МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”  
КАФЕДРА ПРИКЛАДНОЇ МАТЕМАТИКИ

КУРСОВА РОБОТА  
з навчальної дисципліни “Бази даних та інформаційні системи-2”  
на тему  
Система кредитного скорингу (оцінка кредитоспроможності позичальників при видачі кредиту)

Виконали: Керівник:

студенти групи КМ-71  *Шияк Б.А.*

Лисий Павло

Агафонов Дмитро

Ліхачов Артемій

Нікітіна Марина

КИЇВ – 2020

ЗАВДАННЯ НА ВИКОНАННЯ КУРСОВОЇ РОБОТИ

Представлення сервісу з власною скоринговую моделлю для автоматизації процесу видання кредитів. Для цього необхідно виконати предпроектне дослідження (проаналізувати бізнес-процеси, бізнес-правила та елементарні події, бізнес-драйвери, objectives), визначити сутності, що містимуться у базі даних, дані з яких можно бути зробити виносновок о кредитоспроможності, обрати потрібну модель машинного навчання. Більш детально ознайомитись з завданням на проект можна в документі про «Технічне завдання».

РЕФЕРАТ

В ході виконання курсового проекту було створено онлайн-додаток на базі веб-фреймворку Flask. Для виконання була обрана база даних - PostgreSQL. Взаємодія з базою даних була реалізована за допомогою Python та SQLAlchemy.

Веб-сторінки створено за допомогою HTML, з використанням фреймворку Bootstrap.

Створено два віддалених сервери для бази даних та для розгортання виконаного завдання знаходиться на Heroku.

В сервісі використовується модель машинного навчання – модель логістичної регресії.

ЗМІСТ

[ВСТУП](#_heading=h.gjdgxs) 5

[1. АНАЛІЗ ПІДПРИЄМСТВА АВТОМАТИЗАЦІЇ](#_heading=h.gjdgxs) 6

[2. ПОСТАНОВКА ЗАДАЧІ](#_heading=h.gjdgxs) 9

[2.1 Класи даних](#_heading=h.gjdgxs) 9

[2.2 Бізнес правила](#_heading=h.gjdgxs) 13

[2.3 Математичні моделі](#_heading=h.gjdgxs) 14

[3. МОДЕЛЮВАННЯ БІЗНЕС-ПРОЦЕСІВ](#_heading=h.gjdgxs) 19

[4. ТЕСТ КЕЙСИ](#_heading=h.gjdgxs) 22

[5. ВІДПОВІДНІСТЬ ПОСТАВЛЕНИМ ВИМОГАМ](#_heading=h.gjdgxs) 27

[5.1 Відповідність стандарту 12 Factor Apps](#_heading=h.gjdgxs) 27

[5.2 Функціональні і нефункціональні вимоги](#_heading=h.gjdgxs) 28

[5.3 Ролі в системі](#_heading=h.gjdgxs) 29

[5.4 Валідація даних](#_heading=h.gjdgxs) 29

[5.5 REST API](#_heading=h.gjdgxs) 29

[6. РОЗГОРТАННЯ СИСТЕМИ](#_heading=h.gjdgxs) 30

[7. ВІДПОВІДАЛЬНІСТЬ ЧЛЕНІВ КОМАНДИ](#_heading=h.gjdgxs) 31

[8. ВИСНОВКИ](#_heading=h.gjdgxs) 32

[9. СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ](#_heading=h.gjdgxs) 33

ВСТУП

Система кредитного скорингу (оцінка кредитоспроможності позичальників при видачі кредиту) -  курсового проекту.

Business objectives:

* 1 Забезпечити точність сервісу на 80%.
* 2 Час прийняття рішення сервісом обмежити до 2 хв.

Business goals:

-   Представлення сервісу з власною скоринговую моделлю ("унікальна функціональність") для автоматизації процесу видання кредитів.

