НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

«До захисту допущено»
Завідувач кафедри
_____ Олег Чертов
«___» ___ 2023 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Наука про дані та математичне моделювання»
спеціальності 113 «Прикладна математика»
на тему: «Математичне та програмне забезпечення підсистеми аналізу та прогнозування відкриття банківських депозитів та кредитів»

Виконав:
студент IV курсу, групи КМ-93
Данілов Іван Дмитрович _____

Керівник:
Старший викладач
Любашенко Наталія Дмитрівна _____

Консультант з нормоконтролю:
Старший викладач,
Мальчиков Володимир Вікторович _____

Рецензент:
Професор каф. СПіСКС, д-р техн. наук, професор
Терейковський Ігор Анатолійович _____

Засвідчую, що в цій дипломній роботі
немає запозичень із праць інших авторів
без відповідних посилань.
Студент Данілов І.Д.

Київ — 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра прикладної математики

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 113 «Прикладна математика»

Освітньо-професійна програма «Наука про дані та математичне моделювання»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олег Чертов

«___» _____ 2023 р.

ЗАВДАННЯ

на дипломну роботу студенту

Данілову Івану Дмитровичу

1. Тема роботи: «Математичне та програмне забезпечення підсистеми аналізу та прогнозування відкриття банківських депозитів та кредитів», керівник роботи Любашенко Наталія Дмитрівна, старший викладач, затверджені наказом по університету від «31» травня 2023 р. № 2108-С.
2. Термін подання студентом роботи: «12» червня 2023 р.
3. Вихідні дані до роботи: мінімальна точність підсистеми – 70%.
4. Зміст роботи: проаналізувати існуючі програмні та математичні рішення розв’язання поставленої задачі, обрати та пояснити методи моделювання, спроєктувати підсистему аналізу та прогнозування відкриття банківських депозитів та кредитів, описати математичне та програмне забезпечення підсистеми, виконати програмну реалізацію підсистеми аналізу та прогнозування відкриття банківських депозитів та кредитів, провести верифікацію та валідацію розробленої підсистеми.
5. Перелік ілюстративного матеріалу: графіки, діаграми, блок-схеми розроблених алгоритмів.
6. Дата видачі завдання: «06» лютого 2023 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Огляд літератури за тематикою та збір даних	12.11.2022	
2	Проведення порівняльного аналізу математичних методів для створення підсистеми аналізу та прогнозування відкриття банківських депозитів та кредитів	14.12.2022	
3	Проведення порівняльного аналізу математичних методів для створення підсистеми аналізу та прогнозування відкриття банківських депозитів та кредитів	24.12.2022	
4	Підготовка матеріалів першого розділу роботи	01.02.2023	
5	Розроблення математичного забезпечення для створення підсистеми аналізу та прогнозування відкриття банківських депозитів та кредитів	01.03.2023	
6	Підготовка матеріалів другого розділу роботи	15.03.2023	
7	Підготовка матеріалів третього розділу роботи	05.04.2023	
8	Розроблення програмного забезпечення для створення підсистеми аналізу та прогнозування відкриття банківських депозитів та кредитів	15.04.2023	
9	Підготовка матеріалів четвертого розділу роботи	03.05.2023	
10	Оформлення пояснювальної записки	01.06.2023	

Студент _____ Іван ДАНІЛОВ

Керівник роботи _____ Наталія ЛЮБАШЕНКО

АНОТАЦІЯ

Дипломну роботу виконано на 61 аркуші, вона містить 2 додатки та перелік посилань на використані джерела з 11 найменувань. У роботі наведено 42 рисунків та 1 таблиця.

Метою даної дипломної роботи є створення математичного та програмного забезпечення підсистеми аналізу та прогнозування відкриття банківських депозитів та кредитів.

У роботі проведено аналіз існуючих рішень указаної задачі — штучних нейронних мереж, кластерного аналізу, моделей Маркова та машинного навчання. Виконано їх порівняння з погляду точності отримуваних розв’язків, ефективності алгоритмів та пристосованості методів до використання нечітких даних. Для розв’язання задачі в роботі вибрано методи машинного навчання.

Для поставленої задачі роботи створено математичне та програмне забезпечення.

Ключові слова: банківські продукти, кредитування, аналіз банківського ринку, машинне навчання, логістична регресія, дерева рішень, класифікатор випадкових лісів, класифікатор XGBoost.

ABSTRACT

The thesis is presented in 61 pages. It contains 2 appendixes and bibliography of 16 references. Forty two figures and 1 table are given in the thesis.

The aim of this thesis is to develop mathematical and software tools for the subsystem of analysis and prediction of bank deposits and loans.

The thesis analyzes existing solutions for the specified problem, including artificial neural networks, cluster analysis, Markov models, and machine learning. A comparison is made in terms of the accuracy of the obtained solutions, algorithm efficiency, and adaptability of methods to handle fuzzy data. Machine learning methods have been chosen for solving the problem in this thesis.

Mathematical and software tools have been developed for the stated problem in the thesis.

Keywords: banking products, lending, analysis of the banking market, machine learning, logistic regression, decision trees, random forest classifier, XGBoost classifier.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

KNN – k-Nearest Neighbors,

SMOTE – Synthetic Minority Oversampling Technique,

SVM – Support Vector Machines,

XGBoost – eXtreme Gradient Boosting.

ВСТУП

У сучасному світі банківська сфера постійно змінюється та вдосконалюється, оскільки з'являється все більше і більше нових технологій. Разом з цим постає актуальне питання у впровадженні цих технологій в безпосередньо банківську сферу, а саме для прогнозування кредитних ризиків, аналізу банківського ринку тощо. На даний момент існує багато платних та безкоштовних програм, методів та рішень для автоматизації певних банківських процесів, які використовують у свої реалізації статистичні методі для аналізу та обробки даних, методи машинного навчання, та нейронні мережі.

У даній роботі поставлено за мету дослідити методи машинного навчання і створити підсистему для аналізу та прогнозування відкриття банківських депозитів та кредитів. Для кредитування реалізовано прогнозування кредитних ризиків, в той час як для відкриття депозитів створено модель для аналізу банківського ринку і пошуку найкращих клієнтів.

1 ПОСТАНОВКА ЗАДАЧІ

Метою роботи є розробка математичного та програмного забезпечення для створення підсистеми аналізу та прогнозування відкриття банківських депозитів та кредитів.

При розробленні відповідного забезпечення потрібно розв’язати наступні завдання:

а) оглянути та проаналізувати існуючі програмні та математичні рішення;

б) описати математичне забезпечення підсистеми аналізу та прогнозування відкриття банківських депозитів та кредитів;

в) обрати та обґрунтувати методи моделювання;

г) розробити програмне забезпечення підсистеми аналізу та прогнозування відкриття банківських депозитів та кредитів;

д) верифікувати та валідувати підсистему.

Реалізована підсистема має задовольняти такі вимоги:

а) мати високу точність прогнозування кредитних ризиків;

б) мати високу точність аналізу банківського ринку.

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

2.1 Аналіз існуючих програмних рішень

Яскравим прикладом для прогнозування кредитних ризиків та аналізу банківського ринку є робота [1]. В даній роботі використовується новітній метод, який заснований на гібридному методі штучних нейронних мереж та вдосконаленій версії алгоритму пошуку Сови. Також використовується прогнозування ризику C5 кредиту згідно з деревом прийняття рішень. Загалом робота побудована наступним чином:

а) перевірка даних клієнтів банку;
б) підготовка та виправлення даних;
в) кластеризація даних на два кластери;
г) вибір функцій на основі покращеного алгоритму пошуку Сови;
д) побудова двох дерев рішень для кожної кластерної нейронної мережі та тестування датасету.

Для розв’язання поставленої задачі більшість компанії застосовує емпіричний підхід, в якому розглядають створені моделі на основі нейронних мереж, а навчання та перевірка моделей використовується для представлення експериментальних результатах, як це зроблено в роботі [2]. В даній роботі розробники прийшли до того, що створена нейронна мережа є методом «чорної скриньки», через що отриманий результат складно обґрунтувати. Додатково автори виділили, що при використанні нейронних мереж для прогнозування кредитних ризиків користувачеві необхідно виконати додаткові кроки у вигляді нормалізації даних та перевірці атрибутів.

Більшість ІТ-компаній вже створили програмні продукти для аналізу клієнтської бази, прогнозування кредитних ризиків тощо. Однією з них є компанія Moody’s Analytics [3]. Програмний продукт, створений ними, використовує нейронні мережі для прогнозування фінансових показників. Створені моделі кредитних ризиків активно відстежуються та перевіряються на основі поточних економічних

умов та останніх доступних наборів даних для забезпечення високої передбачуваності та точності.

Перевагою продукту є велика варіативність надаваних послуг, а також доволі зручний інтерфейс.

Головним недоліком даного продукту є недостатня прозорість. Деякі критики вказують на недостатню прозорість щодо моделей та методологій, якими користується Moody's Analytics для оцінки ризиків та прогнозування фінансової стійкості компаній і країн. Відсутність детальних пояснень може викликати сумніви щодо достовірності їхніх аналізів.

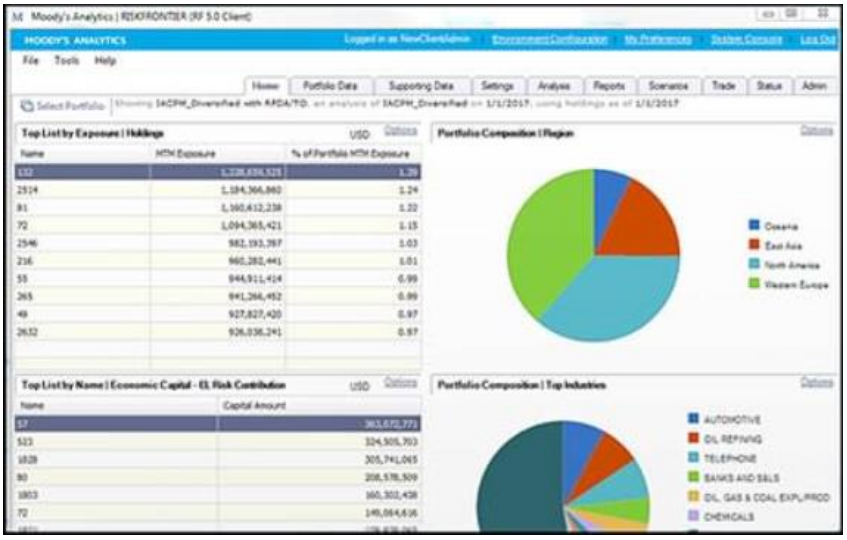


Рисунок 2.1 – Графічний інтерфейс програми Moody’s Analytics

Окрім нейронних мереж, існують програмні рішення, які застосовують методи машинного навчання. Платформа Alteryx Analytics Cloud Platform [4] є саме такою. Дана платформа має вбудовані інструменти для машинного навчання і дозволяє користувачам аналізувати банківські дані, будувати моделі прогнозування тощо.

Хоча Alteryx Analytics Cloud Platform має широкий набір функціональних можливостей, іноді може виникати необхідність в більш специфічних аналітичних інструментах або налаштуваннях, яких не вистачає в платформі. Це може становити проблему для деяких користувачів зі специфічними потребами. Також оскільки Alteryx Analytics Cloud Platform базується на хмарі, для його використання необхідне

стабільне та надійне інтернет-підключення. Відсутність доступу до Інтернету може призвести до обмеження або неможливості використання платформи

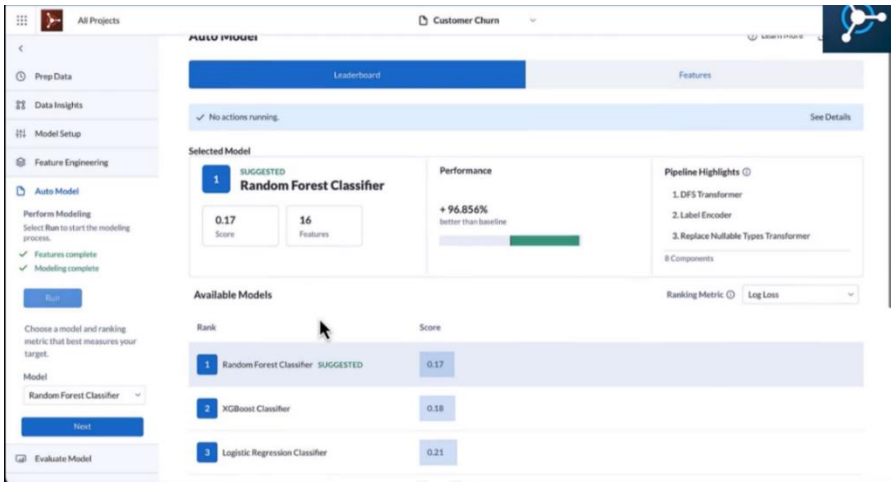


Рисунок 2.2 – Графічний інтерфейс Alteryx Analytics Cloud Platform

Машинне навчання також використовують і автори роботи [5]. Окрім створення моделей автори також будують конвеєр для процедур перехресної валідації та шукають оптимальних параметрів моделі.

2.2 Математичні методи для створення підсистеми аналізу та прогнозування відкриття банківських депозитів та кредитів

2.2.1 Кластерний аналіз

Кластерний аналіз – це метод аналізу даних, який досліджує природні групи всередині набору даних, що називаються кластерами. Цей метод не потребує заздалегідь визначених груп для точок даних, тому він відноситься до неконтрольованого навчання.

Фінансові та банківські організації використовують різні алгоритми кластерного аналізу, щоб ділити клієнтів на різні категорії ризику, враховуючи їх

банківський баланс та заборгованості. Також створені кластери використовується під час затвердження різних банківських послуг, таких як кредити, страхування, депозити тощо.

2.2.2 Моделі Маркова

Моделі Маркова – стохастичні моделі у теорії ймовірностей, що використовуються для моделювання систем, які випадково змінюються. Найпростішим типом моделі Маркова є ланцюг Маркова, де всі стани є спостережуваними, а ймовірність збігається з часом. Основною його перевагою є його можливість прогнозувати майбутній результат, використовуючи лише поточний стан.

Загалом перевагою моделей Маркова є їх простота в реалізації та розумінні. Також моделі Маркова можуть бути ефективними при аналізі послідовностей подій або станів. Вони дозволяють здійснювати передбачення майбутніх станів на основі поточного стану, що робить їх придатними для багатьох задач, включаючи розпізнавання мови, обробку природної мови, аналіз фінансових ринків та багато інших.

Однак з недоліків можна виділити, що моделі Маркова мають обмежену пам'ять і не зберігають історію подій або станів, що сталися до поточного моменту. Це може призвести до втрати корисної інформації, особливо в тих випадках, коли історія має велике значення для подальшого моделювання. Додатково моделі Маркова можуть вимагати великої кількості даних для навчання, особливо у випадку складних систем або великих просторів станів. Якщо дані обмежені, це може призвести до перенавчання або недостатньої точності моделі.

2.2.3 Штучні нейронні мережі

Штучні нейронні мережі – це алгоритми, що створені на основі роботи мозку, а саме його біологічних нейронних мережах. В загальному значенні використовуються для задач прогнозування, моделювання закономірностей різної складності тощо.

Штучні нейронні мережі під час розробки виконують функції нелінійного адаптивного фільтра та класифікатора образів. Після навчання система стає налаштованою на вирішення поставленої задачі.

Головними перевагами нейронних мереж є збереження даних у всій мережі, а не в базі даних, можливість продовжувати роботу навіть, коли якийсь її елемент відмовляє.

Проте з важливих недоліків є вимоги потужних графічних процесорів для тренування моделей, можливість бути чутливими до змін вхідних даних, що може викликати непередбачувані відповіді та потреба у великому обсягу даних для ефективного тренування.

2.2.4 Машинне навчання

Машинне навчання – це вид штучного інтелекту, який спрямований на використання даних та алгоритмів з для імітації процесу навчання людей при цьому поступово підвищуючи його точність.

Модель машинного навчання – це набір кроків, які беруть участь у створенні прогнозів на основі певних даних. У випадку банківських продуктів основними даними для створення моделі машинного навчання будуть дані про клієнтів, а саме їх дохід, робота, вік, заборгованості.

Регресійні методи машинного навчання допомагають спрогнозувати певне числове значення ґрунтуючись на попередніх даних. Найпростішим видом регресійних методів є лінійна регресія, яка використовує математичне рівняння лінії $y = k * x + b$ для моделювання набору даних. Під час роботи цього методу ми обчислюємо нахил і точку перетину лінії, яка найточніше та найкраще апроксимує спостереження в даних.

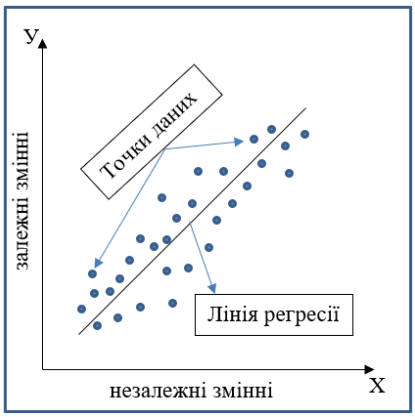


Рисунок 2.3 – Графічне зображення лінійної регресії

Якщо ж необхідно спрогнозувати категоріальні залежні змінні за допомогою заданого набору незалежних змінних, то використовується логістична регресія. Головна відмінність логістичної регресії від лінійної полягає саме в її використанні, оскільки логістична призначена для задач класифікації, в той час як лінійна – для задач регресії. При використанні логістичної регресії ми підбираємо S-подібно логістичну функцію, замість лінії регресії. Дана крива вказує на ймовірність певного заданого факту.

Алгоритм «K-Nearest Neighbors» – алгоритм машинного навчання, який припускає, що схожі речі існують поруч одна з одною. Даний алгоритм використовується зазвичай для задач класифікації. Основна ідея алгоритму полягає в тому, що маючи точку з невідомим класом, ми можемо спробувати зрозуміти, які точки в нашому просторі ознак є найближчими до неї. Такі точки називаються K-найближчими сусідами. Оскільки алгоритм зберігає всі дані і визначає нову точку на

основі подібності, то при появі нових даних, їх можна легко класифікувати. Алгоритм «K-Nearest Neighbors також має назву «алгоритм лінивого навчання», бо він зберігає дані і виконує дії над ними лише під час класифікації, а не навчається на навальній вибірці. По пунктах алгоритм виглядає наступним чином [6]:

1) завантажуємо дані;
2) ініціалізуємо К-значенням обрану кількість сусідів;
3) для кожного прикладу в даних обчислюємо відстань між запитовим прикладом та поточним прикладом з даних;
4) додаємо відстань та індекс прикладу до впорядкованої колекції
5) відсортовуємо впорядковано колекцію відстаней та індексів від найменшої до найбільшої за відстанями;
6) отримуємо мітки обраних К міток;
7) якщо отримуємо регресію, то повертаємо середнє значення К міток;
8) якщо отримуємо класифікацію, то повертаємо моду К міток.

Однак KNN може бути обчислювально витратним для великих наборів даних, особливо якщо кількість ознак велика. Також оскільки даний алгоритм використовує інформацію з найближчих сусідів для класифікації або регресії, то це може призводити до меншої здатності узагальнення до нових, раніше невиданих зразків, зокрема у випадку даних з великою кількістю ознак.

Навчання з підкріпленням – це метод машинного навчання, який дозволяє програмі вчитись в інтерактивному середовищі за допомогою методу спроб і помилок. Після кожного кроку алгоритм отримує зворотній зв’язок, шо дозволяє йому визначити правильність вибору. Даний алгоритм використовується для автоматизованих систем, які мають приймати велику кількість малих рішень без участі людини.

Алгоритм випадкового лісу – це алгоритм машинного навчання, який поєднує вихідні дані декількох дерев рішень, щоб досягнути єдиного результату. Даний алгоритм вирішує і задачі класифікації, і проблеми регресії. За собою алгоритм має декілька основних гіперпараметрів, що мають бути встановлені перед початком

навчання. Цими гіпермараметрами є кількість дерев, кількість вибіркових ознак і класифікатор випадкового лісу. Основні етапи роботи алгоритму є створення випадкового лісу, об'єднуючи N дерев рішень і прогнозування для кожного створеного дерева. Покроково алгоритм приймає такий вигляд [7]:

1) обираємо випадкові K точок даних з навчальної вибірки;
2) будуємо дерева рішення, що пов'язані з обраними точками даних;
3) обираємо число N для дерев рішень, які хочемо побудувати;
4) повторюємо кроки 1-2;
5) для нових точок даних знаходимо прогнози кожного дерева рішень і відносимо нові точки даних до категорії, яка набрала більшість голосів.

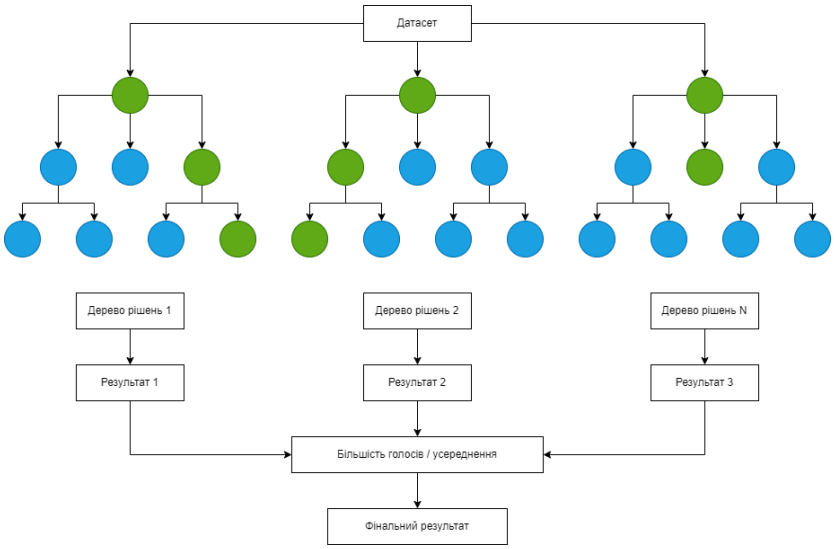


Рисунок 2.4 – Графічне зображення алгоритму випадкового дерева

XGBoost – це реалізація дерев рішень з градієнтним прискоренням. В даному алгоритмі важливу роль відіграють ваги, оскільки вони присвоюються всім незалежним змінним, які потім подаються в дерево рішень, яке прогнозує результати. Алгоритм підходить до процесу послідовної побудови дерева, використовуючи розпаралелену реалізацію, що покращує алгоритмічну продуктивність.

Класифікатор дерева рішень (Decision Tree Classifier) є алгоритмом, який розбиває дані на менші підмножини засновані на різних критеріях, при цьому кожна

підмножина має свою категорію сортування. Кількість об'єктів, що відповідають певному критерію, зменшується з кожним поділом. Процес класифікації закінчується, коли мережа доходить до підмножини з лише одним об'єктом.

Переваги методу:

1) легка інтерпретованість моделі, що дозволяє просто розуміти та пояснювати прийняті рішення;
2) простота візуалізації дерева рішень, що дозволяє не лише представити саму модель, а й зробити прогноз для окремих тестових об'єктів;
3) досить швидке навчання та прогнозування;
4) простота та ефективность, яка здобувається за допомогою мінімальної кількості параметрів у моделі;
5) Підтримка числових і категоріальних ознак.

