означає, що користувачам не потрібно знати, де фізично зберігаються дані на комп'ютері. Їх СУБД може відстежувати кожну запис і повертати її в залежності від конкретної мети. У свою чергу, це означає, що записи не мають певного логічного порядку, і користувачі можуть повертати дані в будь-якому порядку або за допомогою будь-якого фільтра на свій розсуд.

Переваги та обмеження реляційних баз даних

З огляду на організаційну структуру, покладену в основу реляційних баз даних, давайте розглянемо їх деякі переваги і недоліки.

Сьогодні як SQL, так і бази даних, які її використовують, кілька відхиляються від реляційної моделі Кодда. Наприклад, модель Кодда наказує, що кожен рядок в таблиці повинна бути унікальною, а з міркувань практичної доцільності більшість сучасних реляційних баз даних допускають дублювання рядків. Є й ті, хто не вважає бази даних на основі SQL істинними реляційними базами даних, якщо вони не відповідають кожним критерієм реляційної моделі за версією Кодда. Але на практиці будь-яка СУБД, яка використовує SQL і в якійсь мірі відповідає реляційної моделі, може бути віднесена до реляційних систем управління базами даних.

Хоча популярність реляційних баз даних стрімко росла, деякий недоліки реляційної моделі стали проявлятися в міру того, як збільшувалися цінність і обсяги збережених даних. Наприклад, важко масштабувати реляційну базу даних горизонтально. Горизонтальне масштабування або масштабування по горизонталі - це практика додавання більшої кількості машин до існуючого стеку, що дозволяє розподілити навантаження, збільшити трафік і прискорити обробку. Часто це контрастує з вертикальним масштабуванням, яке передбачає модернізацію апаратного забезпечення існуючого сервера, як правило, за допомогою додавання оперативної пам'яті або центрального процесора.

Реляційну базу даних складно масштабувати горизонтально через те, що вона розроблена для забезпечення цілісності, тобто клієнти, які надсилають запити в одну і ту ж базу даних, завжди будуть отримувати однакові дані. Якщо ви масштабіруете реляційну базу даних горизонтально по всім машинам, буде важко забезпечити цілісність, тому що клієнти можуть вносити дані тільки в один вузол, а не в усі. Ймовірно, між початковою записом і моментом поновлення інших вузлів для відображення змін виникне затримка, що призведе до відсутності цілісності даних між вузлами.

Ще одне обмеження, яке існує в РСУБД, полягає в тому, що реляційна модель була розроблена для управління структурованими даними, або даними, які відповідають заздалегідь певного типу даних, або, по крайней мере, будь-яким чином попередньо організовані. Однак з поширенням персональних комп'ютерів і розвитком мережі Інтернет на початку 90-х років з'явилися неструктуровані дані, такі як електронні повідомлення, фотографії, відео та ін.

Але все це не означає, що реляційні бази даних марні. Навпаки, через понад 40 років, реляційна модель все ще є домінуючою основою для управління даними. Поширеність і довголіття реляційних баз даних свідчать про те, що це зріла технологія, яка сама по собі є головною перевагою. Існує багато додатків, призначених для роботи з реляційною моделлю, а також багато кар'єрних адміністраторів баз даних, які є експертами, коли справа доходить до реляційних баз даних. Також існує широкий спектр доступних друкованих та онлайн-ресурсів для тих, хто хоче почати роботу з реляційними базами даних.

Ще одна перевага реляційних баз даних полягає в тому, що майже всі РСУБД підтримують транзакції. Транзакція складається з одного або більше індивідуального вираження SQL, що виконується послідовно, як один блок роботи. Транзакції представляють підхід «все або нічого», що означає, що всі оператори SQL в транзакції повинні бути дійсними. В іншому випадку вся транзакція не буде виконана. Це дуже корисно для забезпечення цілісності даних при внесенні змін до кілька рядків або в таблиці.

Нарешті, реляційні бази даних демонструють надзвичайну гнучкість. Вони використовуються для побудови широкого спектра різних додатків і продовжують ефективно працювати навіть з великими обсягами даних [5]. Мова SQL також має величезний потенціал і дозволяє вам додавати або змінювати дані на льоту, а також вносити зміни в структуру схем баз даних і таблиць, не впливаючи на існуючі дані.

Висновок: завдяки гнучкості і проектного рішення, спрямованого на збереження цілісності даних, через п'ятдесят років після появи такого задуму, реляційні бази даних все ще є основним способом управління даними і їх

зберігання. Навіть зі збільшенням в останні роки числа різноманітних баз даних NoSQL розуміння реляційної моделі і принципів її роботи з РСУБД є ключовим моментом для всіх, хто хоче створювати додатки, що використовують можливості даних [6].

Користувацькі інтерфейси для бази даних

У сучасних системах програмування більшість програм супроводжують користувацьким інтерфейсом, який є невід'ємною частиною роботи СУБД.

Користувацький інтерфейс - це комплекс програм (які недоступні для перегляду користувачу), котрий організовує діалог з користувача з базою даних та реалізує доступ користувача до інформації, яка зберігається с базі даних.

Користувацький інтерфейс створює для користувача можливість зручної взаємодії с даними, наприклад, дає можливість отримувати інформацію зі списків, представлених на екрані, забезпечує швидкий пошук необхідних даних, редагування змісту таблиць тощо. Інтерфейс для користувача невід'ємна частина програми, її дизайн, методи взаємодії с користувачем впливають на ефективність використання програми. Зручний, красивий та ефективний інтерфейс впливає на масовість конкретного програмного забезпечення.

Скорингові моделі СУБД. Скорингова система отримала свою назву від англійського слова «score», що перекладається як рахунок або підрахунок очок. У моїй заліковій роботі застосовується скоринг для оцінки платоспроможності користувача. Клієнт на сайті, при про подачі заявки на навчання, проходить обов'язкове анкетування. Його професійні, демографічні та соціальні характеристики мають певний бал. Програма автоматично «підраховує очки» з анкети і видає результат про потенціал користувача. Менеджер команії на підставі скорингової оцінки вирішує: кому першому зателефонувати зі списку поданих заявок.

В основі скорингових систем існує припущення, що люди зі схожими соціальними показниками поводяться однаково. Апріорно приймаючи такий постулат, можна будувати різні статистичні моделі вельми корисні при веденні будь-якого бізнесу.

Якщо деяким соціальним характеристикам клієнта (стать, вік, місце проживання, посада, тривалість роботи в одному місці тощо) присвоїти певні коефіцієнти ваги, то кожного нового клієнта можна, на основі його анкети, віднести до групи сильно або слабо відповідних бізнесу. Тобто, клієнту автоматично присвоюється певна оцінка, який вказує ступінь довіри і увагу, яку йому слід надавати з боку даного бізнесу.

Можна виділити наступні етапи побудови скорингу:

- визначення певної характеристики, яка нас цікавить;
- збір другорядних відомостей про клієнтів і значення певних характеристик;
- розробка скорингової моделі (присвоєння коефіцієнтів ваг другорядним даними) на основі наявних даних;
- автоматичне ранжування нових клієнтів по пріоритетним групам за допомогою скорингової моделі.

Якщо в якості цікавить характеристики взяти здатність клієнта зробити певну покупку, тоді в результаті ми отримаємо дві групи: клієнти, яким потрібно телефонувати в першу чергу і клієнти, які "почекають".

1.2. Призначення розробки та галузь застосування

Інформаційна система, що виконана для кваліфікаційної роботи, має назву «Розробка додатку для аналітики користувацьких даних та побудови скорингових моделей у СУБД».

Основні терміни та ключові слова:

MySQL - безкоштовна реляційна система управління базами даних. Розробка та підтримка сайта MySQL здійснює корпорація Oracle, яка отримала права на торговельну марку разом з поглиненої Sun Microsystems, яка раніше придбала шведську компанію MySQL AB. Продукт поширюється як під GNU General Public License, так і під власною комерційною ліцензією. Крім цього, розробники створюють функціональність за замовленням ліцензійних

користувачів. Саме завдяки такому замовленню майже в найраніших версіях з'явився механізм реплікації.

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.

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.

Призначення розробки - надати можливість користувачам СУБД експортувати користувацькі дані для їх обробки та аналізу. Моя 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. Підстави для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується з наказом ректора.

Таким чином підставами для розробки (виконанням кваліфікаційної роботи) є:

- освітня програма спеціальності 122 “Комп’ютерні науки”;

- навчальний план та графік навчального процесу;

- наказ ректора Національного технічного університету “Дніпровська політехніка” № 317-с від 07.06.2021р;

- завдання на кваліфікаційну роботу на тему “Розробка мобільного додатку на Android, що вимірює фізичні розміри об’єктів за допомогою камери телефону”.

1.4. Постановка завдання

Метою проекту є розробити додаток для аналітики користувацьких даних та побудови скорингових моделей у СУБД, а також збереження та обробки користувацької інформації.

Даний продукт дозволить зберігати великий об’єм інформацію у зручному для користувача вигляді та прогнозувати вірогідність оплати послуг користувачем.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Кінцевий продукт повинен дотримуватися наступних функціональних вимог:

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

1.5.2. Вимоги до інформаційної безпеки

Організація безпечної і актуальною настройки СУБД. 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 ПО. 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 область входять завдання організації даних з контролем цілісності та інші, специфічні для СУБД проблеми безпеки. Фактично це завдання включає в себе основний обсяг залежать від даних вразливостей і захисту від них.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для підтримки стабільної роботи СУБД, якщо сервер знаходиться локально 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;
- 6 гігабайт вільного місця на диску.

1.5.4. Вимоги до інформаційної та програмної сумісності

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

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

РОЗДІЛ 2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Результатом даної кваліфікаційної роботи має бути застосунок, який за виконує розрахунки платоспрожності користувача на основі певних характеристик, які беруться з СУБД 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. Опис застосованих математичних методів

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].