1. Діаграми

Діаграма компонентів наведена на рисунку 1.1:

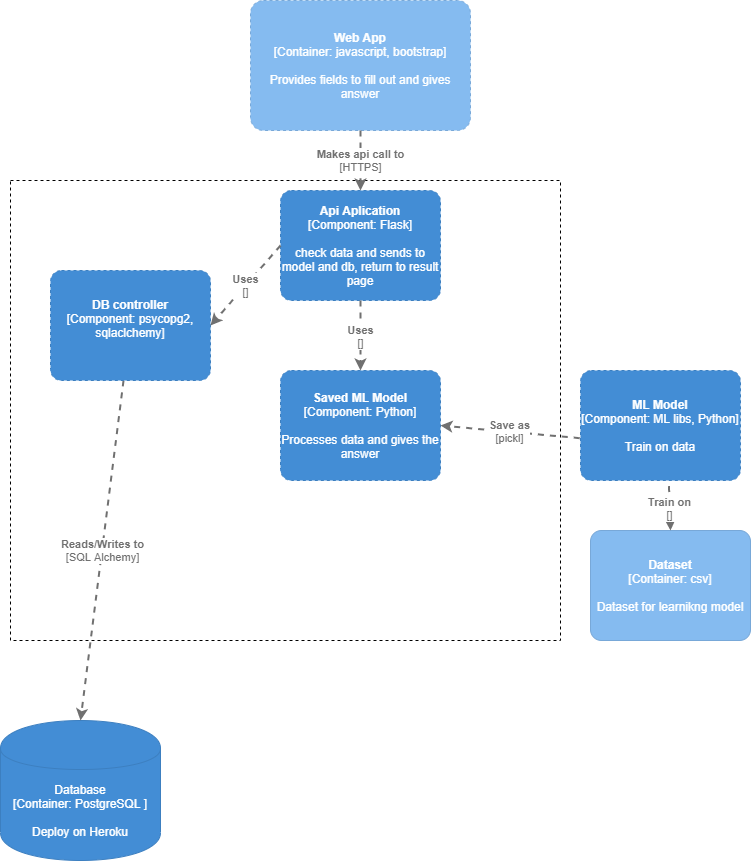


Рисунок 1.1 - Діаграма компонентів

Діаграма розгортання на рисунку 1.2:

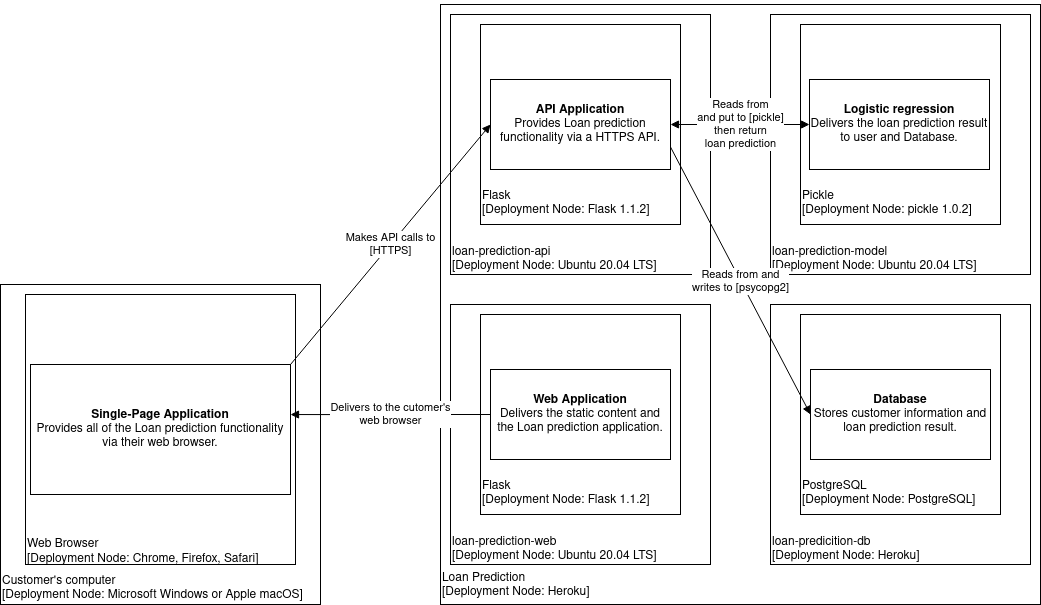


Рисунок 1.2 - Діаграма розгортання

ERD-діаграма наведена на рисунку 1.3:

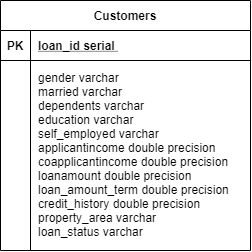


Рисунок 1.3 - ERD-діаграма

2. ПОСТАНОВКА ЗАДАЧІ

Представлення сервісу з власною скоринговую моделлю для автоматизації процесу видання кредитів.