Недоліки:

1) чутливість дерев до шумів у вхідних даних, що може призводити до неточності в прогнозуванні;
2) обмежена гнучкість у розділенні даних, яку може мати дерево рішень, що може вплинути на загальну якість класифікації порівняно з іншими методами.

2.3 Висновки до розділу

Під час роботи над даним розділом було проаналізовано існуючі методи і алгоритми для задачі створення підсистеми аналізу та прогнозування відкриття банківських депозитів та кредитів. Після огляду рішень була обрана реалізація за допомогою методів машинного навчання. Проаналізувавши методи машинного навчання, було обрано: класифікатор XGBoost, логістичну регресію, класифікатор

випадкових лісів та дерев рішень. Під час програмної реалізації дані алгоритми будуть порівнюватись, верифікуватись та валідуватись.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Підготовка та первинна обробка даних

Першим кроком створення підсистем банківських продуктів є підготовка та первинна обробка даних. В даний пункт включені такі кроки:

1) Стандартизація усіх даних;
2) Видалення повторюваних значень;
3) Усунення Nan, Null значень шляхом видалення або заміни на середнє значення.

Далі до набору даних застосовується розвідувальний аналіз, що має за собою на меті:

1) розрахунок елементів описової статистики, а саме середніх арифметичних, мод, медіани, дисперсій;
2) проведення кореляційного та причинно-наслідкового аналізів.

3.2 Масштабування та стандартизація ознак

Перш ніж провести передескритизацію даних необхідно масштабувати та стандартизувати ознаки.

Масштабування ознак – процес зміни масштабу всіх змінних в наборі даних до певного заданого діапазону. У деяких методах машинного навчання, а саме в лінійній та логістичній регресії, використовується оптимізація градієнтного спуску. Це необхідно для того, щоб ці алгоритми працювали ефективно. Формула градієнтного спуску має наступний вигляд:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) x_j^{(i)}, \quad (3.1)$$

Оскільки у формулі присутнє значення ознаки X, що впливає на розмір кроку градієнтного спуску, то різниця в діапазонах ознак призведе до різного розміру кроку для кожної ознак. Масштабування ознак використовується якраз для того, щоб гарантувати, що градієнтний спуск плавно рухається до мінімуму і що кроки градієнтного спуску оновлюються з однаковою швидкістю для всіх ознак.

Стандартизація – це процес масштабування значень даних таким чином, що вони набувають властивостей стандартного нормального розподілу. Тобто це означає, що дані масштабуються так, що середнє значення стає нульовим, а дані мають стандартне відхилення, яке рівне одиниці. Формула стандартизації виглядає наступним чином:

$$x_{stand} = \frac{x - mean(x)}{standard\ deviation(x)}, \tag{3.2}$$

3.3 Передискретизація даних

Під час роботи з методами машинного навчання виникає проблема незбалансованого розподілу даних. Це трапляється, коли спостереження в одному з класів набагато вищі або нижчі, ніж в інших класах. Через те, що методи машинного навчання зменшують похибку, щоб підвищити точність, то вони не враховують розподіл класів. Отже, якщо ми маємо незбалансований розподіл даних, то створена модель стає більш схильною до випадку, в якому клас меншини має малу або дуже незначну згадку.

SMOTE – один з найвідоміших методів передискретизації даних, який верішує проблему дисбалансу. Мета даного методу полягає в балансуванні розподілі класів випадково збільшуючи кількість прикладів класів меншин, реплікуючи їх. Покроково метод виглядає наступним чином [8]:

1) задаємо множину класі меншин A для кожного $x \in A$, k-найближчих сусідів x отримують шляхом обчислення евклідової відстані між x та кожним іншим зразком у множині A;
2) частота дискретизації N задається відповідно до незбалансованої пропорції. Для кожного $x \in A$, N прикладів (тобто $(x_1, x_2, \dots x_n)$ випадковим чином вибираються з його k-найближчих сусідів і будують множину A;
3) для кожного прикладу $x_k \in A_1(k = 1,2, \dots N)$, використовується наступна формула для генерації наступного прикладу:

$$x' = x + rand(0,1) * |x - x_k|, \tag{3.3}$$

де $rand(0,1)$ представляє випадкове число між 0 та 1.

3.4 Методи машинного навчання

3.4.1 Логістична регресія

Даний метод оцінює ймовірність того, що станеться на основі набору даних незалежних змінних. Модель дає не точне значення 0 і 1, а значення в даному проміжку.

Нехай наші дані це (X, Y), де X – матриця значень з m прикладами та n ознаками, а Y – вектор з m прикладами. Завдання логістичної регресії – навчити модель передбачати належність до певного класу майбутні значення.

На початку роботи із логістичною регресією необхідно створити вагову матрицю з випадковою ініціалізацією і помножити її на ознаки.

$$a = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n. \tag{3.4}$$

Результат розрахунку (2.1) передаємо у функцію зв’язку.

$$\widehat{y_i} = \frac{1}{1+e^{-a}}. \quad (3.5)$$

Після цього розраховуємо вартість для даної ітерації, для якої формула має наступний вигляд:

$$cost(w) = \left(-\frac{1}{m}\right)\sum_{i=1}^{1=m} y_i \log(\widehat{y_i}) + (1-y_i)log(1-\widehat{y_i}). \quad (3.6)$$

Далі обчислюємо похідну від даної вартості і після цього ваги оновлюються.

$$cdw_j = \sum_{i=1}^{i=n}(\widehat{y_i} - y)x_j^i, \quad (3.7)$$

$$w_i = w_j - \left(\alpha * dw_j\right). \quad (3.8)$$

Метод логістичної регресії має два класи [9]:

1. бінарний;

2. мультиноміальний.

У бінарному випадку припускається, що цільова функція y_i приймає значення з множини $\{0, 1\}$ для точки даних i. Після підбору прогнозується ймовірніть позитивного класу $P(y_i = 1|X_i)$ як

$$\hat{p}(X_i) = expit(X_i w + w_0) = \frac{1}{1+exp(-X_i w - w_0)}. \quad (3.9)$$

Логістична регресія бінарного класу з регуляризаційним членом $r(w)$ мінімізує наступну функцію витрат:

$$\min_w C\sum_{i=1}^{n}\left(-y_i \log\left(\hat{p}(X_i)\right) - (1-y_i)\log\left(1-\hat{p}(X_i)\right)\right) + r(w). \quad (3.10)$$

Регуляризаційний член $r(w)$ може приймати значення одного з чотирьох наступних варіантів:

1) «None» = 0

2) «l_1» =

$$\|w\|_1. \tag{3.11}$$

3) «l_2» =

$$\frac{1}{2}\|w\|\frac{2}{2} = \frac{1}{2}w^T w\ . \tag{3.12}$$

4) ElasticNet» =

$$\frac{1-p}{2}w^T w + \rho\|w\|_1\ . \tag{3.13}$$

де ρ контролює міцність l_1 регуляризації проти l_2 регуляризації. При $\rho = 1$ «ElasticNet» еквівалентна «l_1», якщо ж $\rho = 0$, то «ElasticNet» еквівалентна «l_2».

Якщо ж бінарний випадок розширити до K класів, то отримаємо мультиноміальний випадок. Припустимо, що $y_i \in 1, \dots, K$ є порядковим кодом цільової змінної для спостереження i, тобто майбутніх передбачуваних класів. На відміну від бінарного випадку, у мультиноміальному присутня матриця коефіцієнтів W, в якій кожен вектор-рядок W_k відповідає класу k. Отже передбачувана ймовірність класів $P(y_i = k|X_i)$ розраховується наступним чином:

$$\hat{p}(X_i) = \frac{\exp(X_i W_k + W_{0,k})}{\sum_{l=0}^{K-1}\exp(X_i W_l + W_{0,l})}\,. \tag{3.14}$$

Логістична регресія мультиноміального класу з регуляризаційним членом $r(w)$ мінімізує наступну функцію витрат:

$$\min_{w} -C \sum_{i=1}^{n} \sum_{k=0}^{K-1} [y_i = k] \log\left(\hat{p}(X_i)\right) + r(w) . \quad (3.15)$$

В даній формулі значення $[y_i = k]$ приймає або значення 0, якщо воно є хибним і значення 1, якщо воно є істинним.

Регуляризаційний член $r(w)$ може приймати значення одного з чотирьох наступних варіантів:

1) «None» = 0

2) «l_1» =

$$\|w\|_{1,1} = \sum_{i=1}^{n} \sum_{j=1}^{K} \left|W_{i,j}\right| . \quad (3.16)$$

3) «l_2» =

$$\frac{1}{2} \|w\| \frac{2}{F} = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{K} W_{i,j}^2 . \quad (3.17)$$

4) «ElasticNet» =

$$\frac{1-p}{2} \|w\| \frac{2}{F} + \rho \|w\|_{1,1} . \quad (3.18)$$

Під час розрахунку цільової функції виконується також задача оптимізація. Для неї існують наступні алгоритми:

1) «lbfgs» – алгоритм, який використовується за замовчунням. Даний алгоритм наближає алгоритм Бройдена-Флетчера-Гольдфарба-Шанно. Перевагою алгоритма є можливість працювати з великою кількістю різних тренувальних даних. В той час як недоліком є погані показники продуктивності під час роботи з погано масштабованими наборами даних.
2) «liblinear» – алгоритм, який використовує алгоритм координатного спуску. В даному алгоритмі задача оптимізації розкладається за принципом «один

проти решти». Як результат у кожному класі проводиться навчання окремих двійкових класифікаторів.

3) «newton-cg» – метод Ньютона-Рафсона, який використовує спряжений градієнт для лінійної оберненої підзадачі.
4) «newton-cholesky» – алгоритм Ньютона, що ґрунтується на обчислені матриці Гессе і розв'язанні отриманої лінійної системи.
5) «sag» – алгоритм, який базується на використанні методу спуску за стохастичним середнім градієнтом. Даний алгоритм добре підходить для наборів даних, які мають великий об’єм.
6) «saga» – різновид алгоритму «sag», який краще підходить для вирішення розрідженої логістичної багатофакторної регресії.

Використання кожного алгоритму залежить від обраного варіанту регуляризаційного члену $r(w)$. Підтримка того чи іншого алгоритму зазнчену у таблиці 3.1.

Таблиця 3.1 – Підтримка регуляризаційних членів за алгоритмами

	Варіант регуляризаційного члену			
Назва алгоритму	«None»	«l_1»	«l_2»	«ElasticNet»
«lbfgs»	+	–	+	–
«liblinear»	–	+	+	–
«newton-cg»	+		+	–
«newton-cholesky»	+	–	+	–
«sag»	+	–	+	–
«saga»	+	+	+	+

3.4.2 Класифікатор випадкових лісів

Використовуючи метод класифікатора випадкових лісів для розв'язання регресійних задач також використовується середня квадратична похибка, яка має назву MSE. Вона необхідна для того, щоб визначити, як наші дані розгалужуються з кожного вузла.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(f_i - y_i)^2 \,, \tag{3.19}$$

де N – кількість точок даних, f_i – значення, що повертається моделлю, y_i – актуальне значення для точки даних i.

Дана формула розраховує відстань від кожного вузла від прогнозованого фактичного значення. Це допомагає знайти найкращу гілку рішень.

Кроки алгоритму випадкового лісу

1) Спочатку необхідно створити підмножини вихідних даних. Для цього зробимо вибірку рядків і вибірку ознак, тобто виберемо рядки і стовпці із заміною і створимо підмножини навчального набору даних;
2) створимо індивідуальне дерево рішень для кожної підмножини, яку ми візьмемо;
3) кожне дерево рішень дасть результат;
4) остаточний результат розглядається на основі голосування більшості, якщо це задача класифікації, і середнього значення, якщо це задача регресії.

3.4.3 Класифікатор XGBoost

Алгоритм методу XGBoost ґрунтується на алгоритмі градієнтного бустингу дерев рішень.

Градієнтний бустинг – це потужний алгоритм машинного навчання, який використовується для табличних наборів даних. Використовується для пошуку будь-якого нелінійного зв'язку між цільовою функцією моделі та ознаками.

Першим кроком градієнтного бустингу є побудова базової моделі для прогнозування спостережень у навчальному наборі даних. Тобто ми беремо середнє значення цільового стовпчика і вважаємо його прогнозованим значення.

$$F_0(x) = \underset{\gamma}{argmin} \sum_{i=1}^{n} L(y_i, \gamma) \,, \tag{3.20}$$

де L – функція втрат, яка в даному випадку регресії є квадратом втрат:

$$L = (y_i - \gamma)^2 \,, \tag{3.21}$$

$argmin$ – функція пошуку значення γ, яке мінімізує $\sum_{i=1}^{n} L(y_i, \gamma)$

Наступним кроком є розрахунок псевдо залишків, які становлять (спостережуване значення – прогнозоване значення).

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \text{ для } i = 1, \dots, n, \,, \tag{3.22}$$

На наступному етапі ми побудуємо модель, використовуючи ці псевдозалишки, і здійснимо прогнозування. Це необхідно з метою мінімізації цих залишків, а зниження залишків у кінцевому підсумку покращить точність нашої моделі та забезпечить більш ефективне прогнозування.

На четвертому кроці ми визначаємо початкові значення для кожного листка у нашому дереві рішень. Це означає, що можуть виникнути ситуації, коли один листок отримує більше одного залишку. Тому нам необхідно знайти кінцевий результат для кожного листка. Для цього ми можемо просто обчислити середнє арифметичне всіх чисел, що знаходяться на листку, незалежно від того, чи міститься там лише одне число чи кілька чисел.

$$\gamma = \frac{1}{n}\sum_{x_i \in R_{jm}} r_{im},, , \tag{3.23}$$

На останньому кроці необхідно оновити прогнози попередньої моделі.

$$F_m(x) = F_{m-1}(x) + v\sum_{j=1}^{Jm} \gamma_{jm} 1\big(x \in R_{jm}\big),, , \tag{3.24}$$

3.4.4 Класифікатор дерева рішень

Класифікатор дерева рішень є методом керованого навчання, що використовується для класифікації та регресії, і є непараметричним. Основною метою цього методу є побудова моделі, яка може передбачати значення цільової змінної шляхом вивчення простих правил прийняття рішень, які випливають з характеристик даних.

Для побудови класифікатора дерева рішень існує багато різних методів, які за собою використовують значення ентропії, що розраховується наступним чином:

$$Entropy = -\sum_{i=1}^{n} p_i * \log(p_i), \tag{3.25}$$

та значення індексу Джині, що розраховується наступним чином:

$$Gini\ index = 1 - \sum_{i=1}^{n} p_i^2, \quad (3.26)$$

В даній роботі використовується алгоритм CART, який використовує індекс Джині для розбиття набору даних на дерево рішень. Він робить це шляхом пошуку найкращої однорідності для підвузлів. CART-моделі формуються шляхом вибору вхідних змінних та оцінки точок розбиття цих змінних, доки не буде побудовано відповідне дерево.

3.5 Показники для оцінки точності створених моделей

Для оцінки точності створених моделей буде використовуватись матриця невідповідностей. Вона надзвичайно корисна для вимірювання кривих Recall, Precision, Specificity, Accuracy і, що найважливіше, для побудови графіків AUC-ROC та Precision-Recall. Сама ж матриця має наступний вигляд :

Рисунок 3.1 – Графічне зображення матриці невідповідностей

Значення Recall використовується для розрахунку ефективності моделі класифікації в пошуку всіх прикладів з набору даних, які підходять умові. Дане значення розраховується наступним чином:

$$Recall = \frac{TP}{TP+FN}. \quad (3.27)$$

Значення Precision використовується для розрахунку точності прогнозів моделі. Дане значення розраховується наступним чином:

$$Precision = \frac{TP}{TP+FP}. \quad (3.28)$$

Значення Accuracy використовується для розрахунку продуктивності моделі. Дане значення розраховується наступним чином:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}. \quad (3.29)$$

Значення F-measure використовується для оцінки загальної ефективності моделі класифікації. Дане значення розраховується наступним чином:

$$F - measure = \frac{2*Recall*Precision}{Recall+Precision}. \quad (3.30)$$

Графік "Precision-Recall" відображає зв'язок між точністю і відтворенням для різних порогових значень [10]. Чим більше площа під кривою, тим вище якість моделі, оскільки це означає високий рівень як точності, так і відтворення. Висока точність вказує на низький рівень помилкових спрацьовувань, тоді як високе відтворення означає низький рівень помилкових негативних спрацьовувань.

Система з високим відтворенням, але низькою точністю, повертає багато результатів, але більшість з них помилкові у порівнянні з наявними мітками. З іншого боку, система з високою точністю, але низьким відтворенням, повертає дуже обмежену кількість результатів, але більшість з них правильні у порівнянні з наявними мітками. Ідеальна система з високою точністю і високим відтворенням повертає багато результатів, причому всі ці результати будуть правильно позначені.

Крива AUC-ROC є метрикою ефективності для задач класифікації при різних порогових значеннях. ROC відображає криву ймовірностей, а AUC (площа під кривою) вказує на рівень відокремленості класів. Високе значення AUC свідчить про те, що модель має здатність вірно класифікувати 0 класи як 0 і 1 класи як 1. Іншими словами, чим більше AUC, тим краще модель відрізняє класи між собою.

ROC-крива будується з TPR проти FPR, де TPR відкладається на осі y, а FPR - на осі x. В свою чергу значення TPR розраховується наступним чином:

$$TPR = \frac{TP}{TP+FN}. \tag{3.31}$$

А значення FPR:

$$FPR = \frac{FP}{TN+FP}, \tag{3.32}$$

3.6 Висновки до розділу

В даному розділі було розглянуто наступні методи машинного навчання: Логістична регресія, класифікатор випадкових лісів, класифікатор XGBoost, класифікатор дерева рішень. Під час аналізу їх теоретичного підґрунтя було створено математичне забезпечення для підсистеми аналізу та прогнозування відкриття банківських депозитів та кредитів.

Також розглянуто та обрано показники для оцінки створених моделей.

Разом з цим описано математичне забезпечення для підготовки та первинної обробки даних, масштабування та стандартизації ознак і передискретизації даних.

4 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

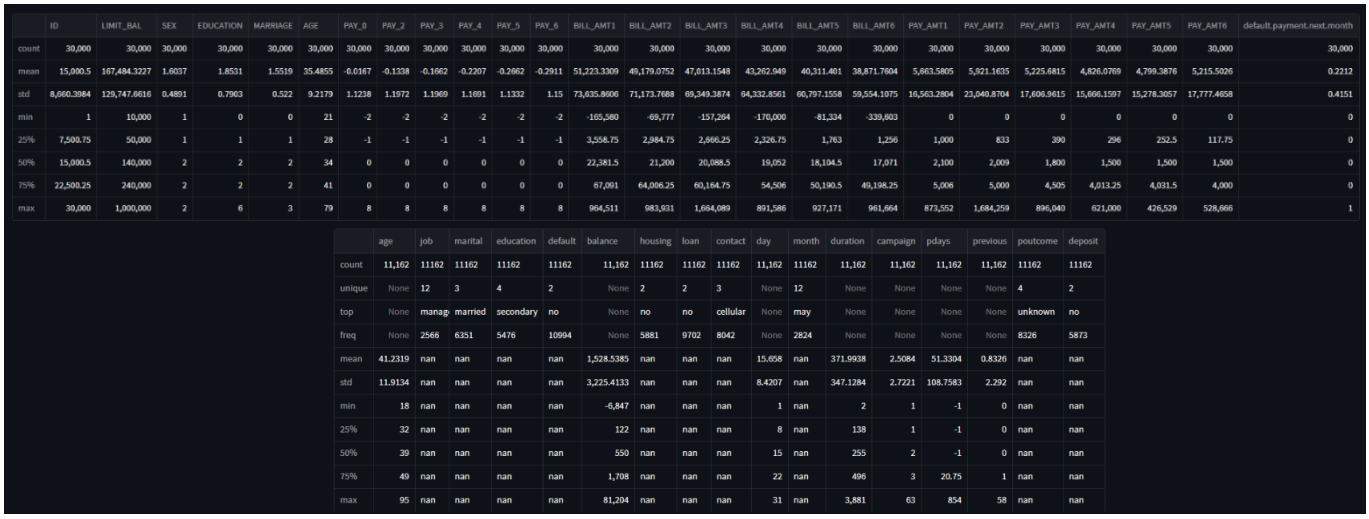
4.1 Підготовка та первинна обробка даних

Пер ніж обробляти дані необхідно їх проаналізувати. Для цього використовуються методи бібліотеки Pandas, такі як describe(), info() та value_counts().