Припустимо, що у нас є два класи і алгоритм, який пророкує приналежність кожного об'єкта одному з класів, тоді матриця помилок класифікації (рис. 2.2) 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. 2.2 матриця помилок класифікаці

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, ми хочемо оцінити роботу спам-фільтра пошти. У нас є 100 НЕ-спам листів, 90 з яких наш класифікатор визначив вірно (True Negative = 90, False Positive = 10), і 10 спам-листів, 5 з яких класифікатор також визначив вірно (True Positive = 5, False Negative = 5).

Тоді accuracy:

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

Однак якщо ми просто будемо передбачати всі листи що не-спам, то отримаємо більш високу accuracy:

$$accuracy = \frac{0 + 100}{0 + 100 + 0 + 10} = 90,9 \quad (2.3)$$

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

Precision, recall і F-міра

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

$$precision = \frac{TP}{TP + FP} \quad (2.4)$$

$$recall = \frac{TP}{TP + FN} \quad (2.5)$$

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.

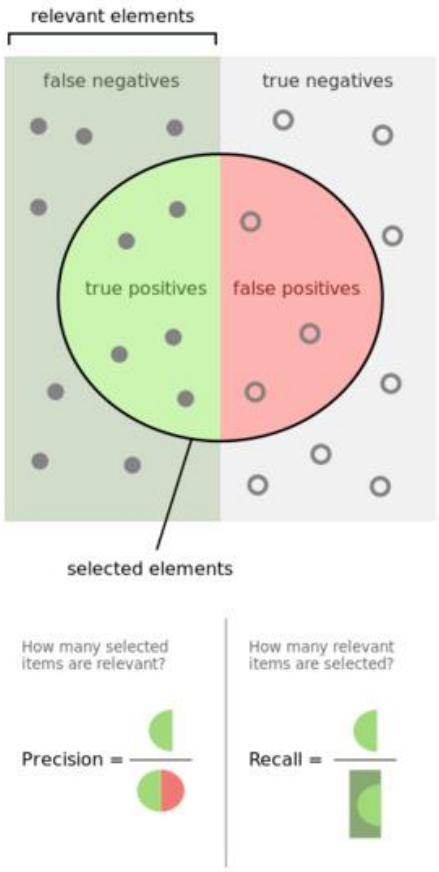

Рис. 2.3 графічне зображення метрик precision і recall

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

Як ми зазначали 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.

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 ми не можемо знаходити всіх клієнтів, які хочуть купити продукт. Але, визначивши стратегію і ресурс для виявлення платоспроможних клієнтів клієнтів, ми можемо підібрати потрібні пороги по precision і recall.

Наприклад, можна зосередитися обзвоні клієнтів лише с високим скорингом, так як ми обмежені в ресурсах колл-центру.

2.3. Опис використаних 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 СУБД підтримують з різним ступенем відповідності стандарт SQL92.

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 - СУБД - мають різні реалізації, такі як ORACLE, MS SQL, MYSQL. Мова SQL в різних СУБД має невеликі відмінності, наприклад, в детальному синтаксисі опису операторів. Такі відмінності присутні в спеціальних функціях, що відносяться до тієї чи іншої СУБД, але все ж в основному мова - це загальний синтаксис, практично ідентичний для будь-якої СУБД.

Реляційна база даних 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 (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;
- 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 С ++.

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 [7]. Мова програмування Python була створена у 1991 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.

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.

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

У своїх кваліфікаційній роботі я використовую Pytnon бібліотеку pandas. Pandas - це високорівнева Python бібліотека для аналізу даних. Чому я її називаю високорівневою, тому що побудована вона 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 [11].

Також я використав бібліотеку matplotlib. Matplotlib - це бібіліотека двовимірної графіки для мови програмування python за допомогою якої можна створювати високоякісні малюнки різних форматів (рис.2.2). Matplotlib є модуль-пакет для python.

Для вирішення завдань класичного машинного навчання я обрав 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 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.

2.4. Опис структури системи та алгоритмів її функціювання

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

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

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

Нижче перераховані деякі рекомендації, які допоможуть домогтися ефективної структури:

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

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

Перша нормальна форма відповідає наступним вимогам:

- в відношенні немає однакових кортежів;
- кортежі не впорядковані;
- атрибути не впорядковані і розрізняються по найменуванню;
- всі значення атрибутів атомарний.

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.

В даній кваліфікаційній роботі фігурують наступні сутності та їх атрибути:

1. Таблиця користувачі (users). Атрибути:
   - ID користувача;
   - електронна адреса;

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

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

4. Таблиця вчителі (teachers). Атрибути:
    - ID вчителя;
    - національність;
    - стать;
    - список мов, якими володіє вчитель;
    - вік;
    - ім’я
    - бінарне поле, яке показує вчитель носій мови чи ні;
    - ID користувача;
    - рівень знань вчителя.
5. Таблиця ввідних занять (skype_lesson_interview). Атрибути:
    - ID ввідного заняття;
    - ID студента;
    - ID вчителя;
    - дата проходження заняття;
    - статус проходження заняття;
    - граматичні навички;
    - навички спілкування;
    - навички аудіювання;
    - рекомендований курс.
6. Таблиця оплат (payment_transaction). Атрибути:
    - ID транзакції;
    - кількість придбаних уроків;
    - тип вчителя;
    - ціна у валюті оплати;
    - валюта оплати;
    - дата транзакції;
    - ID користувача.

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

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

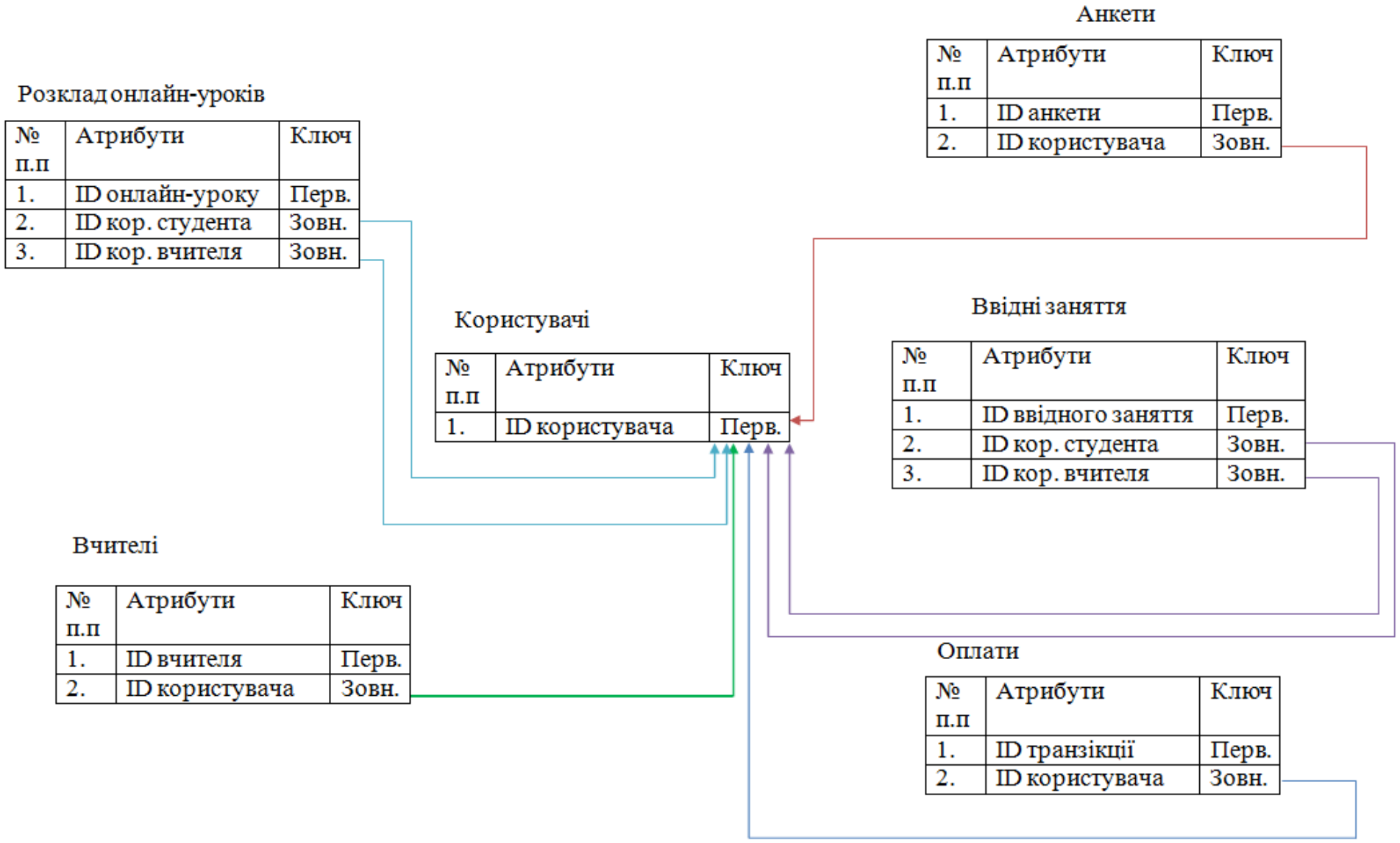

Рис. 2.4 Реляційна БД

На реляційній моделі я показав зв’язки 2-6 таблиць з таблицею користувачів. Усі таблиці поєднані між собою зовнішнім ключем (ID користувача), я не став відображати усі зв’язки заради більшої наглядності 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 [9]. 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 [20]. 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.

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 [15]. 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.

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 [13].

Перенавчання відбувається, коли ми використовуємо дуже гнучку модель (з високою місткістю), яка просто запам'ятовує навчальний набір даних, підганяючи вузли під нього [13]. 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 достатньою гнучкістю для відповідності нелінійним структурам. Ригідна модель може виявитися недостатньо ємною навіть для відповідності тренувальним даними.