2.1 Класи даних

Таблиця 2.1 - Клас даних « Customers »

|  |  |  |
| --- | --- | --- |
| **Сутність** | Customers | |
| **Опис сутності** | Зберігає дані та відповідь моделі | |
| **Атрибути** | **Опис атрибуту** | **Пов’язана сутність з атрибутом** |
| Loan\_id | Unique Loan ID |  |
| **Gender** | Male/ Female |  |
| **Married** | Applicant married (Y/N) |  |
| **Dependents** | Number of dependents |  |
| **Education** | Applicant Education (Graduate/ Under Graduate) |  |
| **Self\_Employed** | Self employed (Y/N) |  |
| **ApplicantIncome** | Applicant income |  |
| **CoapplicantIncome** | Coapplicant income |  |
| **LoanAmount** | Loan amount in thousands of dollars |  |
| **Loan\_Amount\_Term** | Term of loan in months |  |
| **Credit\_History** | credit history meets guidelines yes or no |  |
| **Property\_Area** | Urban/ Semi Urban/ Rural |  |
| **Loan\_Status** | Loan approved (Y/N) this is the target variable |  |

2.3 Математичні моделі

Логіт регресійна модель призначена для вирішення завдань передбачення значень неперервної змінної змінної, за умови, що ця залежна змінна може приймати значення на інтервалі від 0 до 1.

Враховуючи таку специфіку, її часто використовувати для передбачення ймовірності настання деякої події в залежності від значень від деякого числа предикторов.

Можна використовувати логіт регресію і для вирішення завдань з бінарним відгуком. Такі завдання з'являються, коли залежна змінна може приймати тільки два значення.

Математична основа логістичної регресії

Таким чином, як вже було сказано, в залежності від прогнозованої моделі залежною змінною бути залежною, чи більше (або більше) 1, незалежною від незалежних змінних; тому ця модель часто використовується для аналізу бінарних залежних чисел або чисел відгуку.

При цьому використовується рівняння регреcсіі (термін логит був використаний Berkson, 1944):

y = exp (b0 + b1 \* x1 + ... + bn \* xn) / [1 + exp (b0 + b1 \* x1 + ... + bn \* xn)]

Легко побачити, що незалежно від регресійних коефіцієнтів чи величин х, передбачені значення (у) в цій моделі завжди будуть лежати в діапазоні від 0 до 1.

Термін логит походить від того, що цю модель легко линеаризовать за допомогою логит перетворення. Припустимо, що бінарна залежна змінна y є безперервною ймовірністю p, що лежить в діапазоні від 0 до 1. Тоді можна перетворити цю ймовірність наступним чином:

p '= loge {p / (1-p)}

Це перетворення називається логіт або логістичним перетворенням.

Зауважимо, що p 'теоретично може приймати будь-які значення від мінус до плюс нескінченності. Логіт перетворення вирішує проблему 0/1 кордонів для вихідної логічної змінної (ймовірності), можна використовувати ці (логіт перетворені) значення в звичайному лінійному рівнянні регресії.

Фактично, при проведенні логит регресійних рівнянь, наведених вище, ми отримаємо стандартну лінійну модель множинної регресії:

p '= b0 + b1 \* x1 + b2 \* x2 + ... + bn \* xn

Подібне рівняння нам вже знайоме. Вирішивши його, ми отримаємо значення регресійних коефіцієнтів, за яким можна відновити ймовірність р.

Особливості логістичної регресії

Однак, застосування логістичного перетворення до рівняння логіт регресії породжує проблеми.

При завданнях лінійної регресії оцінюється деяку гіперповерхность - пряму в разі простої регресії, площина - в двох незалежних умовах. При переході до рівняння логіт регресії підганяється поверхню вже не матиме такої простої вид. Також, нас не врятує вже і нормальність помилок.

Все це унеможливлює використання методів оцінювання, які застосовуються для лінійних задач.