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 25 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   ID                          30000 non-null  int64
 1   LIMIT_BAL                   30000 non-null  float64
 2   SEX                         30000 non-null  int64
 3   EDUCATION                   30000 non-null  int64
 4   MARRIAGE                    30000 non-null  int64
 5   AGE                         30000 non-null  int64
 6   PAY_0                       30000 non-null  int64
 7   PAY_2                       30000 non-null  int64
 8   PAY_3                       30000 non-null  int64
 9   PAY_4                       30000 non-null  int64
 10  PAY_5                       30000 non-null  int64
 11  PAY_6                       30000 non-null  int64
 12  BILL_AMT1                   30000 non-null  float64
 13  BILL_AMT2                   30000 non-null  float64
 14  BILL_AMT3                   30000 non-null  float64
 15  BILL_AMT4                   30000 non-null  float64
 16  BILL_AMT5                   30000 non-null  float64
 17  BILL_AMT6                   30000 non-null  float64
 18  PAY_AMT1                    30000 non-null  float64
 19  PAY_AMT2                    30000 non-null  float64
 20  PAY_AMT3                    30000 non-null  float64
 21  PAY_AMT4                    30000 non-null  float64
 22  PAY_AMT5                    30000 non-null  float64
 23  PAY_AMT6                    30000 non-null  float64
 24  default.payment.next.month  30000 non-null  int64
dtypes: float64(13), int64(12)
memory usage: 5.7 MB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11162 entries, 0 to 11161
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        11162 non-null  int64
 1   job        11162 non-null  object
 2   marital    11162 non-null  object
 3   education  11162 non-null  object
 4   default    11162 non-null  object
 5   balance    11162 non-null  int64
 6   housing    11162 non-null  object
 7   loan       11162 non-null  object
 8   contact    11162 non-null  object
 9   day        11162 non-null  int64
 10  month      11162 non-null  object
 11  duration   11162 non-null  int64
 12  campaign   11162 non-null  int64
 13  pdays      11162 non-null  int64
 14  previous   11162 non-null  int64
 15  poutcome   11162 non-null  object
 16  deposit    11162 non-null  object
dtypes: int64(7), object(10)
memory usage: 1.4+ MB
```

Рисунок 4.1 – Вивід інформації про назви стовпців, їх тип, кількість записів та кількість ненульових елементів двох датасетів для прогнозування кредитних ризиків та аналізу банківського ринку



	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	default.payment.next.month
count	30,000	30,000	30,000	30,000	30,000	30,000	30,000	30,000	30,000	30,000	30,000	30,000	30,000	30,000	30,000	30,000	30,000	30,000	30,000	30,000	30,000	30,000	30,000	30,000	30,000
mean	15,000.5	167,484.3227	1.6037	1.8531	1.5519	35.4855	-0.0167	-0.1338	-0.1662	-0.2207	-0.2662	-0.2911	51,223.3309	49,179.0752	47,013.1548	43,262.949	40,311.401	38,871.7604	5,663.5805	5,921.1635	5,225.6815	4,826.0769	4,799.3876	5,215.5026	0.2212
std	8,660.3984	129,747.6616	0.4891	0.7903	0.522	9.2179	1.1238	1.1972	1.1969	1.1691	1.1332	1.15	73,635.8606	71,173.7688	69,349.3874	64,332.8561	60,797.1558	59,554.1075	16,563.2804	23,040.8704	17,606.9615	15,666.1597	15,278.3057	17,777.4658	0.4151
min	1	10,000	1	0	0	21	-2	-2	-2	-2	-2	-2	-165,580	-69,777	-157,264	-170,000	-81,334	-339,603	0	0	0	0	0	0	0
25%	7,500.75	50,000	1	1	1	28	-1	-1	-1	-1	-1	-1	3,558.75	2,984.75	2,666.25	2,326.75	1,763	1,256	1,000	833	390	296	252.5	117.75	0
50%	15,000.5	140,000	2	2	2	34	0	0	0	0	0	0	22,381.5	21,200	20,088.5	19,052	18,104.5	17,071	2,100	2,009	1,800	1,500	1,500	1,500	0
75%	22,500.25	240,000	2	2	2	41	0	0	0	0	0	0	67,091	64,006.25	60,164.75	54,506	50,190.5	49,198.25	5,006	5,000	4,505	4,013.25	4,031.5	4,000	0
max	30,000	1,000,000	2	6	3	79	8	8	8	8	8	8	964,511	983,931	1,664,089	891,586	927,171	961,664	873,552	1,684,259	896,040	621,000	426,529	528,666	1

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit
count	11,162	11162	11162	11162	11162	11,162	11162	11162	11162	11,162	11162	11,162	11,162	11,162	11,162	11162	11162
unique	None	12	3	4	2	None	2	2	3	None	12	None	None	None	None	4	2
top	None	manag	married	secondary	no	None	no	no	cellular	None	may	None	None	None	None	unknown	no
freq	None	2566	6351	5476	10994	None	5881	9702	8042	None	2824	None	None	None	None	8326	5873
mean	41.2319	nan	nan	nan	nan	1,528.5385	nan	nan	nan	15.658	nan	371.9938	2.5084	51.3304	0.8326	nan	nan
std	11.9134	nan	nan	nan	nan	3,225.4133	nan	nan	nan	8.4207	nan	347.1284	2.7221	108.7583	2.292	nan	nan
min	18	nan	nan	nan	nan	-6,847	nan	nan	nan	1	nan	2	1	-1	0	nan	nan
25%	32	nan	nan	nan	nan	122	nan	nan	nan	8	nan	138	1	-1	0	nan	nan
50%	39	nan	nan	nan	nan	550	nan	nan	nan	15	nan	255	2	-1	0	nan	nan
75%	49	nan	nan	nan	nan	1,708	nan	nan	nan	22	nan	496	3	20.75	1	nan	nan
max	95	nan	nan	nan	nan	81,204	nan	nan	nan	31	nan	3,881	63	854	58	nan	nan

Рисунок 4.2 – Вивід інформації двох датасетів, яка містить кількість непорожніх значень, середнє значення числових значень, стандартне відхилення, максимальне значення.

Наступним кроком необхідно обробити дані. Для датасету, який використовується для прогнозування кредитних ризиків необхідно:

1) Прибрати стовпчик ID, оскільки його дані непотрібні для тренування моделі;
2) Об’єднати значення «0», «4», «5», «6» стовпчика EDUCATION в одну категорію, оскільки вони мають схожі значення між собою («інше», «невідомо»);
3) Значення «0» стовпичка MARRIAGE додати до значення «1».

Для датасету, який використовується для аналізу банківського ринку необхідно:

1) Замінити текстові значення «так» та «ні» на числові «1» та «0» відповідно;
2) Стовпці з текстовими значеннями (категорійні) перетворюємо на фіктивні змінні;
3) Прибираємо нерелевантні стовпці.

Для кращого розуміння датасету і його значень необхідно провести розвідувальний аналіз EDA.

EDA – це підхід до аналізу наборів даних для узагальнення їх основних характеристик. Для нього часто використовуються методи статистичної графіки.

Проведемо розвідувальний аналіз для датасету, який використовується для прогнозу кредитних ризиків. Результати продемонстровано на наступних рисунках.

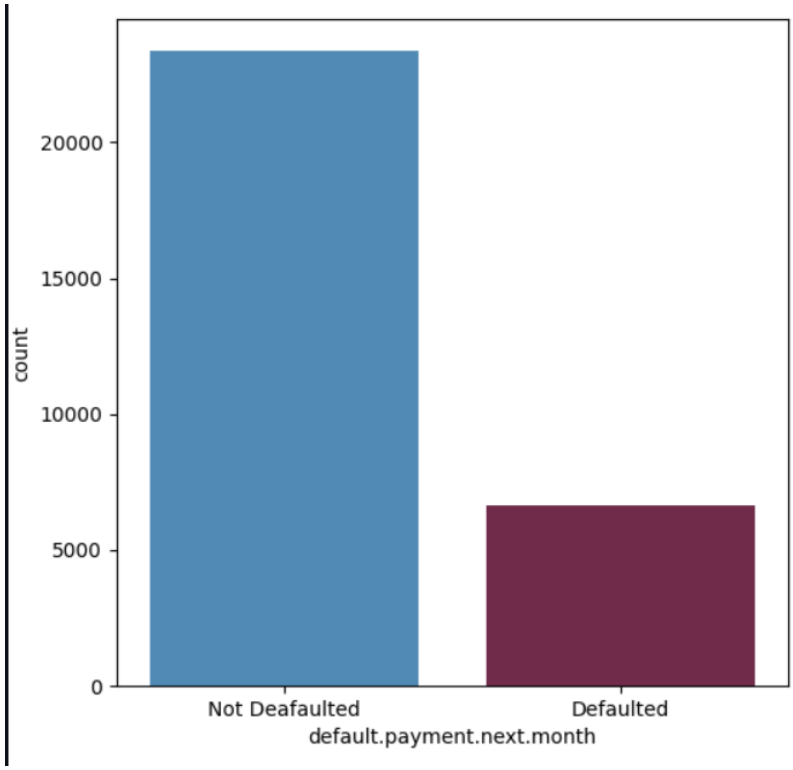


Рисунок 4.3 – Графічне зображення стовпчика default.payment.next.month

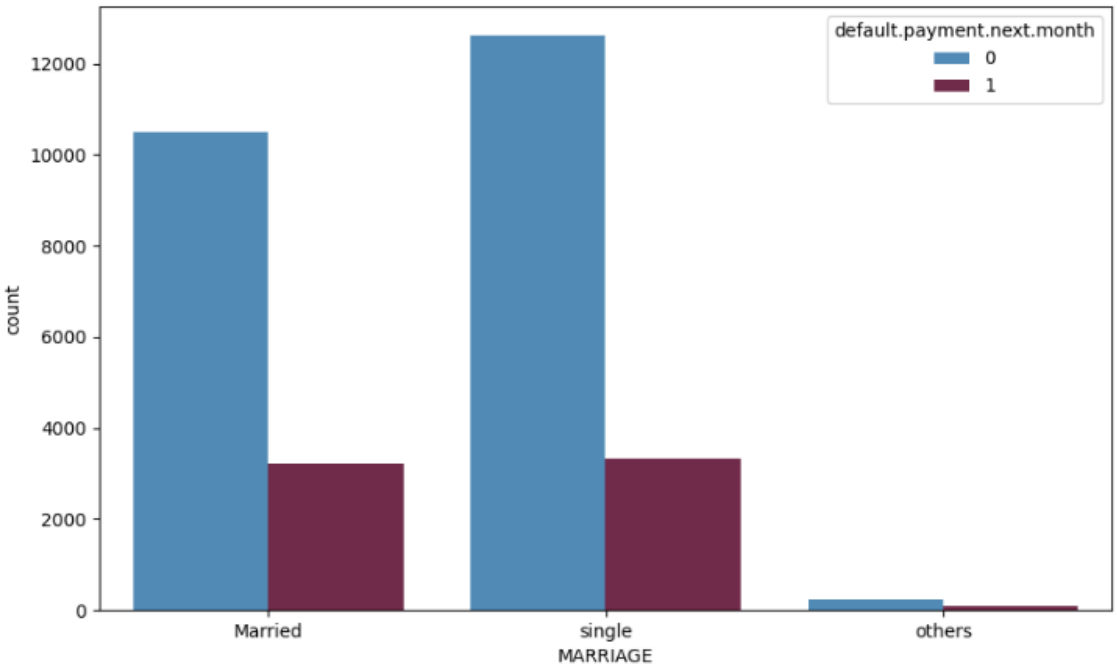


Рисунок 4.4 – Графічне зображення залежності стовпчика MARRIAGE від default.payment.next.month

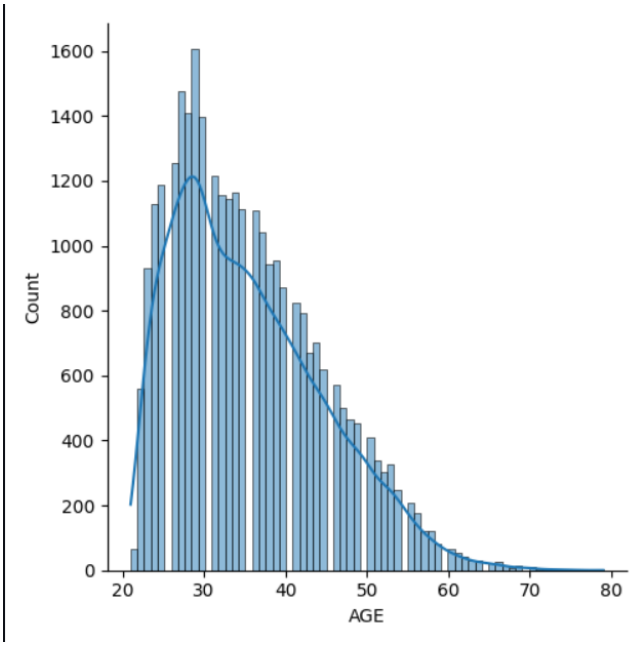


Рисунок 4.5 – Графічне зображення залежності стовпчика AGE

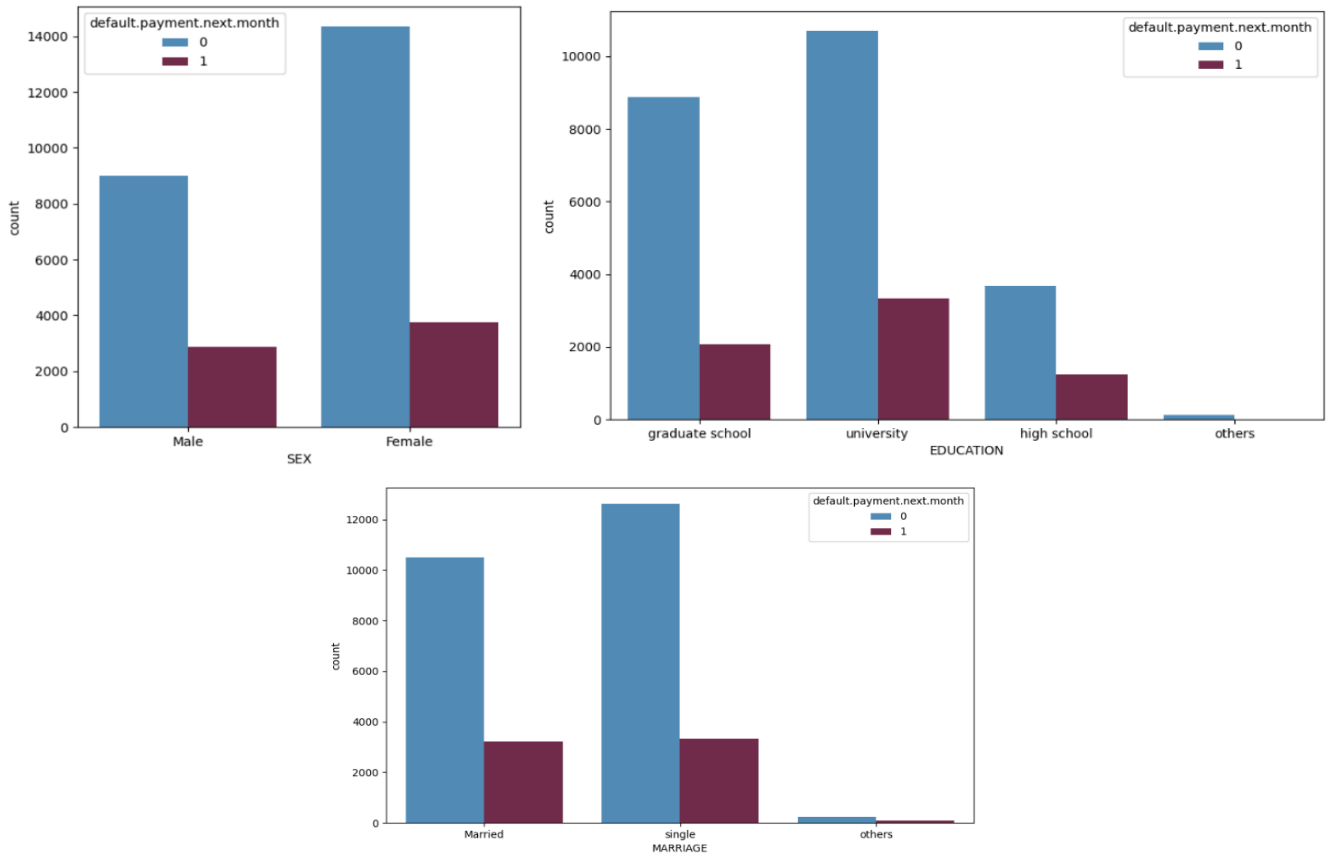


Рисунок 4.6 – Графічне зображення розподілу default.payment.next.month від різних даних, а саме за статтю, освітою та шлюбом

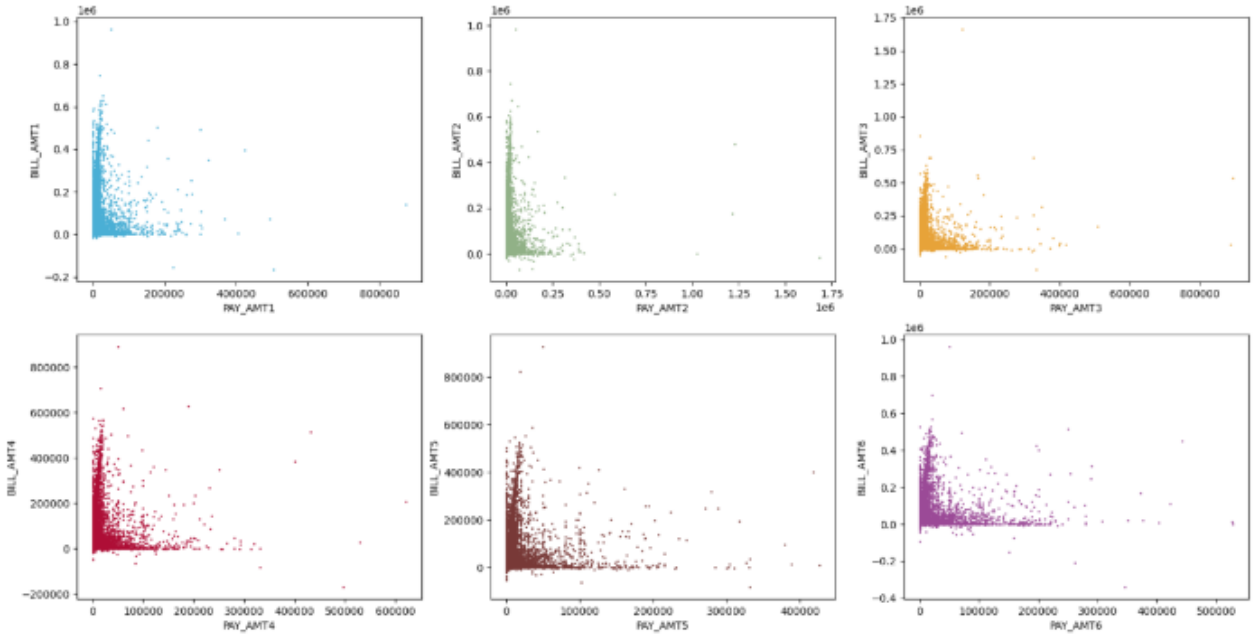


Рисунок 4.7 – Графічне зображення структури платежів у порівнянні з сумою рахунку за останні 6 місяців

Проведемо розвідувальний аналіз для датасету, який використовується для аналізу банківського ринку. Результати продемонстровано на наступних рисунках.

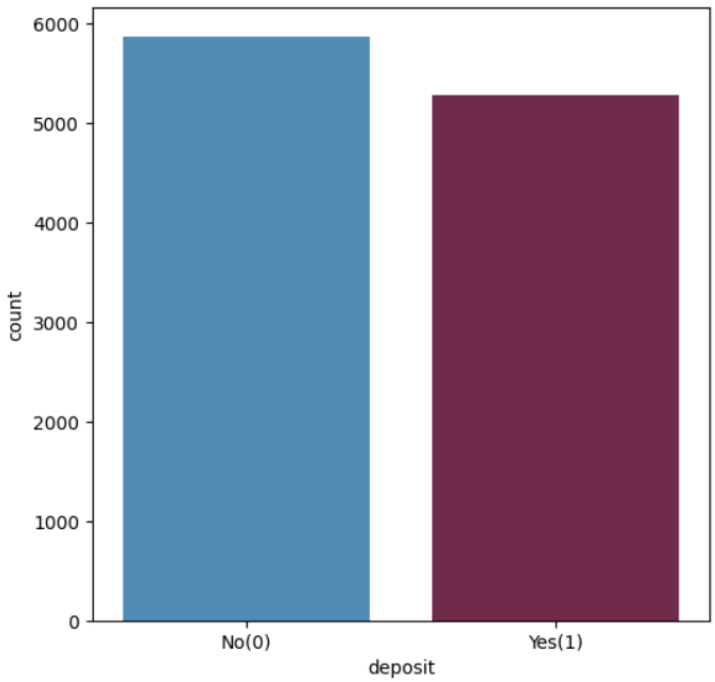


Рисунок 4.8 – Графічне зображення стовпчика deposit

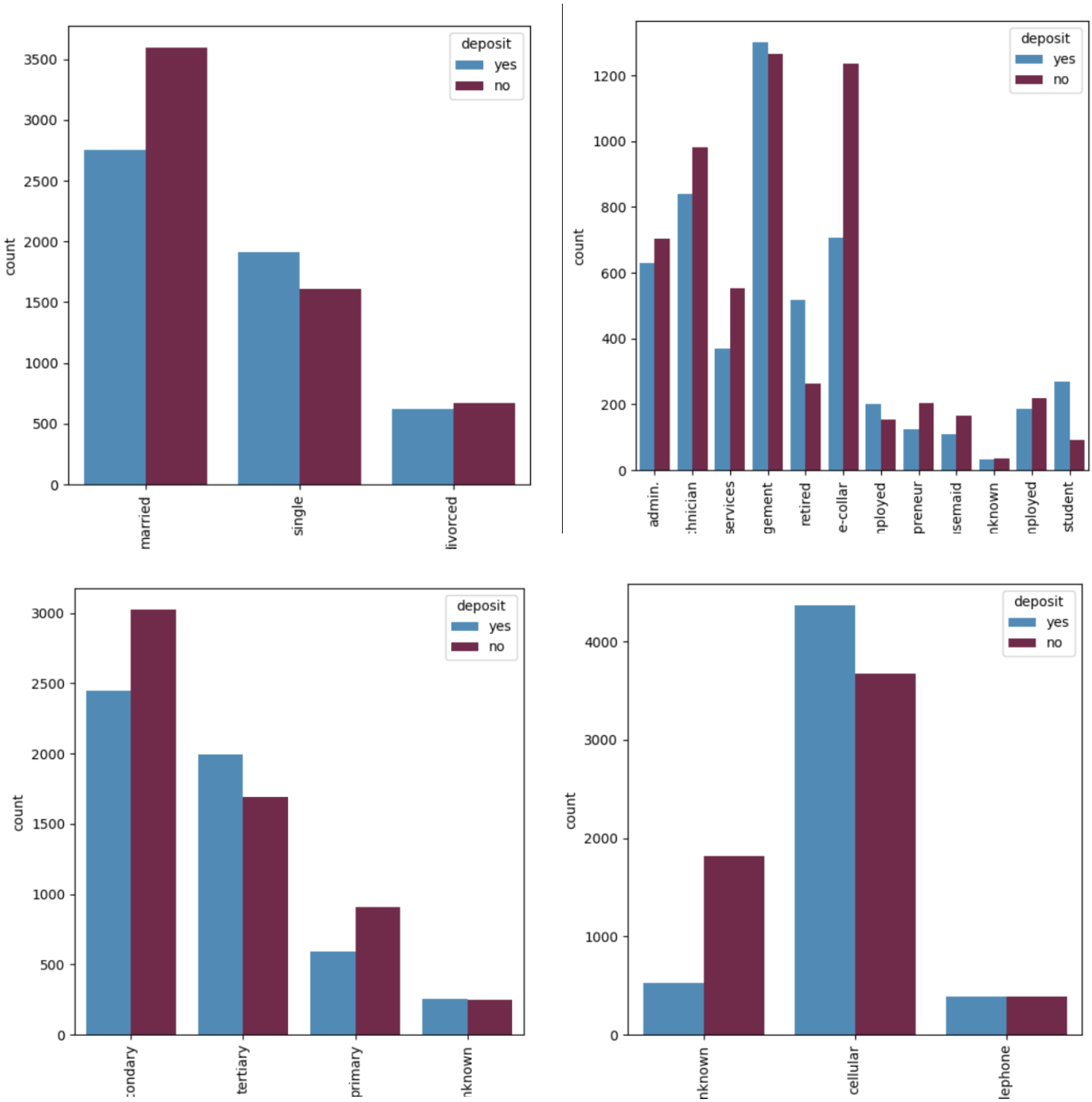


Рисунок 4.9 – Графічне зображення залежностей значення deposit від значень marital, job, education та contact

4.2 Створення незалежних та залежних змінних

Перш ніж стандартизувати змінні необхідно створити незалежні та залежні змінні. В даному контексті під незалежними змінними мається на увазі вхідні дані для процесу, який піддається аналізу, а під залежними змінними мається на увазі вихідні дані процесу.

Для датаесету, який використовується для прогнозу кредитних ризиків залежною змінною є default.payment.next.month, а незалежними є усі інші крім безпосередньо default.payment.next.month.

Для датасету, який використовується для аналізу банківського ринку залежною зміною є deposit_bool, а незалежними є усі інші крім безпосередньо deposit_bool.

4.3 Масштабування та стандартизація змінних

Під час створення моделей необхідно, щоб модель не була зміщена в бік вищого діапазону значень. Для цього використовується масштабування незалежних ознак, щоб зробити всі ознаки в одному діапазоні. Бібліотека sklearn надає можливість використовувати метод StandardScaler() для масштабування ознак [11]. Під час виконання даного методу введеня та масштабування ознак відбуваються незалежно для кожної ознаки. Цей процес має за собою обчислення відповідної стастики для вибірок у навчальній вибірці. При цьому середнє значення та стандартне відхилення зберігаються для використання з подальшими даними за допомогою перетворення.

4.4 Розбиття даних на навчальні та тестові

Після аналізу, обробки та стандартизації даних можна приступати до розбиття їх на навчальні та тестові. Робиться це за допомогою методу train_test_split, який надає бібліотека sklearn. В загальному випадку дані можуть розбиватись у різних співводношеннях, таких як 75 на 25, 70 на 30, 60 на 40 тощо.

Для датаесету, який використовується для прогнозу кредитних ризиків розбиття на навчальні та тестові дані відбувається у співвідношенні 80 на 20.

Для датасету, який використовується для аналізу банківського ринку розбиття на навчальні та тестові дані відбувається у співвідношенні 70 на 30.

4.5 Передискретизація даних

Для передискретизації даних використовується алгоритм SMOTE. Даний алгоритм є одним з найбільш популярних та реалізується за допомогою бібліотеки imblearn. Під час виконання даного алгоритму шляхом створення штучних прикладів клас меншості передискретизується.

Для прикладу успішно проведеної передискретизації даних за допомогою алгоритму SMOTE виведемо кількість даних до та після неї для датасету, який використовується для прогнозування кредитних ризиків.