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.

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 (рис.2.5) - 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.

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. 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:

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

ROC-крива (рис.2.5) - 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 негативних прикладів.

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.

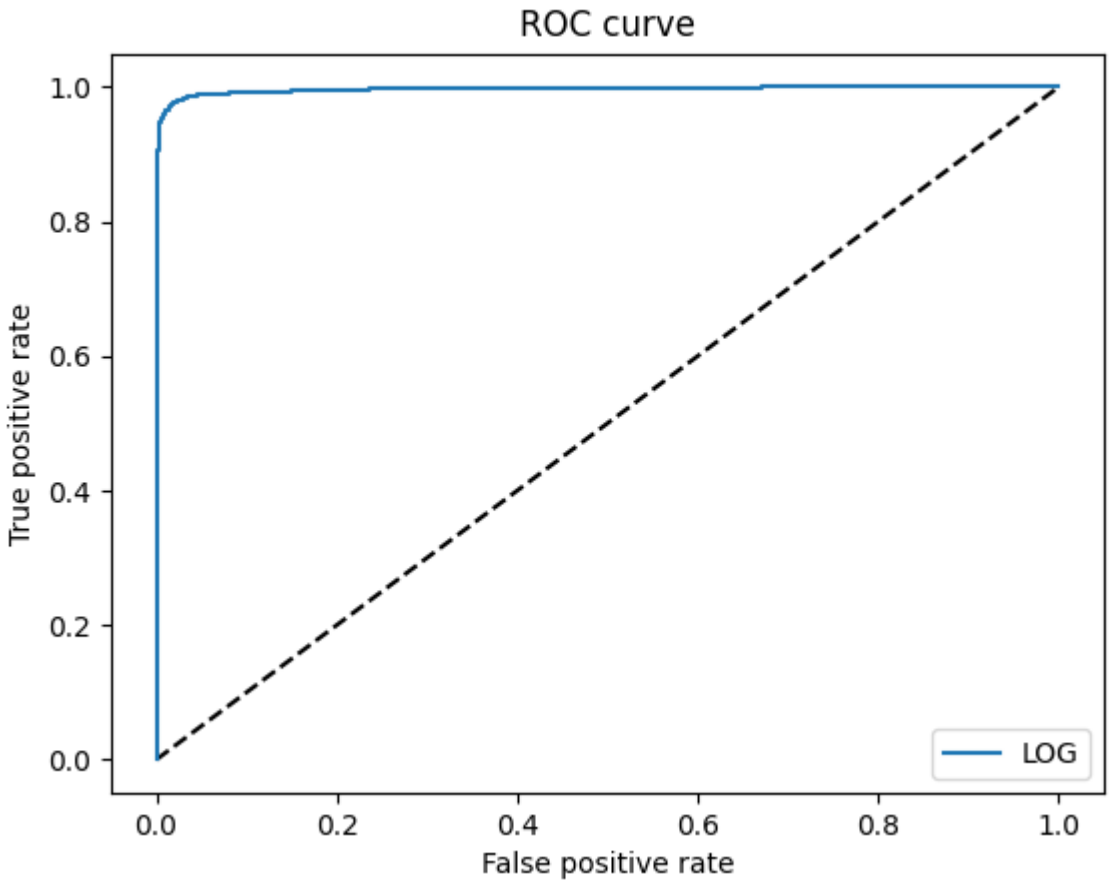

Рис. 2.5 крива ROC

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 лінія відповідає «марної» 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 передбачувальну здатність моделі.

2.5. Обґрунтування та організація вхідних та вихідних даних програми

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 нам необхідні тренувальні данні 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, ми хочемо 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 ми хочемо 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 ми отримуємо графік кривої ROC (рис. 2.5). Оцінка скорингу формується в окремому файлі у форматі «.csv».

2.6. Опис розробленої системи

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. Використані технічні засоби

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 засоби

Для своєї інформаційної системи я обрав середовище MySQL Workbench.

MySQL Workbench (рис. 2.6) - це уніфікований візуальний інструмент для архітекторів баз даних і розробників БД. MySQL Workbench надає можливість моделювання даних, розробку SQL і комплексні інструменти адміністрування для конфігурації сервера, адміністрування користувачів, резервного копіювання та багато іншого. MySQL Workbench доступний на Windows, Linux і Mac OS X.

MySQL Workbench дозволяє DBA, розробнику або архітекторові даних візуально проектувати, моделювати, генерувати і управляти базами даних. Він включає в себе все, що потрібно розробнику моделей даних для створення складних ER-моделей, прямого і зворотного проектування, а також надає ключові функції для виконання складних завдань управління змінами та документування, які зазвичай вимагають багато часу і зусиль.

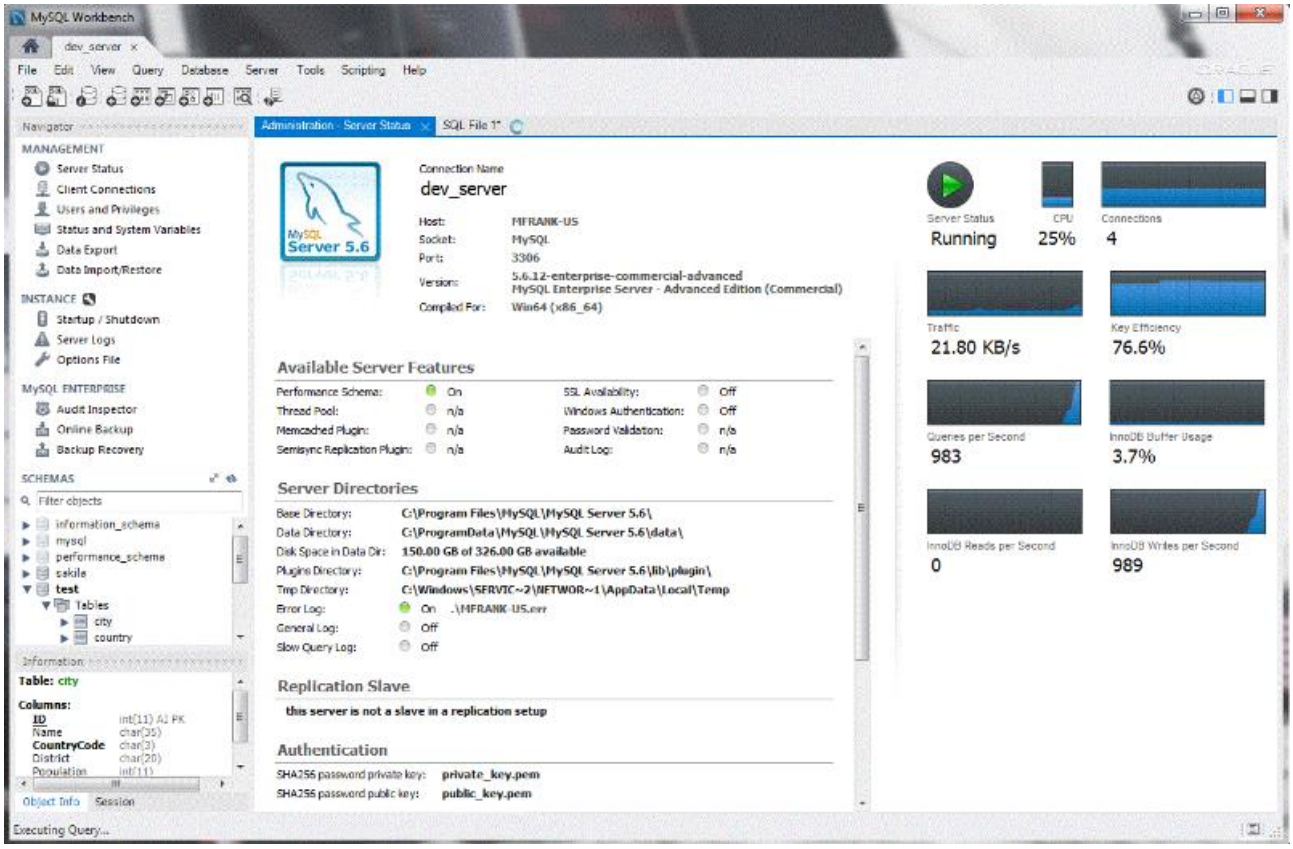

Рис. 2.6 Головна сторінка застосунку MySQL Workbench

У якості середовища програмування я обрав ПЗ Geany (рис. 2.7.2.2). Geany - середовище розробки програмного забезпечення, написана з використанням бібліотеки GTK +. Доступна для наступних операційних систем: BSD, Linux, Mac OS X, Solaris і Windows. Geany поширюється згідно GNU General Public License.

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

Основні функції програмного середовища Geany:

- підсвічування вихідного коду з урахуванням синтаксису мови програмування (мова визначається автоматично по розширенню файлу);
- автозавершення слів;

- автоматична підстановка закривають тегів HTML / XML. Автопідстановка стандартних і існуючих у відкритих файлах функцій;
- простий менеджер проектів;
- підтримка плагінів;
- вбудований емулятор терміналу;
- підтримка великої кількості кодувань;
- гнучкий інтерфейс;
- можливість використання і створення фрагментів. Для цього використовується спеціальний файл snippets.conf в каталозі /home/user/.config/geany дозволяє створювати свої сніппети;
- можливість використання і створення шаблонів файлів. Шаблони повинні бути розташовані в каталозі /home/user/.config/geany/templates/files;
- налагодження коду за допомогою модуля (плагіну) GeanyGDB (використовує відладчик GDB).