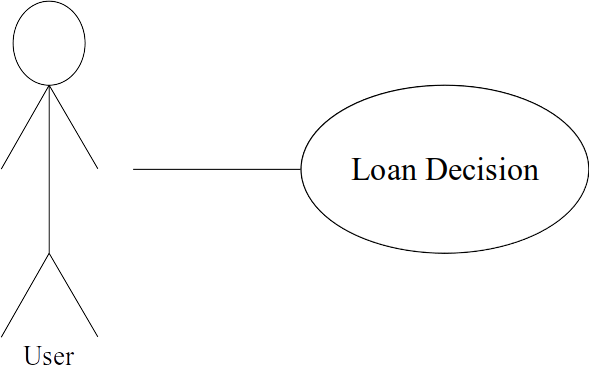
Наприклад, в разі однієї простої регресії, який можна застосовувати метод найменших квадратів. У разі простою логіт регресії такий метод вже непридатний. Непридатними є і методи для вирішення задач з великим числом предикторів.

Тому для вирішення завдань логіт регресії доступне лише обмеження правдоподібності. Якщо коротко, процес оцінки регресійних коефіцієнтів зводиться до максимізації ймовірності відображаються значень вибірки (при заданих спостережуваних значеннях). Це призводить до часто невисокого відсотку бухгалтерської класифікації.

3. МОДЕЛЮВАННЯ БІЗНЕС-ПРОЦЕСІВ

1. **Use case:** Get Loan Decision

**Diagram:**



**Brief Description:**

User fills up the form to get his loan decision and explanation.

**Initial Step-By-Step Description:**

Before this use case can be initiated, the User has already accessed the main page of the Service.

Main Flow

1. The User selects Loan Decision. **[A1: The User wants to get Loan Decision]**

2. The system returns blank form

3. The User fills in the information and submits the form.

4. The system verifies the information and presents a loan decision and explanation.

Action was succeeded

5. The system stays on the main Service page.

Alternatives Flows

**A1: The User wants to get Loan Decision**

1. The system presents a blank form.

Filling in the information

1. The User fills in the information and submits the form. [**R1: Required information wasn`t filled in**]
2. The system presents a loan decision and it’s explanation.

The use case continues at Action was succeeded in the MF.

Recovery Flows

**R1: Required information wasn`t filled in**

1. The system lights up required fields that are still blank and ask the User to fill them with information.

The use case continues at Filling in the information in the A1.

4. ТЕСТ КЕЙСИ

Таблиця 4.1 – Отримання позитивної відповіді

|  |  |
| --- | --- |
| **номер** | 1 |
| **заголовок** | Отримання позитивної відповіді |
|  |  |
|  |  |
| **крок** | **очікуваний** **результат** |
| **Gender** | Male |
| **Married** | Yes |
| **Dependents** | 1 |
| **Education** | Graduate |
| **Self\_Employed** | Yes |
| **ApplicantIncome** | 123.0 |
| **CoapplicantIncome** | 123.0 |
| **LoanAmount** | 1111.0 |
| **Loan\_Amount\_Term** | 12.0 |
| **Credit\_History** | 1.0 |
| **Property\_Area** | Urban |
| **Loan\_Status** | Yes – кредитоспроможність підтверджена |

5. ВІДПОВІДНІСТЬ ПОСТАВЛЕНИМ ВИМОГАМ

Під час виконання важливо було досягти поставлених вимог до роботи, а саме:

1. відповідність стандарту 12 factor apps(дванадцяти факторів);
2. досягти поставлених функціональних і нефункціональних вимог;
3. реалізувати в системі паралельну роботу декількох користувачів;
4. реалізувати валідацію даних на стороні клієнта, сервера та бази даних;
5. притримуватись концепції REST API;
6. написати документацію та створити презентацію.