```
Кількість даних перед передискретизацією:  Counter({0: 18677, 1: 5323})
Кількість даних після передискретизації:  Counter({0: 18677, 1: 18677})
```

Рисунок 4.10 – Вивід створеного програмного застосунку про кількість даних до та після передискретизації даних

4.6 Розробка моделі за допомогою логістичної регресії

Для розробки моделі за допомогою логістичної регресії використовується бібліотека sklearn, а саме метод LogisticRegression(). За замовчуванням гіперпараметри дорівнюють наступним значенням:

1) “tol” = 0.0001
2) “penalty” = “l2”
3) “solver” = “lbfgs”
4) “multi_class” = “auto”

Для логістичної регресії дані значення створюють найкращий результат. Наступним кроком є побудова екземпляра моделі і навчання самої моделі на тренувальних даних.

Після тренування моделей необхідно оцінити їх згідно наступних параметрів: precision (accuracy, macro average, weighted average), recall (accuracy, macro average, weighted average), f1-score (accuracy, macro average, weighted average), support (accuracy, macro average, weighted average). Також необхідно побудувати матрицю невідповідностей та графіки Precion-Recall й AUC-ROC.

Отже, для прогнозування кредитних ризиків було побудовано логістичну регресію з точністю створеної моделі 0.68583. Даний результат не дуже задовольняє поставлену умову, що точність має бути не менше 70%, що робить даний метод не зовсім якісним для прогнозування кредитних ризиків. Інші оцінювальні метрики та графіки представлено на рисунках 4.11-4.13. Покращити точність можна за допомогою більшої кількості даних для тренування моделі.

	precision	recall	f1-score	support
0	0.8784	0.6938	0.7753	4,687
1	0.3755	0.6573	0.478	1,313
accuracy	0.6858	0.6858	0.6858	0.6858
macro avg	0.627	0.6756	0.6266	6,000
weighted avg	0.7684	0.6858	0.7102	6,000

Рисунок 4.11 – Вивід таблиці значень з оцінювальними метриками створеної моделі

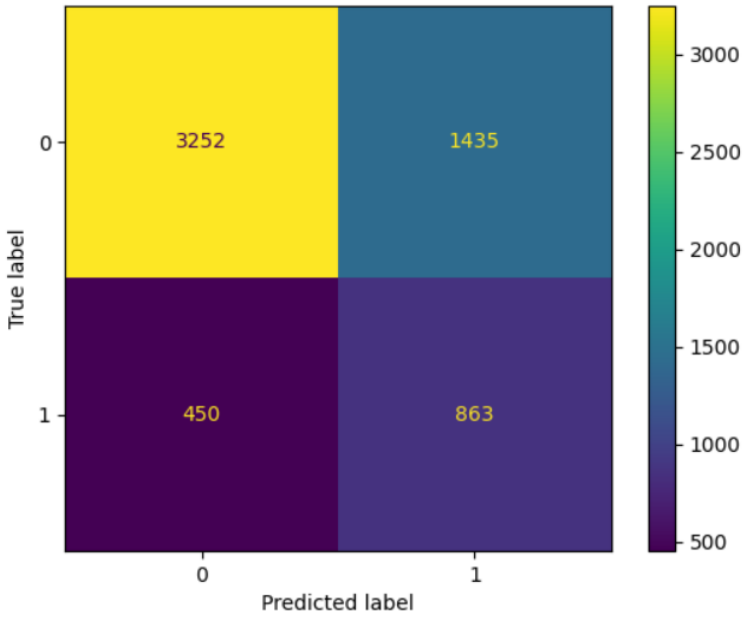


Рисунок 4.12 – Графічне зображення матриці невідповідностей

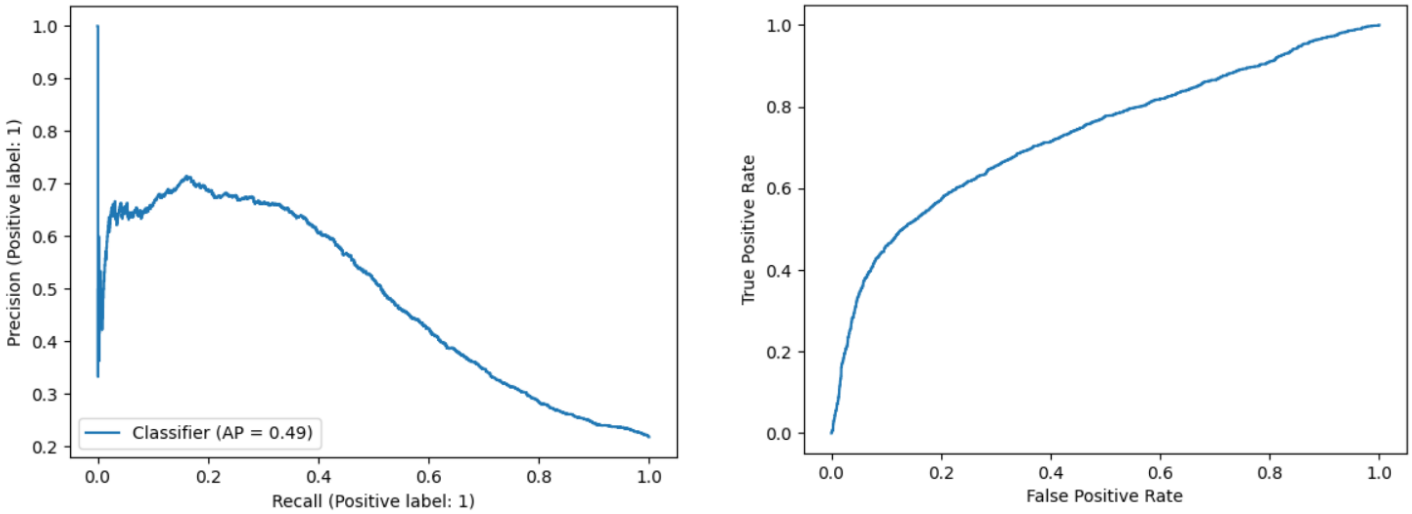


Рисунок 4.13 – Графіки Precision-Recall та AUC-ROC

Для аналізу банківського ринку було побудовано логістичну регресію з точністю створеної моделі 0.7871. Даний результат задовольняє поставлену умову, що точність має бути не менше 70%. Інші оцінювальні метрики та графіки представлено на рисунках 4.14-4.16.

	precision	recall	f1-score	support
0	0.7768	0.8289	0.802	1,742
1	0.8	0.7418	0.7698	1,607
accuracy	0.7871	0.7871	0.7871	0.7871
macro avg	0.7884	0.7853	0.7859	3,349
weighted avg	0.7879	0.7871	0.7865	3,349

Рисунок 4.14 – Вивід таблиці значень з оцінювальними метриками створеної моделі

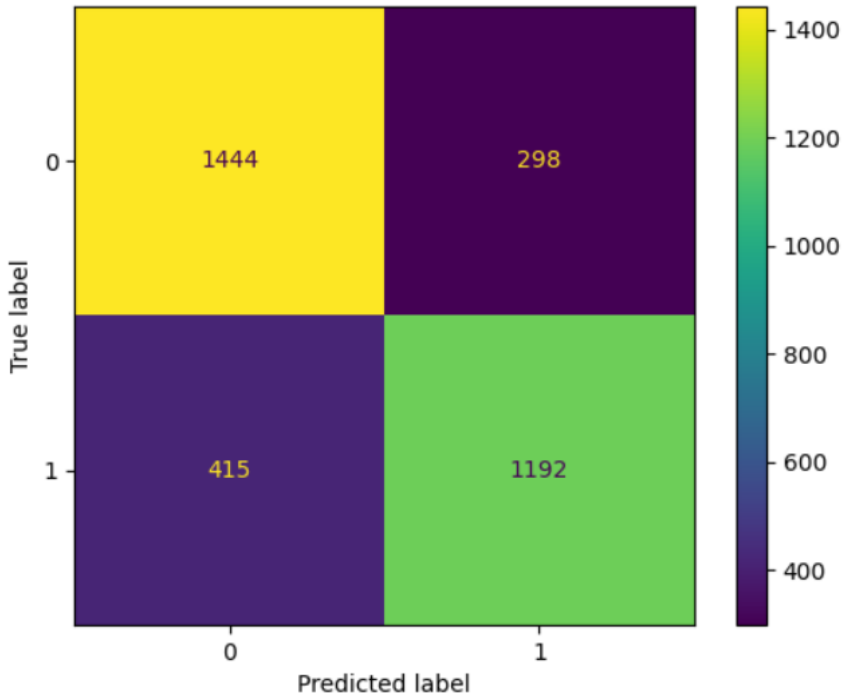


Рисунок 4.15 – Графічне зображення матриці невідповідностей

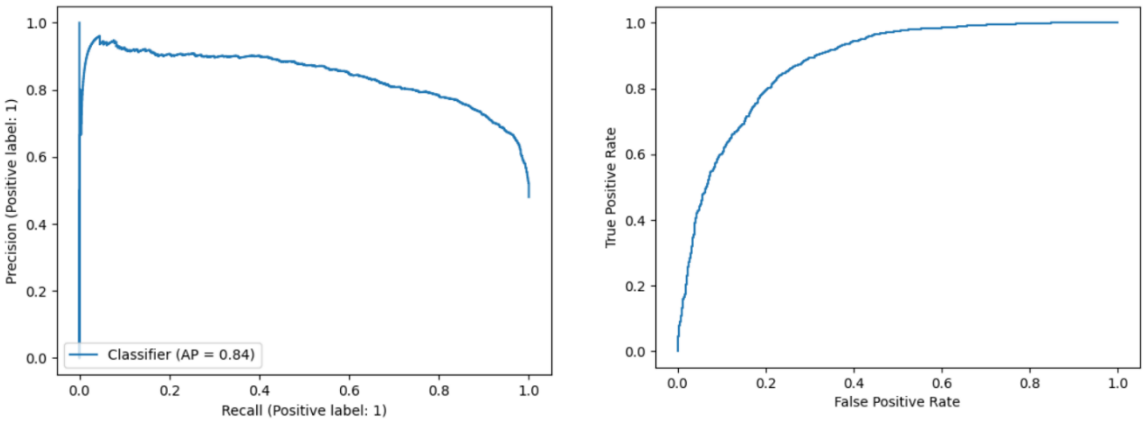


Рисунок 4.16 – Графіки Precision-Recall та AUC-ROC

4.7 Розробка моделі за допомогою класифікатора випадкових лісів

Для розробки моделі за допомогою логістичної регресії використовується бібліотека sklearn, а саме метод RandomForestClassifier(). За замовчуванням гіперпараметри дорівнюють наступним значенням:

1) “criterion” = “gini”,
2) “min_samples_split” = 2,
3) “min_samples_leaf” = 1,
4) “max_features” = “sqrt”.

Для класифікатора випадкових лісів дані значення створюють найкращий результат. Наступним кроком є побудова екземпляра моделі і навчання самої моделі на тренувальних даних.

Після тренування моделей необхідно оцінити їх згідно наступних параметрів: precision (accuracy, macro average, weighted average), recall (accuracy, macro average, weighted average), f1-score (accuracy, macro average, weighted average), support (accuracy, macro average, weighted average). Також необхідно побудувати матрицю невідповідностей та графіки Precion-Recall й AUC-ROC.

Отже, для прогнозування кредитних ризиків було побудовано класифікатор випадкових лісів з точністю створеної моделі 0.7917. Даний результат повністю задовольняє поставлену умову, що точність має бути не менше 70%. Інші оцінювальні метрики та графіки представлено на рисунках 4.17-4.19. Покращити точність можна за допомогою більшої кількості даних для тренування моделі.

	precision	recall	f1-score	support
0	0.8541	0.8844	0.869	4,687
1	0.5275	0.4608	0.4919	1,313
accuracy	0.7917	0.7917	0.7917	0.7917
macro avg	0.6908	0.6726	0.6804	6,000
weighted avg	0.7826	0.7917	0.7865	6,000

Рисунок 4.17 – Вивід таблиці значень з оцінювальними метриками створеної моделі

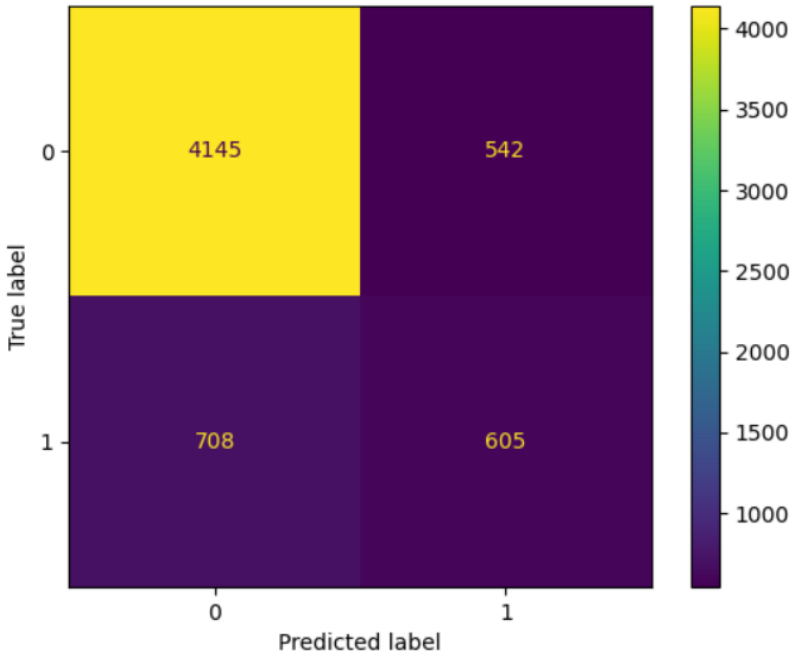


Рисунок 4.18 – Графічне зображення матриці невідповідностей

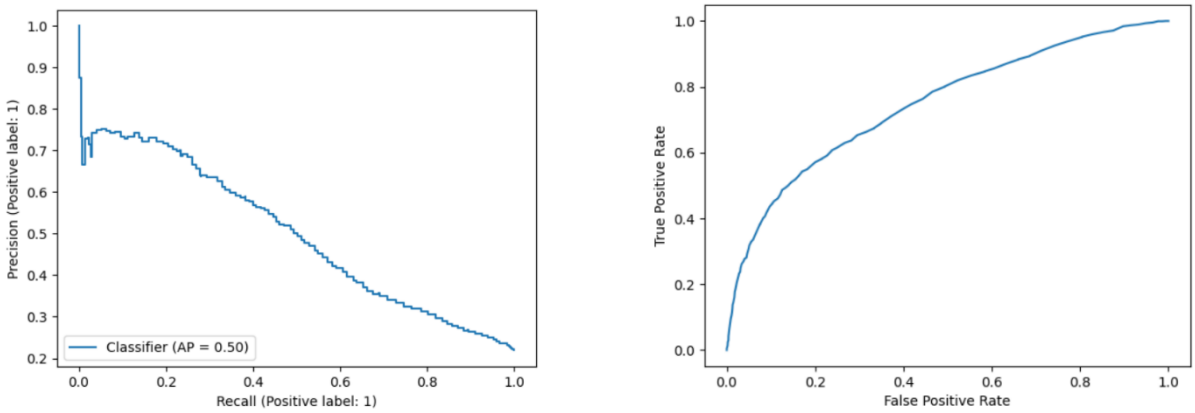


Рисунок 4.19 – Графіки Precision-Recall та AUC-ROC

Для аналізу банківського ринку було побудовано класифікатор випадкових лісів з точністю створеної моделі 0.8456. Даний результат задовольняє поставлену умову, що точність має бути не менше 70%. Інші оцінювальні метрики та графіки представлено на рисунках 4.20-4.22.

	precision	recall	f1-score	support
0	0.8719	0.8243	0.8474	1,742
1	0.8202	0.8687	0.8438	1,607
accuracy	0.8456	0.8456	0.8456	0.8456
macro avg	0.846	0.8465	0.8456	3,349
weighted avg	0.8471	0.8456	0.8457	3,349

Рисунок 4.20 – Вивід таблиці значень з оцінювальними метриками створеної моделі

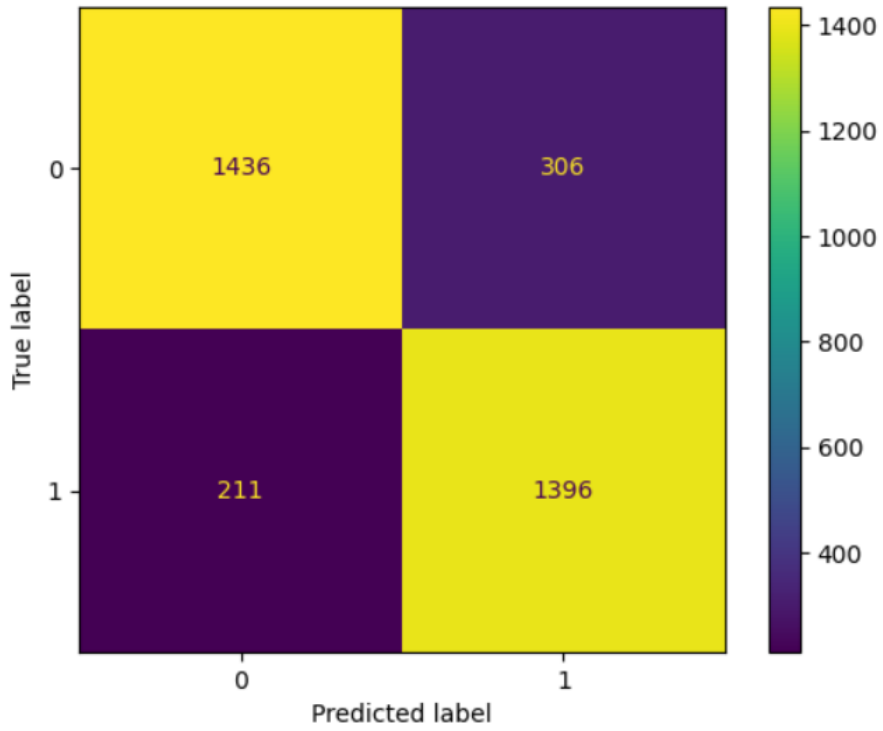


Рисунок 4.21 – Графічне зображення матриці невідповідностей

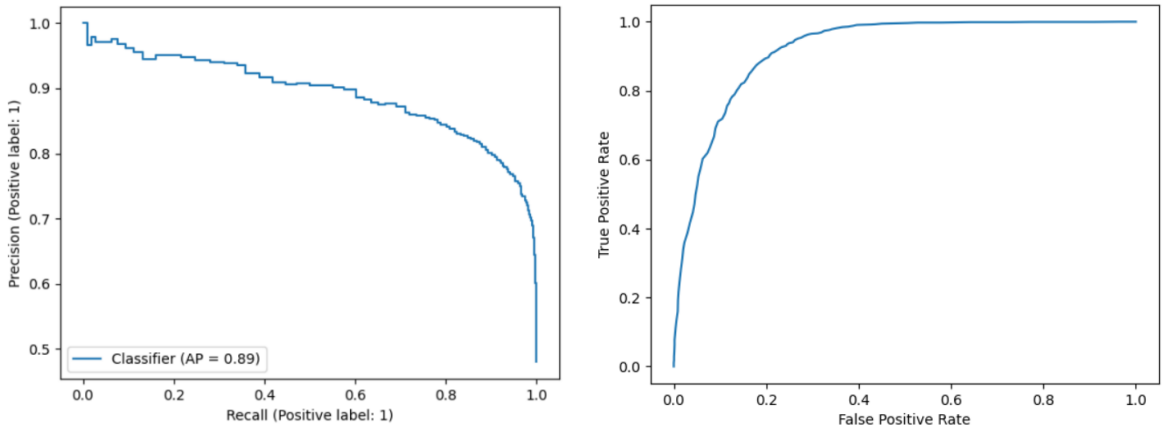


Рисунок 4.22 – Графіки Precision-Recall та AUC-ROC

4.8 Розробка моделі за допомогою класифікатора XGBoost

Для розробки моделі за допомогою логістичної регресії використовується бібліотека sklearn, а саме метод RandomForestClassifier(). За допомогою нього

відбувається побудова екземпляра моделі і навчання самої моделі на тренувальних даних.

Після тренування моделей необхідно оцінити їх згідно наступних параметрів: precision (accuracy, macro average, weighted average), recall (accuracy, macro average, weighted average), f1-score (accuracy, macro average, weighted average), support (accuracy, macro average, weighted average). Також необхідно побудувати матрицю невідповідностей та графіки Precion-Recall й AUC-ROC.

Отже, для прогнозування кредитних ризиків було побудовано класифікатор XGBoost з точністю створеної моделі 0.8123. Даний результат повністю задовольняє поставлену умову, що точність має бути не менше 70%. Інші оцінювальні метрики та графіки представлено на рисунках 4.23-4.25.

	precision	recall	f1-score	support
0	0.8453	0.93	0.8856	4,687
1	0.6109	0.3922	0.4777	1,313
accuracy	0.8123	0.8123	0.8123	0.8123
macro avg	0.7281	0.6611	0.6817	6,000
weighted avg	0.794	0.8123	0.7964	6,000

Рисунок 4.23 – Вивід таблиці значень з оцінювальними метриками створеної моделі

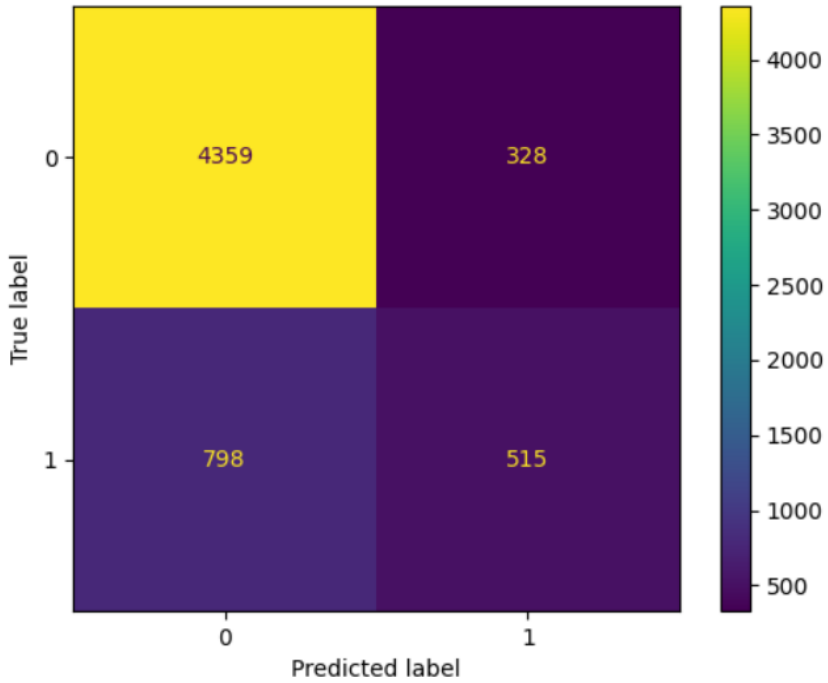


Рисунок 4.24 – Графічне зображення матриці невідповідностей

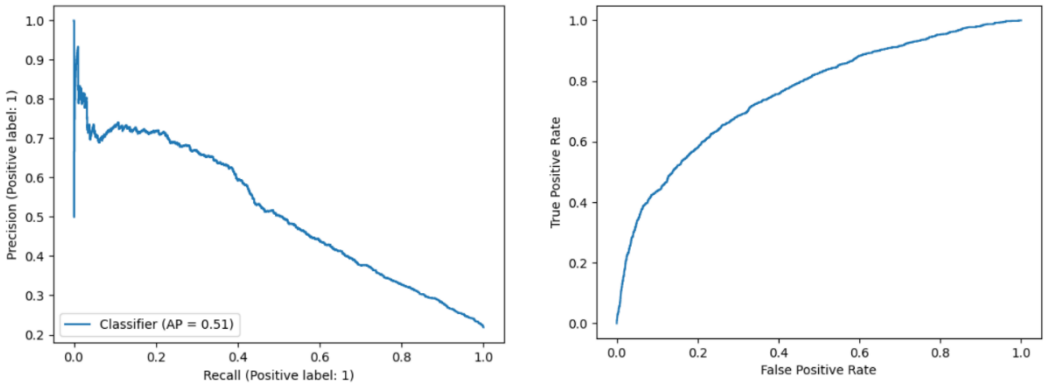


Рисунок 4.25 – Графіки Precision-Recall та AUC-ROC

Для аналізу банківського ринку було побудовано класифікатор XGBoost з точністю створеної моделі 0.8501. Даний результат задовольняє поставлену умову, що точність має бути не менше 70%. Інші оцінювальні метрики та графіки представлено на рисунках 4.26-4.28.

	precision	recall	f1-score	support
0	0.8794	0.8249	0.8513	1,742
1	0.8222	0.8774	0.8489	1,607
accuracy	0.8501	0.8501	0.8501	0.8501
macro avg	0.8508	0.8512	0.8501	3,349
weighted avg	0.852	0.8501	0.8501	3,349

Рисунок 4.26 – Вивід таблиці значень з оцінювальними метриками створеної моделі

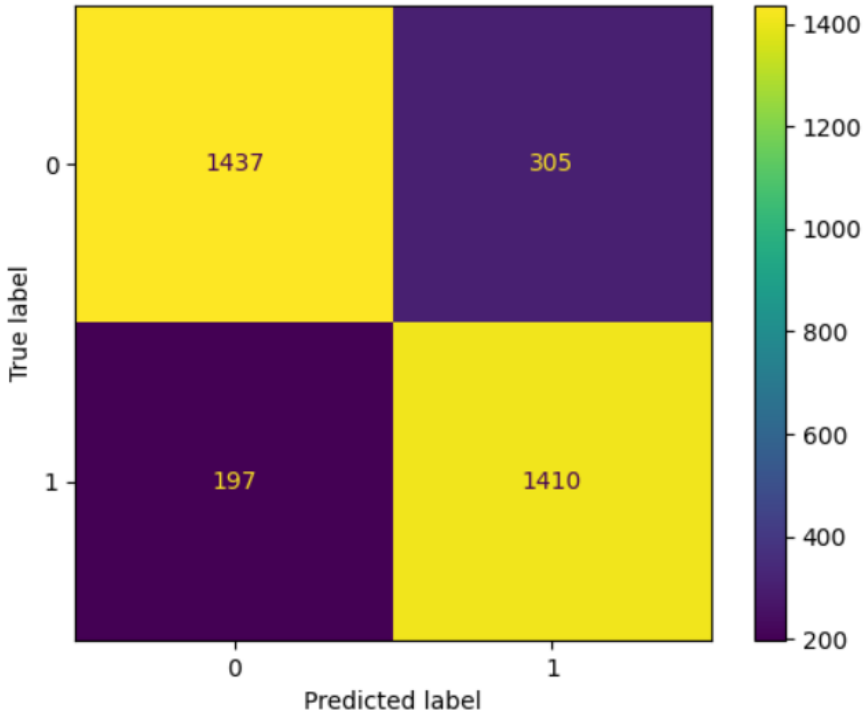


Рисунок 4.27 – Графічне зображення матриці невідповідностей

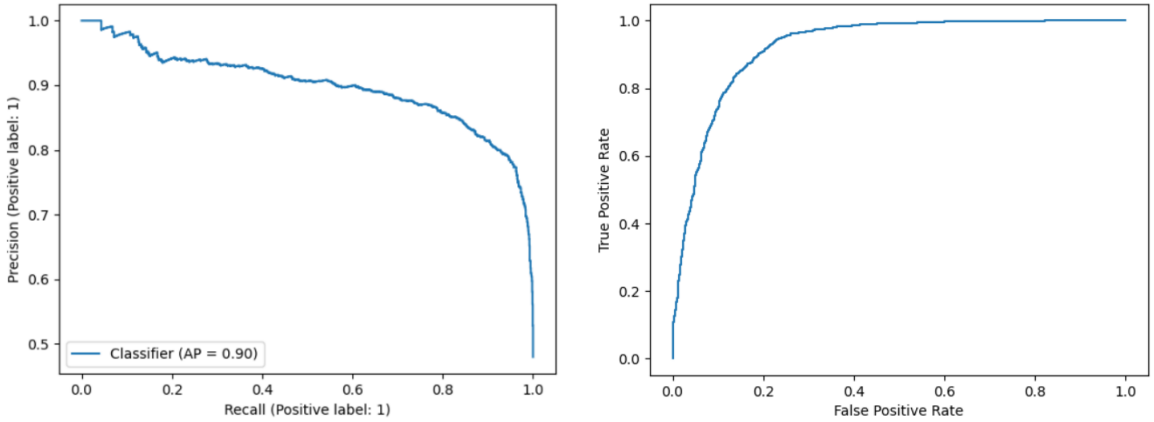


Рисунок 4.28 – Графіки Precision-Recall та AUC-ROC

4.9 Розробка моделі за допомогою класифікатора дерева рішень

Для розробки моделі за допомогою логістичної регресії використовується бібліотека sklearn, а саме метод DecisionTreeClassifier(). За замовчуванням гіперпараметри дорівнюють наступним значенням:

5) “criterion” = “gini”,
6) “criterion” = “best”,
7) “min_samples_split” = 2,
8) “min_samples_leaf” = 1.

Для класифікатора дерева рішень дані значення створюють найкращий результат. Наступним кроком є побудова екземпляра моделі і навчання самої моделі на тренувальних даних.

Після тренування моделей необхідно оцінити їх згідно наступних параметрів: precision (accuracy, macro average, weighted average), recall (accuracy, macro average, weighted average), f1-score (accuracy, macro average, weighted average), support (accuracy, macro average, weighted average). Також необхідно побудувати матрицю невідповідностей та графіки Precion-Recall й AUC-ROC.

Отже, для прогнозування кредитних ризиків було побудовано класифікатор дерева рішень з точністю створеної моделі 0.7043. Даний результат достатньо задовольняє поставлену умову, що точність має бути не менше 70%. Інші оцінювальні метрики та графіки представлено на рисунках 4.29-4.31. Покращити точність можна за допомогою більшої кількості даних для тренування моделі.

