МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ

Система підтримки прийняття рішень при кредитуванні

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма **«Аналітика даних»**

Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи АнД-41

_____Іваніна С. О._____
(прізвище та ініціали)

Керівник _____Самохвалов Ю. Я.____
(прізвище та ініціали)

___д. т. н., професор_____
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту рішенням кафедри *інтелектуальних технологій*
Протокол № 11 від 06.06.2022 р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ – 2022

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
інтелектуальних
технологій
Іларіонов О.Є.

_____
____

"___"_____2022 р.

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Іваніні Саві Олександровичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)

«Система підтримки прийняття рішень при кредитуванні» затверджена протоколом засідання кафедри від затверджена протоколом засідання кафедри від «23» грудня 2021 р. № 4

2. Термін здачі студентом закінченого проекту (роботи) 29 травня 2022 року

3. Вихідні дані до проекту (роботи)

Оцінка кредитоспроможності позичальника.

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

Аналіз предметної області, аналіз проектних рішень, розробка СППР при кредитуванні.

5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій)

Аналіз предметної області(2 слайди), постановка задачі(1 слайд), аналіз проектних рішень(3 слайди), розробка СППР при кредитуванні (6 слайдів), висновки по роботі(1 слайд).

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 15 лютого 2022 року

Керівник _____ / Самохвалов Ю. Я. /
(підпис) (ПІБ)

Завдання прийняв до виконання _____ / Іваніна С. О. /
(підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1	Опрацювання літератури	15.02.2022 – 01.03.2022	
2	Робота над розділом 1. Аналіз предметної області	01.03.2022 – 20.03.2022	
3	Робота над розділом 2. Аналіз проектних рішень	20.03.2022 – 11.04.2022	
4	Робота над розділом 3. Розробка СППР при кредитуванні	11.04.2022 – 15.05.2022	
5	Робота над оформленням пояснювальної записки	25.05.2022	
6	Робота над презентацією	01.06.2022	

Студент-дипломник _____ / Іваніна С. О. /
(підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи _____ / Самохвалов Ю. Я. /
(підпис) (ПІБ)

АНОТАЦІЯ

Іваніна Сава Олександрович виконав випускну кваліфікаційну роботу на тему «Система підтримки прийняття рішень при кредитуванні» за спеціальністю 122 – «Комп'ютерні науки».

У кваліфікаційній роботі було розглянуто основні існуючі методи оцінки кредитоспроможності позичальників та на основі методу скориногової оцінки було реалізовано систему підтримки прийняття рішень при кредитуванні, що визначає кредитоспроможність позичальника.

Ключові слова: Система підтримки прийняття рішень, кредитування, кредитоспроможність.

ANNOTATION

The degree project: « Lending decision support system» has completed by **Ivanina Sava** specialty 122 – «Computer Science».

The main existing methods of assessing the creditworthiness of borrowers were considered in this graduation thesis. Lending decision support system was implemented, which determines the creditworthiness of the borrower, based on the method of peer review.

Keywords: Decision support system, lending, creditworthiness

ЗМІСТ

ВСТУП

Останні роки в Україні проглядається збільшення банківського споживчого кредитування. Це свідчить про позитивні тенденції у цій сфері. У зв'язку з цим банківські установи України зіткнулись із проблемою неповернення населенням отриманих кредитів.

Доходи від кредитних операцій є основним джерелом прибутку банків та фінансових організацій. Відсутність ефективних методик оцінювання кредитоспроможності фізичних осіб і ризику банків в кінцевому результаті може привести банк до виникнення значних проблем, адже кожний недооцінений банківський ризик перетворюється у ризик структурний та завдає збитки банку в цілому. Але повністю уникнути ризиків у банківській діяльності неможливо, тому мета процесу управління ризиками полягає не в повному їх уникненні, а в обмеженні та мінімізації їх впливу.

Саме тому, застосування у банківських установах та фінансових орагнізаціях систем підтримки прийняття рішень при кредитуванні у світлі сучасних кризових явищ у фінансовій сфері є необхідним.

Таким чином, упровадження таких систем у практику українських банків необхідне, як для самих банків щодо впевненості в поверненні кредиту позичальником та відповідно зниженні кредитних ризиків банку, так і для позичальників, для яких система відчутно скоротить час на прийняття банком рішення на видачу кредиту. Також якби рівень неповернень скоротився, це б дозволило знизити і кредитні ставки.

Метою даної дипломної роботи є підвищення ефективності процесу оцінювання кредитоспроможності позичальників шляхом розробки системи підтримки прийняття рішень при кредитуванні.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність проблеми

Останні роки в Україні спостерігається ріст банківського споживчого кредитування. Сума чистих кредитів наданих фізичним особам з кінця 2017 року до 2022 зросла на 163%. Про це свідчать дані Національного банку України (рисунки 1.1, 1.2).

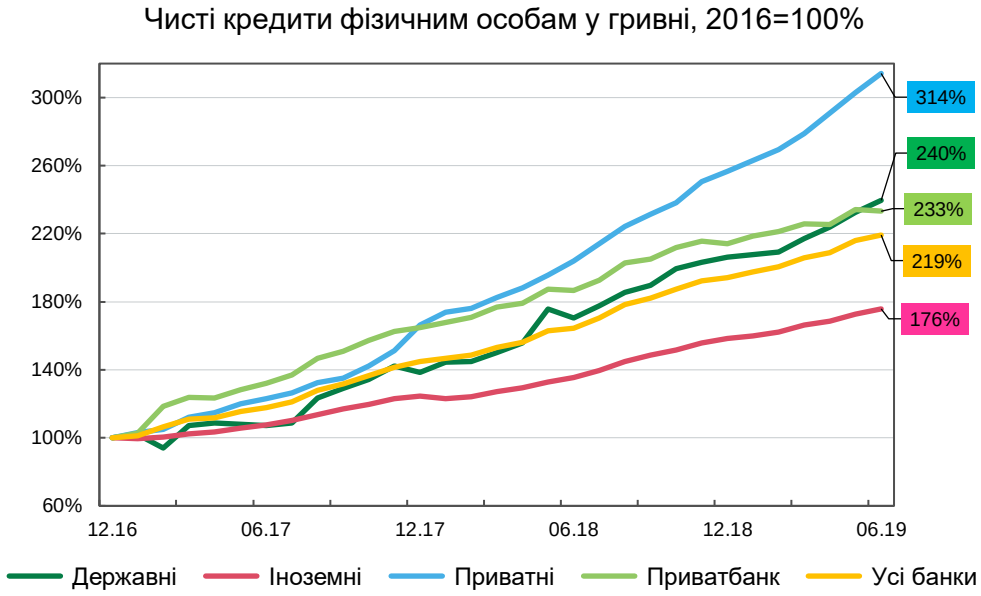

Рисунок 1.1 Чисті кредити фізични особам у гривні, 2017 – 2019 рр.

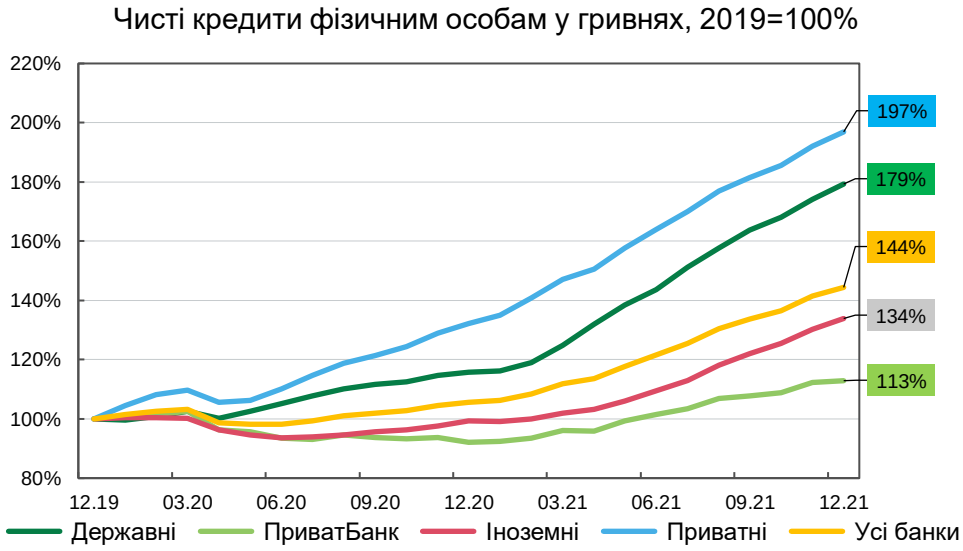

Рисунок 1.2 Чисті кредити фізични особам у гривні, 2020 – 2022 рр.

Згідно з даними, наданими Національним банком України, середня частка непрацюючих кредитів у портфелях банків, що працюють на території України, за останні 5 років складає 48.35% (рисунки 1.3, 1.4).

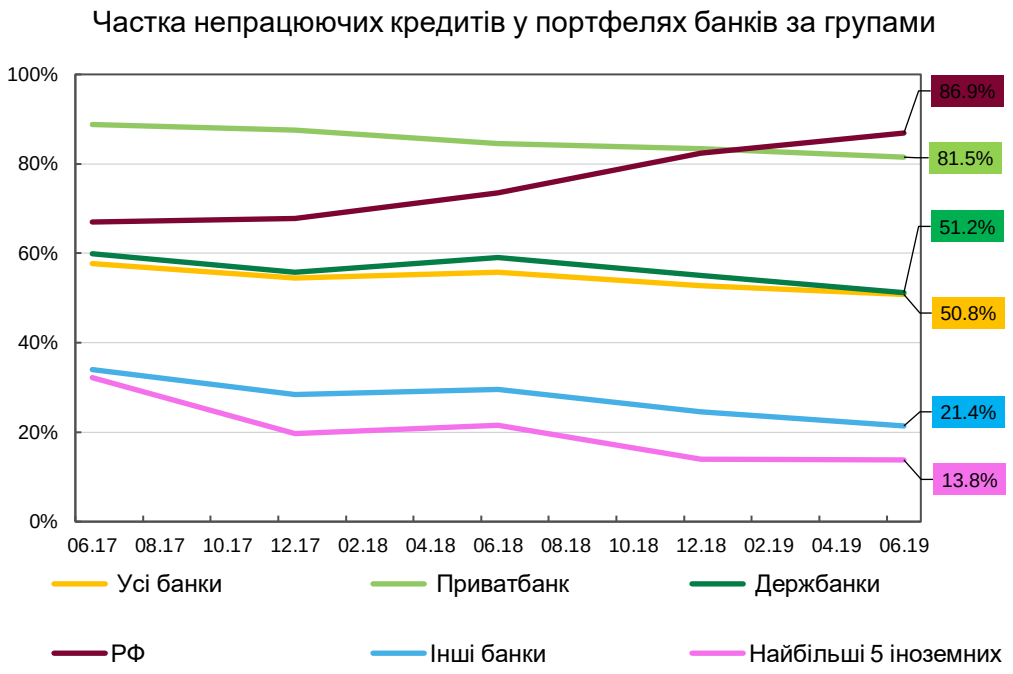

Рисунок 1.3 Частка непрацюючих кредитів, 2017 – 2019 рр.

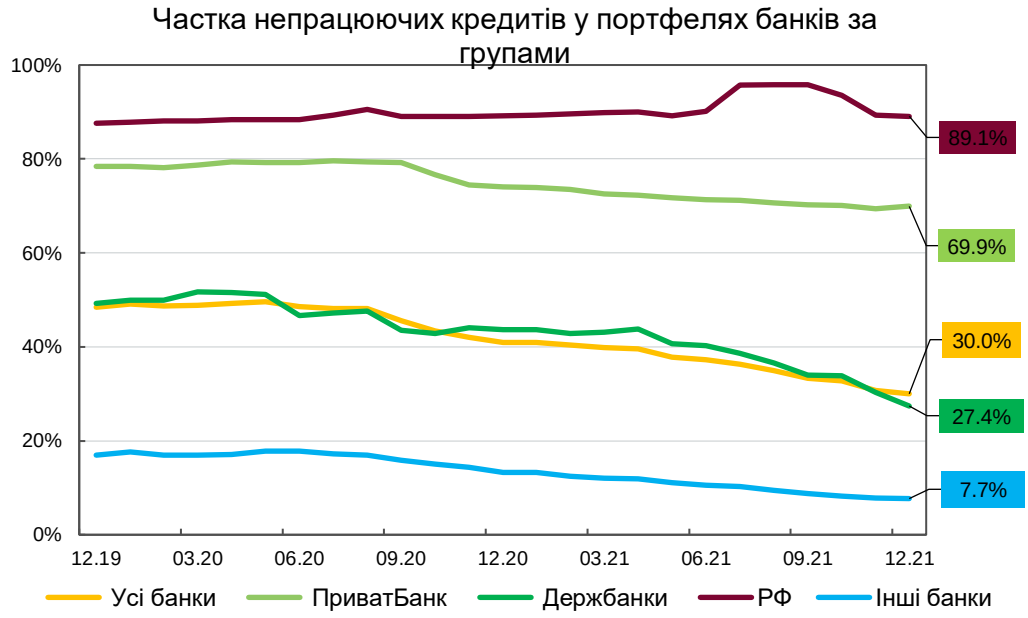

Рисунок 1.4 Частка непрацюючих кредитів, 2020 – 2021 рр.

Тобто майже за кожним другим кредитом, наданим позичальнику в Україні, не відбувається вчасної сплати платежу або погашення боргу.

Неповернені кредити знижують прибутковість банківської установи і обмежують її можливості видавати нові кредити. Вони також ризикують гальмувати довгострокове економічне зростання, що призведе до більшої невизначеності в банківській системі та, у свою чергу, до підвищення ризиків фінансової стабільності.

На даний момент системи підтримки прийняття рішень стали ключовим інструментом для банків та інших фінансових установ при оцінці кредитоспроможності позичальників. Завдяки таким системам банківські установи України зможуть досягти зниження ймовірності ризиків та підвищення ефективності та фінансової стабільності установи. Основна ідея систем підтримки прийняття рішень при кредитуванні полягає у класифікації потенційних позичальників на якісних, які зможуть погасити кредит і неякісних, які не матимуть можливості виплатити кредит.

1.2 Оцінка кредитоспроможності як спосіб зниження ризиків при кредитуванні

Доходи від кредитних операцій є основним джерелом прибутку банків та фінансових організацій. Відсутність ефективних методик оцінювання кредитоспроможності фізичних осіб в кінцевому результаті може привести банк до виникнення значних проблем, адже кожний недооцінений банківський ризик перетворюється у ризик структурний та завдає збитки банку в цілому. Недостатня ефективність менеджменту та відсутність дієвих систем підтримки прийняття рішень у кредитних організаціях в умовах посилення конкуренції на ринку банківських продуктів призводять до збільшення ризиків та, відповідно, до зниження рентабельності банківських операцій та в результаті зниження прибутку банківської установи.

Кредитоспроможність – це здатність позичальника повернути борг банку у зазначені при оформленні кредиту сроки. Якщо кредитор впевнений, що позичальник своєчасно виконає свої боргові зобов'язання, позичальник вважається кредитоспроможним. Якщо позичальник самостійно оцінить свою кредитоспроможність, це призведе до конфлікту інтересів. Тому досвідчені фінансові установи проводять оцінку фізичних осіб та юридичних осіб щоб визначити пов'язаний ризик і ймовірність погашення.

Майже кожний прострочений або неповернений кредит є наслідком недостатньо точної оцінки кредитоспроможності позичальників, і лише незначна частина кредитів - форс-мажорні обставини (стихійні лиха, хвороба кредитоодержувача тощо). Саме тому оцінка кредитоспроможності позичальника є одним з найважливіших процесів при кредитуванні, який потребує постійних досліджень і удосконалень.

1.3 Огляд існуючих методів оцінки кредитоспроможності

Як було зазначено раніше – оцінка кредитоспроможності позичальника є одним з найважливіших процесів банківської установи чи фінансової організації при наданні споживчого кредиту. Завдяки такій оцінці банківські установи та фінансові організації зможуть досягти суттєвого зниження ймовірності ризиків та фінансової стабільності установи.

Кредитоспроможність — це готовність кредитора довірити позичальнику кредит. Кредитоспроможний позичальник — це той, кого кредитор вважає достатньо готовим, здатним і відповідальним для здійснення платежів за кредитом згідно з домовленістю, доки позика не буде погашена. Кредитоспроможність або ймовірність того, що позичальник погасить борг, є основним фактором у тому, чи буде схвалено кредит, автопозику чи іпотеку. Кредитор враховує заборгованість позичальника, його кредитну історію та здатність виплачувати кредити.