5.1 Відповідність стандарту 12 Factor Apps

1. Під час розробки активно використовувалась система контролю версій git.
2. При розробці для ізоляції залежностей було використане віртуальне середовище(virtualenv), всі залежності вказані в файлі requirements.txt за допомогою пакетного менеджера pip (команда: pip freeze requirements.txt).
3. Параметри конфігурацій були збережені в окремих файлах, підключення до бд: credentials.py.
4. Код нашого додатку не робить розділень на локальні та сторонні сервіси, замінивши параметри підключення в credentials.py, або в config.cfg система повинна працювати коректно.
5. В нашій системі всі етапи розділені, неможливо зробити зміни в коді, якщо система знаходиться на етапі виконання, це забезпечується комбінацією git + heroku, для того щоб перемістити код в етап збірки потрібно зробити git commit(кожен commit має унікальний ідентифікатор) після чого git push heroku master. Після виконання останньої команди код перетворюється в робочий пакет і з’являється новий реліз.
6. Наша система не зберігає внутрішнього стану, всі дані зберігаються за допомогою бази даних Postgres, ця стороння служба розміщена на Heroku
7. В проекті прив’язка до портів реалізована за допомогою оголошення залежності для додавання бібліотеки веб сервера Flask.(оскільки додаток по факту один, то і прив’язка для того щоб виконувати всі HTTP запити проходить через порт 5000 за замовченням, хост [0.0.0.0](http://0.0.0.0/)).
8. (Не реалізовано) Можливо досягнути, якщо вкладатися в залізо аби забезпечити хорошу паралельну роботу, або покращений план на хостингу.
9. Система може бути запущена та зупинена в будь-який момент часу, це реалізується за допомогою використання хостингу heroku.
10. Розгортання проекту за 5 хвилин завдяки тому що розробники й займаюсть деплоєм. (різниця між середовищем розробки та середовищем виконання є мінімальною)
11. Система розміщена на heroku - постійний доступ до журналу з можливістю його збереження.
12. Система розміщена на heroku - можливість в терміналі середовища розробки можна проводити адміністрування(виконувати команди запуску/зупинки додатку, проводити міграцію бази даних)

5.2 Функціональні і нефункціональні вимоги

При розробці додатку були враховані функціональні вимоги(окрім реалізації двох чи більше ролей).

|  |  |
| --- | --- |
| **Код вимоги** | **Опис вимоги** |
| FR1 | 1. Користувач має змогу внести свої дані та отримати результат про кредитоспроможність. |
| FR2 | 2. Користувач повинен отримати документацію при натисканні на кнопку "Документація". |
| FR3 | 3. Користувач повинен отримувати інтерпретацію результату (графіки/ аналітику/ і тд) при натисканні на кнопку "Пояснення результату". |
| FR4 | 4. Забезпечити дві ролі - Користувач (Customer) та Адміністратор (Admin) з різними правами доступу. |
| FR5 | 5. Адміністратор повинен мати можливість переглядати базу користувачів та бачити аналітику щодо користування сервісом. |
| FR6 | 6. Користувач повинен ввести наступні поля щоб мати можливість дізнатись кредитоспроможність: |
|  | 1. “Gender”:  Male/ Female |
|  | 2. “Married”:  Applicant married (Y/N) |
|  | 3. “Dependents”:  Number of dependents |
|  | 4. “Education”:  Applicant Education (Graduate/ Under Graduate) |
|  | 5. “Self\_Employed”:  Self employed (Y/N) |
|  | 6. “ApplicantIncome”:  Applicant income |
|  | 7. “CoapplicantIncome”:  Coapplicant income |
|  | 8. “LoanAmount”:  Loan amount in thousands |
|  | 9. “Loan\_Amount\_Term”:  Term of loan in months |
|  | 10. “Credit\_History”:  credit history meets guidelines |
|  | 11. “Property\_Area”:  Urban/ Semi Urban/ Rural |
|  | 12. “Loan\_Status”:  Loan approved (Y/N) |
| FR7 | Реалізовано ml model |
|  |  |

Нефункціональні вимоги:

|  |  |
| --- | --- |
| **Код вимоги** | **Опис вимоги** |
| NFR1 | 1. Користувач при вводі інформації повинен отримати відповідь швидше за 5 секунд (Performance) |
| NFR2 | 2. Вся функціональна система повинна мати документів та презентацію (Documentation) |
| NFR3 | 3. Необхідно мати систему контроля версії (Modifiability) |
| NFR4 | 4. Модель повинна мати точність > 80% (accuracy / relevance) |
| NFR6 | 6. Реалізація в системі валідації даних на рівнях UI, backend, DB: (administrability) |
| NFR7 | Відповідність «12 factor apps» |
| NFR8 | 3 види view в документації |
| NFR9 | Паралельна робота користувачів |
| NFR10 | Реалізація в системі REST API |
| NFR11 | Робота  розміщена на Heroku чи будь-якому іншому hosting сервісу |
| NFR12 | Наданий доступ викладачу до репозиторію з роботою (Використання системи контроля версій) |

Точність та швидкість відповідають поставленим вимогам.

Документованість було досягнуто написанням цього звіту.

Робота розміщенна на Heroku, доступ до репозиторію наданий окремо викладачу.