	precision	recall	f1-score	support
0	0.8358	0.7734	0.8034	4,687
1	0.3614	0.4577	0.4039	1,313
accuracy	0.7043	0.7043	0.7043	0.7043
macro avg	0.5986	0.6156	0.6037	6,000
weighted avg	0.732	0.7043	0.716	6,000

Рисунок 4.29 – Вивід таблиці значень з оцінювальними метриками створеної моделі

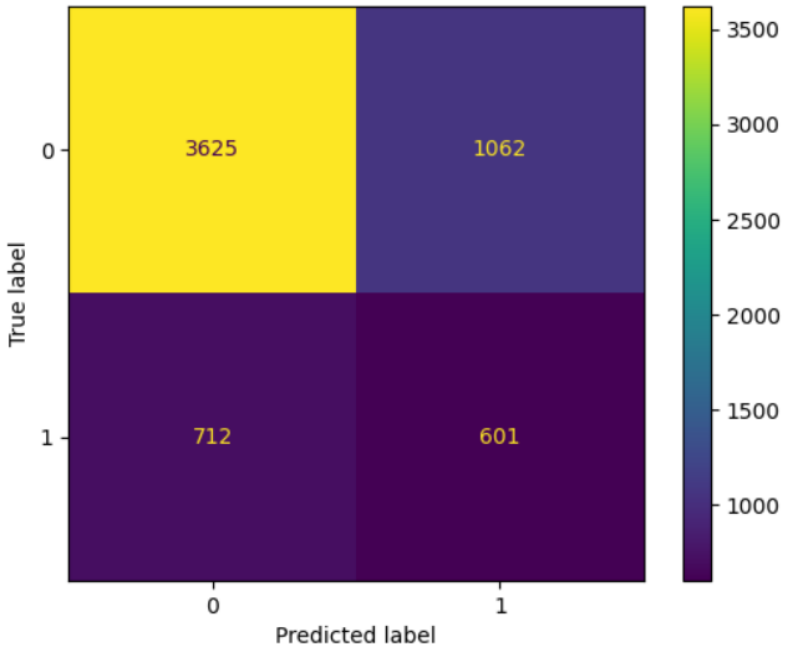


Рисунок 4.30 – Графічне зображення матриці невідповідностей

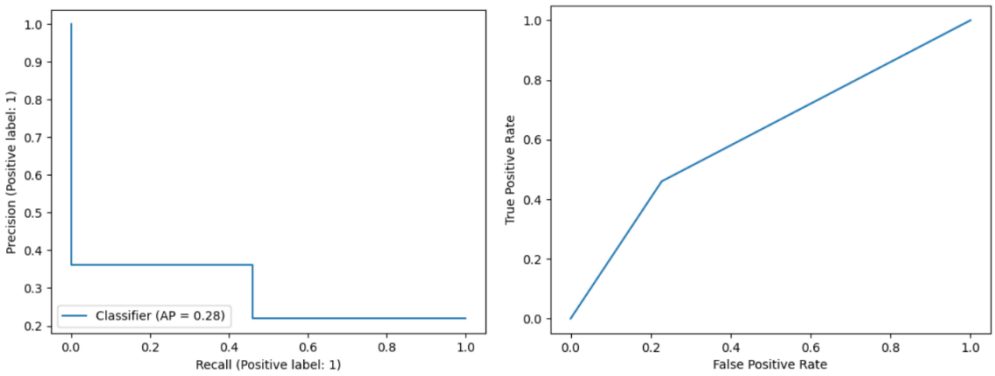


Рисунок 4.31 – Графіки Precision- Recall та AUC-ROC

Для аналізу банківського ринку було побудовано класифікатор дерева рішень з точністю створеної моделі 0.7847. Даний результат задовольняє поставлену умову, що точність має бути не менше 70%. Інші оцінювальні метрики та графіки представлено на рисунках 4.32-4.34.

	precision	recall	f1-score	support
0	0.7892	0.7997	0.7944	1,742
1	0.7797	0.7685	0.7741	1,607
accuracy	0.7847	0.7847	0.7847	0.7847
macro avg	0.7845	0.7841	0.7842	3,349
weighted avg	0.7846	0.7847	0.7846	3,349

Рисунок 4.32 – Вивід таблиці значень з оцінювальними метриками створеної моделі

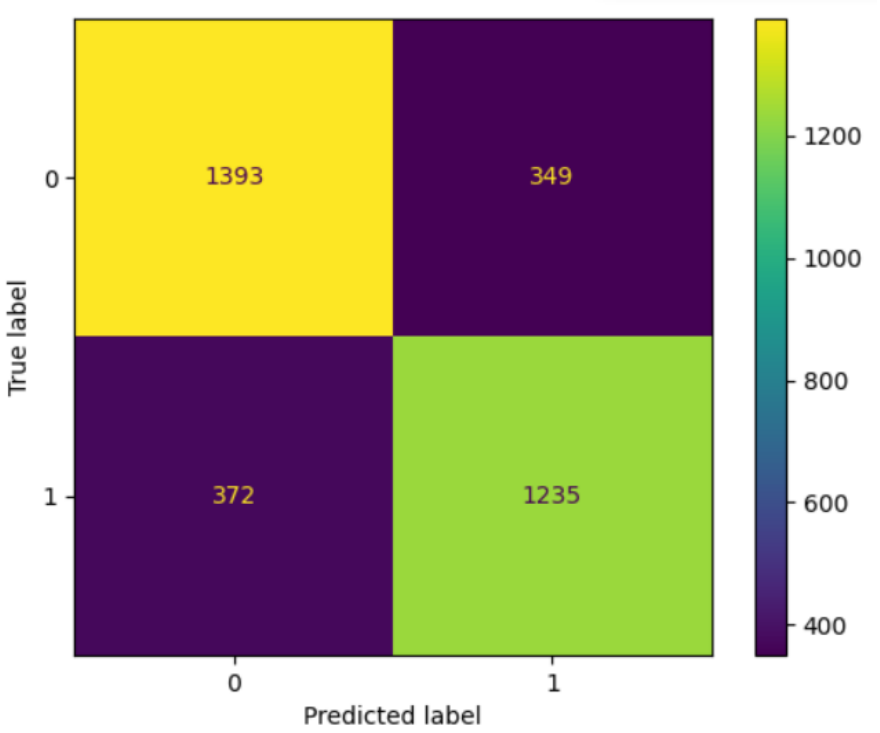


Рисунок 4.33 – Графічне зображення матриці невідповідностей

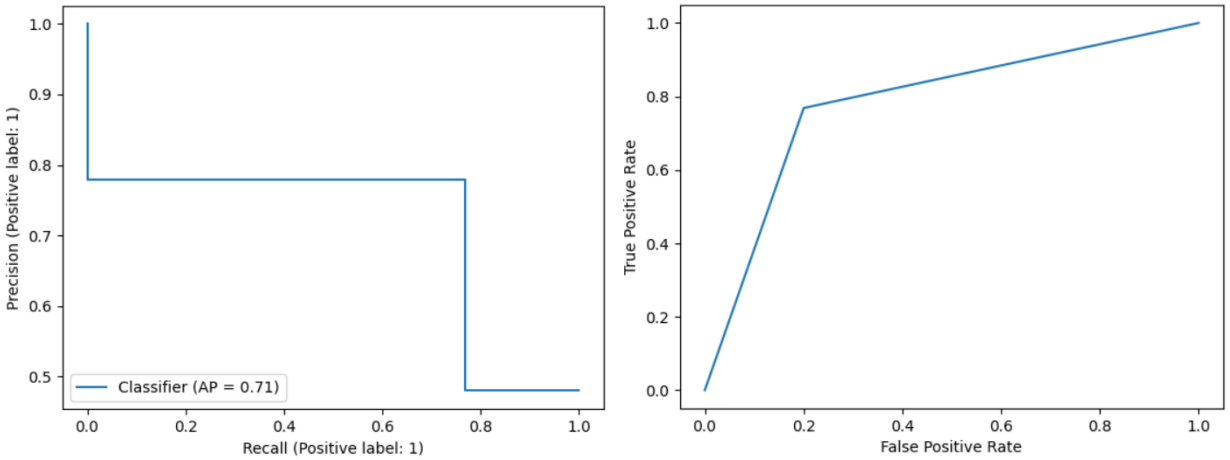


Рисунок 4.34 – Графіки Precision-Recall та AUC-ROC

4.10 Приклад аналізу залежності впливу

За допомогою створених моделей можна проаналізувати, які критерії найбільше впливають на результат. Проведемо аналіз залежності впливу для аналізу

банківського ринку. Для цього візьмемо найкраще створену модель, тобто модель створену за допомогою класифікатора XGBoost. З неї отримаємо графік важливостей критеріїв, що зображений на рисунку 4.35.

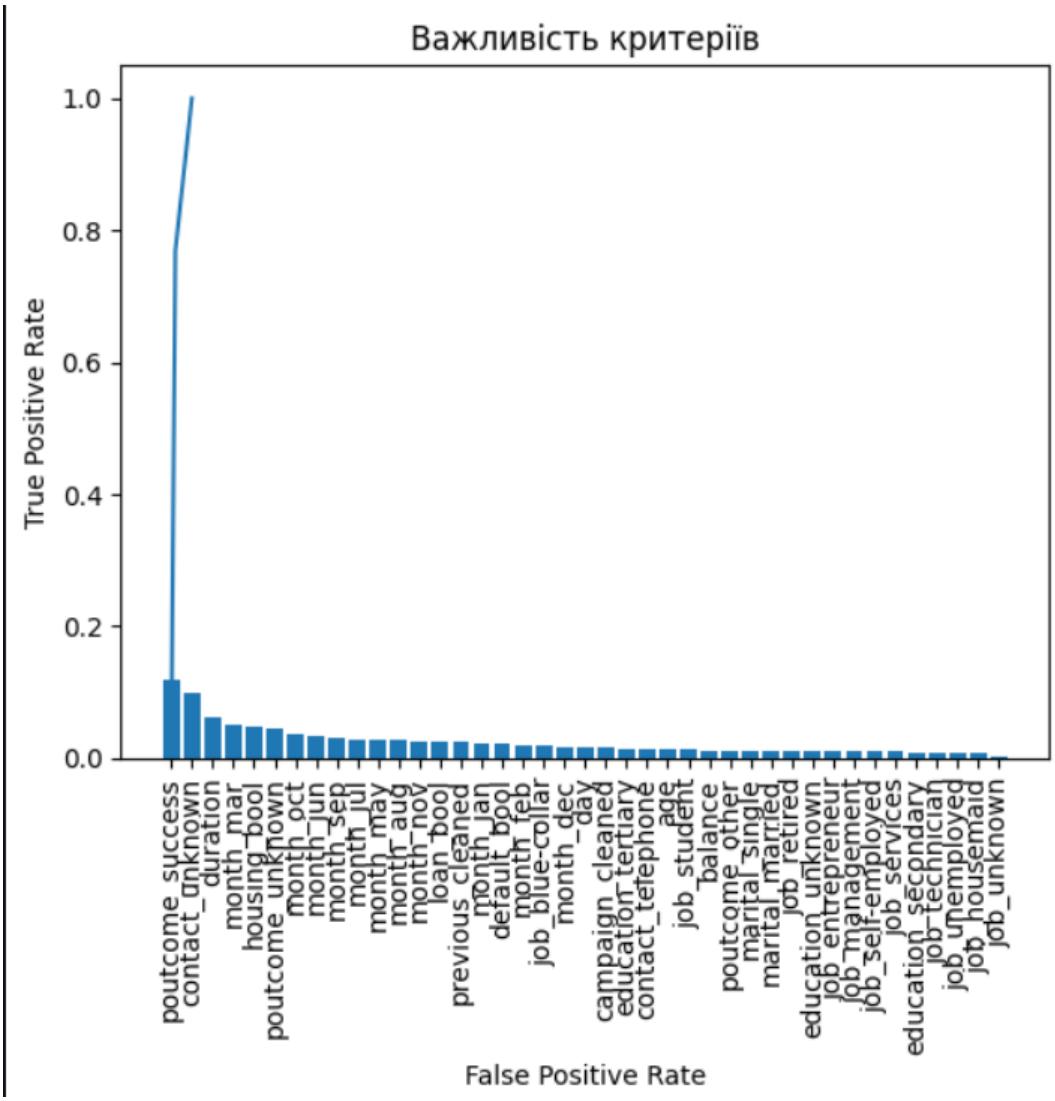


Рисунок 4.35 – Графік важливості критеріїв

З нього можна отримати інформацію про те, що кампанія, яка проводилась в березні, була найуспішнішою та найвпливовішою. Також згідно даного графіку можна зазначити, що серед переліку критеріїв є такі значення як «age» та «balance». Отже проаналізуємо результати кожного з них.

Для критерію «age» необхідно побудувати графік співвідношень важливості до відсотку укладеного строкового депозиту. Отриманий графік продемонстровано на

рисунку 4.36. Проаналізувавши його можемо зазначити, що найбільших відсоток йде в проміжку $[0, 3]$ та $[17, \infty]$. Даним проміжкам відповідає вік 31 та 56 років. Отже з цього робимо висновок, що пропонувати відкриття депозитів краще людям віком молодше 31 років та старше 56 років.

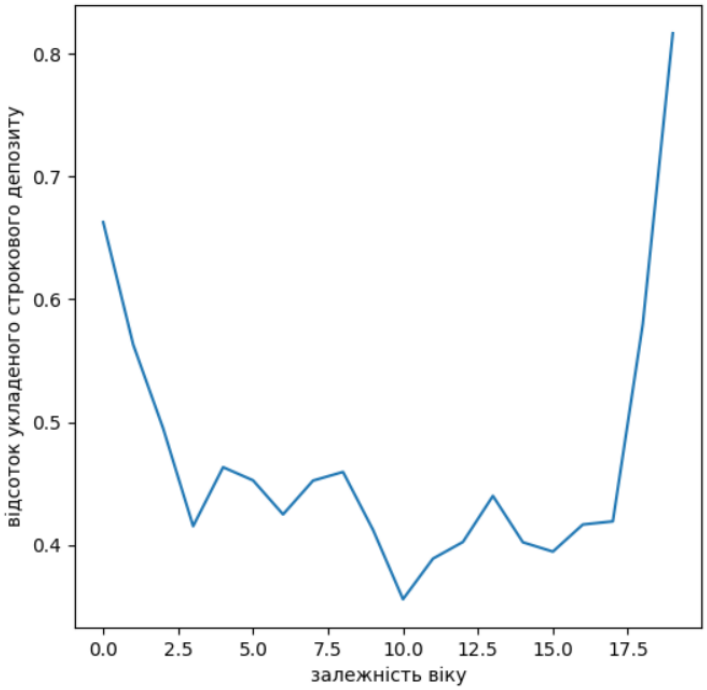


Рисунок 4.36 – Графік відношення залежності віку до відсотку укладеного строкового депозиту

Так само, як і для критерію «age» проведемо аналіз критерію «balance». Отриманий графік продемонстровано на рисунку 4.37. Проаналізувавши його можемо зазначити, що найбільший відсоток починається зі значення 34. Даному значення відповідає сума в 1490 євро. Отже з цього робимо висновок, що пропонувати відкриття депозитів краще людям з балансом більшим за 1490 євро.

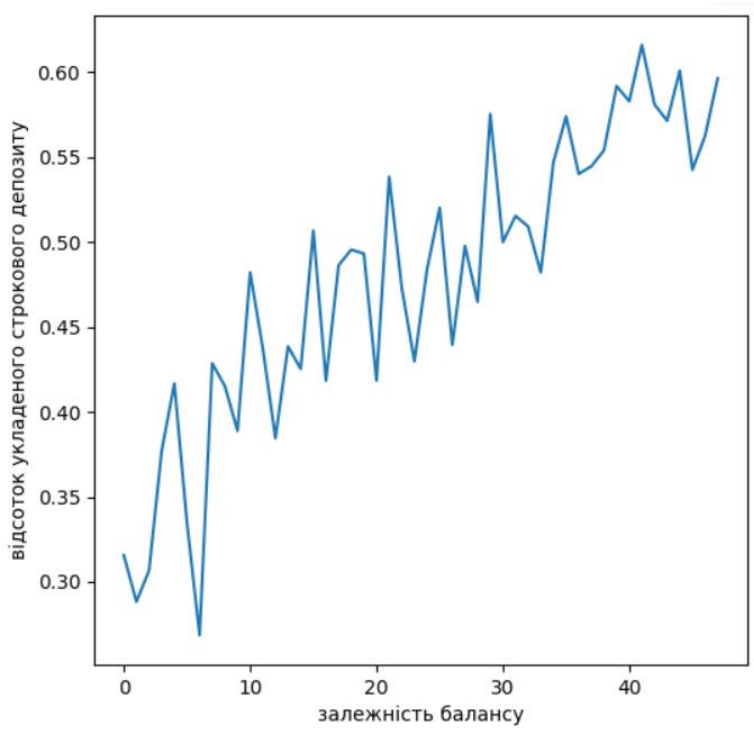


Рисунок 4.37 – Графік відношення залежності балансу до відсотку укладеного строкового депозиту

4.11 Висновки до розділу

В даному розділі було детально описано програмне забезпечення для створення підсистеми аналізу та прогнозування відкриття банківських депозитів та кредитів. Для підготовки даних, їх аналізу та виводі графіків була використана бібліотека Pandas. Для створення моделей для прогнозу кредитних ризиків та аналізу банківського ринку використовувалася бібліотека sklearn. Для створення графічного інтерфейсу було обрано бібліотеку Streamlit. Кожна з обраних бібліотек дає велику можливість для роботи з даними.

В якості методів машинного навчання було розглянуто такі методи: логістична регресія, класифікатор випадкових лісів, класифікатор XGBoost та класифікатор дерева рішень. Найкраще з усіх обраних методів для порівняння показав себе класифікатор XGBoost. Найгірше ж з усіх себе проявила логістична регресія.

Для прогнозування кредитних ризиків було створено модель з максимальною отриманою точністю 0.8123, що задовольняє умову створення моделі з мінімальною точністю у 70%.

Для аналізу банківського ринку було створено модель з максимальною отриманою точністю 0.8501, що задовольняє умову створення моделі з мінімальною точністю у 70%.

5 ВЕРИФІКАЦІЯ ТА ВАЛІДАЦІЯ

5.1 Верифікація

Під час роботи над БАР було створено підсистему для аналізу та прогнозування відкриття банківських депозитів та кредитів. Перед початком аналізу методів машинного навчання, які використовувались при розв’язанні, було проаналізовано існуючі рішення. Після цього було обрано чотири методи машинного навчання для подальшого аналізу та реалізації, а саме логістична регресія, класифікатор випадкових лісів, класифікатор XGBoost та класифікатор дерева рішень. Результати програмних рішень задовольняють поставлену умову про точність не нижчу за 70%. Найкращим методом машинного навчання було встановлено класифікатор XGBoost. Найгіршим методом машинного навчання було встановлено логістичну регресію

5.2 Валідація

Під час роботи над БАР було створено підсистему для аналізу та прогнозування відкриття банківських депозитів та кредитів. Створена підсистема відповідає зазначеній умові про мінімальну точність у 70%. Для прогнозування кредитних ризиків максимальна точність становить 81.23%, а для аналізу банківського ринку 85.01%. Обидва показники є достатньо хорошими і здобуті за допомогою класифікатора XGBoost, що є методом машинного навчання.

ВИСНОВКИ

Як результат виконаної роботи:

а) було розглянуто основні програмні реалізації аналізу та прогнозування відкриття банківських депозитів та кредитів. З них було виділено головні недоліки існуючих рішень, а саме відсутність прозорості розрахунків та тренуванні, відсутність повної звітності, не завжди висока точність передбачень;

б) було розглянуто методи для аналізу та прогнозування відкриття банківських депозитів та кредитів: кластерний аналіз, штучні нейронні мережі, моделі Маркова, методи машинного навчання. У результаті проведеного порівняльного аналізу для вирішення поставленої задачі було обрано використання методів машинного навчання, а саме логістична регресія, класифікатор випадкових лісів, класифікатор XGBoost, класифікатор дерева рішень;

в) розроблено математичне забезпечення, яке включає в себе масштабування та стандартизацію ознак, передискретизацію даних, методи машинного навчання, а саме логістичну регресію, класифікатор випадкових лісів, класифікатор XGBoost, класифікатор дерева рішень. Також розроблено математичне забезпечення для перевірки показників точності створеної підсистеми;

г) розроблено програмне забезпечення підсистеми аналізу та прогнозування відкриття банківських депозитів та кредитів;

д) проведено верифікацію та валідацію створеної підсистеми, під час чого було встановлено відповідність поставленим вимогам, а саме отримання точності створеної підсистеми не менше ніж 70%.

ПЕРЕЛІК ПОСИЛАНЬ

1. Banks Credit Risk Prediction with Optimized ANN Based on Improved Owl Search Algorithm / [P. Sharifi, V. Jain, A. P. Mehdi та ін.]. // Hindawi. – 2021.

2. Kumar Gupta D. Credit Risk Prediction Using Artificial Neural Network Algorithm / D. Kumar Gupta, G. Shruti. // MECS. – 2018. – №5. – С. 9–16.

3. Moody’s Analytics [Електронний ресурс] – Режим доступу до ресурсу: https://www.moodysanalytics.com/.

4. Alteryx Analytics Cloud Platform [Електронний ресурс] – Режим доступу до ресурсу: https://www.alteryx.com/products/alteryx-cloud.

5. G Predictive Analysis on Bank Marketing Campaign Using Machine Learning Algorithms / M.Chenna Keshava, K. Sai Chandana, K. Sai Sagar, M. Naga Harish. // International Journal for Research in Applied Science & Engineering Technology. – 2019. – №7. – С. 266–271.

6. A Brief Review of Nearest Neighbor Algorithm for Learning and Classification, / T.Kashvi, S. De, S. Verma, A. Swetapadma. // International Conference on Intelligent Computing and Control Systems. – 2019. – С. 1255–1260.

7. Random Forests and Decision Trees / A.Jehad, K. Rehanullah, A. Nasir, M. Imran. // International Journal of Computer Science Issues. – 2012. – №9. – С. 272–278.

8. SMOTE and Nearmiss Methods for Disease Classification with Unbalanced Data / A.Alamsyah, S. Rahma, N. Belinda, N. Adi. // International Conference on Data Science and Official Statistics. – 2021. – №1. – С. 305–314.

9. Linear Models [Електронний ресурс] – Режим доступу до ресурсу: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression.

10. Precision-Recall [Електронний ресурс] // scikit-learn. – 2023. – Режим доступу до ресурсу: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html.

11. sklearn.preprocessing.StandardScaler [Електронний ресурс] – Режим доступу до ресурсу: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

Додаток А

Лістинги програм

Лістинг файлу main.py