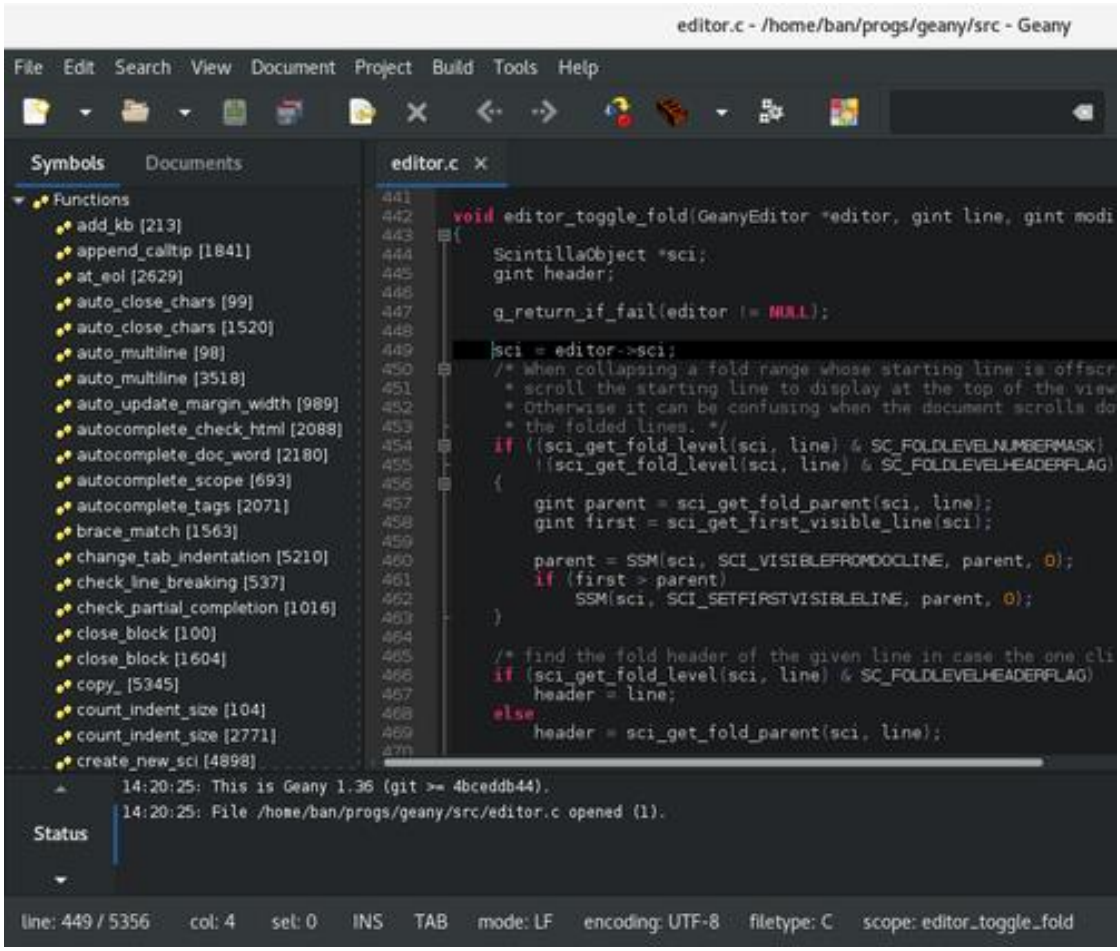

Рис. 2.7 Інтерфейс Geany

2.6.3. Виклик та завантаження програми

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

2.6.4. Опис інтерфейсу користувача

Після запуску додатку з’явиться віконце інтерфейсу (рис. 2.8). Необхідно обрати файли для тестової та тренувальної бази розрахунків, увести значення коефіцієнтів там показник ітерації. Після чого натискаємо на кнопку Submit, або Cancel якщо хочему завершити роботу програми.

Рис. 2.8 Інтерфейс скорингу

Після натиску кнопки Submit починаються розрахунки (рис. 2.9).

Рис. 2.9 Консольне вікно, яке повідомляє про початок розрахунків

Після виконання першої фази розрахунків з’являється два графіки: крива ROC (рис. 2.10) та крива (рис. 2.11), яка показує точку оптимального балансу для розділення бази.

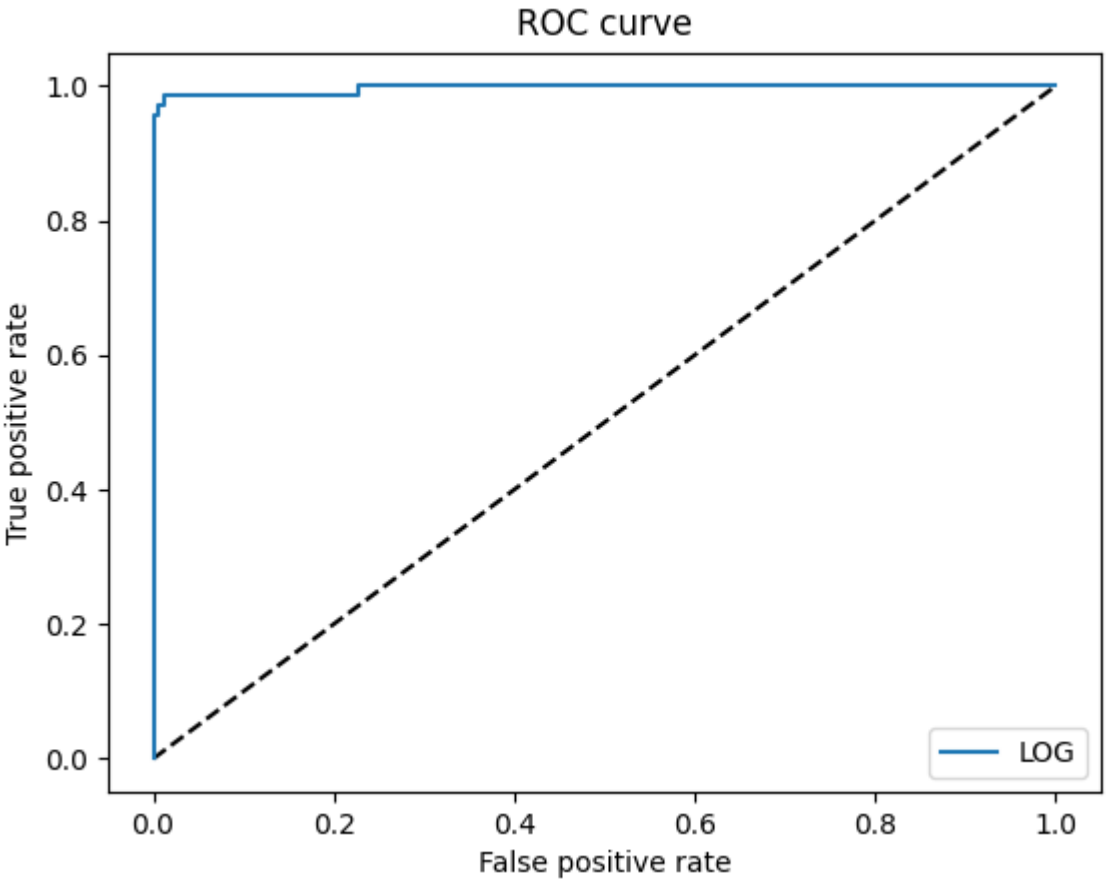

Рис. 2.10 Крива ROC

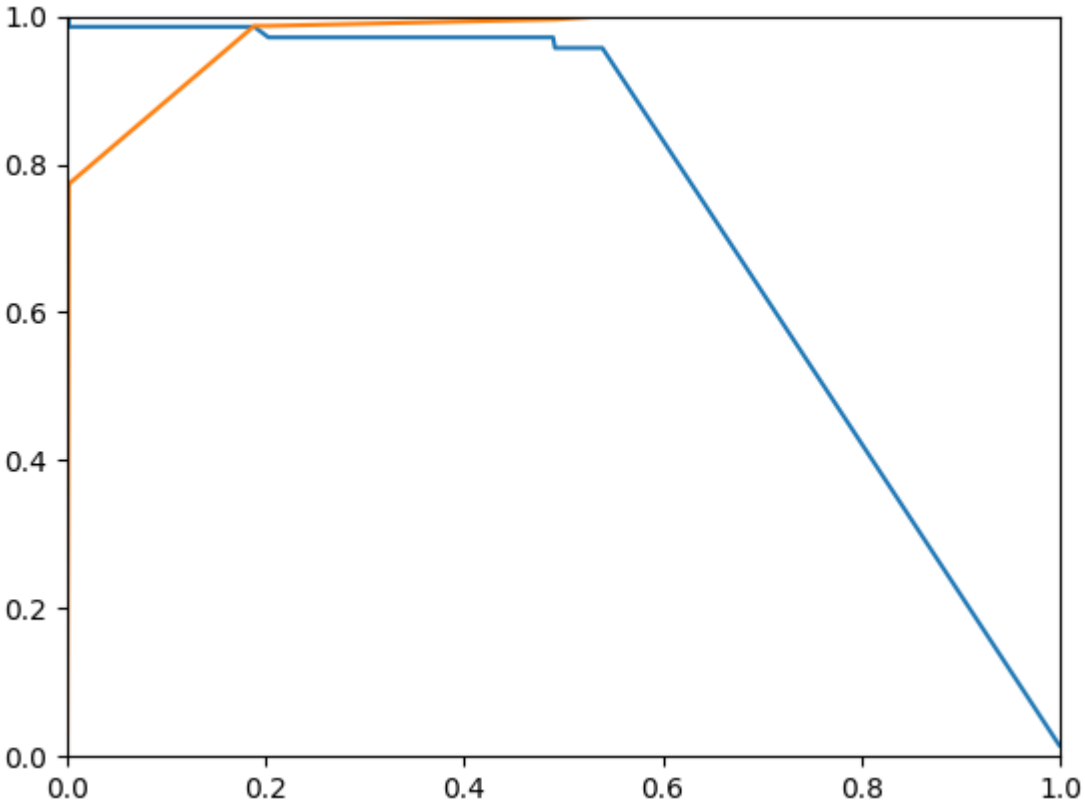

Рис. 2.11 Крива, яка точку оптимального балансу для розділення бази

Після закриття віконець з графіками перед нами з’являється консольне віконце з результатами розрахунків (рис. 2.12).