Хоча кожен кредитор має власне визначення кредитоспроможності та різні способи її вимірювання, можна виділити чотири основні методи оцінки кредитоспроможності фізичних осіб:

1. Оцінка по платоспроможності;
2. Оцінка за кредитною історією;
3. Андеррайтинг;
4. Скорингова оцінка.

1.3.1 Оцінка по платоспроможності

Банківські та фінансові установи, які надають кредити на кілька років, перевіряють платоспроможність на основі комерційних і статистичних процедур. Тому, коли клієнт банку подає заявку на кредит, банк запитує поточну довідку про доходи. Позичальник в свою чергу зобов'язаний надати інформацію про свої регулярні платіжні зобов'язання. У рамках кредитної оцінки банк визначає, чи достатньо різниці між доходами та витратами позичальника для регулярної виплати кредиту. У цьому процесі, банківська або фінансова установа, що надає кредит, також враховує витрати на життя та можливі активи позичальника, які можна використовувати як заставу. Також показник платоспроможності залежить від ступеню ризику втрати основного джерела доходу.

1.3.2 Оцінка за кредитною історією

В основі оцінки кредитоспроможності за кредитною історією лежить вивчення та аналіз кредитної історії позичальника. На основі даних, що містяться у заявці позичальника на кредит: ПІБ, адреса місця проживання, номер паспорту та інші, банківська або фінансова установа збирають інформацію про кредити позичальника: інформація про непогашені кредити, кількість кредитних карток клієнта, кількість активних та погащених кредитів, на скільки вчасно клієнт

оплачує свої рахунки. На основі зібраних даних формується кредитна історія.

В Україні діє закон «Про організацію формування та обігу кредитних історій»[3]. На основі даного закону створюються бюро кредитних історій, до яких звертаються банки та фінансові установи для отримання даних про кредитну історію позичальника. Згідно з законом «Про організацію формування та обігу кредитних історій»[3], кредитна історія - це сукупність інформації про юридичну або фізичну особу, що її ідентифікує, відомостей про виконання нею зобов'язань за кредитними правочинами, іншої відкритої інформації відповідно до Закону.

Бюро кредитних історій - юридична особа, виключною діяльністю якої є збір, зберігання, використання інформації, яка складає кредитну історію. Бюро кредитних історій виступають як інформаційні посередники. Вони можуть бути засновані і належати самим банківським та фінансовим організаціям, що кредитують, або вести свою діяльність та отримувати від неї прибуток незалежно. Кредитори забезпечують бюро кредитних історій даними про своїх клієнтів. Бюро зіставляє їх з інформацією, отриманою з інших джерел (суди, державні реєстраційні та податкові органи тощо) та формує картотеку на кожного позичальника. При регулярному та достовірному наданні інформації бюро кредитних історій про своїх клієнтів кредитори можуть постійно отримувати звіти про кредитні операції потенційних позичальників.

1.3.3 Андеррайтинг

Підвищення ефективності кредитування безпосередньо залежить від результатів андеррайтингу.

Андеррайтинг (англ. underwriting) буквально означає «підпис під документом / підпис під ризиком»; його головною функцією є виявлення можливих ризиків під час укладання кредитного договору, оцінка ймовірності погашення кредиту.

Процес андеррайтингу складається з таких операцій:

1) аналіз ризиків (збір інформації про фінансовий стан особи, яка потребує кредиту; вивчення відповідних документів; консультації зі сторонніми експертами – медиками, юристами, технологами тощо; аналіз кредитної історії позичальника; розрахунки оптимальних умов надання послуг, тарифних ставок і виплат по зобов'язаннях);

2) прийняття рішення про надання кредиту або відмову в кредитуванні;

3) компромісне коригування умов кредиту (зменшення суми кредиту, яка була запрошена; зміна терміну погашення кредиту; встановлення додаткового забезпечення – поруки, застави, страхового покриття тощо);

6) розробка заходів щодо зниження ризиків;

7) контроль факторів, що сприяють чи перешкоджають реалізації ризиків;

8) підтвердження безпеки кредитування для банку або фінансової установи.

При управлінні андеррайтингом беруться до уваги фактори мінливого зовнішнього середовища (зокрема, рівень економічного розвитку країни, зміни реальних доходів позичальника, його купівельної спроможності тощо), що впливають на успіх кредитування.

За результатами проведення андеррайтингу потенційного позичальника банк приймає позитивне чи компромісне рішення або відмовляє у наданні кредиту.

1.3.4 Скорингова оцінка

Скоринг — це сучасна ефективна технологія для кредитних рішень. Її сутність полягає у визначенні сукупного кредитного балу позичальника. Формально технологія скорингу ґрунтується на оцінках позичальників по ряду критеріїв, які мають різну значимість, після чого на підставі отриманих оцінок визначається сукупний кредитний бал.

Скорингова оцінка може базуватися на різних показниках — соціально-демографічних (вік, кількість утриманців, дохід, вартість житла тощо),

професійно-кваліфікаційних (рід занять, стаж, посада), поведінкових (моральні якості клієнта, його бажання і здатність заробити кошти на погашення позички). Найсильнішою передбачуваністю характеризуються поведінкові характеристики, тому що вони відображають ставлення позичальника до виплати боргу в минулому та пов’язані з надійністю чи ненадійністю клієнта.

Перелік показників, що характеризують кредитоспроможність позичальника, залежить від виду кредиту, строку кредитування, кредитної історії позичальника. Значення оптимальних показників для різних позичальників можуть відрізнятися залежно від їх діяльності та індивідуальних умов угоди.

Таким чином, скорингова оцінка позичальника є математично-статистичним інструментом, що за допомогою спеціальних формул та коригувальних коефіцієнтів дозволяє в інтегральному чисельному вигляді визначити ймовірність виплати позичальником кредиту у встановлені терміни в повному обсязі.

Кредитний скоринг необхідний для якісної оцінки кредитного ризику (запобігання шахрайства, підтвердження платоспроможності і платіжної дисципліни), зменшення витрат і часу для обробки заяв на отримання кредиту і прийняття по ним неупереджених рішень, мінімізації операційного ризику неповернення кредиту.

Спільно з кредитними звітами і верифікацією позичальника скоринг дозволяє вибудувати ефективну систему прийняття рішень про кредитування.

1.4 Переваги та недоліки існуючих методів

Як було зазначено раніше, кредитоспроможність є основним фактором у тому, чи буде схвалено кредит клієнту банківської або фінансової установи. Існує ряд основних методів оцінки кредитоспроможності позичальника, що були розглянуті раніше. Кожний метод має свої переваги і недоліки.

Такі методи оцінки кредистоспроможності як оцінка по платоспроможності, за кредитною історією позичальника та андрейтинг зазвичай використовуються при наданні довгострокових кредитів, іпотек, кредитів на придбання дорогих речей тощо.

Основною і найбільш значимою перевагою даних методів є те, що вони дозволяють сформувати кредитний портфель більш високої якості та значно знизити кредитні ризики. Дана перевага обумовлена тим, що у процесі оцінки кредитоспроможності позичальника даними методами приймають участь широке коло банківських підрозділів: юридична служба, служба безпеки, відділ цінних паперів, відділ житлового будівництва та інші. У кінцевому результаті оцінювання банк має можливість виробити до будь-якого потенційного позичальника індивідуальний підхід кредитування і тим самим знизити ризики.

Попри перевагу формування кредитного портфелю високої якості і зниження кредитних ризиків банківської установи, використання таких методів має деякі недоліки, а саме:

1. Низька швидкість прийняття рішень.

Для оцінки платоспроможності клієнта кредитним інспекторам потрібно проаналізувати дуже багато документів. Список їх досить великий і налічує близько п’ятнадцяти найменувань;

2. Зниження кількості потенційних позичальників.

Обов’язкове надання клієнтом широкого списку необхідних документів для підтвердження його платоспроможності обмежує коло потенційних позичальників банку;

3. Трудомісткість процесу оцінки.

Операціями з іпотечного кредитування фізичних осіб у банку займається досить широке коло банківських підрозділів: юридична служба, служба безпеки, відділ цінних паперів, відділ житлового будівництва та ін. оцінки та умови надання іпотечних кредитів. Мінус цієї оцінки – трудомісткість її виконання, що вимагає особливої кваліфікації банківських працівників.

Скорингові моделі як єдиний метод при оцінці кредитоспроможності в основному використовуються банківською установою при наданні експрес кредитів або при видачі кредитних карток.

Скоринг – це математична модель, яка на базі наявних даних про кредитну історію попередніх клієнтів банку або фінансової установи визначає ймовірність того, що позичальник поверне борг у межах зазначених термінів. Можна виділити основні переваги та недоліки скорингових моделей для оцінюваннння кредитоспроможності позичальника.

Переваги скориногових моделей:

1. Зниження ймовірності ризиків.

Завдяки скоринговим системам банківські та фінансові установи можуть досягти зниження ймовірності ризиків та підвищити фінансову стабільність установи;

2. Висока швидкість прийняття рішень.

За допомогою впровадження скорингової моделі в процес прийняття рішень, обґрунтовані рішення можна приймати швидше або взагалі повністю автоматизувати процес;

3. Об’єктивність прийнятого рішення.

Оскільки критерії, які використовуються для обчислення балів, заздалегідь визначені та визначені в моделі оцінки, кожна оцінка обчислюється об’єктивним і однорідним способом. Рішення більше не підлягають різним інтерпретаціям або особистому досвіду працівника банку;

4. Відсутність потреби довгого навчання робітників кредитного департаменту.

Ефективно реалізована скорингова система прийняття рішень дає можливість приймати рішення щодо надання кредиту працівникам без вищої економічної освіти. Завдяки скоринговій системі нефінансові працініки мають можливість приймаюти рішення щодо надання кредиту. Скориногова система дозволяє їм приймати обґрунтовані рішення, які базуються на знаннях та досвіді колег у фінансах.

Попри суттєві переваги скорингових моделей серед інших методів оцінки кредитоспроможності позичальника, використання таких моделей має деякі недоліки:

1. Оцінка позичальника проводиться на базі інформації про клієнтів, яким вже видали кредит.

Для розробки скорингової моделі потрібні дані про клієнтів яким вже видали кредит. Для навчання скорингової моделі потрібні дані як про надійних позичальників так і про тих, які не виконали вчасної сплати платежу або не погасили борг. Тобто перед розробкою скорингової моделі оцінки кредитоспроможності позичальника банківській або фінансовій установі необхідно видати певну кількість кредитів серед яких будуть і непрацюючі кредити;

2. Короткочасність терміну дії моделі.

Оскільки скорингові моделі будуються на основі даних найбільш ранніх клієнтів, з часом результати роботи скорингової моделі погіршуються. Через це з'являється потреба періодичної перевірки якості роботи моделі співробітниками банку. У разі погіршення якості роботи моделі, її треба оновити або розробити нову;

3. Нездатність чітко враховувати економічні умови.

Необхідно зазначити, що на даний момент банківські та фінансові установи можуть використовувати як одну так і декілька методик одночасно для оцінювання кредитоспроможності позичальника. При оцінюванні кредитоспроможності позичальника кредитор обирає конкретну методику оцінювання залежно від цілей аналізу позичальника, виду кредиту на який претендує клієнт, термінів кредитування та інших обставин.

1.5 Постановка задачі

На даний момент системи підтримки прийняття рішень стали ключовим інструментом для банків та інших фінансових установ при оцінці

кредитоспроможності позичальників. Завдяки таким системам банківські установи України зможуть досягти зниження ймовірності ризиків та підвищення ефективності та фінансової стабільності установи. Основна ідея систем підтримки прийняття рішень при кредитуванні полягає у класифікації потенційних позичальників на якісних, які зможуть погасити кредит і неякісних , які не матимуть можливості виплатити кредит.

Темою даної дипломної роботи є система підтримки прийняття рішень при кредитуванні та її розробка.

Метою даної дипломної роботи є підвищення ефективності процесу оцінювання кредитоспроможності позичальників шляхом розробки системи підтримки прийняття рішень при кредитуванні.

Об’єктом дослідження є клієнти банку, що претендують на кредит, представлені даними їх характеристик.

Предметом дослідження даної роботи є моделі машинного навчання для проведення класифікації позичальників на основі обраних даних.

Архітектуру системи підтримки прийняяття рішень при кредитуванні у вигляді контекстної діаграми IDEF0 можна побачити на наступному рисунку (рисунок 1.5).

Рисунок 1.5 Ахітектура СППР при кредитуванні у вигляді контекстної діаграми IDEF0

Основними заікавленими сторонами у розробці та впровадженні даної системи підтримки прийняття рішень при кредитуванні є банківські та фінансові установи, що займаються кредитування фізичних осіб та самі позичальники. Для банківських та фінансових установ впровадження такої системи дасть змогу знизити ризики при кредитуванні і тим самим збільшити прибуковість організації. Для позичальників така система значно скоротить час прийнятя та розгляду їх зявок на кредит.

Розглядаючи задачу розробки системи підтримки прийняття рішень при кредитуванні, основні завдання, що стоять перед банком можна сформулювати наступним чином:

1. Вибір методів для розробки моделі;
2. Вибір метрик для оцінювання точності моделі на вибраних методах;
3. Збір, аналіз та попередня обробка даних про клієнтів банку;
4. Оцінка методів;
5. Навчання моделі через метод, що дав найбільшу точність;
6. Створення архітектури бази даних;
7. Програмна реалізація СППР;
8. Створення інтерфейсу СППР;

1.6 Висновки до розділу

У даному розділі було розглянуто актуальність дослідження, а саме стан кредитного портфелю українських банківських установ. Було виявлено, що майже за кожним другим кредитом, наданим позичальнику в Україні, не відбувається вчасної сплати платежу або погашення боргу, що обумовлює актуальність розробки СППР при кредитуванні. Також у першому розділі було наведено означення кредитоспроможності і виявлено, що оцінка кредитоспроможності позичальника є одним з ключових факторів для зниження ризиків при кредитуванні. Були розглянуті основні існуючі методи для оцінки

кредитоспроможності і виявлені їх переваги та недоліки. Було доведено, що оцінка кредитоспроможності за допомогою скорингових технологій є потужним інструментарієм кредитного ризик-менеджементу, що дозволить значно знизити ризики банківської або фінансової установи при наданні кредитів.

РОЗДІЛ 2. АНАЛІЗ ПРОЕКТНИХ РІШЕНЬ

2.1 Огляд основних методів реалізації алгоритму кредитного скорингу

За останні роки було розроблено багато моделей кредитного скорингу для оцінки кредитоспроможності претендентів на кредит. Традиційно рішення про надання кредиту заявнику ґрунтувалося на суб'єктивних судженнях, зроблених експертами з використанням минулого досвіду та деяких керівних принципів. Однак цей метод страждає від високих витрат на навчання, частих неправильних рішень і суперечливих рішень, прийнятих різними експертами для одного і того ж застосування. Протягом останніх кількох десятиліть класифікація кредитів постійно привертає велику увагу академічних дослідників та фінансових установ, що призвело до створення різноманітних алгоритмів, відомих як моделі класифікації кредитів. Методи класифікації кредитів зазвичай оцінюються за трьома властивостями, а саме: точністю, інтерпретацією та ефективністю обчислень. Точність є суттєвою вимогою, яка означає, що може бути згенеровано максимально можливу кількість правильних рішень. Незначне підвищення точності означає значну економію для фінансової установи. Інтерпретація є досить важливою не лише для осіб, які приймають рішення, але й для претендентів на кредит, оскільки вона являє собою здатність генерувати зрозумілий для заявників механізм оцінки. Ефективність обчислень являє собою швидкість класифікації. Оцінювачам корисно якнайшвидше прийняти рішення, надавати кредит чи ні, відповідно до результату класифікації.

Розглянемо методи, що використовуються для реалізації моделі кредитного скорингу:

1) Генетичні алгоритми;
2) Імітаційне моделювання;
3) Міркування на основі прецедентів;
4) Штучні нейронні мережі;
5) Методи машинного навчання для вирішення задач класифікації.

Генетичні алгоритми.

Генетичний алгоритм — це адаптивний евристичний алгоритм пошуку. Він використовується для вирішення завдань оптимізації в машинному навчанні. Це один із важливих алгоритмів, оскільки він допомагає вирішувати складні проблеми, вирішення яких займе багато часу. Так як логіка кредитування є складною, її побудова за допомогою генетичних алгоритмів буде дуже трудомістким завданням адже моделювання відбувається шляхом випадкового підбору властивостей.

Імітаційне моделювання.