```
# Імпорт всіх необхідних бібліотек та модулів
import xgboost as xgb
from sklearn.ensemble import RandomForestClassifier
from sklearn import tree
from sklearn import metrics
from sklearn.metrics import PrecisionRecallDisplay
from sklearn.metrics import classification_report, accuracy_score, precision_recall_curve
from sklearn.linear_model import LogisticRegression
from scipy.stats import randint
import sklearn as skl
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
import streamlit as st
from PIL import Image
import numpy as np
import io
import warnings
from collections import Counter

# Ігнорувати попередження
warnings.filterwarnings("ignore")

# Встановлюємо випадкове насіння для повторюваності
np.random.seed(42)

# Вибір назв сторінок
data_options = ['Головна', 'Прогнозування кредитних ризиків',
                'Аналіз банківського ринку']
app_mode = st.sidebar.selectbox('Меню', options=data_options)

# Головна сторінка
if app_mode == "Головна":
    # Заголовок "Дипломна робота"
```

```
        st.title('Дипломна робота')
        st.markdown('на тему')  # Текст "на тему"
        st.markdown('"Математичне та програмне забезпечення підсистеми аналізу та прогнозування відкриття банківських депозитів та кредитів"')
        # Стилізація бокової панелі
        st.markdown(
            """
            <style>
            [data-testid="stSidebar"][aria-expanded="true"] > div:first-child{
                width: 350px
            }
            [data-testid="stSidebar"][aria-expanded="false"] > div:first-child{
                width: 350px
                margin-left: -350px
            }
            </style>
            """,

            unsafe_allow_html=True,
        )

        # Текст "студента групи КМ-93"
        st.markdown('студента групи КМ-93')
        # Текст "Данілов Івана Дмитровича"
        st.markdown('Данілов Івана Дмитровича')

    # Сторінка "Прогнозування кредитних ризиків"
    elif app_mode == 'Прогнозування кредитних ризиків':
        # Заголовок "Прогнозування кредитних ризиків"
        st.title('Прогнозування кредитних ризиків')
        # Текст "Головним завданням даного розділу є створення моделі для прогнозування кредитних ризиків"
        st.markdown('Головним завданням даного розділу є створення моделі для прогнозування кредитних ризиків')
        # Заголовок "Що таке кредитний ризик?"
        st.header('Що таке кредитний ризик?')
        # Текст "Кредитний ризик - це можливість сплати відсотків та основної суми боргу з відхиленням від умов кредитної угоди або повним їх не виконанням"
        st.markdown("Кредитний ризик - це можливість сплати відсотків та основної суми боргу з відхиленням від умов кредитної угоди або повним їх не виконанням")
        # Заголовок "Етапи розробки"
        st.header('Етапи розробки')
        # Текст "1. Підготовка та первинна обробка даних"
        st.markdown(
            "1. Підготовка та первинна обробка даних")
        # Текст "2. Відбір необхідних характеристик"
        st.markdown(
```

```
    "2. Відбір необхідних характеристик")
# Текст "3. Розробка та оцінка моделі"
st.markdown("3. Розробка та оцінка моделі")
st.header('Датасет')  # Заголовок "Датасет"
# Завантаження датасету "credit_risks.csv" із збереженням його в змінну "credit_risk"
credit_risk = pd.read_csv("credit_risks.csv")
# Копіювання датасету "credit_risk" в змінну "df"
df = credit_risk.copy()
# Відображення датасету у вигляді таблиці
st.dataframe(df)
# Відкривається розгорнутий блок "Опис даних датасету"
with st.expander("Опис даних датасету"):
    st.write("ID - ID кожного клієнта")
    st.write("LIMIT_BAL - Сума наданого кредиту в новозеландських доларах (включає індивідуальний та сімейний/додатковий кредит")
    st.write("SEX - стать (1=чоловік, 2=жінка)")
    st.write("EDUCATION - (1=аспірантура, 2=університет, 3=середня школа, 4=інше, 5=невідомо, 6=невідомо)")
    st.write(
        "MARRIAGE - Сімейний стан (1=заміжня, 2=незаміжня, 3=інші)")
    st.write("AGE - вік")
    st.write("PAY_0 - Стан погашення заборгованості у вересні 2005 року (-1=плачу вчасно, 1=затримка платежу на один місяць, 2=затримка платежу на два місяці, ... 8=затримка платежу на вісім місяців, 9=затримка платежу на дев'ять місяців і більше)")
    st.write("PAY_2 - Статус погашення у серпні 2005 року (шкала така ж, як і вище)")
    st.write("PAY_3 - Стан погашення заборгованості у липні 2005 року (шкала така ж, як і вище)")
    st.write("PAY_4 - Статус погашення у червні 2005 року (шкала така ж, як і вище)")
    st.write("PAY_5 - Статус погашення у травні 2005 року (шкала така ж, як і вище)")
    st.write("PAY_6 - Статус погашення у квітні 2005 року (шкала така ж, як і вище)")
    st.write("BILL_AMT1 - Сума виписки по рахунку у вересні 2005 року (долар США)")
    st.write("BILL_AMT2 - Сума виписки за рахунком у серпні 2005 року (долар США)")
    st.write("BILL_AMT3 - Сума виписки по рахунку за липень 2005 року (долар США)")
    st.write("BILL_AMT4 - Сума виписки по рахунку за червень 2005 року (долар США)")
    st.write("BILL_AMT5 - Сума виписки по рахунку за травень 2005 року (долар США)")
    st.write("BILL_AMT6 - Сума виписки по рахунку за квітень 2005 року (долар США)")
    st.write("PAY_AMT1 - Сума попереднього платежу у вересні 2005 року (долар США)")
    st.write("PAY_AMT2 - Сума попереднього платежу в серпні 2005 року (долар США)")
    st.write("PAY_AMT3 - Сума попереднього платежу в липні 2005 року (долар США)")
    st.write("PAY_AMT4 - Сума попереднього платежу в червні 2005 року (долар США)")
    st.write("PAY_AMT5 - Сума попереднього платежу в травні 2005 року (долар США)")
    st.write("PAY_AMT6 - Сума попереднього платежу в квітні 2005 року (долар США)")
    st.write(
        "default.payment.next.month - Платіж за замовчуванням (1 = так, 0 = ні)")
# Заголовок "Підготовка та первинна обробка даних"
st.header('Підготовка та первинна обробка даних')
# Відкривається розгорнутий блок "Аналіз даних"
with st.expander("Аналіз даних"):
```

```
st.write("Інформаці про данні датасету:")
buffer = io.StringIO()
# Отримання інформації про датасет
df.info(buf=buffer)
s = buffer.getvalue()
# Виведення інформації у текстовому форматі
st.text(s)
st.divider()  # Додавання роздільника
st.write("Статистика з датасету:")
# Отримання описової статистики
df_description = df.describe(include='all')
# Виведення описової статистики
st.write(df_description)
st.divider()
st.write(
    "Перевіряємо кількість нульових елементів")
# Отримання кількості нульових значень
df_isnull = df.isnull().sum()
# Виведення кількості нульових значень
st.write(df_isnull)
st.divider()
st.write("Стовпчик EDUCATION")
# Виведення інформації про стовпчик EDUCATION
df_description = df['EDUCATION'].value_counts()
st.write(df_description)
st.divider()
# Виведення інформації про стовпчик MARRIAGE
st.write("Стовпчик MARRIAGE")
df_description = df['MARRIAGE'].value_counts()
st.write(df_description)
st.divider()
# Виведення інформації про стовпчик PAY_0
st.write("Стовпчик PAY_0")
df_description = df['PAY_0'].value_counts()
st.write(df_description)
st.divider()
# Виведення інформації про стовпчик PAY_2
st.write("Стовпчик PAY_2")
df_description = df['PAY_2'].value_counts()
st.write(df_description)
# Виведення інформації про стовпчик PAY_3
st.write("Стовпчик PAY_3")
df_description = df['PAY_3'].value_counts()
st.write(df_description)
st.divider()
```

```
    # Виведення інформації про стовпчик PAY_4
    st.write("Стовпчик PAY_4")
    df_description = df['PAY_4'].value_counts()
    st.write(df_description)
    st.divider()
    # Виведення інформації про стовпчик PAY_5
    st.write("Стовпчик PAY_5")
    df_description = df['PAY_5'].value_counts()
    st.write(df_description)
    st.divider()
    # Виведення інформації про стовпчик PAY_6
    st.write("Стовпчик PAY_6")
    df_description = df['PAY_6'].value_counts()
    st.write(df_description)
# Відкривається розгорнутий блок "Підготовка даних"
with st.expander("Підготовка даних"):
    # Видаляємо стовпець ID, оскільки його дані не потрібні для навчання моделі
    st.write("1. Приберемо стовпчик ID, оскільки його дані непотрібні для тренування моделі")
    df.drop(["ID"], axis=1, inplace=True)
    # Об'єднуємо значення 0, 4-6 стовпця EDUCATION в одну категорію
    st.write("2. Об'єднаємо значення 0, 4-6 стовпчика EDUCATION в одну категорію")
    df['EDUCATION'].replace(
        {0: 1, 1: 1, 2: 2, 3: 3, 4: 4, 5: 1, 6: 1}, inplace=True)
    # Замінюємо значення 0 стовпця MARRIAGE на значення 1
    st.write(
        "3. Значення 0 стовпичка MARRIAGE перетворимо в значення 1")
    df['MARRIAGE'].replace({0: 1, 1: 1, 2: 2, 3: 3}, inplace=True)
# Відкривається розгорнутий блок "Графічне зображення даних"
with st.expander("Графічне зображення даних"):
    # Графічне зображення даних стовпця 'default.payment.next.month'
    st.write(
        "Графічне зображення даних стовпця default.payment.next.month")
    plt.figure(figsize=(6, 6))
    sns.countplot(x=df['default.payment.next.month'], order=df['default.payment.next.month'].value_counts(
    ).index, palette=['#408EC6', "#7A2048"])
    plt.xticks([0, 1], labels=["Not Defaulted", "Defaulted"])
    plt.savefig('target_distribution.png')
    image = Image.open('target_distribution.png')
    st.image(image, caption='default.payment.next.month')
    st.divider()

    # Графічне зображення даних стовпця 'AGE'
    st.write(
        "Графічне зображення даних стовпця AGE")
    plt.figure(figsize=(6, 6))
```

```
sns.displot(df['AGE'], kde=True)
plt.xticks(rotation=0)
plt.ylabel('Count')
plt.savefig('age_distribution.png')
image = Image.open('age_distribution.png')
st.image(image, caption='Розподіл за віком')
st.divider()

# Графічне зображення даних стовпця 'SEX'
st.write(
   "Графічне зображення даних стовпця SEX")
plt.figure(figsize=(6, 6))
sns.countplot(x=df['SEX'], hue=df['default.payment.next.month'], palette=[
            '#408EC6', "#7A2048"])
plt.xticks([0, 1], labels=["Male", "Female"])
plt.savefig('sex_distribution.png')
image = Image.open('sex_distribution.png')
st.image(image, caption='Розподіл за статтю')
st.divider()

# Графічне зображення даних стовпця 'EDUCATION'
st.write(
   "Графічне зображення даних стовпця EDUCATION")
plt.figure(figsize=(10, 6))
sns.countplot(x=df['EDUCATION'], hue=df['default.payment.next.month'], palette=[
            '#408EC6', "#7A2048"])
plt.xticks([0, 1, 2, 3], labels=["graduate school",
         "university", 'high school', 'others'])
plt.savefig('education_distribution.png')
image = Image.open('education_distribution.png')
st.image(image, caption='Розподіл за освітою')
st.divider()

# Графічне зображення даних стовпця 'MARRIAGE'
st.write(
   "Графічне зображення даних стовпця MARRIAGE")
plt.figure(figsize=(10, 6))
sns.countplot(x=df['MARRIAGE'], hue=df['default.payment.next.month'], palette=[
            '#408EC6', "#7A2048"])
plt.xticks([0, 1, 2], labels=["Married", "Single", 'Others'])
plt.savefig('marriage_distribution.png')
image = Image.open('marriage_distribution.png')
st.image(image, caption='Розподіл за шлюбом')
st.divider()
```

```
# Графічне зображення даних стовпця 'LIMIT_BAL'
st.write(
    "Графічне зображення даних стовпця LIMIT_BAL")
sns.displot(df.LIMIT_BAL, kde=True)
plt.savefig('limit_bal_distribution.png')
image = Image.open('limit_bal_distribution.png')
st.image(
    image, caption='Розподіл за сумою наданого кредиту')
st.divider()

# Графічне зображення даних стовпців 'PAY_AMT1' - 'PAY_AMT6' та 'BILL_AMT1' - 'BILL_AMT6'
st.write("Графічне зображення даних стовпциків PAY_AMT1 - PAY_AMT6 та BILL_AMT1 - BILL_AMT6")
plt.subplots(figsize=(20, 10))
plt.subplot(231)
plt.scatter(x=df.PAY_AMT1, y=df.BILL_AMT1, c='#4AAFD5', s=1)
plt.xlabel('PAY_AMT1')
plt.ylabel('BILL_AMT1')

plt.subplot(232)
plt.scatter(x=df.PAY_AMT2, y=df.BILL_AMT2, c='#91B187', s=1)
plt.xlabel('PAY_AMT2')
plt.ylabel('BILL_AMT2')

plt.subplot(233)
plt.scatter(x=df.PAY_AMT3, y=df.BILL_AMT3, c='#E7A339', s=1)
plt.xlabel('PAY_AMT3')
plt.ylabel('BILL_AMT3')

plt.subplot(234)
plt.scatter(x=df.PAY_AMT4, y=df.BILL_AMT4, c='#AE0E36FF', s=1)
plt.xlabel('PAY_AMT4')
plt.ylabel('BILL_AMT4')

plt.subplot(235)
plt.scatter(x=df.PAY_AMT5, y=df.BILL_AMT5, c='#783937FF', s=1)
plt.xlabel('PAY_AMT5')
plt.ylabel('BILL_AMT5')

plt.subplot(236)
plt.scatter(x=df.PAY_AMT6, y=df.BILL_AMT6, c='#9B4A97FF', s=1)
plt.xlabel('PAY_AMT6')
plt.ylabel('BILL_AMT6')
plt.savefig('payment_distribution.png')
image = Image.open('payment_distribution.png')
st.image(image, caption='Структура платежів у порівнянні з сумою рахунку за останні 6 місяців')
```

```
# Виводить заголовок "2. Відбір необхідних характеристик"
st.header('2. Відбір необхідних характеристик')
# Виводить повідомлення про створення незалежних та залежних змінних
st.write("1. створено незалежні та залежні змінні")
# Вилучає стовпець 'default.payment.next.month' із DataFrame ізмінну X
X = df.drop(['default.payment.next.month'], axis=1)
# Записує стовпець 'default.payment.next.month' у змінну y
y = df['default.payment.next.month']
# Виводить повідомлення про стандартизацію змінних
st.write("2. Стандартизовано змінні")
# Створює об'єкт класу StandardScaler для стандартизації
scaler = StandardScaler()
# Застосовує стандартизацію до змінної X
X = scaler.fit_transform(X)
# Розбиває дані на навчальний і тестовий набори
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20, random_state=42)
# Виводить кількість даних перед передискретизацією за допомогою лічильника Counter
print("Кількість даних перед передискретизацією: ", Counter(y_train))
SMOTE = SMOTE()  # Створює об'єкт класу SMOTE для передискретизації
# Передискретизує дані за допомогою SMOTE
X_train, y_train = SMOTE.fit_resample(X_train, y_train)
# Виводить кількість даних після передискретизації за допомогою лічильника Counter
print("Кількість даних після передискретизації: ", Counter(y_train))
# Виводить повідомлення про розбиття даних на навчальний і тестовий набори
st.write(
    "3. Данні розбито на навчальні та тестові")
# Виводить повідомлення про передискретизацію даних
st.write("4. Передискретизовано дані")
# Виводить заголовок "3. Розробка та оцінка моделі"
st.header('3. Розробка та оцінка моделі')
tab1, tab2, tab3, tab4 = st.tabs(["Логістична регресія", "Класифікатор випадкових лісів", "Класифікатор XGBoost",
                                  "Класифікатор дерева рішень"])  # Створює вкладки для різних класифікаторів
# Виводить дані розділу Логістична регресія
with tab1:
    # Виводить заголовок "Логістична регресія" в межах першої вкладки
    st.header("Логістична регресія")
    # Створює об'єкт класу LogisticRegression для моделі логістичної регресії
    logit = LogisticRegression()
    # Навчає модель логістичної регресії на навчальних даних
    logit.fit(X_train, y_train)
    # Здійснює передбачення за допомогою моделі логістичної регресії на тестових даних
    pred_logit = logit.predict(X_test)
    # Формує рядок з точністю моделі
    string = f'Точність створеної моделі = {accuracy_score(y_test, pred_logit)}'
```