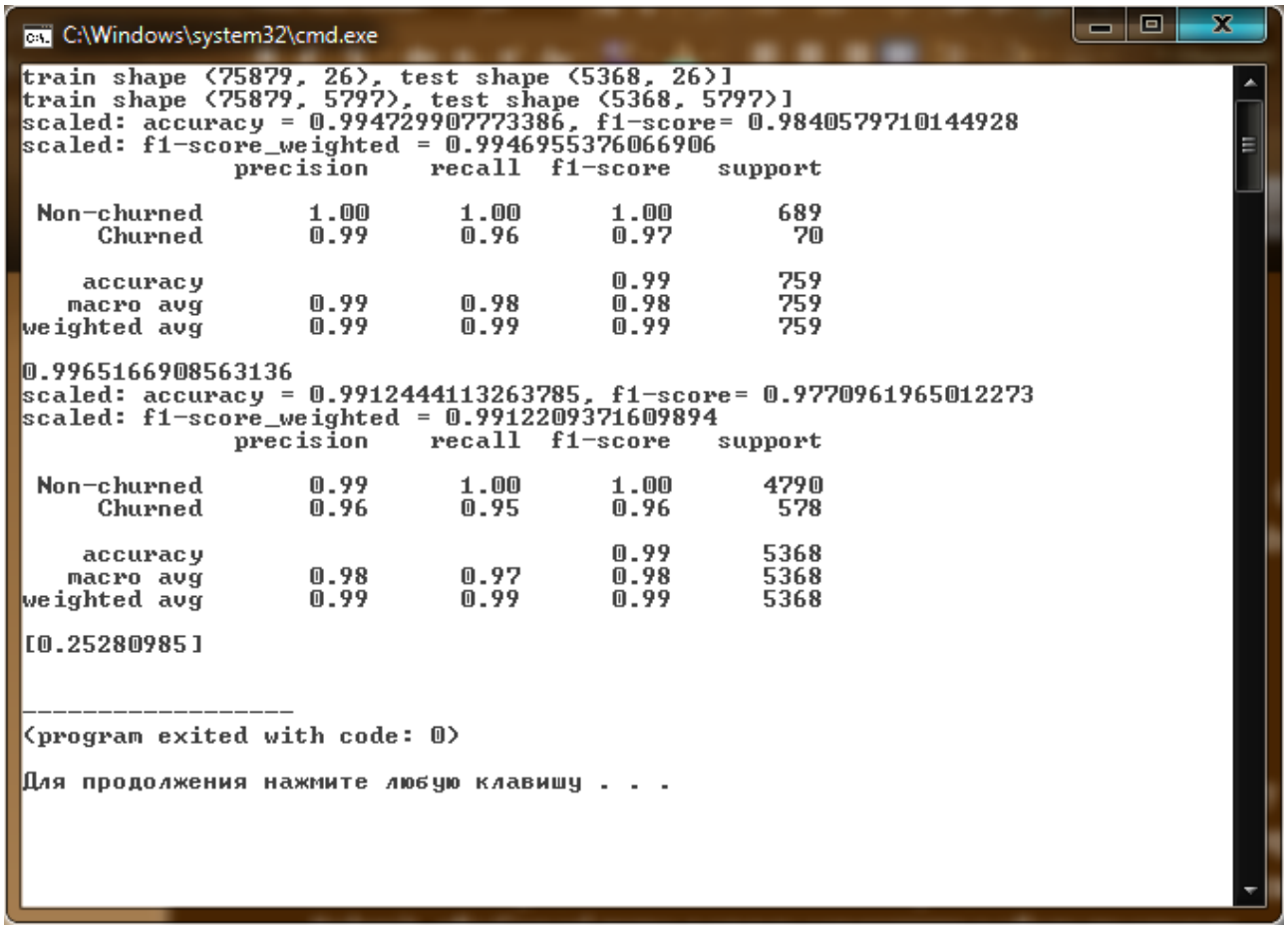

Рис. 2.12 Консольне віконце з мат. показниками

Після завершення розрахунків формується файл у форматі «.csv». з результатами розрахунків та скорингом для користувачів тестової бази. Файл формується у спільній директорії нашого програмного засобу.

РОЗДІЛ 3 ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми - 1750;
2. коефіцієнт складності програми – 1,3;
3. коефіцієнт корекції програми в ході її розробки – 0,08;
4. годинна заробітна аналітика бази даних – 124 грн/год;

Середня година зарплати аналітика бази даних була вирахувана виходячи з даних «Української спільноти програмістів (DOU)». Середньоукраїнська заробітна плата аналітика, який володіє мовою запитів SQL та мовою програмування Pyhton з досвідом роботи близько року дорівнює 800 американських доларів у місяць. При курсі валют НБУ на початок червня 2021 року один американський долар дорівнює 27,34 грн, тому середня зарплата в гривнях дорівнює 21870 грн. При восьмигодинному робочому дні (176 годин у місяць в середньому) середня зарплата за годину буде становити 124 грн.

5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,3;
7. вартість машино-години ЕОМ – 14 грн/год

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин}, \qquad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

$t_и$– витрати праці на дослідження алгоритму рішення задачі;

$t_а$– витрати праці на розробку блок-схеми алгоритму;

$t_п$– витрати праці на програмування по готовій блок-схемі;

$t_{отл}$– витрати праці на налагодження програми на ЕОМ;

$t_д$– витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q=q\cdot C\cdot(1+p), \text{ де} \qquad (3.2)$$

q– передбачуване число операторів;

C– коефіцієнт складності програми;

p– коефіцієнт кореляції програми в ході її розробки.

$$Q= 1750\cdot 1{,}3\cdot(1+0{,}08)=2457;$$

Витрати праці на вивчення опису задачі $t_и$ визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_и = \frac{Q \bullet B}{(75\ldots85) \bullet K}, \text{ людино-годин}, \qquad (3.3)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

K – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності;

$$t_и = \frac{2457\cdot 1{,}2}{78\cdot 1{,}3} = 29{,}07, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_n=\frac{Q}{(20\ldots25)\cdot k}, \text{ людино-годин.} \qquad (3.4)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.4), людино-годин:

$t_a = \frac{2457}{20 \cdot 1,3} = 94,5$, людино-годин.

Витрати на складання програми по готовій блок-схемі:

$t_n = \frac{Q}{(20 \ldots 25) \cdot k}$, людино-годин. (3.5)

$t_a = \frac{2457}{25 \cdot 1,3} = 75,6$, людино-годин.

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$t_{отл} = \frac{Q}{(4..5) \cdot k}$, людино-годин. (3.6)

$t_n = \frac{2457}{5 \cdot 1,3} = 378$, людино-годин,

- за умови комплексного налагодження завдання:

$t_{отл}^{к} = 1,2 \cdot t_{отл}$; (3.7)

$t_{отл}^{к} = 1,2 \cdot 378 = 453,6$, людино-годин

Витрати праці на підготовку документації:

$t_д = t_{др} + t_{до}$; (3.8)

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису

$t_д = \frac{Q}{(15 \ldots 20) \cdot K}$; (3.9)

$t_д = \frac{2457}{18 \cdot 1,3} = 105$ людино-годин.

$t_{до}$– трудомісткість редагування, печатки й оформлення документації

$t_{до} = 0,75 \cdot t_{др}$; (3.10)

$t_{до} = 0,75 \cdot 105 = 78,75$, людино-годин.

$t_д = 105 + 78,75 = 183,75$, людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення:

t =50+29,07+75,6+378+453,6+183,75=1170,02, людино-годин.

У результаті ми розрахували, що в загальній складності необхідно 1170,02 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \quad (3.11)$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн,} \quad (3.12)$$

де t – загальна трудомісткість, людино-годин;

$C_{ПР}$– середня годинна заробітна плата програміста, грн/година

$З_{ЗП} = 1170{,}02 \cdot 124 = 145330{,}48$, грн.

$З_{МВ}$– Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot C_{М}, \text{ грн,} \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$C_{МЧ}$– вартість машино-години ЕОМ, грн/год.

$З_{МВ} = 1170{,}02 \cdot 14 = 16380{,}28$, грн.

$К_{ПО} = 145330{,}48 + 16380{,}28 = 161710{,}76$, грн.

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ мес.} \quad (3.14)$$

де B_k- число виконавців;

F_p– місячний фонд робочого часу (при 40 годинному робочому тижні Fp=176 годин).

$$T = \frac{1170,02}{1 \cdot 176} = 6,6 \text{міс.}$$

Висновки. На розробку даного програмного забезпечення піде 1170,02 людино-годин. Тобто, ймовірна очікувана тривалість розробки складатиме 6,6 місяці при стандартному 40-годинному робочому тижні і 176-годинному робочому місяці. Очікувані витрати на створення програмного забезпечення складатимуть 145330,48 грн.

ВИСНОВКИ

В даній кваліфікаційній роботі був розроблений додаток для аналітики користувацьких даних та побудови скорингових моделей у СУБД.

Дане програмне забезпечення призначене для аналізу великих об’ємів користувацьких даних для отримання оцінки скорингу для кожного користувача за певний період часу. Додаток надасть можливість побудови скорингових моделей для оцінки якості виконання аналізу. На основі цієї оцінки, ми зможемо зрозуміти коректність вхідних данних, вдалість підбору вагових коефіцієнтів чи показника ітерації.