Імітаційне моделювання безпечно та ефективно вирішує реальні проблеми. Даний метод аналізу легко перевірити, передати та зрозуміти. У різних галузях і дисциплінах імітаційне моделювання надає цінні рішення, даючи чітке уявлення про складні системи. Проте для використання цього методу необхідно чітко розуміти структуру процесу, а у задачі оцінки кредитоспроможності - логіка кредитування важко формалізується.

Міркування на основі прецедентів.

Даний метод виконує пошук рішень шляхом адаптації подібних ситуацій у базі даних, де зібарний минулий досвід вирішення даної задачі. Основними недоліками міркувань на основі прецендентів є невміння узагальнювати дані, складність пошуку рішень подібних ситуацій та адаптацій рішення.

Штучні нейронні мережі.

Нейронні мережі дозволяють моделювати найскладніші процеси. Позитивною особливістю використання нейромереж у задачі оцінювання кредоспроможності є механізм її навчання. Нейронні мережі не потребують формалізації логіки кредитування. Користувачеві потрібно лише відібрати потрібні дані, визначити необхідну архітектуру нейронної мережі і запустити її навчання. Попри високу точність результатів, нейронні мережі мають деякі недоліки, а саме складність в інтерпретуванні результатів, низька швидкість прийняття рішень.

Методи машинного навчання для вирішення задач класифікації

У кредитному скорингу є дві основні проблеми: надання позики поганому позичальнику і відмова хорошому. Однак існує багато способів вирішення цієї проблеми, і деякі з них ефективніші за інші. Передові статистичні та математичні методи забезпечують швидкі й автоматичні інструменти, які допомагають приймати ефективні рішення. Вважається, що моделі, засновані на алгоритмах машинного навчання є більш ефективними для підтримки прийняття рішень у фінансових компаніях. Комбінація методів машинного навчання може зробити великий внесок у систему кредитування і буде набагато простішою з точки зору використання. Оскільки прогнози машинного навчання більш адаптивні та гнучкі до змін, вони можуть давати точніші результати. Найбільшою перевагою використання методів машинного навчання у задачі оцінки кредитоспроможнсті є те, що існує безліч різних алгоритмів класифікації для навчання моделі. Залежно від вхідних даних, кожний алгоритм машинноо навчання має свої переваги та недоліки. Наприклад на одних даних конкретний алгоритм машиннго навчання є ефективним, а на інших даних його точність нижча за інші алгоритми машинного навчання. Велика кількість різноманітих алгоритмів машинного навчання дають змогу підібрати найефективніший у тих чи інших вхідних даних. Також одною с суттєвих переваг використання алгоритмів класифікації машинного навчання для реалізації моделі кредитного скорингу є те, що реалізація моделі займає менше часу в порівнянні з іншими методами, адже усі алгоритми давно реалізовані, а деякі програмні засоби мають пакети з вже реалізованими алгоритмами. Тому перед розробником стоїть задача встановлення пакету з вже реалізованими алгоритмами, ознайомитися с принципами роботи кожного з них і перевірити їх роботу на своїх даних.

Для розробки моделі кредитного скорингу було прийнято рішення використовувати методи машинного навчання для вирішення задачі класифікації через їх велике різноманіття та легкість в реалізації.

2.2 Огляд алгоритмів машинного навчання для вирішення задачі класифікації

Класифікація є одним із двох основних типів методів навчання з вчителем, іншим є регресія. Моделі класифікації передбачають мітку класу до якого належить об’єкт. Підходи до класифікації корисні для бізнес-задач, які мають велику кількість даних, включаючи мітки, які вказують, чи входить щось у ту чи іншу групу. У даній роботі буде розглянуто основні алгоритми машинного навчання для задачі класифікації, а саме:

- Наївний Байєсівський класифікатор
- Логістична регресія
- Дискримінантний аналіз
- K-найближчих сусідів
- Дерева прийняття рішень
- Випадковий ліс

2.2.1 Наївний Байєсівський класифікатор

Наївний Байєс — це ймовірнісний алгоритм, який зазвичай використовується для задач класифікації. Алгоритм є простим, інтуїтивно зрозумілим та в багатьох випадках працює дуже ефективно. Алгоритм оснований на теоремі Байеса. Основною ідеєю алгоритму є те, що ефект кожної функції на відповідний клас цільової функції не залежить від інших функцій. Тобто алгоритм припускає що всі функції є незалежними, навіть якщо функції залежні між собою, алгоритм їх буде розглядати незалежними одна від одної. Так, за допомогою цього наївного припущення (наївного, оскільки ознаки рідко є незалежними) ми можемо зробити класифікацію з набагато меншою кількістю параметрів. Через це алгоритм спрощує складні обчислення і має назву «наївний».

Теорема Байеса (формула 2.1):

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \quad (2.1)$$

де:

P(c|x) – апостеріорна ймовірність класу «*c*» (значення цільової змінної) при даному значенні ознаки x.

P(c) – апріорна ймовірність класу «*c*».

P(x|c) –ймовірність даного значення признака при даному класі «c».

P(x) – апріорна ймовірність даного значення ознаки.

Переваги методу:

- Простота і легкість реалізації.
- Має значні переваги у швидкості і точності серед інших методів класифікації, коли припущення щодо незалежності функцій виконується.
- Може обробляти як безперервні, так і дискретні дані.

Недоліки методу:

- Через припущення, що всі ознаки є незалежними (що рідко трапляється в реальному житті) розрахована ймовірність часто неточна.
- При великій кількості даних інші складніші моделі, як правило перевершують даний алгоритм.

2.2.2 Логістична регресія

Логістична регресія — це метод класифікації, запозичений машинним навчанням зі сфери статистики. Логістична регресія — це статистичний метод для аналізу набору даних, у якому є одна або кілька незалежних змінних, які визначають результат. Мета використання логістичної регресії полягає в тому, щоб знайти найкращу модель для опису зв’язку між залежною та незалежною

змінною. Так як результатом лінійної регресії є лінійна інтерполяція між точками, тобто немає значення порога при якому ми можемо віднести записи до певного класу. Рішенням для задачі класифікації є логістична регресія. На заміну апроксимації прямої лінійної регресії, логістична регресія стискає дані лінійної функції в діапазон [0 ; 1]. Логістична функція має вигляд (формула 2.2):

$$logistic(\eta) = \frac{1}{1 + \exp(-\eta)} \tag{2.2}$$

де η – лінійне рівняння зв'язку між класом цільової функції і її характеристиками.

Переваги методу:

- Логістична регресія виявляється дуже ефективною, коли набір даних має властивості, які лінійно розділені.
- Висока швидкість навчання в порівнянні з більшістю інших складних методів
- Окрім самої класифікації, модель дає ймовірність з якою вона відносить запис до певного класу.

Недоліки методу:

- Алгоритм чутливий до викидів.
- Оскільки логістична регресія має лінійну поверхню рішення, вона працює не так ефективно з нелінійно роздільними даними, що на практиці зустрічаються набагато частіше.
- Логістична регресія вимагає великого набору даних , а також достатніх навчальних прикладів для всіх класів, які необхідно визначити.

2.2.3 Дискримінантний аналіз

Лінійний дискримінантний аналіз (Linear Discriminant Analysis) і квадратичний дискримінантний аналіз (Quadratic Discriminant Analysis) є двома

класичними класифікаторами, які мають, як випливає з назви, лінійну та квадратичну поверхню рішення відповідно. Ці класифікатори є привабливими, оскільки вони мають закриті рішення, які можна легко обчислити, добре працюють на практиці та не мають гіперпараметрів для налаштування. Лінійний дискримінантний аналіз може вивчати лише лінійні кордони, тоді як квадратичний дискримінантний аналіз може вивчати квадратичні межі і тому є більш гнучким.

Можна виділити основні завдання дискримінантного аналізу:

- З'ясування, які ознаки найкраще дозволяють класифікувати об'єкти.
- З'ясування правил класифікації існуючих об'єктів.
- Класифікація нових невідомих об'єктів згідно правилу класифікації.

Переваги методу:

- Простота і легкість реалізації
- Легкість в інтерпретуванні результатів

Недоліки методу:

- Чутливість до розподілу даних
- Вимагає припущення нормального розподілу за ознаками

2.2.4 K-найближчих сусідів

Алгоритм k-найближчих сусідів (KNN) — це простий, легкий у реалізації алгоритм машинного навчання з учителем, який можна використовувати для вирішення проблем як класифікації, так і регресії. Суть алгоритму полягає в тому, що елемент присвоюється тому класу, який є найпоширенішим серед схожих йому елементів(сусідів).

Алгоритм k-найближчих сусідів складається з таких етапів:

1. Вибір значення параметру k (кількість елементів);
2. Обчислення відстані між елементом та іншими елементами;
3. Сортування відстаней у зростаючому порядку;

4. Вибір перших k елементів у відсортованому списку;

5. Отримання значень цільової функції даних елементів;

6. Рішення до якого класу належить елемент, на основі класу більшості сусідніх йому елементів.

Зазвичай схожість елементів вимірюється за формулою Евклідової відстані (формула 2.3):

$$D_E = \sqrt{\sum_{i}^{n} (x_i - y_i)^2} \tag{2.3}$$

де:

n – кількість атрибутів;

x, y – точки між якими треба виміряти відстань

Переваги алгоритму:

- Висока швидкість алгоритму за рахунок відсутність процесу навчання
- Додавання нових даних не впливає на точність алгоритму
- Легка реалізація алгоритму

Недоліки алгоритму:

- Висока чутливість до зашумлених даних, пропущених значень і викидів.
- Низька швидкість роботи алгоритму при великих наборах даних

2.2.5 Дерева рішень

Дерево рішень - алгорити машинного навчання, який окрім вирішення задачі класифікації також може використовуватися для розв'язування задач регресії. Дерева рішень також мають назву CART (Classification and Regression Tree), що означає дерева класифікації та регресії. Хоча це один з найпростіших алгоритмів класифікації, якщо його параметри правильно налаштовані, він може дати неймовірно точні результати. Суть алгоритма полягає у безперервному

розбитті точок даних відповідно до певних параметрів та проблеми, яку намагається вирішити алгоритм.

Кожне дерево рішень містить:

- Вузли
- Ребра (гілки дерева)
- Кінцеві вузли

Внутрішні вузли, присутні в дереві, описують різні тестові випадки та перевіряють значення відповідного аргументу. Ребра або гілки дерева відповідають за результати тесту Кожне ребро або гілка дерева відповідає результату тесту, а результат представлений листковим вузлом або кінцевим вузлом. Кінцевий вузол який представляє кінцевий розподіл класів.

Алгоритм можна розглядати як графічну деревоподібну структуру, яка використовує різні налаштовані параметри для прогнозування результатів. Дерева рішень застосовують підхід зверху вниз до набору даних, який подається під час навчання. Алгоритм дерева рішень працює як набір вкладених операторів «if-else», де перевіряються послідовні умови поки модель не досягне висновку (віднесення до певного класу). Дерева рішень будуються за допомогою евристики, яка називається рекурсивним розбиттям (зазвичай іменується як Розділяй і володарюй). Кожен вузол, наступний за кореневим вузлом, розбивається на кілька вузлів. Ключова ідея полягає в тому, щоб використовувати дерево рішень для поділу простору даних на щільні області та розріджені області. Розщеплення бінарного дерева може бути як двійковим, так і багатостороннім. Алгоритм продовжує розбивати дерево, поки дані не стануть достатньо однорідними. Наприкінці навчання повертається дерево рішень, яке можна використовувати для створення оптимальних категоризованих прогнозів.

Розбиття вузлів виконується, щоб розбити елементи на групи так, щоб зменшилася ентропія, тобто щоб групи даних були більш однорідні. Для визначення ентропії зазвичай використовується формула ентропії Шенона (формула 2.4):

$$S = -\sum_{i=1}^{N} p_i \log_2 p_i \qquad (2.4)$$

де:

p_i – частота входження елементу у набір даних;

N – кількість класів цільової функції.

Переваги методу:

- Оптимальне відношення точності і швидкості алгоритму в порівнянні з іншими методами класифікації
- Досить легкий в розумінні і інтерпретуванні
- Стійкість до викидів і пропущених значень

Недоліки методу:

- Схильність до перенавчання
- Не стійкість до змін навчальної вибірки

Приклад дерева рішень (Рисунок 2.1).

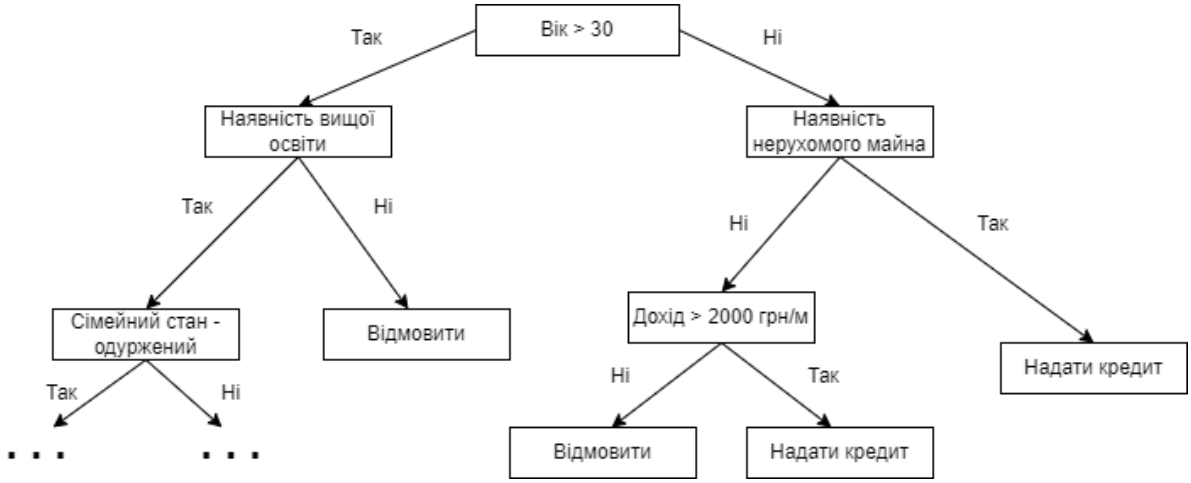

Рисунок 2.1 Приклад дерева рішень

2.2.6 Випадковий ліс

Випадковий ліс — це широко використовуваний алгоритм машинного навчання, який об’єднує результати кількох дерев рішень для досягнення єдиного результату. Через простоту використання та гнучкість алгоритм є досить поширеним у використанні в машинному навчанні. Так як і дерева рішень алгоритм вирішує як задачу класифікації, так і з регресії. Хоча дерева рішень є звичайними алгоритмами навчання з наглядом, вони можуть бути схильними до проблем, таких як зміщення та переобладнання. Однак, коли кілька дерев рішень утворюють ансамбль в алгоритмі випадкового лісу, вони передбачають більш точні результати, особливо коли окремі дерева не корелюють одне з одним.

Алгоритм випадкового лісу складається з таких кроків:

1. Вибір випадкових вибірок із заданого набору даних.
2. Побудова дерев рішень для кожної вибірки
3. Отримання прогнозу з кожного дерева рішень
4. Вибір результату прогнозу за найбільшою кількістю голосів

Переваги алгоритму:

- Знижений ризик переобладнання
- Добре працює як з категоріальними так із безперервними даними
- Висока точність алгоритму
- Стійкість до викидів у даних

Недоліки алгоритму:

- Потребує більшої обчислювальної потужності в порівнянні з іншими методами класифікації
- Тривалий період навчання

2.3 Метрики оцінювання алгоритмів

Для оцінки точності класифікації методів машинного навчання використовують матрицю помилок (Confusion Matrix) (таблиця 2.1).

	Y=1	Y=0
Y*predict*=1	True Positive (TP)	False Positive (FP)
Y*predict*=0	False Negative (FN)	True Negative (TN)

Таблиця 2.1 Матриця помилок

Матриця помилок дає краще уявлення про те, що створена модель класифікації працює правильно і які типи помилок вона допускає. Матриця помилок складається з стовпців, що відповідають справжнім класам елементів, і рядків – клас до якого розроблена модель віднесла елемент. Значеннями даної матриці є кількість елементів одного з класів, шо були віднесені моделлю до певного класу.

Значення матриці помилок:

- True Positive (TP) – модель віднесла елемент, що належить класу 1 до класу 1.
- False Negative (FN) – модель віднесла елемент, що належить класу 1 до класу 0.
- False Positive (FP) - модель віднесла елемент, що належить класу 0 до класу 1.
- True Negative (TN) - модель віднесла елемент, що належить класу 0 до класу 0.

Accuracy – це метрика оцінювання точності моделі класифікації, яка обраховується як відношення правильних прогнозів моделі до числа всіх елементів у вибірці. Accuracy обраховується за формулою (формула 2.5):

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.5}$$

Дана метрика є досить простою і очевидною. Проте вона рідко використовується у реальних задачах через її неточність при оцінюванні моделей, навчених на вибірці з нерівними класами.