Паралельна робота користувачів достягнена за допомогою архітектурних рішень та перевірена для 3 користувачів

Всі дані зберігаються в базі даних postgres.

Для досягнення 60% валідації даних на стороні клієнта було використано можливості HTML та мови Javascript, на стороні сервера валідація проводилась можливостями мови Python та фремворку Flask.

Досягнуто REST API level 1. Спілкування між сервером було організовано шляхом використання POST, GET запитів та JSON.

6. РОЗГОРТАННЯ СИСТЕМИ

**Необхідні інструменти та акаунти:** акаунти github та heroku, git-bash, PYCHARM, Python 3.8+. Деплой було виконано на операційній системі Ubuntu LTS 20.04 Focal та Windows 10 для субд, PgAdmin4.

**Кроки для розгортання**

1. Якщо при розробці не було створена віртуальне середовище, то створити його командою: python -m venv venv/

2. Обрати інтерпретатор venv в середовищі виконання, та встановити всі необхідні в проекті бібліотеки.

3. Встановити Gunicorn в консолі:

pip install gunicorn

3. Створити в папці проекту Procfile без формату та прописати там рядок(app.py - це ім’я файла який містить фласк частину, тому значенню app присвоюємо значення app):

web: gunicorn app:app

4. В консолі середовища виконання виконати комнаду:

pip freeze > requirements.txt

5. В консолі середовища виконати команди ініціалізації репозиторію, додавання всіх файлів папки до нього, та створення commit:

git init .

git add .

git commit -m "first commit"

6. Увійти в акаунт хероку з середовища розробки, команда в консолі:

heroku login -i

7. Створити проект на хероку командою:

heroku create

8. Виконати пуш на heroku master:

git push heroku master

9. Тепер проект опинився на heroku.

10 На іншому віддаленому сервері розгоратємо базу данних

10. Після встановлення у розпорядженні розробника опиниться можливість під’єднатися до віддаленої бази даних.

11. Під’єднатися до віддаленої бази даних використовуючи pgadmin.

12. Виконати в цій базі даних необхідні sql файли.

13. Повернутися в локальне середовище виконання, замінити всі credentials бд, на credentials бази даних на хероку(логін, пароль, хост, uri).

14. Виконати крок 5 і одразу після нього крок 8.

15. Перейти за посиланням проекту і пересвідчитись що все працює.

16. Проект розміщено.

7. ВІДПОВІДАЛЬНІСТЬ ЧЛЕНІВ КОМАНДИ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Nikitina | Likhachov | Lysyi | Agafonov |
| Preprocessing dataset & EDA | I | C | **RA** | С |
| Model development | I | C | **RA** | С |
| Bulding analytics | R | I | **RA** | C |
| Security through obscurity | R | **RA** | I | R |
| DB development | **RA** | CI | CI | CI |
| DB deploy&migration | **RA** | C | I | I |
| WEB development | CI | R | C | **RA** |
| Multithreading | CI | **RA** | CI | R |
| Integration with web | I | **RA** | C | C |
| Heroku deployment | R | **RA** | I | I |
| Testing | R | R | R | **RA** |
| Collecting requirements | **RA** | R | R | R |
| Processing Requirements | **RA** | R | R | R |

8. ВИСНОВКИ

В ході виконання курсової роботи була створена система для реалізації кредитоспроможності клієнта, Як результат виконання проекту досягнуто: інформаційна система, що дозволяє клієнтам отримати знання про їх кредитоспроможність. База даних, що взята за основу – PostgreSQL, а на Python – SQLAlchemy. Веб-застосунок збудований на базі фреймворків Flask та Bootstrap-4. Ресурс розгорнутий на хостингу Heroku. Математична модель працює на алгоритмі логістичної регресії. Точність моделі 86%. Розроблена система потребує допрацювання. Для того, щоб далі з нею могли працювати реальні банківські системи, потрібно додати шифрування даних, аби забезпечити безпеку даних користувачів. А для адміністрації можна розробити спеціальний аккаунт адміна, який матиме доступ до всіх корисутувачів. Також необхідно мати можливість підключати різні моделі/бд для різних банківських систем

9. СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. PostgreSQL 13.1 Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.postgresql.org/docs/13/index.html.
2. Flask Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://flask.palletsprojects.com/en/1.1.x/
3. SQLAlchemy Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.sqlalchemy.org/en/13/