```
# Виводить рядок з точністю моделі
st.write(string)
# Обчислює звіт про класифікацію
class_report = classification_report(
    y_test, pred_logit, output_dict=True)
# Створює DataFrame зі звітом про класифікацію
class_report_df = pd.DataFrame(class_report).transpose()
# Виводить DataFrame зі звітом про класифікацію
st.dataframe(class_report_df)
# Обчислює точність моделі
accuracy_score_logit = accuracy_score(y_test, pred_logit)
# Створює матрицю невідповідностей за допомогою моделі логістичної регресії
disp = skl.metrics.ConfusionMatrixDisplay.from_estimator(
    logit, X_test, y_test, cmap="Blues_r")
disp.plot()  # Відображає матрицю невідповідностей
# Зберігає зображення матриці невідповідностей
plt.savefig('disp_logistic.png')
# Завантажує зображення матриці невідповідностей
image = Image.open('disp_logistic.png')
# Виводить зображення матриці невідповідностей з підписом
st.image(image, caption='Матриця невідповідностей')
# Здійснює передбачення ймовірностей класу 1 (позитивного класу) на тестових даних
y_pred = logit.predict_proba(X_test)[:, 1]
# Відображає графік точності-повноти
PrecisionRecallDisplay.from_predictions(y_test, y_pred)
# Зберігає зображення графіку точності-повноти
plt.savefig('PrecisionRecallDisplay_logistic.png')
# Завантажує зображення графіку точності-повноти
image = Image.open('PrecisionRecallDisplay_logistic.png')
# Виводить зображення графіку точності-повноти з підписом
st.image(image, caption='Графік Precision-Recall')
# Обчислює характеристики для побудови кривої ROC
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred)
plt.clf()  # Очищає поточний графік
plt.plot(fpr, tpr)  # Побудова кривої ROC
# Встановлює підпис для осі y
plt.ylabel('True Positive Rate')
# Встановлює підпис для осі x
plt.xlabel('False Positive Rate')
# Зберігає зображення кривої ROC
plt.savefig('ROCAUC_logistic.png')
# Завантажує зображення кривої ROC
image = Image.open('ROCAUC_logistic.png')
# Виводить зображення кривої ROC з підписом
st.image(image, caption='Крива помилок AUC-ROC')
```

```
# Виводить дані розділу Класифікатор випадкових лісів
with tab2:
    # Виводить заголовок "Класифікатор випадкових лісів" в межах другої вкладки
    st.header("Класифікатор випадкових лісів")
    # Створює об'єкт класу RandomForestClassifier для моделі випадкових лісів
    random_forest = RandomForestClassifier()
    # Навчає модель випадкових лісів на навчальних даних
    random_forest.fit(X_train, y_train)
    # Здійснює передбачення за допомогою моделі випадкових лісів на тестових даних
    pred_random_forest = random_forest.predict(X_test)
    # Формує рядок з точністю моделі
    string = f'Точність створеної моделі = {accuracy_score(y_test, pred_random_forest)}'
    # Виводить рядок з точністю моделі
    st.write(string)
    # Обчислює звіт про класифікацію
    class_report = classification_report(
        y_test, pred_random_forest, output_dict=True)
    # Створює DataFrame зі звітом про класифікацію
    class_report_df = pd.DataFrame(class_report).transpose()
    # Виводить DataFrame зі звітом про класифікацію
    st.dataframe(class_report_df)
    # Обчислює точність моделі
    accuracy_score_logit = accuracy_score(y_test, pred_logit)
    # Створює матрицю невідповідностей за допомогою моделі випадкових лісів
    disp = skl.metrics.ConfusionMatrixDisplay.from_estimator(
        random_forest, X_test, y_test, cmap="Blues_r")
    disp.plot()  # Відображає матрицю невідповідностей
    # Зберігає зображення матриці невідповідностей
    plt.savefig('disp_random_forest.png')
    # Завантажує зображення матриці невідповідностей
    image = Image.open('disp_random_forest.png')
    # Виводить зображення матриці невідповідностей з підписом
    st.image(image, caption='Матриця невідповідностей')
    # Здійснює передбачення ймовірностей класу 1 (позитивного класу) на тестових даних
    y_pred = random_forest.predict_proba(X_test)[:, 1]
    # Відображає графік точності-повноти
    PrecisionRecallDisplay.from_predictions(y_test, y_pred)
    # Зберігає зображення графіку точності-повноти
    plt.savefig('PrecisionRecallDisplay_random_forest.png')
    # Завантажує зображення графіку точності-повноти
    image = Image.open('PrecisionRecallDisplay_random_forest.png')
    # Виводить зображення графіку точності-повноти з підписом
    st.image(image, caption='Графік Precision-Recall')
    # Обчислює характеристики для побудови кривої ROC
    fpr, tpr, _ = metrics.roc_curve(y_test, y_pred)
```

```
    plt.clf()  # Очищає поточний графік
    plt.plot(fpr, tpr)  # Побудова кривої ROC
    # Встановлює підпис для осі y
    plt.ylabel('True Positive Rate')
    # Встановлює підпис для осі x
    plt.xlabel('False Positive Rate')
    # Зберігає зображення кривої ROC
    plt.savefig('ROCAUC_random_forest.png')
    # Завантажує зображення кривої ROC
    image = Image.open('ROCAUC_random_forest.png')
    # Виводить зображення кривої ROC з підписом
    st.image(image, caption='Крива помилок AUC-ROC')
# Виводить дані розділу Класифікатор XGBoost
with tab3:
    # Виводить заголовок "Класифікатор XGBoost" в межах третьої вкладки
    st.header("Класифікатор XGBoost")
    # Створює об'єкт класу XGBClassifier для моделі XGBoost
    xgb_classifier = xgb.XGBClassifier()
    # Навчає модель XGBoost на навчальних даних
    xgb_classifier.fit(X_train, y_train)
    # Здійснює передбачення за допомогою моделі XGBoost на тестових даних
    xgb_predict = xgb_classifier.predict(X_test)
    # Формує рядок з точністю моделі
    string = f'Точність створеної моделі = {accuracy_score(y_test, xgb_predict)}'
    # Виводить рядок з точністю моделі
    st.write(string)
    # Обчислює звіт про класифікацію
    class_report = classification_report(
        y_test, xgb_predict, output_dict=True)
    # Створює DataFrame зі звітом про класифікацію
    class_report_df = pd.DataFrame(class_report).transpose()
    # Виводить DataFrame зі звітом про класифікацію
    st.dataframe(class_report_df)
    # Створює матрицю невідповідностей за допомогою моделі XGBoost
    disp = skl.metrics.ConfusionMatrixDisplay.from_estimator(
        xgb_classifier, X_test, y_test, cmap="Blues_r")
    disp.plot()  # Відображає матрицю невідповідностей
    # Зберігає зображення матриці невідповідностей
    plt.savefig('disp_XGBoost.png')
    # Завантажує зображення матриці невідповідностей
    image = Image.open('disp_XGBoost.png')
    # Виводить зображення матриці невідповідностей з підписом
    st.image(image, caption='Матриця невідповідностей')
    # Здійснює передбачення ймовірностей класу 1 (позитивного класу) за допомогою моделі XGBoost на тестових даних
    y_pred = xgb_classifier.predict_proba(X_test)[:, 1]
```

```
    # Відображає графік точності-повноти
    PrecisionRecallDisplay.from_predictions(y_test, y_pred)
    # Зберігає зображення графіку точності-повноти
    plt.savefig('PrecisionRecallDisplay_XGBoost.png')
    # Завантажує зображення графіку точності-повноти
    image = Image.open('PrecisionRecallDisplay_XGBoost.png')
    # Виводить зображення графіку точності-повноти з підписом
    st.image(image, caption='Графік Precision-Recall')
    # Обчислює характеристики для побудови кривої ROC
    fpr, tpr, _ = metrics.roc_curve(y_test, y_pred)
    plt.clf()  # Очищає поточний графік
    plt.plot(fpr, tpr)  # Побудова кривої ROC
    # Встановлює підпис для осі y
    plt.ylabel('True Positive Rate')
    # Встановлює підпис для осі x
    plt.xlabel('False Positive Rate')
    # Зберігає зображення кривої ROC
    plt.savefig('ROCAUC_xgb_classifier.png')
    # Завантажує зображення кривої ROC
    image = Image.open('ROCAUC_xgb_classifier.png')
    # Виводить зображення кривої ROC з підписом
    st.image(image, caption='Крива помилок AUC-ROC')
# Виводить дані розділу Класифікатор дерева рішень
with tab4:
    # Виводить заголовок "Класифікатор дерева рішень" в межах четвертої вкладки
    st.header("Класифікатор дерева рішень")
    # Створює об'єкт класу DecisionTreeClassifier для моделі дерева рішень
    DecisionTree = tree.DecisionTreeClassifier()
    # Навчає модель дерева рішень на навчальних даних
    DecisionTree.fit(X_train, y_train)
    # Здійснює передбачення за допомогою моделі дерева рішень на тестових даних
    pred_DecisionTree = DecisionTree.predict(X_test)
    # Формує рядок з точністю моделі
    string = f'Точність створеної моделі = {accuracy_score(y_test, pred_DecisionTree)}'
    # Виводить рядок з точністю моделі
    st.write(string)
    # Обчислює звіт про класифікацію
    class_report = classification_report(
        y_test, pred_DecisionTree, output_dict=True)
    # Створює DataFrame зі звітом про класифікацію
    class_report_df = pd.DataFrame(class_report).transpose()
    # Виводить DataFrame зі звітом про класифікацію
    st.dataframe(class_report_df)
    # Створює матрицю невідповідностей за допомогою моделі дерева рішень
    disp = skl.metrics.ConfusionMatrixDisplay.from_estimator(
```

```
                DecisionTree, X_test, y_test, cmap="Blues_r")
            disp.plot()  # Відображає матрицю невідповідностей
            # Зберігає зображення матриці невідповідностей
            plt.savefig('disp_DecisionTree.png')
            # Завантажує зображення матриці невідповідностей
            image = Image.open('disp_DecisionTree.png')
            # Виводить зображення матриці невідповідностей з підписом
            st.image(image, caption='Матриця невідповідностей')
            # Здійснює передбачення ймовірностей класу 1 (позитивного класу) за допомогою моделі дерева рішень на тестових даних
            y_pred = DecisionTree.predict_proba(X_test)[:, 1]
            # Відображає графік точності-повноти
            PrecisionRecallDisplay.from_predictions(y_test, y_pred)
            # Зберігає зображення графіку точності-повноти
            plt.savefig('PrecisionRecallDisplay_DecisionTree.png')
            # Завантажує зображення графіку точності-повноти
            image = Image.open('PrecisionRecallDisplay_DecisionTree.png')
            # Виводить зображення графіку точності-повноти з підписом
            st.image(image, caption='Графік Precision-Recall')
            # Обчислює характеристики для побудови кривої ROC
            fpr, tpr, _ = metrics.roc_curve(y_test, y_pred)
            plt.clf()  # Очищає поточний графік
            plt.plot(fpr, tpr)  # Побудова кривої ROC
            # Встановлює підпис для осі y
            plt.ylabel('True Positive Rate')
            # Встановлює підпис для осі x
            plt.xlabel('False Positive Rate')
            # Зберігає зображення кривої ROC
            plt.savefig('ROCAUC_DecisionTree.png')
            # Завантажує зображення кривої ROC
            image = Image.open('ROCAUC_DecisionTree.png')
            # Виводить зображення кривої ROC з підписом
            st.image(image, caption='Крива помилок AUC-ROC')
    # Сторінка "Аналіз банківського ринку"
    elif app_mode == 'Аналіз банківського ринку':
        # Виводить заголовок "Аналіз банківського ринку"
        st.title('Аналіз банківського ринку')
        # Виводить текстовий блок з поясненням головної мети розділу
        st.markdown('Головним завданням даного розділу є пошук найкращої стратегії для покращення для наступної маркетингової кампанії')
        # Виводить заголовок "Етапи розробки"
        st.header('Етапи розробки')
        # Виводить текстовий блок з першим етапом розробки
        st.markdown(
            "1. Підготовка та первинна обробка даних")
```

```
# Виводить текстовий блок з другим етапом розробки
st.markdown(
    "2. Відбір необхідних характеристик")
# Виводить текстовий блок з третім етапом розробки
st.markdown("3. Розробка та оцінка моделі")
# Виводить заголовок "Датасет"
st.header('Датасет')
# Завантажує датасет "bank_marketing.csv" із файлу
credit_risk = pd.read_csv("bank_marketing.csv")
df = credit_risk.copy()  # Створює копію датасету
# Виводить датасет у вигляді таблиці
st.dataframe(df)
# Виводить розгорнутий блок з назвою "Опис даних датасету"
with st.expander("Опис даних датасету"):
    # Виводить опис першої колонки "age"
    st.write("age - вік кожного клієнта")
    # Виводить опис другої колонки "job"
    st.write("job - вид роботи")
    # Виводить опис третьої колонки "marital"
    st.write("marital - сімейний стан")
    # Виводить опис четвертої колонки "education"
    st.write(
        "education - освіта та кваліфікація клієнта")
    # Виводить опис п'ятої колонки "default"
    st.write(
        "default - Чи є у клієнта дефолтний (несплачений) кредит")
    # Виводить опис шостої колонки "balance"
    st.write("balance - середньорічний залишок, в євро")
    # Виводить опис сьомої колонки "housing"
    st.write("housing - Чи є у клієнта кредит на житло?")
    # Виводить опис восьмої колонки "loan"
    st.write("loan - Чи є у клієнта особистий кредит?")
    # Виводить опис дев'ятої колонки "contact"
    st.write("contact - контактний тип зв'язку")
    # Виводить опис десятої колонки "day"
    st.write(
        "day - останній день коли було встановлено контакт")
    # Виводить опис одинадцятої колонки "month"
    st.write(
        "month - останній місяць коли було встановлено контакт")
    # Виводить опис дванадцятої колонки "duration"
    st.write(
        "duration - тривалість останнього контакту, у секундах")
    # Виводить опис тринадцятої колонки "campaign"
    st.write("campaign - кількість контактів, здійснених під час цієї кампанії та для цього клієнта")
```

```
    # Виводить опис чотирнадцятої колонки "pdays"
    st.write("pdays - кількість днів, що минули з моменту останнього контакту з клієнтом з попередньої кампанії")
    # Виводить опис п'ятнадцятої колонки "previous"
    st.write("previous - кількість контактів, здійснених до цієї кампанії та для цього клієнта")
    # Виводить опис шістнадцятої колонки "poutcome"
    st.write(
        "poutcome - результат попередньої маркетингової кампанії")
    # Виводить опис сімнадцятої колонки "deposit"
    st.write(
        "deposit - чи оформив клієнт строковий депозит?")
# Виведення заголовка "Підготовка та первинна обробка даних"
st.header('Підготовка та первинна обробка даних')
# Виводить розгорнутий блок з назвою "Аналіз даних"
with st.expander("Аналіз даних"):
    # Виведення тексту "Інформаці про данні датасету:"
    st.write("Інформаці про данні датасету:")
    buffer = io.StringIO()  # Створення буфера
    # Отримання інформації про датасет
    df.info(buf=buffer)
    s = buffer.getvalue()  # Отримання значення з буфера
    st.text(s)  # Виведення значення з буфера
    st.divider()  # Виведення розділювача
    # Виведення тексту "Статистика з датасету:"
    st.write("Статистика з датасету:")
    # Отримання статистики з датасету
    df_description = df.describe(include="all")
    st.write(df_description)  # Виведення статистики
    st.divider()  # Виведення розділювача

    # Виведення тексту "Перевіряємо кількість нульових елементів"
    st.write(
        "Перевіряємо кількість нульових елементів")
    # Отримання кількості нульових елементів
    df_isnull = df.isnull().sum()
    # Виведення кількості нульових елементів
    st.write(df_isnull)
    st.divider()  # Виведення розділювача

    # Виведення тексту "Графічне зображення стовпчиків з числовим значенням"
    st.write(
        "Графічне зображення стовпчиків з числовим значенням")
    # Список стовпчиків з числовими значеннями
    num_columns = ['balance', 'day', 'duration',
                   'campaign', 'pdays', 'previous']
    # Створення підграфіків для відображення графіків
```

```
fig, axs = plt.subplots(
    2, 3, sharex=False, sharey=False, figsize=(20, 15))
counter = 0  # Лічильник
for num_column in num_columns:  # Для кожного стовпчика з числовими значеннями
    # Розраховується розташування рядка підграфіка
    trace_x = counter // 3
    # Розраховується розташування стовпчика підграфіка
    trace_y = counter % 3
    # Побудова гістограми для стовпчика
    axs[trace_x, trace_y].hist(df[num_column])
    # Встановлення заголовка для підграфіка
    axs[trace_x, trace_y].set_title(num_column)
    counter += 1  # Збільшення лічильника

# Збереження графічних зображень
plt.savefig('numerical_columns.png')
# Відкриття графічного зображення
image = Image.open('numerical_columns.png')
# Виведення графічного зображення
st.image(image)
st.divider()  # Виведення розділювача

# Виведення тексту "Графічне зображення стовпчиків зі значенням типу string"
st.write(
    "Графічне зображення стовпчиків зі значенням типу string")
# Список стовпчиків зі значеннями типу string
category_columns = ['job', 'marital', 'education', 'default',
                    'housing', 'loan', 'contact', 'month', 'poutcome']
# Створення підграфіків для відображення графіків
fig, axs = plt.subplots(
    3, 3, sharex=False, sharey=False, figsize=(20, 15))
counter = 0  # Лічильник
for cat_column in category_columns:  # Для кожного стовпчика зі значеннями типу string
    # Отримання кількості значень
    value_counts = df[cat_column].value_counts()
    # Розраховується розташування рядка підграфіка
    trace_x = counter // 3
    # Розраховується розташування стовпчика підграфіка
    trace_y = counter % 3
    # Позиції для стовпців графіку
    x_pos = np.arange(0, len(value_counts))
    # Побудова стовпчикової діаграми
    axs[trace_x, trace_y].bar(
        x_pos, value_counts.values, tick_label=value_counts.index)
    # Встановлення заголовка для підграфіка
```

```
                axs[trace_x, trace_y].set_title(cat_column)
                # Поворот позначок на осі X
                for tick in axs[trace_x, trace_y].get_xticklabels():
                    tick.set_rotation(90)
                counter += 1  # Збільшення лічильника

            # Збереження графічних зображень
            plt.savefig('category_columns.png')
            # Відкриття графічного зображення
            image = Image.open('category_columns.png')
            # Виведення графічного зображення
            st.image(image)
        # Виводить розгорнутий блок з назвою "Підготовка даних"
        with st.expander("Підготовка даних"):
            # Виведення тексту про перше кроку підготовки даних
            st.write(
                "1. Приберемо нульові та невірні значення")
            # Виведення тексту про другий крок підготовки даних
            st.write(
                "2. Замінимо значення 'так' на 1 і значення 'ні' на 0")
            # Виведення тексту про третій крок підготовки даних
            st.write(
                "3. Значення типу string змінимо на числові")

            def convert_yes_no_into_0_1(row, column_name):
                # Функція для заміни значень 'yes' на 1 і 'no' на 0
                return 1 if row[column_name] == 'yes' else 0

            def get_correct_values(row, column_name, threshold, df):
                # Функція для отримання коректних значень стовпчика
                return row[column_name] if row[column_name] <= threshold else df[df[column_name] <=
threshold][column_name].mean()

            def clean_data(df):
                # Створення копії вихідного датасету
                cleaned_df = df.copy()

                # Список стовпчиків з булевими значеннями
                bool_columns = ['default', 'housing', 'loan', 'deposit']
                for bool_col in bool_columns:
                    # Заміна булевих значень на числові
                    cleaned_df[bool_col + '_bool'] = df.apply(
                        lambda row: convert_yes_no_into_0_1(row, bool_col), axis=1)

                # Видалення булевих стовпчиків
```

```
        cleaned_df = cleaned_df.drop(columns=bool_columns)

        # Список стовпчиків зі значеннями типу string
        cat_columns = ['job', 'marital', 'education',
                       'contact', 'month', 'poutcome']
        for col in cat_columns:
            cleaned_df = pd.concat([cleaned_df.drop(col, axis=1),
                                    pd.get_dummies(cleaned_df[col], prefix=col, prefix_sep='_', drop_first=True, dummy_na=False)],
                                   axis=1)  # Заміна стовпчиків зі значеннями типу string на числові за допомогою one-hot encoding

        # Видалення стовпчика 'pdays'
        cleaned_df = cleaned_df.drop(columns=['pdays'])

        # Отримання коректних значень для стовпчика 'campaign'
        cleaned_df['campaign_cleaned'] = df.apply(
            lambda row: get_correct_values(row, 'campaign', 34, cleaned_df), axis=1)
        # Отримання коректних значень для стовпчика 'previous'
        cleaned_df['previous_cleaned'] = df.apply(
            lambda row: get_correct_values(row, 'previous', 34, cleaned_df), axis=1)

        # Видалення стовпчиків 'campaign' і 'previous'
        cleaned_df = cleaned_df.drop(columns=['campaign', 'previous'])

        return cleaned_df

    # Очищення даних і отримання очищеного датасету
    cleaned_df = clean_data(df)
# Виводить розгорнутий блок з назвою "Графічне зображення даних"
with st.expander("Графічне зображення даних"):
    # Графічне зображення даних стовпця 'deposit'
    st.write(
        "Графічне зображення даних стовпчика deposit")
    plt.figure(figsize=(6, 6))
    sns.countplot(x=df['deposit'],
                  order=df['deposit'].value_counts().index, palette=['#408EC6', "#7A2048"])
    plt.xticks([0, 1], labels=["No(0)", "Yes(1)"])
    plt.savefig('deposit_distribution.png')
    image = Image.open('deposit_distribution.png')
    st.image(image, caption='deposit')
    st.divider()
    # Графічне зображення залежності даних стовпчика deposit від інших
    st.write(
        "Графічне зображення залежності даних стовпчика deposit від інших")
    plt.figure(figsize=(6, 6))
    sns.countplot(x=df['marital'], hue=df['deposit'], palette=[
```