Precision, recall – метрики, що дають оцінку точності роботи моделі по кожному з класів. Метрика precision визначає відношення кількості об’єктів, що класифікатор відніс до позитивного класу і вони дійсно мають позитивний клас до усіх об’єктів, яких класифікатор відніс до позитивного класу. Метрика Recall визначає відношення кількості об’єктів, що класифікатор відніс до позитивного класу і вони дійсно мають позитивний клас до усіх об’єктів даного класу вибірки. Метрика precision обраховуються за формулою (формула 2.6):

$$precision = \frac{TP}{TP + FP} \tag{2.6}$$

Метрика recall обраховуються за формулою (формула 2.7):

$$recall = \frac{TP}{TP + FN} \tag{2.7}$$

На відміну від метрики accuracy - precision і recall не залежать від незбалансованості класів і тому вони є застосованими для таких даних.

F1 Score – це метрика є показником ефективності для алгоритму класифікації і розраховується як середнє гармонійне значення precision і recall. Так як і метрики precision і recall, метрика F1 Score не залежить від незбалансованості класів вибірки. Метрика F1 Score обраховуються за формулою (формула 2.8):

$$F_1 = \frac{2}{precision^{-1} + recall^{-1}} \tag{2.8}$$

AUC-ROC – це метрика оцінювання моделі класифікації, що дозволяє оцінити модель загалом, не прив'язуючись до конкретного порогу. AUC-ROC - площа під кривою помилок. Дана крива являє собою лінію від (0,0) до (1,1) в координатах True Positive Rate (TPR) та False Positive Rate (FPR).
TPR обчислюються за формулою (формула 2.9):

$$TPR = \frac{TP}{TP + FN} \tag{2.9}$$

FPR обчислюються за формулою (формула 2.10):

$$FPR = \frac{FP}{FP + TN} \tag{2.10}$$

У загальному випадку, AUC ROC для бінарного рішення обчислюється за формулою (формула 2.11):

$$AUC = \frac{(1 + TPR - FPR)}{2} \tag{2.11}$$

2.4 Вибір програмних засобів для реалізації моделі

Працювати в індустрії машинного навчання означає мати справу з величезною кількістю даних, які потрібно обробляти найбільш зручним та ефективним способом. Мова програмування Python є найпопулярнішим програмним засобом, що використовується для машинного навчання. Причиною цього є те, що мова програмування Python містить великий вибір бібліотек, що використовуються у машинному навчанні. Також вона є доволі простою і гнучкою мовою програмування, що дозволяє реалізувати програмний продукт доволі просто і без витрат.

Саме через велику кількість переваг мови програмування Python у машинному навчанні для розробки моделі оцінки кредитоспроможності позичальника була обрана дана мова програмування.

2.4.1. Python

Понад 8,2 мільйона розробників у всьому світі використовують Python для кодування. Python п’ять років поспіль займає перше місце в щорічному рейтингу популярних мов програмування за IEEE Spectrum з результатом 100 балів [9]. Python став одним з найкращих вибором для аналізу даних, машинного навчання та штучного інтелекту – все завдяки його величезній кількості бібліотек, що дозволяють фахівцям машинного навчання з легкістю обробляти дані, перетворювати їх та використовувати методи машинного навчання. Також велика кількість інженерів машинного навчання обирають мову програмування Python через її незалежністю від платформи, меншою складністю та кращою читабельністю. Можна виділити основні переваги мови програмування Python:

- Велика колекція бібліотек і пакетів

Вбудовані бібліотеки та пакети Python забезпечують код базового рівня, тому інженерам машинного навчання не доведеться починати писати з нуля. Машинне навчання вимагає безперервної обробки даних, а Python має вбудовані бібліотеки та пакети майже для кожного завдання. Це допомагає інженерам з машинного навчання скоротити час розробки та підвищити продуктивність під час роботи зі складними програмами машинного навчання

- Читабельність

Математика машинного навчання зазвичай складна і неочевидна. Таким чином, читабельність коду надзвичайно важлива для успішної реалізації складних алгоритмів машинного навчання та різноманітних робочих процесів. Простий синтаксис Python дозволяють інженерам машинного навчання легко зосередитися на тому, що писати, замість того, щоб думати про те, як писати. Читабельність коду полегшує фахівцям, які практикують машинне

навчання, легко обмінюватися ідеями, алгоритмами та інструментами зі своїми колегами.

- Гнучкість

Багатопарадигмність і гнучкість Python дозволяють інженерам машинного навчання підійти до проблеми найпростішим способом. Він підтримує процедурний, функціональний, об'єктно-орієнтований та імперативний стиль програмування, що дозволяє експертам з машинного навчання комфортно працювати над тим, який підхід найкраще підходить. Гнучкість, яку пропонує Python, допомагає інженерам з машинного навчання вибрати стиль програмування на основі типу проблеми

2.4.2 Scikit-learn

Scikit learn - це бібліотека, яка використовується для виконання машинного навчання на Python. Scikit learn - бібліотека з відкритим вихідним кодом, яка ліцензована під BSD і може використовуватися як в академічних так і комерційних цілях.

Scikit-Learn дає доступ до безлічі різних алгоритмів класифікації. Основні з яких:

- Метод k-найближчих сусідів (K-Nearest Neighbors);
- Метод опорних векторів (Support Vector Machines);
- Класифікатор дерева рішень (Decision Tree Classifier) / Випадковий ліс (Random Forests);
- Наївний байесовский метод (Naive Bayes);
- Лінійний дискримінантний аналіз (Linear Discriminant Analysis);
- Логістична регресія (Logistic Regression).

2.5. Висновки до розділу

У другому розділі були розглянуті основні методи реалізації моделі кредитного скорингу. Для розробки моделі були обрані основні алгоритми машинного навчання для задачі класифікації. Було описано суть кожного з алгоритмів машинного навчання та наведено основні переваги та недоліки кожного з них. Також були розглянуті основні метрики для оцінювання точності моделі, побудованої на обраних методах. Було обґрунтовано та прийняте рішення використовувати мову програмування Python як основний програмний засіб для реалізації моделі кредитного скорингу.

РОЗДІЛ 3. РОЗРОБКА СППР ПРИ КРЕДИТУВАННІ

3.1. Збір та аналіз даних

Для розробки моделі оцінювання кредитоспроможності позичальника були взяті дані поведінки клієнтів, що оформили кредит в одному з Індійських банків. Вперше ці дані були були опубліковані компанією Univ.ai на щорічному конкурсі з машинного навчання.

Оригінальний набір даних містить 250000 записів, 12 незалежних атрибутів і одну цільову змінну. Кожний запис представляє дані надані позичальником про себе при заявці на кредит та цільову ознаку, що містить інформацію про надійність позичальника. Датасет складається з числових змінних і категоріальних (рисунок 3.1).

ID	INCOME	AGE	EXP	MARRIED	HOUSE_OWN	CAR_OWN	PROFESSION	CITY	STATE	JOB_YEAR	HOUSE_YEAR	RISK
1	1303835	23	3	single	rented	no	Mechanical_engineer	Rewa	Madhya_Pradesh	3	13	0
2	7574516	40	10	single	rented	no	Software_Developer	Parbhani	Maharashtra	9	13	0
3	3991815	66	4	married	rented	no	Technical_writer	Alappuzha	Kerala	4	10	0
4	6256451	41	2	single	rented	yes	Software_Developer	Bhubaneswar	Odisha	2	12	1
5	5768871	47	11	single	rented	no	Civil_servant	Tiruchirappalli[10]	Tamil_Nadu	3	14	1
...	...	...	...	...	...	...	...	...	...	...	...	...
251996	8154883	43	13	single	rented	no	Surgeon	Kolkata	West_Bengal	6	11	0
251997	2843572	26	10	single	rented	no	Army_officer	Rewa	Madhya_Pradesh	6	11	0
251998	4522448	46	7	single	rented	no	Design_Engineer	Kalyan-Dombivli	Maharashtra	7	12	0
251999	6507128	45	0	single	rented	no	Graphic_Designer	Pondicherry	Puducherry	0	10	0
252000	9070230	70	17	single	rented	no	Statistician	Avadi	Tamil_Nadu	7	11	0

Рисунок 3.1 Вхідні дані

Числові змінні:

1. «ID» – Індивідуальний порядковий номер кожного позичальника.
2. «INCOME» – річний дохід позичальника.
3. «AGE» – вік позичальника.
4. «EXP» – Професійний досвід позичальника.
5. «JOB_YEAR» – Кількість років, що позичальник працює на останній роботі.

6. «HOUSE_YEAR» – кількість років, що позичальник проживає у вказаному житлі.

Категоріальні змінні:

1. «MARRIED» – Сімейний стан позичальника.
2. «PROFESSION» - Професія позичальника
3. «HOUSE_OWN» – тип житла позичальника.
4. «CAR_OWN» - чи є особа власником автомобіля .
5. «CITY» - місто, в якому проживає позичальник
6. «STATE» - область, в якій проживає позичальник

Цільова ознака:

«RISK» – чи було виявлено прострочення кредиту за даним позичальником (1 – Так, 0 – Ні)

3.2. Попередня обробка даних

Попередня обробка даних є важливим кроком в процесі інтелектуального аналізу даних. Методи збору даних часто погано контролюються, що призводить до неприпустимих значень (таким як, дохід: -100), неможливим комбінаціям даних, відсутнім значенням та інше. При аналізі даних, що не захищені від такого роду проблем, можна прийти до неправильних висновків.

Якщо є багато зайвої інформації або зашумлених і недостовірних даних, то витяг знань під час тренування стає скрутним. Крок підготовки та фільтрації даних може зайняти значний час.

Попередня підготовка даних включає:

- очистку
- нормалізацію
- перетворення даних
- відбір ознак
- перевірку незбалансованості класів

Очистка даних.

Для очистки даних треба перевірити наявність пропущених значень у них. Дана таблиця показує скільки пропущених значень містить кожна ознака даного набору даних. Оскільки пропущених значень не виявлено, можна продовжувати процес попередньої обробки даних.

Нормалізація.

Деякі методи розробки моделі потребують нормалізації даних. Суть нормалізації полягає в тому, що усі дані переводяться в діапазон [0;1] шляхом ділення значення кожної змінної на максимальне значення цієї змінної.

Перетворення даних:

Іноді у наборі даних трапляються текстові ознаки, наприклад: Стать клієнта записана символами «Ч»(чоловік) і «Ж»(жінка). Такі ознаки потребують перетворення (чоловік - 1, жінка - 0). Оскільки цей набір даних містить такі ознаки, він потребує перетворення даних:

Змінна «MARRIED» є текстовою ознакою та набуває одного з двух значень 'single' – не заручений та 'married' – заручений. Після перетворення змінна married набуває значень: 0 – не заручений, 1 – заручений.

Змінна car_ownership є теж текстовою ознакою та набуває одного з двух значень 'yes' – позичальник має власний автомобіль та 'no' – позичальник не має авто. Після перетворення змінна car_ownership набуває значень: 0 – позичальник не має авто, 1 – позичальник має власний автомобіль.

Змінна «HOUSE_OWN» може набувати одного з трьох значень 'rented', 'norent_noown', 'owned'. У випадках коли змінна може набувати більше двох значень – значення змінної доречно перетворити на нові змінні (атрибути). Після перетворення даних до оригінального датасету додались такі нові атрибути: house_rented, house_norent_noown, house_owned, які набувають значень 1 або 0, в залежності який тип житла має позичальник.

Відбір ознак.

Основною передумовою використання техніки вибору ознак є те, що дані містять деякі ознаки, які є, або зайвими, або невідповідними і таким чином вони можуть бути видалені без значної втрати інформації.

Перевірка незбалансованості класів.

Досить поширеною проблемою у машинному навчанні, особливо в задачах класифікації є незбалансованість класів. Більшість алгоритмів машинного навчання найкраще працюють, коли кількість зразків у кожному класі приблизно однакова. Це пов’язано з тим, що більшість алгоритмів розроблені для отримання максимальної точності та зменшення помилок. Якщо набір даних знаходиться в дисбалансі, навчена модель показує досить високу точність, просто передбачивши клас більшості , але через дисбаланс класів вона не може захопити менший клас. Тому перевірка набору даних на дисбаланс класів є необхідною.

Відношення класів цільової змінної «RISK» (рисунок 3.2):

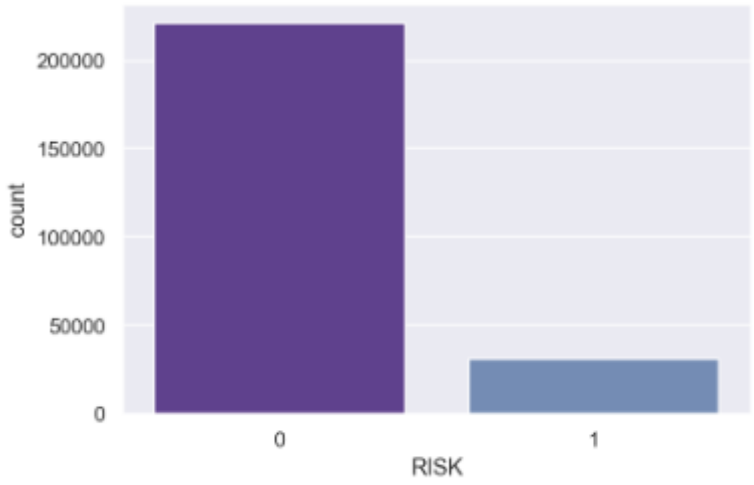

Рисунок 3.2 Відношення класів цільової змінної «RISK»

З діаграми видно, що кількість записів про кредитоспроможних позичальників перевищує кількість записів про клієнтів по яким було виявлено прострочення кредиту приблизно в 7 разів. Тобто у даному наборі даних

присутня проблема незбалансованості класів цільової змінної, що неприпустима для даної задачі.

Існує декілька методів вирішення проблеми незбалансованості класів. Основні методи збалансування класів полягають у вилученні записів із класу більшості та/або додаванні інших прикладів із класу меншості. Мінусом, який слід враховувати при вилученні записів із класу більшості, є те, що це може призвести до втрати інформації, яка може бути цінною. Тому було прийнято рішення випадково вибрати записи з класу меншості і додати їх до набору даних.

Результат збалансування класів (Рисунок 3.3):

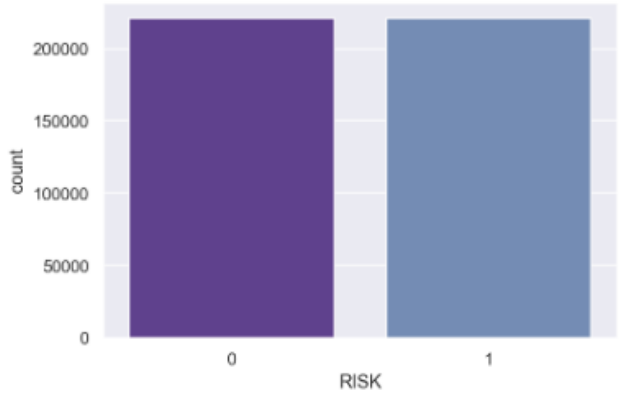

Рисунок 3.3 Результат збалансування класів цільової функції «RISK»

3.3. Аналіз результатів роботи алгоритмів класифікації

Після навчання моделей за описаними у другому розділі алгоритмами машинного навчання, створені моделі потребують перевірки точності за основними метриками оцінки моделей класифікації, що були наведені у другому розділі. Після перевірки точності були отримані такі результати (рисунок 3.4):

model	accuracy	precision	recall	f1	roc_auc
Random Forest	0.947460	0.908881	0.994638	0.949827	0.947460
Decision Tree	0.935791	0.889111	0.995774	0.939424	0.935791
K-Nearest Neighbors	0.839661	0.871630	0.796651	0.832452	0.839661
Bayes	0.532658	0.522501	0.758430	0.618674	0.532658
QuadraticDiscriminantAnalysis	0.536635	0.526879	0.718702	0.608001	0.536635
Linear Discriminant Analysis	0.538445	0.535962	0.572931	0.553830	0.538445
Logistic Regression	0.532228	0.530557	0.559601	0.544691	0.532228

Рисунок 3.4 Точнічть розроблених моделей

З таблиці результатів видно, що найбільшу точність класифікації показали такі методи: k-найближчих сусідів, дерево рішень та випадковий ліс.

Метод класифікації випадковий ліс дав найбільшу точність серед усіх інших методів. Через найбільшу точність серед інших методів по всім метрикам було прийняте рішення використовувати модель побудовану на алгоритмі випадковий ліс.