На практиці подібний аналіз даних дозволить оптимізувати роботу call-центру, адже людський ресурс доволі обмежений. Мій додаток дозволить присвоїти оцінку скорингу кожному користувачу окремо, таке рішення дозволить менеджерам працювати більш ефективно та контактувати з цільовою аудиторією онлайн-школи.

Під час виконання даного кваліфікаційної роботи були виконані наступні задачі:

- проаналізовано предметну область задачі, що розв’язується;
- з’ясовані вимоги та підстави для розробки даної інформаційної системи;
- обрано раціональну структуру і параметри програми;
- створена реляційна база данних;
- створено інтерфейс для аналізу користувацьких даних;
- розроблено рекомендації щодо використання програми.

База даних створена за допомогою мови SQL у середовищі MySQL Workbench. Механізм аналізу даних реалізовано за допогою мови програмування Python, графічний інтерфейс додатку був створений за допогою Python-бібліотеки PySimpleGUI.

Також у кваліфікаційній роботі було визначено трудомісткість розробленого програмного продукту (1170,02 люд-год), проведений підрахунок вартості роботи по створенню програми (145330,48) грн. та розраховано час на його створення (6,6 міс).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Anthony Molinaro. SQL Cookbook: Query Solutions and Techniques for Database Developers, 1st Edition. - O'Reilly Media, 2005. - 877 p.
2. Alan Beaulieu. Learning SQL: Master SQL Fundamentals, 2nd Edition. - O'Reilly Media, 2009. - 388p.
3. Itzik Ben-Gan. T-SQL Fundamentals, 3rd Edition. Microsoft Press, 2016. - 464p.
4. Stephane Faroult, Peter Robson. The art of SQL, 1st Edition. - O'Reilly Media, 2006. - 370 p.
5. Upon Malik, Matt Goldwasser, Benjamin Johnston. SQL for Data Analytics: Perform fast and efficient data analysis with the power of SQL, 1st Edition. - Packt Publishing, 2019. - 437p.
6. Upon Malik, Matt Goldwasser, Benjamin Johnston. The Applied SQL Data Analytics Workshop: Develop your practical skills and prepare to become a professional data analyst, 2nd Edition. - Packt Publishing, 2020. - 546p.
7. Michael Walker. Python Data Cleaning Cookbook: Modern techniques and Python tools to detect and remove dirty data and extract key insights, 1st Edition. - Packt Publishing, 2020. - 436p.
8. Felix Zumstein. Python for Excel: A Modern Environment for Automation and Data Analysis 1st Edition. - O'Reilly Media, 2021. - 565p.
9. Imran Ahmad. 40 Algorithms Every Programmer Should Know: Hone your problem-solving skills by learning different algorithms and their implementation in Python, 1st Edition. - Packt Publishing, 2020. - 384p.
10. Hong Zhou. Learn Data Mining Through Excel: A Step-by-Step Approach for Understanding Machine Learning Methods, 1st Edition. Apress, 2020. - 236p.
11. Stephen Klosterman. Data Science Projects with Python: A case study approach to successful data science projects using Python, pandas, and scikit-learn, 1st Edition. - Packt Publishing, 2019. - 374p.

12. Matt Harrison. Pandas 1.x Cookbook: Practical recipes for scientific computing, time series analysis, and exploratory data analysis using Python, 2nd Edition. - Packt Publishing, 2020. - 816p.

13. Jaime Buelta. Python Automation Cookbook: 75 Python automation ideas for web scraping, data wrangling, and processing Excel, reports, emails, and more, 2nd Edition. - Packt Publishing, 2020. - 528p.

14. Gordon S. Linoff. Data Analysis Using SQL and Excel, 2nd Edition. - Wiley, 2015. - 729p.

15. David Mertz. Cleaning Data for Effective Data Science: Doing the other 80% of the work with Python, R, and command-line tools, 1st Edition. - Packt Publishing, 2021. - 498p.

16. Jim Frost. Regression Analysis: An Intuitive Guide for Using and Interpreting Linear Models, 1st Edition. - Statistics By Jim Publishing, 2020. -367 p.

17. Jim Frost. Introduction to Statistics: An Intuitive Guide for Analyzing Data and Unlocking Discoveries, 1st Edition. - Statistics By Jim Publishing, 2020.

-276 p.

18. Giuseppe Ciaburro. Hands-On Simulation Modeling with Python: Develop simulation models to get accurate results and enhance decision-making processes, 1st Edition. - Packt Publishing, 2020. - 431p.

19. Hyatt Saleh. The Machine Learning Workshop: Get ready to develop your own high-performance machine learning algorithms with scikit-learn, 1st Edition. - Packt Publishing, 2020. - 309p.

20. Tproger «Реалізація та розбір алгоритму «випадковий ліс» на Python».[Електронний ресурс]. Режим доступу: https://tproger.ru/translations/python-random-forest-implementation/

21. Loginom «Логістична регресія і ROC-аналіз - математичний апарат». [Електронний ресурс]. Режим доступу: https://loginom.ru/blog/logistic-regression-roc-auc

22. Open Net «Основи мови SQL». [Електронний ресурс]. Режим доступу: https://www.opennet.ru/docs/RUS/sql/

ДОДАТОК А

КОД ПРОГРАМИ

Лістнг створення БД

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

  PRIMARY KEY (`teacher_id`));

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

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

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

```
  ON UPDATE CASCADE;

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


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

  PRIMARY KEY (`user_id`));

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

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

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

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

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


  PRIMARY KEY (`application_id`));

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

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

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

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

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

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

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



  PRIMARY KEY (`lesson_id`));

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

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

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

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

  PRIMARY KEY (`skype_lesson_interview_id`));

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

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

```
  ON DELETE CASCADE
  ON UPDATE CASCADE;

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

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

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


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

Запит для виводу вхідних даних у БД:

Відраховуємо рік від того місяця, скоринг якого потрібно розрахувати (приклад - потрібно розрахувати скоринг для листопада 2020 року - тоді дати в скрипті будуть 2019-11-01 - 2020-11-01), місяць розрахунку не включаємо. Встановлюємо @dt_now як сьогодні мінус 1 місяць (тобто для розрахунку 5.12 @dt_now = '2020-11-05').

```
SET @dt_now = '2020-06-03';

SELECT
  user_id, stud,
  datediff(dt_request,dt_registration) as days_before_req,
  ifnull(datediff(dt_max_intro,dt_request),'(no_intro)') as days_btw_req_intro,
  #ifnull(datediff(@dt_now,dt_max_intro),'(no_intro)') as days_after_intro,
  #first_lesson,
  ifnull(if(stud=1,datediff(first_lesson,dt_max_intro),datediff(@dt_now,dt_max_intro)),'(no_intro)') as days_after_intro,
  ifnull(if(stud=1,datediff(first_payment,dt_request),datediff(@dt_now,dt_request)),'(no_paymetns)') as days_btw_req_pay,
  dt_request,hour(dt_request) as hour_req,
  dayofweek(dt_request) as dayofweek_req,
  month(dt_request) as month_req,
  edwords_user_before_appl,
  gmt,
  IF(from_page_pretty = '',
      '/',
      IF(from_page_pretty REGEXP '#',
      '/',
      from_page_pretty)) AS from_page_pretty,
  from_page_hash,
  (select count(distinct student_id) from ebook_lessons_interview where student_id=v.user_id and dt_start>=v.dt_request and dt_start<@dt_now) request_intro,
```

```
(select count(distinct student_id) from ebook_lessons_interview where status='complete'
and student_id=v.user_id and dt_start>=v.dt_request and dt_start<@dt_now) complete_intro,
#(select count(distinct student_id) from ebook_lessons_interview where student_id=v.user_id
and dt_start>=v.dt_request) request_intro_all,
#(select count(distinct student_id) from ebook_lessons_interview where status='complete'
and student_id=v.user_id and dt_start>=v.dt_request ) complete_intro_all,
(select count(user_id) from payment_log where lesson_type='individual' and
user_id=v.user_id and create_date>=v.dt_request and create_date<@dt_now) payments_ind_all,
#(select count(user_id) from payment_log where lesson_type='individual' and
order_state='wait' and user_id=v.user_id and create_date>=v.dt_request) payments_ind_wait,
#(select count(user_id) from payment_log where lesson_type='individual' and
order_state='accepted' and user_id=v.user_id and create_date>=v.dt_request)
payments_ind_accepted,
(select count(user_id) from payment_log where lesson_type<>'individual' and
order_state='wait' and user_id=v.user_id and create_date>=v.dt_request and create_date<@dt_now)
payments_others_wait,
(select count(user_id) from payment_log where lesson_type<>'individual' and
order_state='accepted' and user_id=v.user_id and create_date>=v.dt_request and
create_date<@dt_now) payments_others_accepted,
#           (select teacher_id from ebook_lessons_interview where status='complete' and
student_id=v.user_id and dt_start>=v.dt_request and dt_start<date_add(dt_request,interval 7 day))
complete_intro,


if((datediff(@dt_now,birthday)/365)>=18 and
segment='kids','individual',if((datediff(@dt_now,birthday)/365)<18 and
segment='individual','kids',segment)) as segment,
country,
gender,
#ifnull(round(datediff(@dt_now,birthday)/365),'(not_set)') as age,


case
        when round(datediff(@dt_now,birthday)/365)<10 then "<10"
```

```
                when round(datediff(@dt_now,birthday)/365)>=10 and
round(datediff(@dt_now,birthday)/365)<15 then "[10-15)"

                when round(datediff(@dt_now,birthday)/365)>=15 and
round(datediff(@dt_now,birthday)/365)<20 then "[15-20)"

                when round(datediff(@dt_now,birthday)/365)>=20 and
round(datediff(@dt_now,birthday)/365)<25 then "[20-25)"

                when round(datediff(@dt_now,birthday)/365)>=25 and
round(datediff(@dt_now,birthday)/365)<30 then "[25-30)"
                when round(datediff(@dt_now,birthday)/365)>=30 and
round(datediff(@dt_now,birthday)/365)<35 then "[30-35)"
                when round(datediff(@dt_now,birthday)/365)>=35 and
round(datediff(@dt_now,birthday)/365)<40 then "[35-40)"
                when round(datediff(@dt_now,birthday)/365)>=40 and
round(datediff(@dt_now,birthday)/365)<45 then "[40-45)"
                when round(datediff(@dt_now,birthday)/365)>=45 and
round(datediff(@dt_now,birthday)/365)<50 then "[45-50)"
                when round(datediff(@dt_now,birthday)/365)>=50 and
round(datediff(@dt_now,birthday)/365)<55 then "[50-55)"
                when round(datediff(@dt_now,birthday)/365)>=55 and
round(datediff(@dt_now,birthday)/365)<60 then "[55-60)"
                when round(datediff(@dt_now,birthday)/365)>=60 and
round(datediff(@dt_now,birthday)/365)<65 then "[60-65)"
                when round(datediff(@dt_now,birthday)/365)>=65 then ">=65"
                else "(not_set)"
                end as age_range,

        Source_app,
        Medium_app,
        if(Medium='invite','friend',Source) as Source,
        Medium,
```

```
        if(english_level_id is null or english_level_id='',2,english_level_id) as
english_level_id,
        if(field_of_activity = '','(empty)',field_of_activity) as field_of_activity,
        if(target_choice = '','(empty)',target_choice) as target_choice,
        if(level_history = '','(empty)',level_history) as level_history,
        if(level_difficulties = '','(empty)',level_difficulties) as level_difficulties,
        if(lessons_count = '','(empty)',lessons_count) as lessons_count,
        if(lesson_time = '','(empty)',lesson_time) as lesson_time,
        if(lesson_days =
'','(empty)',if(right(lesson_days,1)=';',substring_index(lesson_days,';',length(lesson_days)-
length(replace(lesson_days,';',''))),lesson_days)) as lesson_days,
        if(teacher_lang_type = '','(empty)',teacher_lang_type) as teacher_lang_type,
        if(teacher_sex = '','(empty)',teacher_sex) as teacher_sex,
        if(teacher_age = '','(empty)',teacher_age) as teacher_age,
        if(teacher_char_activity = '','(empty)',teacher_char_activity) as teacher_char_activity,
        if(teacher_char_loyality = '','(empty)',teacher_char_loyality) as
teacher_char_loyality,
        if(teacher_char_humor = '','(empty)',teacher_char_humor) as teacher_char_humor,
        if(goals = '','(empty)',goals) as goals
    FROM
      (SELECT
        ap.user_id,
        (SELECT
            COUNT(DISTINCT student_user_id)
        FROM
            individual_lesson_history_classes
        WHERE
            student_user_id = ap.user_id) AS stud,
            (SELECT
            min(dt_meeting)
        FROM
            individual_lesson_history_classes
        WHERE
            student_user_id = ap.user_id) AS first_lesson,
            (SELECT
```

```
                min(dt_created)
            FROM
                payment_transaction
            WHERE
                user_id = ap.user_id
                and form_study='individual' and type_transaction in
('payment','payment_manually','activate_individual_certificate')) AS first_payment,
            u.dt_created AS dt_registration,
            ap.dt_created AS dt_request,
            (select max(dt_start) from ebook_lessons_interview where student_id=ap.user_id
and dt_start>=ap.dt_created and dt_start<@dt_now) dt_max_intro,
            IF(u.dt_logined_ed_words <= ap.dt_created, 1, 0) edwords_user_before_appl,
            ap.gmt,
            CASE
            WHEN from_page = '' AND connection_type = '' THEN '(not_set)'
            WHEN from_page = '' AND connection_type <> '' THEN connection_type
            WHEN from_page REGEXP '/l/' THEN CONCAT('/l/',
SUBSTRING_INDEX(SUBSTRING_INDEX(SUBSTRING_INDEX(from_page, '?', 1),
'm.com/l/', - 1), '/', 1))
            ELSE
SUBSTRING_INDEX(SUBSTRING_INDEX(SUBSTRING_INDEX(from_page, '?', 1), 'm.com/',
- 1), '/', 1)
            END AS from_page_pretty,
            IF(from_page REGEXP '/#', CONCAT('#',
SUBSTRING_INDEX(SUBSTRING_INDEX(from_page, '?', 1), '#', - 1)), '(not_set)')
from_page_hash,

            ap.segment,
            u.country,
            IFNULL(IF(u.gender = '', IFNULL(apr.gender, '(not_set)'), u.gender), '(not_set)') AS
gender,
            IF(u.birthday = '', IFNULL(apr.birthday, '(not_set)'), u.birthday) AS birthday,
            if(from_page
regexp'utm_source=',SUBSTRING_INDEX(SUBSTRING_INDEX(from_page, 'utm_source=', -
1),'&',1),'(not_set)') as Source_app,
```

```
        if(from_page
regexp'utm_medium=',SUBSTRING_INDEX(SUBSTRING_INDEX(from_page, 'utm_medium=', -
1),'&',1),'(not_set)') as Medium_app,
        SUBSTRING_INDEX(IF(u.source_medium IS NULL, '(not_set) / (not_set)',
u.source_medium), ' / ', 1) AS Source,
        SUBSTRING_INDEX(IF(u.source_medium IS NULL, '(not_set) / (not_set)',
u.source_medium), ' / ', - 1) AS Medium,
        IFNULL(IF(u.english_level_id = '', IFNULL(apr.english_level_id, '(not_set)'),
u.english_level_id), '(not_set)') AS english_level_id,
        IFNULL(apr.field_of_activity, '(not_set)') as field_of_activity,
        IFNULL(apr.target_choice, '(not_set)') as target_choice,
        IFNULL(apr.level_history, '(not_set)') as level_history,
        IFNULL(apr.level_difficulties, '(not_set)') as level_difficulties,
        IFNULL(apr.lessons_count, '(not_set)') as lessons_count,
        IFNULL(apr.lesson_time, '(not_set)') as lesson_time,

        IF(apr.lesson_days is null, '(not_set)',concat(if(lesson_days regexp 'ПН',
'ПН;',''),if(lesson_days regexp 'ВТ', 'ВТ;',''),if(lesson_days regexp 'СР', 'СР;',''),if(lesson_days
regexp 'ЧТ', 'ЧТ;',''),if(lesson_days regexp 'ПТ', 'ПТ;',''),if(lesson_days regexp 'СБ',
'СБ;',''),if(lesson_days regexp 'ВС', 'ВС',''))) as lesson_days,

        IFNULL(apr.teacher_lang_type, '(not_set)') as teacher_lang_type,
        IFNULL(apr.teacher_sex, '(not_set)') as teacher_sex,
        IFNULL(apr.teacher_age, '(not_set)') as teacher_age,
        IFNULL(apr.teacher_char_activity, '(not_set)') as teacher_char_activity,
        IFNULL(apr.teacher_char_loyality, '(not_set)') as teacher_char_loyality,
        IFNULL(apr.teacher_char_humor, '(not_set)') as teacher_char_humor,
        IFNULL(apr.goals, '(not_set)') as goals
    FROM
        applications ap
    LEFT JOIN users u ON ap.user_id = u.user_id
    LEFT JOIN application_profile apr ON ap.user_id = apr.user_id
    WHERE
        ap.dt_created >= '2019-01-01'
        AND ap.dt_created < '2020-04-01'
```

```
        AND ap.application_type NOT IN ('group' , 'multilang', 'dashboard_individual', 'hr',
'corporate', 'corporate_online_courses', 'corp')
        AND ap.first_name NOT REGEXP 'test'
        AND ap.first_name NOT REGEXP 'Test'
        AND ap.first_name NOT REGEXP 'TEST'
        AND ap.first_name NOT REGEXP 'тест'
        AND ap.first_name NOT REGEXP 'Тест'
        AND ap.first_name NOT REGEXP 'ТЕСТ'
        AND ap.email NOT REGEXP 'test'
        AND ap.email NOT REGEXP 'Test'
        AND ap.email NOT REGEXP 'TEST'
        AND ap.email NOT REGEXP 'qa.eng'
        AND ap.email NOT REGEXP 'englishdom'
        AND ap.segment NOT LIKE 'b2b'
        AND ap.from_page NOT REGEXP '/corp/'
        AND ap.from_page NOT REGEXP '/corporate/'
        AND ap.crm_id IS NOT NULL
  ORDER BY ap.dt_created ASC) v
```

Для тренувальной бази змінюємо дату подання заявки **ap.dt_created** на потрібний місяць.