Також було проведено аналіз найбільш впливових атрибутів на результат класифікації (Рисунок 3.5).

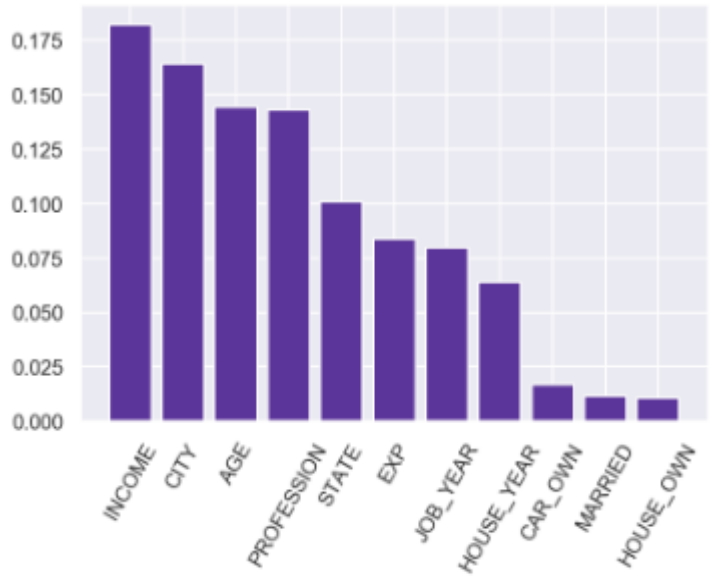

Рисунок 3.5 Діагарама найбільш впливових атрибутів

З діаграми видно, що найвпливовішими атрибутами у даній вибірці є дохід позичальника, місто його проживання та його вік.

3.4. Архітектура бази даних

Базою даних для побудови СППР при кредитуванні було обрано реляційну базу даних MySQL. MySQL — це система баз даних з відкритим вихідним кодом і безкоштовна у використанні, яка допомагає полегшити правильне та ефективне керування базами даних. Це також дуже надійний і стабільний спосіб віддати перевагу рішенням для управління базами даних із розширеними функціями.

Архітектура бази даних має вигляд(рисунок 3.6):

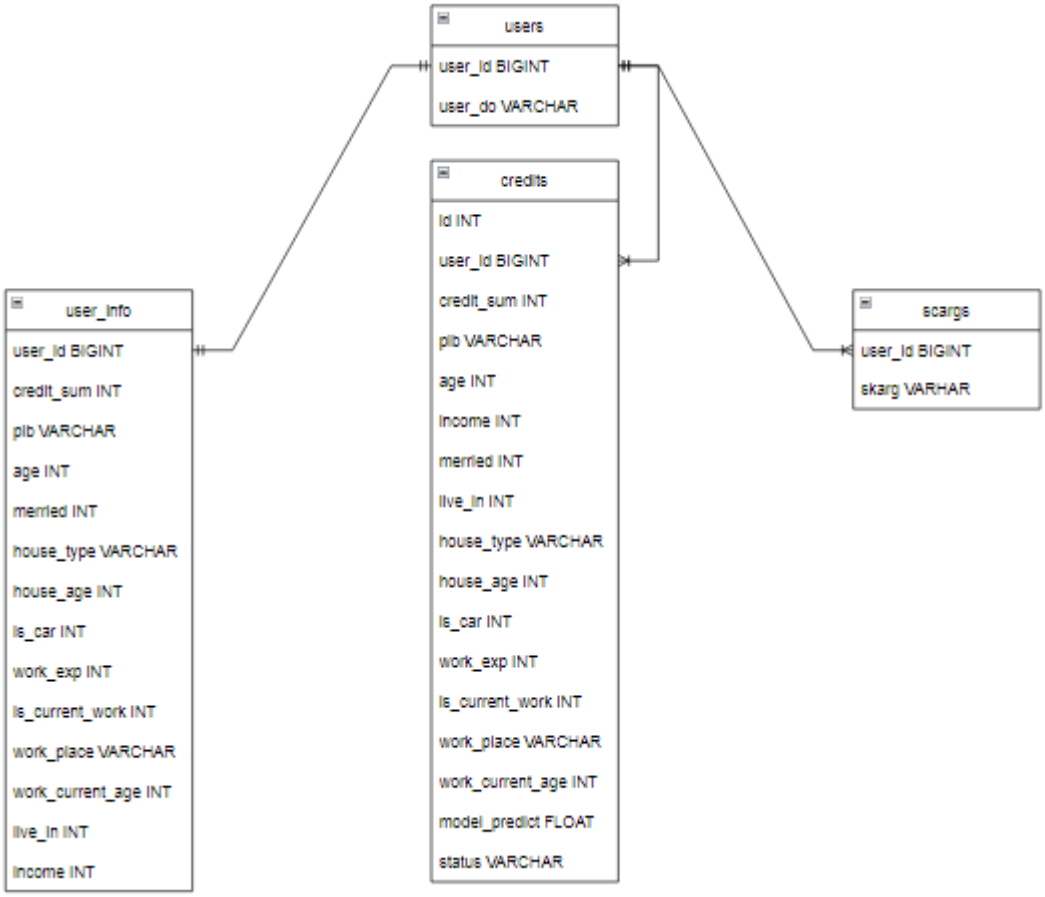

Рисунок 3.6 Архітектура бази даних

Опис таблиць створеної бази даних:

«users» - таблиця бази даних, що слугує для визначення на якому етапі заповнення заявки знаходиться позичальник.

«user_info» - таблиця бази даних, що слугує для збереження тимчасової інформації про кредитну заявку користувача.

«credits» - таблиця бази даних, що містить інформацію про усі заявки на кредит.

«scargs» - таблиця бази даних, що містить скарги та побажання клієнтів.

3.5. Програмна реалізація системи

Система була реалізована у вигляді чат ботів месенджеру Telegram. Telegram — популярний міжплатформний додаток для обміну повідомленнями, який широко використовується. Боти – це сторонні програми, які працюють всередині Telegram. Користувачі можуть взаємодіяти з ботами, надсилаючи їм повідомлення, команди та вбудовані запити.

Керування ботами здійснюється за допомогою запитів до їх API (Application Programming Interface). API – це набір визначених правил, які пояснюють, як комп’ютери або програми взаємодіють один з одним. API знаходяться між програмою та веб-сервером, діючи як проміжний рівень, який обробляє передачу даних між системами. Для створення ботів було використано прикладниий програмний інтерфейс додатку Telegram – Telegram API.

Мовою програмування для написання функціональної частини та логіки боту було обрано мову програмування Python через наявну у ній бібліотеку telebot. Бібліотека telebot забезпечує чистий асинхронний інтерфейс Python для API бота Telegram, ця бібліотека містить ряд високорівневих класів, щоб зробити розробку ботів легкою та зрозумілою. Для реалізації СППР при кредитуванні було створено два чат боти: клієнтський і менеджерський.

Головною функцією клієнтського боту є оформлення клієнтом заявки на кредит. Після відправлення клієнтом заявки, вона подається на вхідні дані до

розробленої моделі оцінки кредитоспроможності позичальника. Уся інформація про позичальника і результат роботи моделі зберігаються у базі даних.

Головною функцією менеджерського боту є прийняття менеджером банку остаточного рішення щодо надання кредиту або відмову в кредитуванні на основі даних про клієнта та результату роботи моделі оцінки кредитоспроможності. Після прийняття остаточного рішенння менеджером банку позичальнику надходить повідомлення про одобрення чи відхилення його заявки.

3.6. Інтерфейс кінцевого користувача

Інтерфейс системи має вигляд Telegram ботів з притаманними їм елементами керування. Користувачі можуть взаємодіяти з ботами, надсилаючи їм повідомлення, команди та вбудовані запити.

3.6.1. Інтерфейс клієнтського боту

При першому відкритті бота клієнт бачить банер, що містить основну інформацію про бота. Після натиску на кнопку старт або введення команди /start у діалогове вікно користувач переходить у меню бота. Меню бота містить чотири кнопки під діалоговим вікном: «Нова заявка», «Мої кредити», «Інформація», «Скарги та пропозиції» (рисунок 3.7).

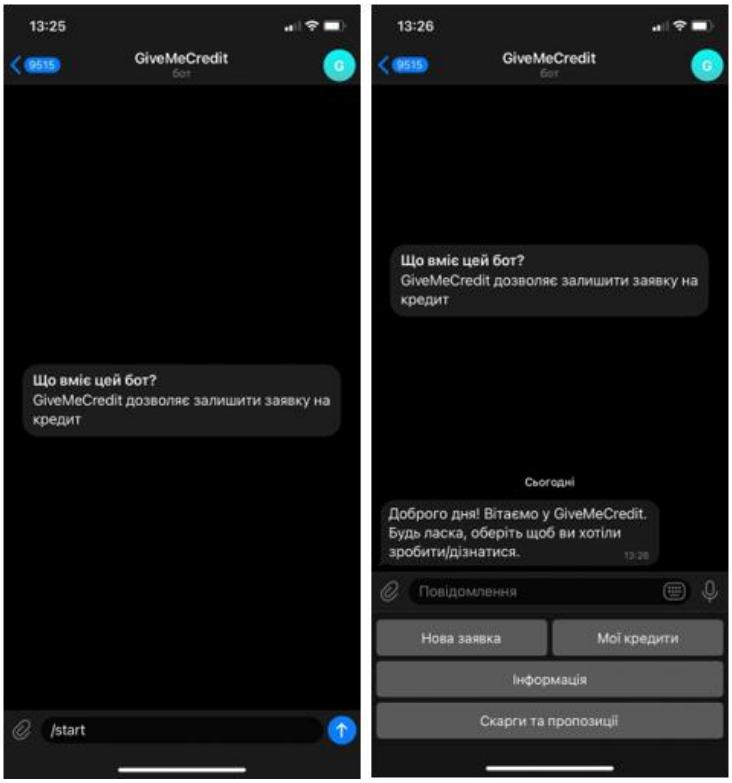

Рисунок 3.7 Інтерфейс меню клієнтського боту

Кнопка «Нова заявка» запускає процес оформлення заявки на кредит, а саме введення потрібної інформації клієнтом щодо кредиту та його інформації. Заповнення заявки проходить у режимі переписки між клієнтом і ботом. Після того як клієнт відповів на всі питання система дає можливість перевірити вказану інформацію і вибрати одну з двох кнопок: «Все вірно» або «Заповнити заново». При виборі першої, заявка приймається, користувач отримує повідомлення про успішне прийняття заявки на обробку і автоматично переходить до головного меню. Після підтвердження користувачем введених даних, інформація подається на вхідні дані моделі оцінки кредитоспроможності, що оцінює кредитоспроможність даного позичальника на основі інформації, що він надав. Після оцінки моделлю кредитоспроможності дані про клієнта і його оцінка зберігаються у базі даних. При виборі кнопки «Заповнити заново» запускається процес оформлення заявки заново (рисунок 3.8).

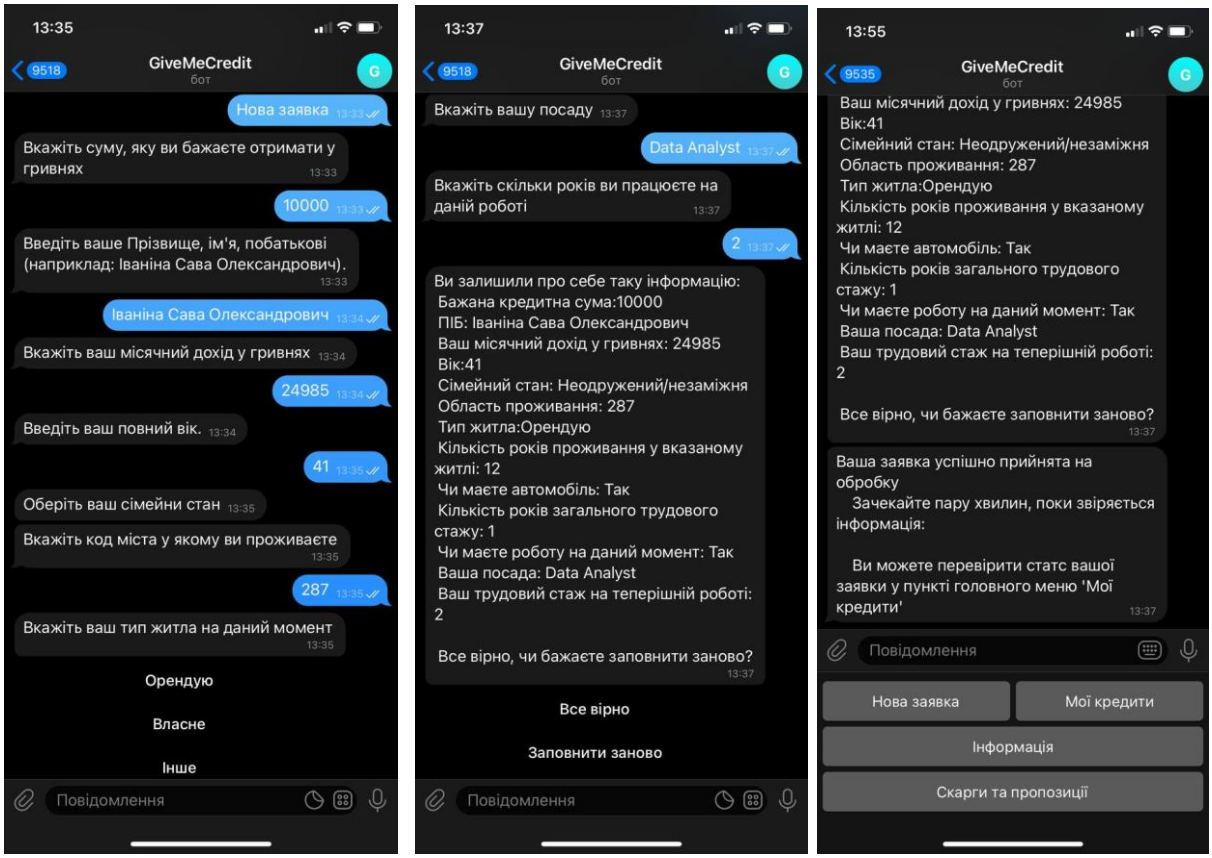

Рисунок 3.8 Інтерфейс оформлення заявки у клієнтському боті

Кнопка «Інформація» виводить інформацію про компанію у вигляді повідомлення.

Кнопка «Скарги та пропозиції» дозволяє користувачу залишити повідомлення для менеджера банку. Після введення необхідного повідомлення, воно зберігається у базі даних разом з індивідуальним номером Telegram аккаунту клієнта (рисунок 3.9).

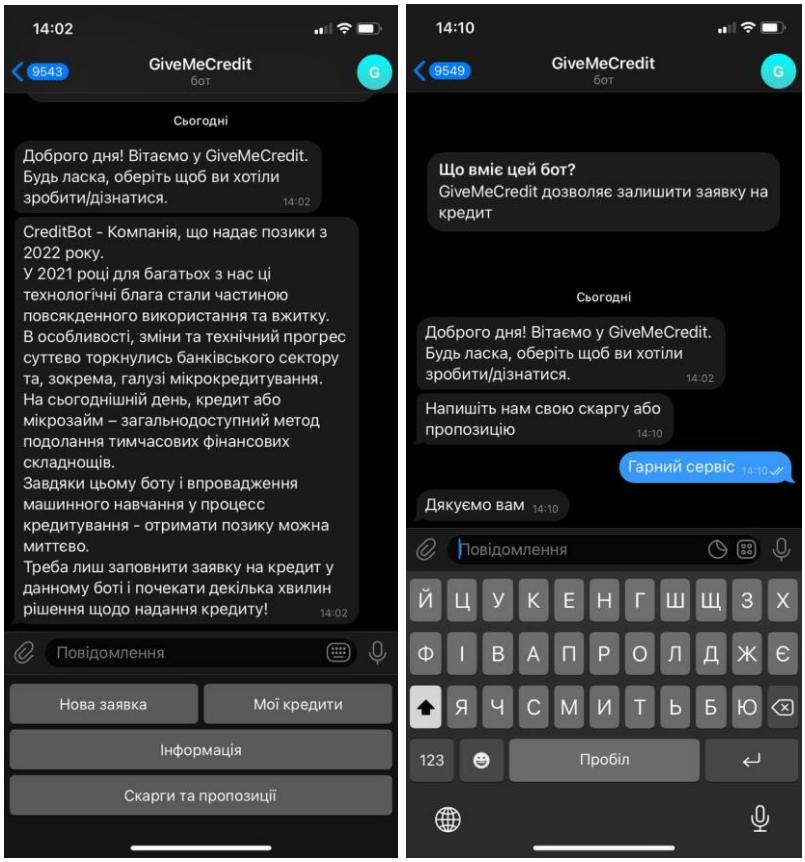

Рисунок 3.9 Інтерфейс «Скарги та пропозиції» клієнтського бота

Кнопка «Мої кредити» дозволяє користувачу перевірити всі наявні його кредитні заявки та їх статус (рисунок 3.10).

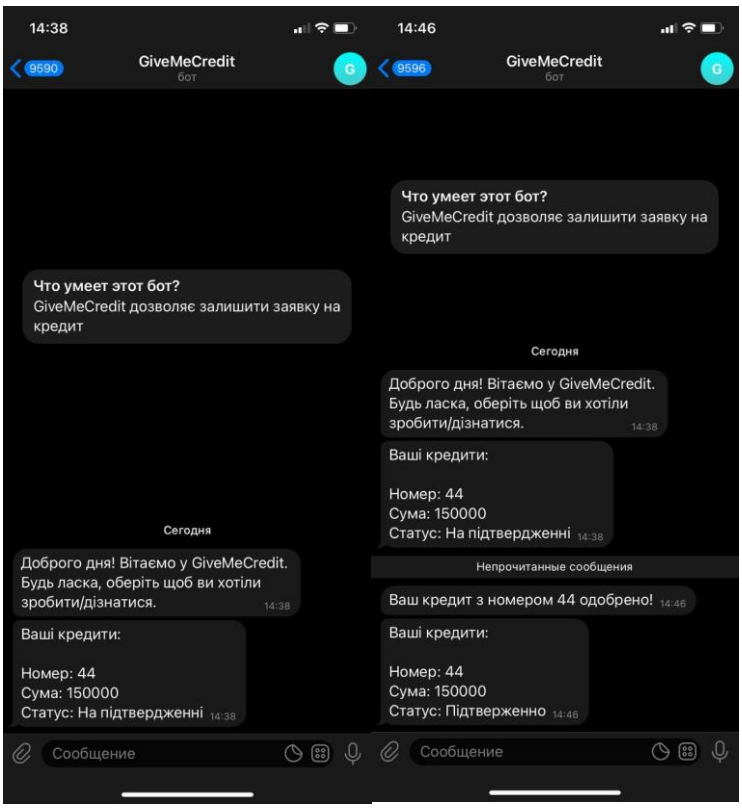

Рисунок 3.10 Інтерфейс «Мої кредити» клієнтського бота

3.6.2. Інтерфейс менеджерського боту

При першому відкритті бота клієнт бачить банер, що містить основну інформацію про бота. Після натиску на кнопку старт або введення команди /start у діалогове вікно користувач переходить у меню бота. Меню бота містить чотири кнопки під діалоговим вікном: «Нова заявка», «Мої кредити», «Інформація», «Скарги та пропозиції» (рисунок 3.11).

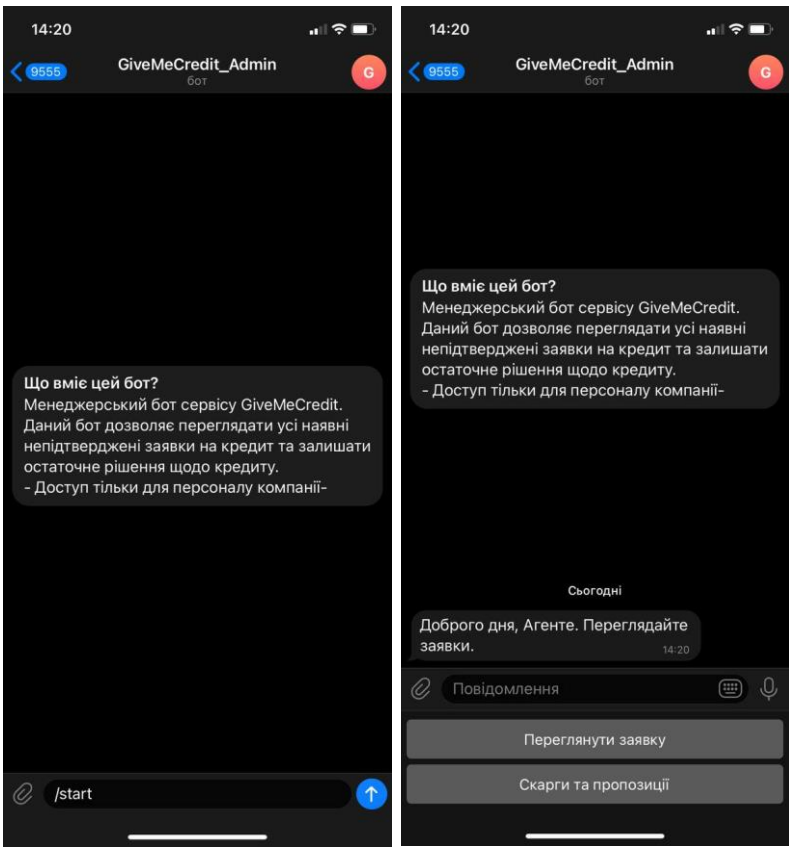

Рисунок 3.11 Інтерфейс меню менеджерського боту

Кнопка «Переглянути заявку» почергово виводить усі наявні заявки, що потребують підтвердження. Система дає можливість перевірити вказану позичальником інформацію і вибрати одну з двох кнопок: «Схвалити» або «Відмова» (рисунок 3.12). Після вибору однієї із кнопок позичальник отримає повідомлення про схвалення або відмови його заявки на кредит. Інформація про усі заявки та рішення, що прийняв менеджер банку зберігається у базі даних.

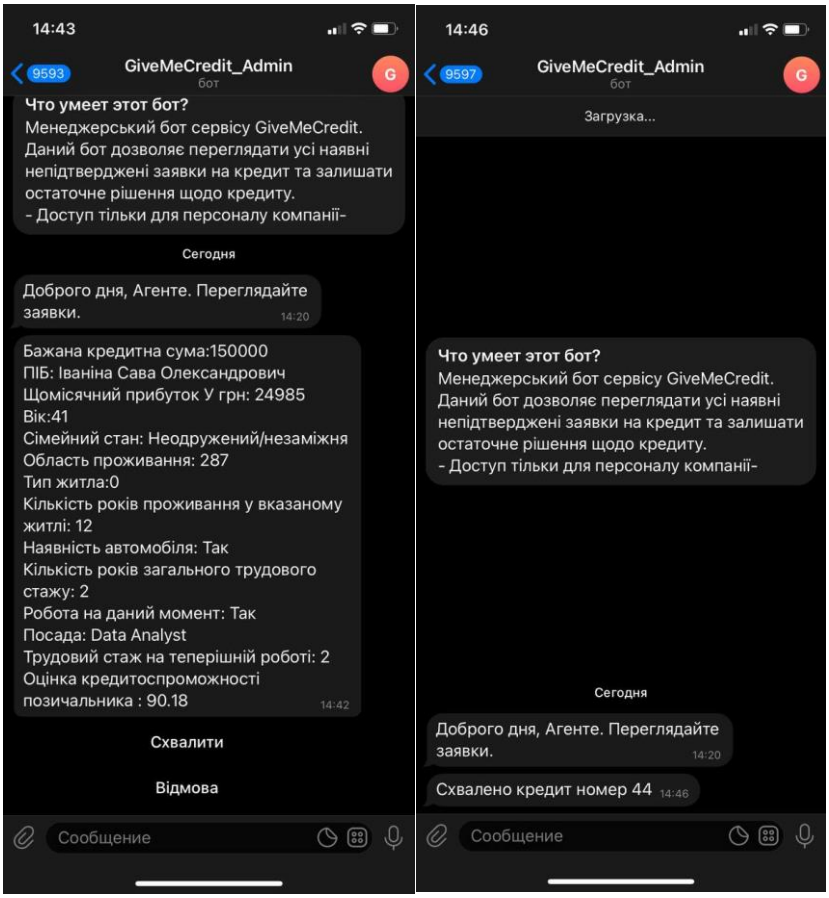

Рисунок 3.12 Інтерфейс «Переглянути заявку» менеджерського боту

Кнопка «Скарги та пропозиції» показує усі повідомлення, що залишили користувачі клієнтського боту. Кожне повідомлення має інформацію про клієнта, а саме його індивідуальний номер акаунту Telegram (рисунок 3.13).

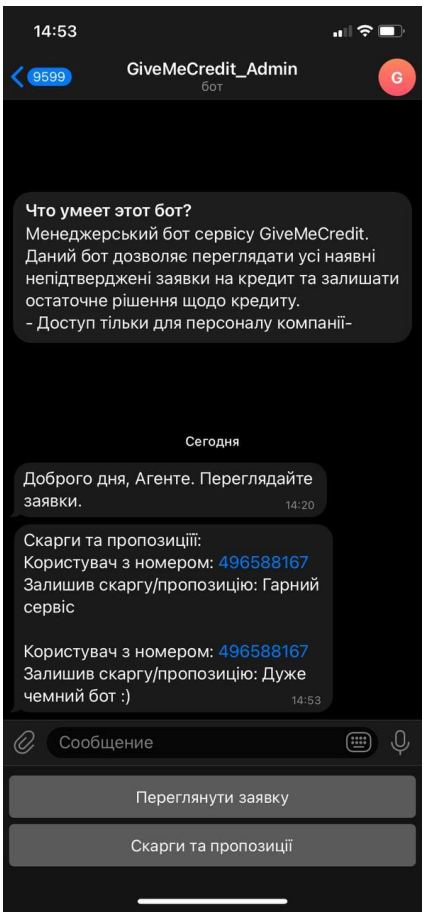

Рисунок 3.13 Інтерфейс «Скарги та пропозиції» менеджерського боту

3.7. Висновки до розділу

У третьому розділі роботи був описаний процес розробки системи підтримки прийняття рішень при кредитуванні. Був проведений етап збору, аналізу та попередньої обробки даних на якому були описані та оброблені вхідні дані перед процесом навчання моделі.

Також був проведений процес навчання моделей на методах машинного навчання для вирішення задачі класифікації, які були описані у другому розділі. При аналізі результатів роботи моделей навчених за різними методами машинного навчання було виявлено, що метод випадкового лісу виявився найбільш ефективним на обраних даних серед інших методів у даній задачі. Через найбільшу точність серед інших методів по всім метрикам було прийняте рішення використовувати модель побудовану на алгоритмі випадковий ліс для

оцінки кредитоспроможності позичальників у системі підтримки прийняття рішень при кредитуванні.

Для роботи СППР з даними була створена база даних та описана її архітектура. Базою даних для побудови СППР при кредитуванні було обрано реляційну базу даних MySQL.

Також у третьому розділі була описана програмна реалізація розроблювальної СППР. Система була реалізована у вигляді чат ботів популярного міжплатформного додатоку для обміну повідомленнями Telegram з функціями написаними на мові програмування Python. Для реалізації СППР при кредитуванні було створено два чат боти: клієнтський і менеджерський, був розроблений і описаний інтерфейс кожного з них. Також було проведено тестування розробленої системи підтримки прийняття рішень при кредитуванні.

ВИСНОВКИ

У першому розділі даної роботи було розглянуто актуальність дослідження, а саме стан кредитного портфелю українських банківських установ. Було виявлено, що майже за кожним другим кредитом, наданим позичальнику в Україні, не відбувається вчасної сплати платежу або погашення боргу, що обумовлює актуальність розробки СППР при кредитуванні. Також у першому розділі було наведено означення кредитоспроможності і виявлено, що оцінка кредитоспроможності позичальника є одним з ключових факторів для зниження ризиків при кредитуванні. Були розглянуті основні існуючі методи для оцінки кредитоспроможності і виявлені їх переваги та недоліки. Було доведено, що оцінка кредитоспроможності за допомогою скорингових технологій є потужним інструментарієм кредитного ризик-менеджементу, що дозволить значно знизити ризики банківської або фінансової установи при наданні кредитів.

У другому розділі даної роботи були розглянуті основні методи реалізації моделі кредитного скорингу. Для розробки моделі були обрані основні алгоритми машинного навчання для задачі класифікації. Було описано суть кожного з алгоритмів машинного навчання та наведено основні переваги та недоліки кожного з них. Також були розглянуті основні метрики для оцінювання точності моделі, побудованої на обраних методах. Було обґрунтовано та прийняте рішення використовувати мову програмування Python як основний програмний засіб для реалізації моделі кредитного скорингу.

У третьому розділі роботи був описаний процес розробки системи підтримки прийняття рішень при кредитуванні. Був проведений етап збору, аналізу та попередньої обробки даних на якому були описані та оброблені вхідні дані перед процесом навчання моделі. Також був проведений процес навчання моделей на методах машинного навчання для вирішення задачі класифікації, які були описані у другому розділі. При аналізі результатів роботи моделей навчених за різними методами машинного навчання було виявлено, що метод

випадкового лісу виявився найбільш ефективним на обраних даних серед інших методів у даній задачі. Через найбільшу точність серед інших методів по всім метрикам було прийняте рішення використовувати модель побудовану на алгоритмі випадковий ліс для оцінки кредитоспроможності позичальників у системі підтримки прийняття рішень при кредитуванні. Для роботи СППР з даними була створена база даних та описана її архітектура. Базою даних для побудови СППР при кредитуванні було обрано реляційну базу даних MySQL. Також у третьому розділі була описана програмна реалізація розроблювальної СППР. Система була реалізована у вигляді чат ботів популярного міжплатформного додатоку для обміну повідомленнями Telegram з функціями написаними на мові програмування Python. Для реалізації СППР при кредитуванні було створено два чат боти: клієнтський і менеджерський, був розроблений і описаний інтерфейс кожного з них. Також було проведено тестування розробленої системи підтримки прийняття рішень при кредитуванні.

У ході роботи була сформована та досягнута мета, а саме підвищення ефективності процесу оцінювання кредитоспроможності позичальників шляхом розробки системи підтримки прийняття рішень при кредитуванні. Слід зазначити, що в Україні дуже важливе вживання заходів щодо розвитку ефективних методик оцінювання кредитоспроможності фізичних осіб. Активізація роботи таких методик призведе до зростання надійності банківської системи України. Розроблена система підтримки прийняття рішень при кредитуванні допоможе знизити ризики банку при кредитуванні шляхом допомоги менеджеру банку або кредитному спеціалісту у прийнятті рішення щодо кредитування. Подальше дослідження, покращення та автоматизація систем підтримки прийняття рішень при кредитуванні в перспективі може виключити менеджера або кредитного спеціаліста с процесу прийняття рішень щодо надання кредиту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. «МЕТОДИ ОЦІНКИ КРЕДИТОСПРОМОЖНОСТІ ПОЗИЧАЛЬНИКА – ФІЗИЧНОЇ ОСОБИ (Кучеренко С.Ю.,Кучеренко О. М.)»: https://economic-bulletin.com/index.php/journal/article/view/697/716

2. «What Is Creditworthiness? by Jim Akin»; Experian: https://www.experian.com/blogs/ask-experian/what-is-creditworthiness/

3. «Про організацію формування та обігу кредитних історій»; Верховна рада України: https://zakon.rada.gov.ua/laws/show/2704-15#Text

4. «Машинне навчання - методи, типи, завдання та приклади»; New day crypto: https://newdaycrypto.com/uk/machine-learning-methods-types-tasks-and-examples/

5. «Огляд банкывського сектору»; Національний банк України: https://bank.gov.ua/admin_uploads/article/Banking_Sector_Review_2021-08.pdf?v=4

6. «pyTelegramBotAPI»; Python Package Index: https://pypi.org/project/pyTelegramBotAPI/

7. «Telegram Bot API»; Telegram: https://core.telegram.org/bots/api

8. «scikit-learn Machine Learning in Python»; Scikit Learn: https://scikit-learn.org/stable/

9. «Top Programming Languages 2021»; IEEE Spectrum: https://spectrum.ieee.org/top-programming-languages/

10. «Кредитоспроможність»; Wikipedia: https://uk.wikipedia.org/wiki/Кредитоспроможність

11. «Кредитний скоринг»; Wikipedia: https://uk.wikipedia.org/wiki/Кредитний_скоринг

12. «Кредитна історія»; Wikipedia: https://uk.wikipedia.org/wiki/Кредитна_історія

13. «CREDIT SCORING APPROACHES GUIDELINES»; The World Bank Group: https://thedocs.worldbank.org/en/doc/935891585869698451-0130022020/original/CREDITSCORINGAPPROACHESGUIDELINESFINALWEB.pdf