Interface.py

```
#import libraries



from pandas import read_csv, DataFrame, get_dummies, concat
import numpy as np
import matplotlib.pyplot as plt
import PySimpleGUI as sg
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.metrics import roc_curve
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC, SVC, SVR
```

```
from sklearn.metrics import accuracy_score,f1_score,classification_report,roc_auc_score
layout = [
   [sg.Text('Test'), sg.InputText(), sg.FileBrowse(),


    ],
   [sg.Text('Train'), sg.InputText(), sg.FileBrowse(),


    ],
     [sg.Text('Fill C1'), sg.InputText(),


    ],
     [sg.Text('Fill C2'), sg.InputText(),


    ],
     [sg.Text('Fill max_iter'), sg.InputText(),


    ],
   [sg.Submit(), sg.Cancel()]
]
window = sg.Window('Scoring', layout)
while True:                         # The Event Loop
   event, values = window.read()
   # print(event, values) #debug
   if event in (None, 'Exit', 'Cancel'):
      break



#чтение датасетов
train_df=read_csv('train_2019-11-01 - 2021-05-01.csv',index_col=None) #тренировочный 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) #тренировочный датасет
test_df=read_csv('test_2020-11-01 - 2020-12-01.csv',index_col=None) #тестовый датасет
```

```
#test_df=read_csv('test_2020-11-01 - 2020-12-01.csv',index_col=None) #тестовый датасет
result = DataFrame(test_df.user_id) #для записи результата в файл
print('train shape {0}, test shape {1}]'. format(train_df.shape, test_df.shape)) #нужно для дальнейшей контрольной точки
#убираем из датасетат ненужные данные (столбцы)
#если есть clientid - убрать!!!
trainy = train_df['is_student'] #выбираем бинарное поле, которое является нашим ключевым признаком (в данном случае stud=1 означает что данная заявка сконвертилась в запуск)
testy = test_df['is_student'] #выбираем бинарное поле, которое является нашим ключевым признаком (в данном случае stud=1 означает что данная заявка сконвертилась в запуск)
train_df = train_df.drop(['user_id','is_student','date_appl','month', 'source_reg',
  'medium_reg', 'campaign_reg'],axis=1)
test_df = test_df.drop(['user_id','is_student','date_appl','month', 'source_reg',
  'medium_reg', 'campaign_reg'],axis=1)
#следующие строки для корректной кодировки качественных полей (треин+тест)
mergedata = train_df.append(test_df) #соединяем тренировочный и тестовый датасеты
testcount = len(test_df)
count = len(mergedata)-testcount
#выбираем и кодируем качественные поля
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','is_intro_set',
  'is_intro_complete'])
#mergedata = mergedata.drop(['hour_req','dayofweek_req','month_req','gmt','from_page_pretty','country','Source','Medium','from_page_hash'],axis=1)
'''
#нормализируем датасет по столбцам типа 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)
'''
#разделяем закодированный датасет обратно на треин и тест датасеты
#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)) #используется в качестве точки контроля для проверки корректности предыдущих шагов
#приводим закодированные датасеты к стандартному виду
#trainy = train['is_student'] #выбираем бинарное поле, которое является нашим ключевым признаком (в данном случае stud=1 означает что данная заявка сконвертилась в запуск)
#trainx = train.drop(['is_student'], axis=1) #дропаем из датасета ключевое поле из предыдущей строки
#testy = test['is_student'] #выбираем бинарное поле, которое является нашим ключевым признаком (в данном случае stud=1 означает что данная заявка сконвертилась в запуск)
#testx = test.drop(['is_student'], axis=1) #дропаем из датасета ключевое поле из предыдущей строки
#разделяем тренировочный датасет рандомно на два датасета (в соотношении test_size) для тестирования алгоритмов и оценки их точности (и других параметров)
## 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=0.01, 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:.4, 1:.6}, 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) #обучаем модель
pred=model_log.predict(X_test) #делаем предсказание на тестовом датасете с параметрами, которые получили в результате обучения модели
#print( scaled:  results (pred, real): ,list(zip(pred,y_test_part))) - если хотим вывести на экран реальное значение искомого ключевого признака и его предсказанное значение
#вывод на экран метрик: точность, ф-мера, precision, recall
print('scaled: accuracy = {}, f1-score= {}'. format( accuracy_score(y_test,pred), f1_score(y_test,pred, average='macro')))
print('scaled: f1-score_weighted = {}'. format(f1_score(y_test,pred, average='weighted')))
report = classification_report(y_test, pred, target_names=['Non-churned', 'Churned'])
print(report)

y_pred_rf = model_log.predict_proba(X_test)[:, 1]

fpr_log, tpr_log, d = roc_curve(y_test, y_pred_rf)

print(roc_auc_score(y_test, y_pred_rf))
plt.figure(1)

plt.plot([0, 1], [0, 1], 'k--')

plt.plot(fpr_log, tpr_log, label='LOG')

plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.figure(2)
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.plot(d, tpr_log, label='Se (TPR)')
plt.plot(d, 1-fpr_log, label='Sp (1-FPR)')
#plt.legend(loc='Se&Sp')
```

```
plt.show()
pred=model_log.predict(testx) #делаем предсказание на тестовом датасете с параметрами, которые получили в результате обучения модели
#print( scaled:  results (pred, real): ,list(zip(pred,y_test_part))) - если хотим вывести на экран реальное значение искомого ключевого признака и его предсказанное значение
#вывод на экран метрик: точность, ф-мера, 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)
#выводим вероятность
pred_probability = model_log.predict_proba(testx)
#записываем в файл рассчитанные данные: предсказанные данные, вероятности и скоринг
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 считается как умножение вероятности положительного исхода на коэффициент
result.to_csv('test_result.csv', index=False)
#определение коэффициентов
#coefs = np.abs(model_log.coef_[0])
inter=model_log.intercept_
print(inter)
coefs = model_log.coef_[0]
indices = np.argsort(coefs)[::-1]
'''
plt.figure()
plt.title(Feature importances (Logistic Regression))
plt.bar(range(15), coefs[indices[:15]],
    color=b, align=center)
numerical_columns = trainx.columns
plt.xticks(range(15), trainx.columns[indices[:15]], rotation=45, ha='right')
```

```
    plt.subplots_adjust(bottom=0.3)
    #plt.tight_layout()
    #plt.show()
    #rev0=get_dummies(mergedata).idxmax(1)
    '''
    res_importances = concat([DataFrame(trainx.columns[indices[:]]),
DataFrame(coefs[indices[:]])], axis=1)
    res_importances.columns=['parameter','importance_value']
    res_importances.to_csv('importances_range.csv', index=False)
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

Перелік файлів на диску

Перелік документів на магнітному носії

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота Немченко.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота Неиченко.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Немченко.rar	Архів. Містить коди програми і скомпільовану програму
Презентація	
Немченко.ppt	Презентація кваліфікаційної роботи

ВІДГУК

на дипломний проект бакалаврана тему:
"Розробка додатку для аналітики користувацьких даних та побудови скорингових моделей у СУБД"
студента групи 122-17-3 Немченка Максима Ігоровича

Розроблене в кваліфікаційній роботі програмне забезпечення призначене для створення інформаційної системи, яке спроможно виконувати аналіз та побудову скорингових моделей великих об’ємів даних.

Актуальність розробленого програмного продукту полягає в створенні такого додатку, що дозволяє оптимізувати роботу менеджерів call-центру.

Переваго запропонованої програми в тому, що вона дозволяє зрозуміти користувачу як проаналізувати користувацькі дані.

Для вирішення поставлених задач при розробці додатку були здійснені наступні дії:

1. Розробка логічної моделі програми.
2. Створення бази даних.
3. Проектування та розробка програмного забезпечення.
4. Розробка простого й зрозумілого інтерфейсу програми.

Додаток розроблений на мові Python у середовищі Geany, а для створення бази даних була використана мова SQL у середовищі MySQL Workbench.

Практична значимість створення данного програмного продукту полягає в можливості змусити користувача проаналізувати дані клієнтів та виявити цільову аудиторію онлайн-школи.

Працездатність представленої програми підтверджена налагоджувальними випробуваннями та тестуванням програми.

У економічному розділі визначено трудомісткість розробленого додатку, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Тема кваліфікаційної роботи безпосередньо пов’язана з об’єктом діяльності бакалавра за напрямом підготовки 122 Комп’ютерні науки.

Оформлення пояснювальної записки до кваліфікаційної роботи викано відповідно до стандартів на програмну документацію.

В кваліфікаційній роботі були виявлені недоліки за оформлення, але вона виконана самостійно та заслуговує оцінки 95 балів «відмінно», а студенту Немченко Максиму Ігоровичу присвоєння йому кваліфікації бакалавра за спеціальністю «фахівець з розробки та тестування програмного забезпечення».

Керівник кваліфікаційної роботи
ас. каф. ПЗКС, к.т.н. **Приходченко С.Д.**

РЕЦЕНЗІЯ
на дипломний проект бакалавра
на тему:
" Розробка додатку для аналітики користувацьких даних та побудови скорингових моделей у СУБД"
студента групи 122-17-3 Немченка Максима Ігоровича

Кваліфікаційна роботана тему «Розробка додатку для аналітики користувацьких даних та побудови скорингових моделей у СУБД» виконаний в повному обсязі, відповідно до технічного завдання.

Мета кваліфікаційної роботи: розробка додатку для аналітики даних з користувацькими налаштуваннями вхідних даних.

У пояснювальній записці розглянуто необхідність створення і сфера застосування розробленої, виконано постановку завдання, опис вхідних і вихідних даних, розроблено логічну структуру програми, розроблено інформаційне забезпечення системи, наведені загальні відомості про додаток та його функціональне призначення, зазначені використовувані технічні засоби, визначені джерела, використані при розробці.

Для реалізації інформаційної системи була використана мова програмування Python. Реляційна база даних була створена за допомогою мови SQL.

Вважаю завдання і зміст дипломного проекту відповідним для перевірки ступеня підготовленості Немченко М.І. за напрямом 122 «Комп'ютерні науки».

Список літератури, наведений в роботі, налічує більше 20 джерел, що свідчить про вміння автора працювати з літературою та іншими джерелами інформації. Якість оформлення дипломного проекту можна визнати задовільною, він супроводжується достатньою кількістю малюнків . В роботі присутні заключні висновки. Працездатність цієї інформаційної системи підтверджується експлуатаційними випробуваннями.

Рівень теоретичної та практичної підготовки автора, логіка і стиль викладу матеріалу в цілому відповідає кваліфікаційним вимогам.

Дипломний проект виконаний самостійно та заслуговує оцінки «відмінно», а студент Немченко Максим Ігорович присвоєння йому кваліфікації бакалавра за спеціальністю «фахівець з розробки та тестування програмного забезпечення».

Рецензент кваліфікаційної роботи
к.т.н., доцент каф. БІТ **Герасіна О.В.**