14. «MySQL»; Wikipedia: https://uk.wikipedia.org/wiki/MySQL

15. «Metrics to Evaluate your Classification Model to take the right decisions»; Analytics Vidhya: https://www.analyticsvidhya.com/blog/2021/07/metrics

16. «Data pre-processing»; Wikipedia: https://en.wikipedia.org/wiki/Data_pre-processing

17. «10 Techniques to deal with Imbalanced Classes in Machine Learning»; Analytics Vidhya: https://www.analyticsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-imbalance-in-machine-learning/

18. «Задача класифікації»; Wikipedia: https://uk.wikipedia.org/wiki/Задача_класифікації

19. «Introduction to Classification Algorithms»; edureka!: https://www.edureka.co/blog/classification-algorithms/

20. «Advantages and Disadvantages of different Classification Models»; GeeksforGeeks: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-different-classification-models/

ДОДАТКИ

Програмний код

Реалізація моделі оцінки кредитоспроможності:

```
import gc
import pickle

import pandas as pd
import numpy as np


from sklearn.metrics import *
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis, QuadraticDiscriminantAnalysis
from sklearn.cluster import KMeans
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import precision_recall_fscore_support
from sklearn.metrics import mean_squared_error as mse
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_validate, cross_val_score, KFold, StratifiedKFold
from sklearn.model_selection import train_test_split
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.feature_selection import SelectFromModel
from sklearn.svm import LinearSVC
```

```
from sklearn.ensemble import VotingClassifier
from sklearn.feature_selection import RFECV

import warnings
warnings.filterwarnings('ignore')

import seaborn as sns
import matplotlib.pyplot as plt



%config InlineBackend.figure_format = 'retina'
sns.set(color_codes=True, rc={'figure.figsize':(15, 10)})
sns.set_palette(sns.color_palette('twilight_shifted'))

data = pd.read_csv("Training Data.csv")
data = data.drop(['Id'], axis=1)
data.rename(columns = {'Id' : 'ID', 'age' : 'AGE', 'income' : 'INCOME', 'experience' : 'EXP', 'married' : 'MARRIED', 'house_ownership' : 'HOUSE_OWN', 'car_ownership': 'CAR_OWN', 'profession':'PROFESSION', 'city': 'CITY', 'state':'STATE','current_job_years': 'JOB_YEAR','current_house_years': 'HOUSE_YEAR', 'risk_flag': 'RISK'}, inplace = True)
data.head()

def map_feature(df, col):
  feature = list(df[col].value_counts().index)
  mapped_feature = {value: i for i, value in zip(range(len(feature)), feature)}
  return df[col].map(mapped_feature)

cat_features = [
            t['feature'] for t in data.dtypes.reset_index().rename(
```

```
                columns={"index": "feature", 0: "type"}
).astype({"feature": str, "type": str}).to_dict('records') if t['type'] == "object"]

for col in cat_features:
    data[col] = map_feature(data, col)

data

fig, axes = plt.subplots(3, 2, figsize=(30, 10))
for ax, col in zip(axes.reshape(-1), cat_features):
    sns.histplot(data[col], ax=ax)
sns.histplot(data['INCOME'], kde=True)

X = data.drop(['RISK'], axis = 1)
y = data['RISK']

from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(random_state=42)
x_ros, y_ros = ros.fit_resample(X, y)

print('Original dataset shape', y.value_counts())
print('Resample dataset shape', y_ros.value_counts())

sns.countplot(y_ros)

scoring = ['accuracy', 'precision', 'recall_macro' , 'f1_weighted', 'roc_auc']

models = {
    "Bayes": GaussianNB(),
    "Random Forest": RandomForestClassifier(),
```

```
    "Decision Tree": DecisionTreeClassifier(),
    "Logistic Regression": LogisticRegression(),
    "Linear Discriminant Analysis": LinearDiscriminantAnalysis(),
    "QuadraticDiscriminantAnalysis": QuadraticDiscriminantAnalysis(),
    "K-Nearest Neighbors":  KNeighborsClassifier()
}

folds_result = []
splits = 2
cv = StratifiedKFold(n_splits=splits, random_state=492939191, shuffle=True)
for (train_index, test_index), fold in zip(cv.split(x_ros, y_ros), range(splits)):
  print("TRAIN:", train_index, "TEST:", test_index)
  X_train, X_test = x_ros.iloc[train_index], x_ros.iloc[test_index]
  y_train, y_test = y_ros.iloc[train_index], y_ros.iloc[test_index]

  fold_result = []

  for k, v in models.items():
    _model_pred = v.fit(X_train, y_train.values).predict(X_test)
    fold_result.append(
        {
            "fold": fold+1,
            "model": k,
            "accuracy": accuracy_score(y_test, _model_pred),
            "precision": precision_score(y_test, _model_pred),
            "recall": recall_score(y_test, _model_pred),
            "f1": f1_score(y_test, _model_pred),
            "roc_auc": roc_auc_score(y_test, _model_pred),
        }
    )
```

```
    gc.collect()

  folds_result.append({f"fold_{fold+1}": fold_result})
  gc.collect()

results = pd.concat([pd.DataFrame(res[f'fold_{i+1}']) for res, i in zip(folds_result, range(len(folds_result)))], axis=0)
results

results_by_model = results.groupby(['model']).mean().drop(['fold'], axis=1)
results_by_model.sort_values(by='f1', ascending=False)

best_model = results_by_model[results_by_model['f1'] == results_by_model['f1'].max()]
best_model

best_fit = models[list(best_model.index)[-1]].fit(x_ros, y_ros)
fe_dict = pd.DataFrame([{"feature": col, "importance": v} for col,v in zip(list(X.columns), best_fit.feature_importances_)])

fig, ax = plt.subplots(1, 1)
ax.bar(fe_dict['feature'], fe_dict['importance'])
ax.tick_params(axis='x', rotation=60)

filename = 'finalized_model_NEEEEEEEEEEW.sav'
pickle.dump(best_fit, open(filename, 'wb'))

loaded_model = pickle.load(open(filename, 'rb'))
loaded_model_result = loaded_model.predict_proba(test.to_numpy().reshape(1,-1))
loaded_model_result, real
```

Програмний код клієнтського боту:

```
print('Started')
from sqlalchemy import create_engine
import telebot
from telebot import types
import re
import pandas as pd
import pickle



token = '5368576777:AAEEYq_O-5R_BcWDbB7gGj3VKF-saSCIsdg'

bot = telebot.TeleBot(token)

engine =
create_engine('mysql://sava:C8e6W8g5@178.63.77.140:3306/credit?charset=utf8mb
4', pool_recycle=3600)
conn = engine.raw_connection()
cursor = conn.cursor()

with open('finalized_model_NEEEEEEEEEEW.sav', 'rb') as file:
    model = pickle.load(file)

hello_text = """Доброго дня! Вітаємо у GiveMeCredit.
Будь ласка, оберіть щоб ви хотіли зробити/дізнатися."""

info_text = """CreditBot - Компанія, що надає позики з 2022 року.
```

У 2021 році для багатьох з нас ці технологічні блага стали частиною повсякденного використання та вжитку.

В особливості, зміни та технічний прогрес суттєво торкнулись банківського сектору та, зокрема, галузі мікрокредитування.

На сьогоднішній день, кредит або мікрозайм – загальнодоступний метод подолання тимчасових фінансових складнощів.

Завдяки цьому боту і впровадження машинного навчання у процесс кредитування - отримати позику можна миттєво.

Треба лиш заповнити заявку на кредит у данному боті і почекати декілька хвилин рішення щодо надання кредиту!

"""

```
stages_text = {
    'credit_sum':'Вкажіть суму, яку ви бажаєте отримати у гривнях',
    'pib':"Введіть ваше Прізвище, ім'я, побатькові (наприклад: Іваніна Сава Олександрович).",
    'income':"Вкажіть ваш місячний дохід у гривнях",
    'age':'Введіть ваш повний вік.',
    'merried':'Оберіть ваш сімейни стан',
    'live_in':'Вкажіть код міста у якому ви проживаєте',
    'house_type':'Вкажіть ваш тип житла на даний момент',
    'house_age':'Вкажіть скільки років ви проживаєте у вказаному житлі',
    'is_car':'Вкажіть чи маєте ви атомобіль',
    'work_exp':'Вкажіть ваш трудовий стаж у роках',
    'is_current_work':'Вкажіть чи маєте ви роботу в даний момент',
    'work_place':'Вкажіть вашу посаду',
    'work_current_age':'Вкажіть скільки років ви працюєте на даній роботі',
    'end':"""Ваша заявка успішно прийнята на обробку
Зачекайте пару хвилин, поки звіряється інформація:
```

```
Ви можете перевірити статс вашої заявки у пункті головного меню 'Мої кредити"""",
   'skarg':"Напишіть нам свою скаргу або пропозицію"
}

@bot.message_handler(commands=["start", "help"])
def login(message):
   row = pd.read_sql("SELECT * FROM users where user_id = "+str(message.from_user.id), con=conn)

   text = hello_text
   markup = types.ReplyKeyboardMarkup(one_time_keyboard = False, resize_keyboard = True)
   markup.row('Нова заявка', 'Мої кредити')
   markup.row('Інформація')
   markup.row('Скарги та пропозиції')

   bot.delete_message(message.from_user.id, message.message_id)
   if row.empty:
      sql = "INSERT INTO users (user_id) VALUES ({0})".format(message.from_user.id)
      cursor.execute(sql)

      sql = "INSERT INTO user_info (user_id) VALUES ({0})".format(message.from_user.id)
      cursor.execute(sql)
      conn.commit()
      bot.send_message(chat_id=message.chat.id, text=text, parse_mode="Markdown", reply_markup=markup)
   else:
```

```
        query = """ UPDATE users
                    SET user_do = '{0}'
                    WHERE user_id = {1} """.format('start', message.from_user.id)
        cursor.execute(query)
        conn.commit()
        bot.send_message(chat_id=message.chat.id, text=text, parse_mode="Markdown", reply_markup=markup)
        conn.commit()


@bot.message_handler(content_types=["text"], regexp = r'^Мої кредити$')
def my_credits(message):
    conn.commit()
    credit_info = pd.read_sql("SELECT id, credit_sum, `status` FROM credits where user_id = "+str(message.from_user.id), con=conn)
    bot.delete_message(message.from_user.id, message.message_id)
    if not credit_info.empty:
        text = "Ваші кредити:\n\n"
        for credit in credit_info.values:
            text = text + f"Номер: {credit[0]}\nСума: {credit[1]}\nСтатус: {credit[2].replace('on_approve', 'На підтвердженні').replace('approved', 'Підтверженно').replace('abandoned', 'Відмовлено')}\n\n"
    else:
        text = "Наразі у вас немає кредитів"

    bot.send_message(chat_id=message.chat.id, text=text)


@bot.message_handler(content_types=["text"], regexp = r'^Інформація$')
def company_info(message):
```

```
    text = info_text
    bot.delete_message(message.from_user.id, message.message_id)
    bot.send_message(chat_id=message.chat.id, text=text)

@bot.message_handler(content_types=["text"], regexp = r'^Скарги та пропозиції$')
def skarg(message):
    stage = 'skarg'
    bot.delete_message(message.from_user.id, message.message_id)
    query = """ UPDATE users
            SET user_do = '{0}'
            WHERE user_id = {1} """.format(stage, message.from_user.id)
    cursor.execute(query)
    conn.commit()
    bot.send_message(chat_id=message.chat.id, text=stages_text[stage])

@bot.message_handler(content_types=["text"], regexp = r'^Нова заявка$')
def new_credit(message):
    stage = 'credit_sum'
    query = """ UPDATE users
            SET user_do = '{0}'
            WHERE user_id = {1} """.format(stage, message.from_user.id)
    cursor.execute(query)
    conn.commit()
    bot.send_message(chat_id=message.chat.id, text=stages_text[stage])

@bot.message_handler(content_types=["text"])
def text(message):
    row = pd.read_sql("SELECT * FROM users where user_id =
"+str(message.from_user.id), con=conn)
    if row['user_do'].values[0]== 'credit_sum':
```

```
        if re.search(r'^\d+\.?\d{0,2}$', message.text):
            query = """ UPDATE user_info
                    SET {0} = {1}
                    WHERE user_id = {2} """.format(row['user_do'].values[0], message.text, message.from_user.id)
            cursor.execute(query)

            stage = 'pib'
            query = """ UPDATE users
                    SET user_do = '{0}'
                    WHERE user_id = {1} """.format(stage, message.from_user.id)
            cursor.execute(query)
            conn.commit()

            bot.send_message(chat_id=message.chat.id, text=stages_text[stage])
        else:
            bot.send_message(chat_id=message.chat.id, text='Не коректні дані, спробуйте ще раз')

    elif row['user_do'].values[0]== 'pib':
        query = """ UPDATE user_info
                SET {0} = '{1}'
                WHERE user_id = {2} """.format(row['user_do'].values[0], message.text, message.from_user.id)
        cursor.execute(query)

        stage = 'income'
        query = """ UPDATE users
                SET user_do = '{0}'
```

```
                WHERE user_id = {1} """.format(stage, message.from_user.id)
        cursor.execute(query)
        conn.commit()

        bot.send_message(chat_id=message.chat.id, text=stages_text[stage])

    elif row['user_do'].values[0]== 'income':
        if re.search(r'^\d+$', message.text):
            query = """ UPDATE user_info
                    SET {0} = {1}
                    WHERE user_id = {2} """.format(row['user_do'].values[0],
message.text, message.from_user.id)
            cursor.execute(query)
            stage = 'age'
            query = """ UPDATE users
                    SET user_do = '{0}'
                    WHERE user_id = {1} """.format(stage, message.from_user.id)
            cursor.execute(query)
            conn.commit()

            bot.send_message(chat_id=message.chat.id, text=stages_text[stage])
        else:
            bot.send_message(chat_id=message.chat.id, text='Не коректні дані,
спробуйте ще раз')


    elif row['user_do'].values[0]== 'live_in':
        if re.search(r'^\d+$', message.text):
            query = """ UPDATE user_info
                    SET {0} = '{1}'
```

```
                WHERE user_id = {2} """.format(row['user_do'].values[0],
message.text, message.from_user.id)
        cursor.execute(query)
        stage = 'house_type'
        query = """ UPDATE users
                SET user_do = '{0}'
                WHERE user_id = {1} """.format(stage, message.from_user.id)
        cursor.execute(query)
        conn.commit()

        markup = types.InlineKeyboardMarkup(row_width=1)
        rent = types.InlineKeyboardButton(text='Орендую', callback_data='rent')
        own = types.InlineKeyboardButton(text='Власне', callback_data='own')
        other = types.InlineKeyboardButton(text='Інше', callback_data='other')
        markup.add(rent, own, other)

        bot.send_message(chat_id=message.chat.id, text=stages_text[stage],
reply_markup=markup)
    else:
        bot.send_message(chat_id=message.chat.id, text='Не коректні дані,
спробуйте ще раз')

  elif row['user_do'].values[0]== 'age':

    if re.search(r'^\d+$', message.text):
        query = """ UPDATE user_info
                SET {0} = {1}
                WHERE user_id = {2} """.format(row['user_do'].values[0],
message.text, message.from_user.id)
        cursor.execute(query)
```

```
            stage = 'merried'
            query = """ UPDATE users
                    SET user_do = '{0}'
                    WHERE user_id = {1} """.format(stage, message.from_user.id)
            cursor.execute(query)
            conn.commit()

            markup = types.InlineKeyboardMarkup(row_width=1)
            merried = types.InlineKeyboardButton(text='Одружений/заміжня', callback_data='merried')
            single = types.InlineKeyboardButton(text='Неодружений/незаміжня', callback_data='single')
            markup.add(merried, single)

            bot.send_message(chat_id=message.chat.id, text=stages_text[stage], reply_markup=markup)
        else:
            bot.send_message(chat_id=message.chat.id, text='Не коректні дані, спробуйте ще раз')

    elif row['user_do'].values[0]== 'house_age':
        if re.search(r'^\d+$', message.text):
            query = """ UPDATE user_info
                    SET {0} = {1}
                    WHERE user_id = {2} """.format(row['user_do'].values[0], message.text, message.from_user.id)
            cursor.execute(query)

            stage = 'is_car'
```

```
            query = """ UPDATE users
                    SET user_do = '{0}'
                    WHERE user_id = {1} """.format(stage, message.from_user.id)
            cursor.execute(query)
            conn.commit()

            markup = types.InlineKeyboardMarkup(row_width=1)
            car_yes = types.InlineKeyboardButton(text='Так', callback_data='car_yes')
            car_no = types.InlineKeyboardButton(text='Ні', callback_data='car_no')
            markup.add(car_yes, car_no)

            bot.send_message(chat_id=message.chat.id, text=stages_text[stage],
reply_markup=markup)
        else:
            bot.send_message(chat_id=message.chat.id, text='Не коректні дані,
спробуйте ще раз')

    elif row['user_do'].values[0]== 'work_exp':
        if re.search(r'^\d+$', message.text):
            query = """ UPDATE user_info
                    SET {0} = {1}
                    WHERE user_id = {2} """.format(row['user_do'].values[0],
message.text, message.from_user.id)
            cursor.execute(query)

            stage = 'is_current_work'
            query = """ UPDATE users
                    SET user_do = '{0}'
                    WHERE user_id = {1} """.format(stage, message.from_user.id)
            cursor.execute(query)
```

```
            conn.commit()

            markup = types.InlineKeyboardMarkup(row_width=1)
            work_yes = types.InlineKeyboardButton(text='Так', callback_data='work_yes')
            work_no = types.InlineKeyboardButton(text='Hi', callback_data='work_no')
            markup.add(work_yes, work_no)

            bot.send_message(chat_id=message.chat.id, text=stages_text[stage], reply_markup=markup)
        else:
            bot.send_message(chat_id=message.chat.id, text='Не коректні дані, спробуйте ще раз')

    elif row['user_do'].values[0]== 'work_place':
        query = """ UPDATE user_info
                    SET {0} = '{1}'
                    WHERE user_id = {2} """.format(row['user_do'].values[0], message.text, message.from_user.id)
        cursor.execute(query)

        stage = 'work_current_age'
        query = """ UPDATE users
                    SET user_do = '{0}'
                    WHERE user_id = {1} """.format(stage, message.from_user.id)
        cursor.execute(query)
        conn.commit()

        bot.send_message(chat_id=message.chat.id, text=stages_text[stage])
```

```
    elif row['user_do'].values[0]== 'work_current_age':
        if re.search(r'^\d+$', message.text):
            query = """ UPDATE user_info
                    SET {0} = {1}
                    WHERE user_id = {2} """.format(row['user_do'].values[0], message.text, message.from_user.id)
            cursor.execute(query)

            stage = 'end'
            query = """ UPDATE users
                    SET user_do = '{0}'
                    WHERE user_id = {1} """.format(stage, message.from_user.id)
            cursor.execute(query)
            conn.commit()

            user_info = pd.read_sql("SELECT * FROM user_info where user_id = "+str(message.from_user.id), con=conn)

            text = f"""Ви залишили про себе таку інформацію:
 Бажана кредитна сума:{user_info['credit_sum'].values[0]}
 ПІБ: {user_info['pib'].values[0]}
 Ваш місячний дохід у гривнях: {user_info['income'].values[0]}
 Вік:{user_info['age'].values[0]}
 Сімейний стан: {str(user_info['merried'].values[0]).replace('1','Одружений/заміжня').replace('0', 'Неодружений/незаміжня')}
 Область проживання: {user_info['live_in'].values[0]}
 Тип житла:{user_info['house_type'].values[0].replace('rent','Орендую').replace('own','Власне').replace('other','Інше')}
```

```
Кількість років проживання у вказаному житлі: {user_info['house_age'].values[0]}
Чи маєте автомобіль: {str(user_info['is_car'].values[0]).replace('1', 'Так').replace('1', 'Ні')}
Кількість років загального трудового стажу: {user_info['work_exp'].values[0]}
Чи маєте роботу на даний момент: Так
Ваша посада: {user_info['work_place'].values[0]}
Ваш трудовий стаж на теперішній роботі: {user_info['work_current_age'].values[0]}

Все вірно, чи бажаєте заповнити заново?"""

            markup = types.InlineKeyboardMarkup(row_width=1)
            all_correct = types.InlineKeyboardButton(text='Все вірно', callback_data='all_correct')
            again = types.InlineKeyboardButton(text='Заповнити заново', callback_data='again')
            markup.add(all_correct, again)


            bot.send_message(chat_id=message.chat.id, text=text, reply_markup=markup)

        else:
            bot.send_message(chat_id=message.chat.id, text='Не коректні дані, спробуйте ще раз')

    elif row['user_do'].values[0]== 'skarg':
        sql = "INSERT INTO skargs (user_id, skarg) VALUES ({0}, '{1}')".format(message.from_user.id, message.text)
        cursor.execute(sql)
```

```
        stage = 'start'
        query = """ UPDATE users
                SET user_do = '{0}'
                WHERE user_id = {1} """.format(stage, message.from_user.id)
        cursor.execute(query)
        conn.commit()

        bot.send_message(chat_id=message.chat.id, text='Дякуємо вам')




@bot.callback_query_handler(func=lambda call: True)
def call_back(call):
    row = pd.read_sql("SELECT * FROM users where user_id = "+str(call.from_user.id), con=conn)
    data = call.data
    bot.edit_message_reply_markup(call.from_user.id, message_id = call.message.message_id, reply_markup = '')
    if data in ['merried', 'single']:
        query = """ UPDATE user_info
                SET {0} = {1}
                WHERE user_id = {2} """.format(row['user_do'].values[0], int(data.replace('merried', '1').replace('single', '0')), call.from_user.id)
        cursor.execute(query)
        stage = 'live_in'
        query = """ UPDATE users
                SET user_do = '{0}'
                WHERE user_id = {1} """.format(stage, call.from_user.id)
        cursor.execute(query)
```

```
        conn.commit()

        bot.send_message(chat_id=call.from_user.id, text=stages_text[stage])

    elif data in ['rent', 'own', 'other']:
        query = """ UPDATE user_info
                SET {0} = '{1}'
                WHERE user_id = {2} """.format(row['user_do'].values[0], data,
call.from_user.id)
        cursor.execute(query)

        stage = 'house_age'
        query = """ UPDATE users
                SET user_do = '{0}'
                WHERE user_id = {1} """.format(stage, call.from_user.id)
        cursor.execute(query)
        conn.commit()

        bot.send_message(chat_id=call.from_user.id, text=stages_text[stage])

    elif data in ['car_yes', 'car_no']:
        query = """ UPDATE user_info
                SET {0} = {1}
                WHERE user_id = {2} """.format(row['user_do'].values[0],
data.replace('car_yes','1').replace('car_no','0'), call.from_user.id)
        cursor.execute(query)

        stage = 'work_exp'
        query = """ UPDATE users
                SET user_do = '{0}'
```

```
                WHERE user_id = {1} """.format(stage, call.from_user.id)
        cursor.execute(query)
        conn.commit()

        bot.send_message(chat_id=call.from_user.id, text=stages_text[stage])

    elif data in ['work_yes', 'work_no']:
        query = """ UPDATE user_info
                SET {0} = {1}
                WHERE user_id = {2} """.format(row['user_do'].values[0],
data.replace('work_yes','1').replace('work_no','0'), call.from_user.id)
        cursor.execute(query)

        if data == 'work_no':

            query = """ UPDATE user_info
                    SET work_place = '', work_current_age = 0
                    WHERE user_id = {0} """.format(call.from_user.id)
            cursor.execute(query)
            stage = 'end'
            query = """ UPDATE users
                    SET user_do = '{0}'
                    WHERE user_id = {1} """.format(stage, call.from_user.id)
            cursor.execute(query)
            conn.commit()

            user_info = pd.read_sql("SELECT * FROM user_info where user_id =
"+str(call.message.chat.id), con=conn)

            text = f"""Ви залишили про себе таку інформацію:
```

```
 Бажана кредитна сума:{user_info['credit_sum'].values[0]}
 ПІБ: {user_info['pib'].values[0]}
 Ваш місячний дохід у гривнях: {user_info['income'].values[0]}
 Вік:{user_info['age'].values[0]}
 Сімейний стан: {str(user_info['merried'].values[0]).replace('1','Одружений/заміжня').replace('0', 'Неодружений/незаміжня')}
 Область проживання: {user_info['live_in'].values[0]}
 Тип житла:{user_info['house_type'].values[0].replace('rent','Орендую').replace('own','Власне').replace('other','Інше')}
 Кількість років проживання у вказаному житлі: {user_info['house_age'].values[0]}
 Чи маєте автомобіль: {str(user_info['is_car'].values[0]).replace('1', 'Так').replace('1', 'Ні')}
 Кількість років загального трудового стажу: {user_info['work_exp'].values[0]}
 Чи маєте роботу на даний момент: Ні

 Все вірно, чи бажаєте заповнити заново?"""

        markup = types.InlineKeyboardMarkup(row_width=1)
        all_correct = types.InlineKeyboardButton(text='Все вірно', callback_data='all_correct')
        again = types.InlineKeyboardButton(text='Заповнити заново', callback_data='again')
        markup.add(all_correct, again)


        bot.send_message(chat_id=call.message.chat.id, text=text, reply_markup=markup)
```

```
        elif data == 'work_yes':
            stage = 'work_place'
            query = """ UPDATE users
                        SET user_do = '{0}'
                        WHERE user_id = {1} """.format(stage, call.from_user.id)
            cursor.execute(query)
            conn.commit()
            bot.send_message(chat_id=call.message.chat.id, text=stages_text[stage])

    elif data == 'again':
        stage = 'credit_sum'
        query = """ UPDATE users
                    SET user_do = '{0}'
                    WHERE user_id = {1} """.format(stage, call.message.chat.id)
        cursor.execute(query)
        conn.commit()
        bot.send_message(chat_id=call.message.chat.id, text=stages_text[stage])

    elif data == 'all_correct':

        if row['user_do'].values[0] == 'end':
            user_info = pd.read_sql("SELECT * FROM user_info where user_id = "+str(call.message.chat.id), con=conn)
            raw_house_type = user_info['house_type'].values[0]
            house_type = int(raw_house_type.replace('own', '1').replace('rent', '0').replace('other', '2'))
            user_info['house_type'] = [house_type]

            df = user_info[['income', 'age', 'work_exp', 'merried', 'house_type', 'is_car', 'live_in', 'work_current_age', 'house_age']]
```

```
            for i in df:
                df[i]=float(df[i])
            user_info['income'] = user_info['income']

            model_predict = round(model.predict_proba(df.to_numpy())[0][1]*100,2)

            sql = "INSERT INTO credits (user_id, credit_sum, pib, age, merried,
house_type, house_age, is_car, work_exp, is_current_work, work_place,
work_current_age, model_predict, `status`, live_in, income) VALUES ({0}, {1}, '{2}',
{3}, {4}, '{5}', {6}, {7}, {8}, {9}, '{10}', {11}, {12}, '{13}', '{14}',
{15})".format(call.message.chat.id, user_info['credit_sum'].values[0],
user_info['pib'].values[0], user_info['age'].values[0], user_info['merried'].values[0],
user_info['house_type'].values[0], user_info['house_age'].values[0],
user_info['is_car'].values[0], user_info['work_exp'].values[0],
user_info['is_current_work'].values[0], user_info['work_place'].values[0],
user_info['work_current_age'].values[0], model_predict, 'on_approve',
user_info['live_in'].values[0], user_info['income'].values[0])
            cursor.execute(sql)
            conn.commit()

            text = """Ваша заявка успішно прийнята на обробку
        Зачекайте пару хвилин, поки звіряється інформація:

        Ви можете перевірити статс вашої заявки у пункті головного меню 'Мої
кредити"""
            bot.send_message(chat_id=call.message.chat.id, text=text)
        else:
            text = 'Щось пішло не так, спробуйте знову'
            bot.send_message(chat_id=call.message.chat.id, text=text)
```

```
bot.polling(none_stop=True)
```

Програмний код менеджерського боту:

```
print('Started')
from sqlalchemy import create_engine
import telebot
from telebot import types
import re
import pandas as pd



token = '5439251300:AAHAHNSolwK3pHKJ5luJFuveVTLgLyJN_jE'
client_token = '5368576777:AAEEYq_O-5R_BcWDbB7gGj3VKF-saSCIsdg'
bot = telebot.TeleBot(token)
client_bot = telebot.TeleBot(client_token)

engine = create_engine('mysql://sava:C8e6W8g5@178.63.77.140:3306/credit?charset=utf8mb4', pool_recycle=3600)
conn = engine.raw_connection()
cursor = conn.cursor()

@bot.message_handler(commands=["start"])
def login(message):
    if message.from_user.id == 496588167 or message.from_user.id == 353662267:
        markup = types.ReplyKeyboardMarkup(one_time_keyboard = False, resize_keyboard = True)
```

```
        markup.row('Переглянути заявку')
        markup.row('Скарги та пропозиції')
        bot.delete_message(message.from_user.id, message.message_id)
        bot.send_message(chat_id=message.chat.id, text='Доброго дня, Агенте. Переглядайте заявки.', parse_mode="Markdown", reply_markup=markup)

@bot.message_handler(content_types=["text"], regexp = r'^Скарги та пропозиції$')
def skargs(message):
    conn.commit()
    row = pd.read_sql("SELECT * FROM skargs", con=conn)
    bot.delete_message(message.from_user.id, message.message_id)
    if not row.empty:
        text = "Скарги та пропозиції:\n"
        for i in row.values:
            text = text + 'Користувач з номером: {0}\nЗалишив скаргу/пропозицію: {1}\n\n'.format(i[0], i[1])
    else:
        text = " Наразі немає скарг та пропозицій"

    bot.send_message(chat_id=message.chat.id, text=text)

@bot.message_handler(content_types=["text"], regexp = r'^Переглянути заявку$')
def see(message):
    if message.from_user.id == 496588167 or message.from_user.id == 353662267:
        conn.commit()
        bot.delete_message(message.from_user.id, message.message_id)
        user_info = pd.read_sql("SELECT * FROM credits where `status` = 'on_approve'", con=conn)
        if not user_info.empty:
            user_info = user_info.sample()
```

```
        text = f"""Бажана кредитна сума:{user_info['credit_sum'].values[0]}
ПІБ: {user_info['pib'].values[0]}
Щомісячний прибуток У грн: {user_info['income'].values[0]}
Вік:{user_info['age'].values[0]}
Сімейний стан: {str(user_info['merried'].values[0]).replace('1','Одружений/заміжня').replace('0', 'Неодружений/незаміжня')}
Область проживання: {user_info['live_in'].values[0]}
Тип житла:{user_info['house_type'].values[0].replace('rent','Орендую').replace('own','Власне').replace('other','Інше')}
Кількість років проживання у вказаному житлі: {user_info['house_age'].values[0]}
Наявність автомобіля: {str(user_info['is_car'].values[0]).replace('1', 'Так').replace('1', 'Ні')}
Кількість років загального трудового стажу: {user_info['work_exp'].values[0]}
Робота на даний момент: Так
Посада: {user_info['work_place'].values[0]}
Трудовий стаж на теперішній роботі: {user_info['work_current_age'].values[0]}
Оцінка кредитоспроможності позичальника : {user_info['model_predict'].values[0]}"""

        markup = types.InlineKeyboardMarkup(row_width=1)
        approve = types.InlineKeyboardButton(text='Схвалити', callback_data='approved_{0}'.format(user_info['id'].values[0]))
        abandon = types.InlineKeyboardButton(text='Відмова', callback_data='abandoned_{0}'.format(user_info['id'].values[0]))
        markup.add(approve, abandon)

        bot.send_message(chat_id=message.chat.id, text=text, reply_markup=markup)
    else:
```

```
        text = "Наразі немає заявок на кредит"
        bot.send_message(chat_id=message.chat.id, text=text)
@bot.callback_query_handler(func=lambda call: True)
def call_back(call):
    row = pd.read_sql("SELECT * FROM credits where `status` = 'on_approve'", con=conn)
    data = call.data
    bot.delete_message(call.from_user.id, call.message.message_id)
    if 'approved' in data:
        credit_id = int(data.split('_')[-1])
        user_credit = row[row['id']==credit_id]
        query = """ UPDATE credits
                SET `status` = 'approved'
                WHERE id = {0} """.format(credit_id)
        cursor.execute(query)
        conn.commit()
        client_bot.send_message(chat_id=user_credit['user_id'].values[0], text='Ваш кредит з номером {0} одобрено!'.format(credit_id))
        bot.send_message(chat_id=call.from_user.id, text='Схвалено кредит номер {0}'.format(credit_id))
    elif 'abandoned' in data:
        credit_id = int(data.split('_')[-1])
        user_credit = row[row['id']==credit_id]
        query = """ UPDATE credits
                SET `status` = 'abandoned'
                WHERE id = {0} """.format(credit_id)
        cursor.execute(query)
        conn.commit()
        client_bot.send_message(chat_id=user_credit['user_id'].values[0], text='Ваш кредит з номером {0} відхилено'.format(credit_id))
```

```
bot.send_message(chat_id=call.from_user.id, text='Відмовлено кредит номер {0}'.format(credit_id))
```