```
                '#408EC6', "#7A2048"])
            plt.xticks(rotation=90)
            plt.savefig('marital_distribution.png')
            image = Image.open('marital_distribution.png')
            st.image(image, caption='Залежність deposit від marital')
            st.divider()
            plt.figure(figsize=(6, 6))
            sns.countplot(x=df['job'], hue=df['deposit'], palette=[
                '#408EC6', "#7A2048"])
            plt.xticks(rotation=90)
            plt.savefig('job_distribution.png')
            image = Image.open('job_distribution.png')
            st.image(image, caption='Залежність deposit від job')
            st.divider()
            plt.figure(figsize=(6, 6))
            sns.countplot(x=df['education'], hue=df['deposit'], palette=[
                '#408EC6', "#7A2048"])
            plt.xticks(rotation=90)
            plt.savefig('education_distribution.png')
            image = Image.open('education_distribution.png')
            st.image(image, caption='Залежність deposit від education')
            st.divider()
            plt.figure(figsize=(6, 6))
            sns.countplot(x=df['contact'], hue=df['deposit'], palette=[
                '#408EC6', "#7A2048"])
            plt.xticks(rotation=90)
            plt.savefig('contact_distribution.png')
            image = Image.open('contact_distribution.png')
            st.image(image, caption='Залежність deposit від contact')
        st.header('3. Розробка та оцінка моделі')
        tab1, tab2, tab3, tab4 = st.tabs(["Логістична регресія",
                                          "Класифікатор випадкових лісів", "Класифікатор XGBoost", "Класифікатор дерева рішень"])  # Данні
4 розділів
        # Вилучення стовпчика 'deposit_bool' з даних
        X = cleaned_df.drop(columns='deposit_bool')
        # Створення цільової змінної 'deposit_bool'
        y = cleaned_df[['deposit_bool']]
        # Ініціалізація об'єкта для стандартизації даних
        scaler = StandardScaler()
        X = scaler.fit_transform(X)  # Стандартизація даних
        # Розбиття даних на тренувальний та тестовий набори
        X_train, X_test, y_train, y_test = train_test_split(
            X, y, test_size=0.3, random_state=42)

        SMOTE = SMOTE()  # Ініціалізація об'єкта для використання методу SMOTE
```

```
# Застосування методу SMOTE для балансування класів в тренувальному наборі
X_train, y_train = SMOTE.fit_resample(X_train, y_train)
with tab1:  # Вкладка для моделі "Логістична регресія"
    # Виведення заголовку "Логістична регресія"
    st.header("Логістична регресія")
    # Ініціалізація моделі логістичної регресії
    logit = LogisticRegression()
    # Навчання моделі на тренувальних даних
    logit.fit(X_train, y_train)
    # Прогнозування на тестових даних
    pred_logit = logit.predict(X_test)
    # Розрахунок та виведення точності моделі
    string = f'Точність створеної моделі = {accuracy_score(y_test, pred_logit)}'
    # Виведення рядка з точністю моделі
    st.write(string)
    # Розрахунок звіту по класифікації
    class_report = classification_report(
        y_test, pred_logit, output_dict=True)
    # Створення DataFrame звіту по класифікації
    class_report_df = pd.DataFrame(class_report).transpose()
    # Виведення DataFrame звіту по класифікації
    st.dataframe(class_report_df)
    # Розрахунок точності моделі
    accuracy_score_logit = accuracy_score(y_test, pred_logit)
    disp = skl.metrics.ConfusionMatrixDisplay.from_estimator(
        logit, X_test, y_test, cmap="Blues_r")  # Створення матриці невідповідностей
    disp.plot()  # Виведення матриці невідповідностей
    # Збереження зображення матриці невідповідностей
    plt.savefig('disp_logistic.png')
    # Завантаження зображення матриці невідповідностей
    image = Image.open('disp_logistic.png')
    # Виведення зображення матриці невідповідностей з підписом
    st.image(image, caption='Матриця невідповідностей')
    # Прогнозування ймовірностей на тестових даних
    y_pred = logit.predict_proba(X_test)[:, 1]
    # Створення графіка Precision-Recall
    PrecisionRecallDisplay.from_predictions(y_test, y_pred)
    # Збереження графіка Precision-Recall
    plt.savefig('PrecisionRecallDisplay_logistic.png')
    # Завантаження графіка Precision-Recall
    image = Image.open('PrecisionRecallDisplay_logistic.png')
    # Виведення графіка Precision-Recall з підписом
    st.image(image, caption='Графік Precision-Recall')
    # Створення ROC-кривої
    fpr, tpr, _ = metrics.roc_curve(y_test, y_pred)
```

```
    plt.clf()
    plt.plot(fpr, tpr)
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    # Збереження графіка ROC-AUC
    plt.savefig('ROCAUC_logistic.png')
    # Завантаження графіка ROC-AUC
    image = Image.open('ROCAUC_logistic.png')
    # Виведення графіка ROC-AUC з підписом
    st.image(image, caption='Крива помилок AUC-ROC')

with tab2:  # Вкладка для моделі "Класифікатор випадкових лісів"
    # Виведення заголовку "Класифікатор випадкових лісів"
    st.header("Класифікатор випадкових лісів")
    # Ініціалізація моделі класифікатора випадкових лісів
    random_forest = RandomForestClassifier()
    # Навчання моделі на тренувальних даних
    random_forest.fit(X_train, y_train)
    # Прогнозування на тестових даних
    pred_random_forest = random_forest.predict(X_test)
    # Розрахунок та виведення точності моделі
    string = f'Точність створеної моделі = {accuracy_score(y_test, pred_random_forest)}'
    # Виведення рядка з точністю моделі
    st.write(string)
    # Розрахунок звіту по класифікації
    class_report = classification_report(
        y_test, pred_random_forest, output_dict=True)
    # Створення DataFrame звіту по класифікації
    class_report_df = pd.DataFrame(class_report).transpose()
    # Виведення DataFrame звіту по класифікації
    st.dataframe(class_report_df)
    # Розрахунок точності моделі
    accuracy_score_logit = accuracy_score(y_test, pred_logit)
    disp = skl.metrics.ConfusionMatrixDisplay.from_estimator(
        random_forest, X_test, y_test, cmap="Blues_r")  # Створення матриці невідповідностей
    disp.plot()  # Виведення матриці невідповідностей
    # Збереження зображення матриці невідповідностей
    plt.savefig('disp_random_forest.png')
    # Завантаження зображення матриці невідповідностей
    image = Image.open('disp_random_forest.png')
    # Виведення зображення матриці невідповідностей з підписом
    st.image(image, caption='Матриця невідповідностей')
    # Прогнозування ймовірностей на тестових даних
    y_pred = random_forest.predict_proba(X_test)[:, 1]
    # Створення графіка Precision-Recall
```

```
    PrecisionRecallDisplay.from_predictions(y_test, y_pred)
    # Збереження графіка Precision-Recall
    plt.savefig('PrecisionRecallDisplay_random_forest.png')
    # Завантаження графіка Precision-Recall
    image = Image.open('PrecisionRecallDisplay_random_forest.png')
    # Виведення графіка Precision-Recall з підписом
    st.image(image, caption='Графік Precision-Recall')
    # Розрахунок значень False Positive Rate і True Positive Rate для ROC-кривої
    fpr, tpr, _ = metrics.roc_curve(y_test, y_pred)
    plt.clf()  # Очистка поточної фігури
    plt.plot(fpr, tpr)  # Побудова ROC-кривої
    plt.ylabel('True Positive Rate')  # Підпис вісі Y
    plt.xlabel('False Positive Rate')  # Підпис вісі X
    # Збереження графіка ROC-AUC
    plt.savefig('ROCAUC_random_forest.png')
    # Завантаження графіка ROC-AUC
    image = Image.open('ROCAUC_random_forest.png')
    # Виведення графіка ROC-AUC з підписом
    st.image(image, caption='Крива помилок AUC-ROC')

with tab3:  # Вкладка для моделі "Класифікатор XGBoost"
    # Виведення заголовку "Класифікатор XGBoost"
    st.header("Класифікатор XGBoost")
    xgb_classifier = xgb.XGBClassifier(n_estimators=100, learning_rate=0.08, gamma=0, subsample=0.75,
                                       colsample_bytree=1, max_depth=7)  # Ініціалізація моделі класифікатора XGBoost
    # Навчання моделі на тренувальних даних
    xgb_classifier.fit(X_train, y_train.squeeze().values)
    # Прогнозування на тестових даних
    xgb_predict = xgb_classifier.predict(X_test)
    # Розрахунок та виведення точності моделі
    string = f'Точність створеної моделі = {accuracy_score(y_test, xgb_predict)}'
    # Виведення рядка з точністю моделі
    st.write(string)
    # Розрахунок звіту по класифікації
    class_report = classification_report(
        y_test, xgb_predict, output_dict=True)
    # Створення DataFrame звіту по класифікації
    class_report_df = pd.DataFrame(class_report).transpose()
    # Виведення DataFrame звіту по класифікації
    st.dataframe(class_report_df)
    disp = skl.metrics.ConfusionMatrixDisplay.from_estimator(
        xgb_classifier, X_test, y_test, cmap="Blues_r")  # Створення матриці невідповідностей
    disp.plot()  # Виведення матриці невідповідностей
    # Збереження зображення матриці невідповідностей
    plt.savefig('disp_XGBoost.png')
```

```
    # Завантаження зображення матриці невідповідностей
    image = Image.open('disp_XGBoost.png')
    # Виведення зображення матриці невідповідностей з підписом
    st.image(image, caption='Матриця невідповідностей')
    # Прогнозування ймовірностей на тестових даних
    y_pred = xgb_classifier.predict_proba(X_test)[:, 1]
    # Створення графіка Precision-Recall
    PrecisionRecallDisplay.from_predictions(y_test, y_pred)
    # Збереження графіка Precision-Recall
    plt.savefig('PrecisionRecallDisplay_XGBoost.png')
    # Завантаження графіка Precision-Recall
    image = Image.open('PrecisionRecallDisplay_XGBoost.png')
    # Виведення графіка Precision-Recall з підписом
    st.image(image, caption='Графік Precision-Recall')
    # Розрахунок значень False Positive Rate і True Positive Rate для ROC-кривої
    fpr, tpr, _ = metrics.roc_curve(y_test, y_pred)
    plt.clf()  # Очистка поточної фігури
    plt.plot(fpr, tpr)  # Побудова ROC-кривої
    plt.ylabel('True Positive Rate')  # Підпис вісі Y
    plt.xlabel('False Positive Rate')  # Підпис вісі X
    # Збереження графіка ROC-AUC
    plt.savefig('ROCAUC_xgb_classifier.png')
    # Завантаження графіка ROC-AUC
    image = Image.open('ROCAUC_xgb_classifier.png')
    # Виведення графіка ROC-AUC з підписом
    st.image(image, caption='Крива помилок AUC-ROC')

with tab4:  # Вкладка для моделі "Класифікатор дерева рішень"
    # Виведення заголовку "Класифікатор дерева рішень"
    st.header("Класифікатор дерева рішень")
    # Ініціалізація моделі класифікатора дерева рішень
    DecisionTree = tree.DecisionTreeClassifier()
    # Навчання моделі на тренувальних даних
    DecisionTree.fit(X_train, y_train)
    # Прогнозування на тестових даних
    pred_DecisionTree = DecisionTree.predict(X_test)
    # Розрахунок та виведення точності моделі
    string = f'Точність створеної моделі = {accuracy_score(y_test, pred_DecisionTree)}'
    # Виведення рядка з точністю моделі
    st.write(string)
    # Розрахунок звіту по класифікації
    class_report = classification_report(
        y_test, pred_DecisionTree, output_dict=True)
    # Створення DataFrame звіту по класифікації
    class_report_df = pd.DataFrame(class_report).transpose()
```

```
    # Виведення DataFrame звіту по класифікації
    st.dataframe(class_report_df)
    disp = skl.metrics.ConfusionMatrixDisplay.from_estimator(
        DecisionTree, X_test, y_test, cmap="Blues_r")  # Створення матриці невідповідностей
    disp.plot()  # Виведення матриці невідповідностей
    # Збереження зображення матриці невідповідностей
    plt.savefig('disp_DecisionTree_mark.png')
    # Завантаження зображення матриці невідповідностей
    image = Image.open('disp_DecisionTree_mark.png')
    # Виведення зображення матриці невідповідностей з підписом
    st.image(image, caption='Матриця невідповідностей')
    # Прогнозування ймовірностей на тестових даних
    y_pred = DecisionTree.predict_proba(X_test)[:, 1]
    # Створення графіка Precision-Recall
    PrecisionRecallDisplay.from_predictions(y_test, y_pred)
    # Збереження графіка Precision-Recall
    plt.savefig('PrecisionRecallDisplay_DecisionTree_mark.png')
    # Завантаження графіка Precision-Recall
    image = Image.open('PrecisionRecallDisplay_DecisionTree_mark.png')
    # Виведення графіка Precision-Recall з підписом
    st.image(image, caption='Графік Precision-Recall')
    # Розрахунок значень False Positive Rate і True Positive Rate для ROC-кривої
    fpr, tpr, _ = metrics.roc_curve(y_test, y_pred)
    plt.clf()  # Очистка поточної фігури
    plt.plot(fpr, tpr)  # Побудова ROC-кривої
    plt.ylabel('True Positive Rate')  # Підпис вісі Y
    plt.xlabel('False Positive Rate')  # Підпис вісі X
    # Збереження графіка ROC-AUC
    plt.savefig('ROCAUC_DecisionTree_mark.png')
    # Завантаження графіка ROC-AUC
    image = Image.open('ROCAUC_DecisionTree_mark.png')
    # Виведення графіка ROC-AUC з підписом
    st.image(image, caption='Крива помилок AUC-ROC')

# Відкриття розгортки з заголовком "Приклад залежностей впливу"
with st.expander("Приклад залежностей впливу"):
    headers = ["name", "score"]  # Заголовки стовпців
    # Сортування значень впливу за спаданням
    values = sorted(zip(
        X_train.columns, xgb_classifier.feature_importances_), key=lambda x: x[1] * -1)
    # Створення DataFrame з впливом критеріїв
    xgb_feature_importances = pd.DataFrame(values, columns=headers)
    # Позиції для стовпчиків на графіку
    x_pos = np.arange(0, len(xgb_feature_importances))
    # Побудова стовпчикової діаграми впливу
```

```
plt.bar(x_pos, xgb_feature_importances['score'])
# Позначення назв критеріїв на осі X
plt.xticks(x_pos, xgb_feature_importances['name'])
# Поворот назв критеріїв для кращої читабельності
plt.xticks(rotation=90)
# Заголовок графіку
plt.title('Важливість критеріїв')
# Виведення тексту
st.write("Графік важливості критеріїв")
# Збереження зображення
plt.savefig('feature_deposit.png', bbox_inches='tight')
# Завантаження зображення
image = Image.open('feature_deposit.png')
st.image(image)  # Виведення зображення
# Виведення тексту
st.write("Залежність депозиту від віку")
df_new = cleaned_df.copy()  # Створення копії даних
# Розділення віку на групи
df_new['age_dependencies'] = pd.qcut(
    df_new['age'], 20, labels=False, duplicates='drop')
# Розрахунок середнього депозиту для кожної групи віку
mean_deposit = df_new.groupby(['age_dependencies'])[
    'deposit_bool'].mean()
plt.figure(figsize=(6, 6))  # Розмір фігури
# Побудова графіку
plt.plot(mean_deposit.index, mean_deposit.values)
plt.xlabel('залежність віку')  # Підпис вісі X
# Підпис вісі Y
plt.ylabel(
    'відсоток укладеного строкового депозиту')
# Збереження зображення
plt.savefig('mean_age_deposit.png')
# Завантаження зображення
image = Image.open('mean_age_deposit.png')
st.image(image)  # Виведення зображення
# Мінімальний вік для відповідної групи
age_min = df_new[df_new['age_dependencies'] == 3]['age'].max()
# Максимальний вік для відповідної групи
age_max = df_new[df_new['age_dependencies'] == 17]['age'].min()
# Текст залежності для молодших віку
age_min_text = f"-людям віком молодше {age_min}"
# Текст залежності для старших віку
age_max_text = f"-людям віком старше {age_max}"
# Виведення тексту
st.write("Згідно отриманого графіку можна зробити висновок, що вигідніше пропонувати відкрити депозит:")
```

```
# Виведення тексту про молодших віку
st.write(age_min_text)
# Виведення тексту про старших віку
st.write(age_max_text)
st.divider()  # Вставка розділювача
# Виведення тексту
st.write("Залежність депозиту від балансу")
df_new_balance = cleaned_df.copy()  # Створення копії даних
# Розділення балансу на групи
df_new_balance['balance_dependencies'] = pd.qcut(
    df_new_balance['balance'], 50, labels=False, duplicates='drop')
# Розрахунок середнього депозиту для кожної групи балансу
mean_deposit = df_new_balance.groupby(['balance_dependencies'])[
    'deposit_bool'].mean()
plt.clf()  # Очистка поточної фігури
# Побудова графіку
plt.plot(mean_deposit.index, mean_deposit.values)
# Підпис вісі X
plt.xlabel('залежність балансу')
# Підпис вісі Y
plt.ylabel(
    'відсоток укладеного строкового депозиту')
# Збереження зображення
plt.savefig('mean_balance_deposit.png')
# Завантаження зображення
image = Image.open('mean_balance_deposit.png')
st.image(image)  # Виведення зображення
# Мінімальний баланс для відповідної групи
min_balance = df_new_balance[df_new_balance['balance_dependencies'] == 34]['balance'].min(
)
# Текст залежності для більших балансів
min_balance_text = f"-людям з балансом більшим {min_balance}"
# Виведення тексту
st.write("Згідно отриманого графіку можна зробити висновок, що вигідніше пропонувати відкрити депозит:")
# Виведення тексту про більші баланси
st.write(min_balance_text)
```

Додаток Б

Ілюстративний матеріал

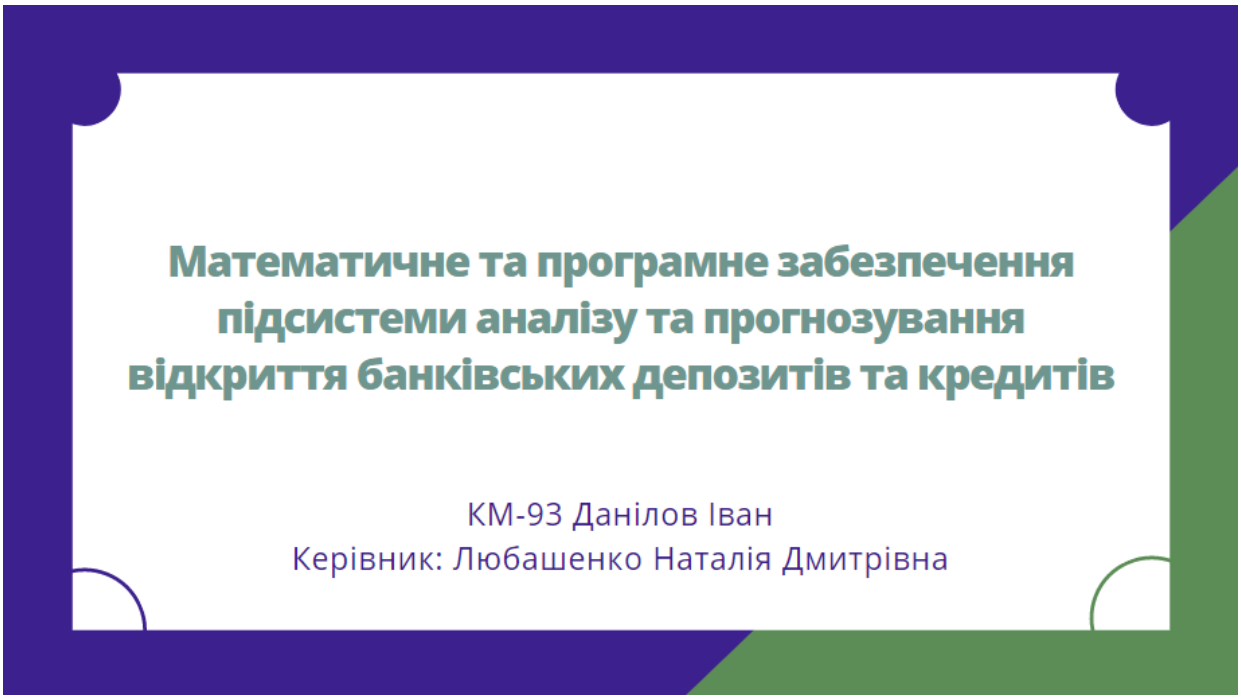


Рисунок Б.1 – Слайд 1

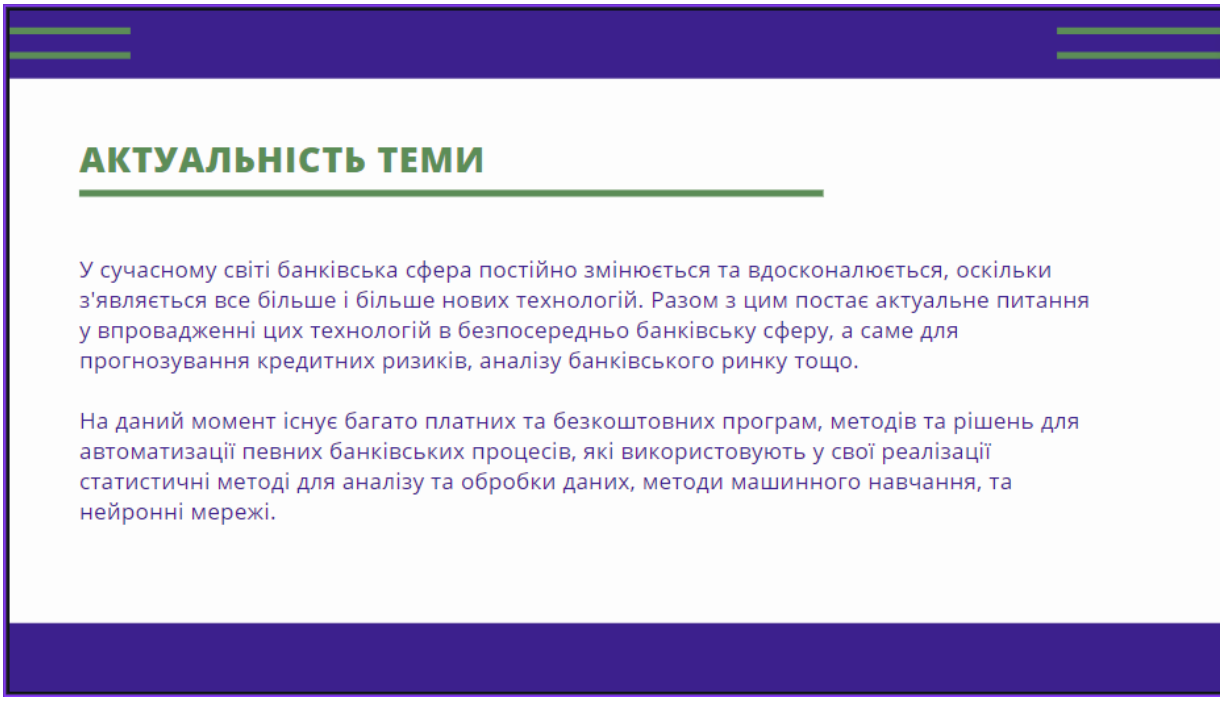


Рисунок Б.2 – Слайд 2

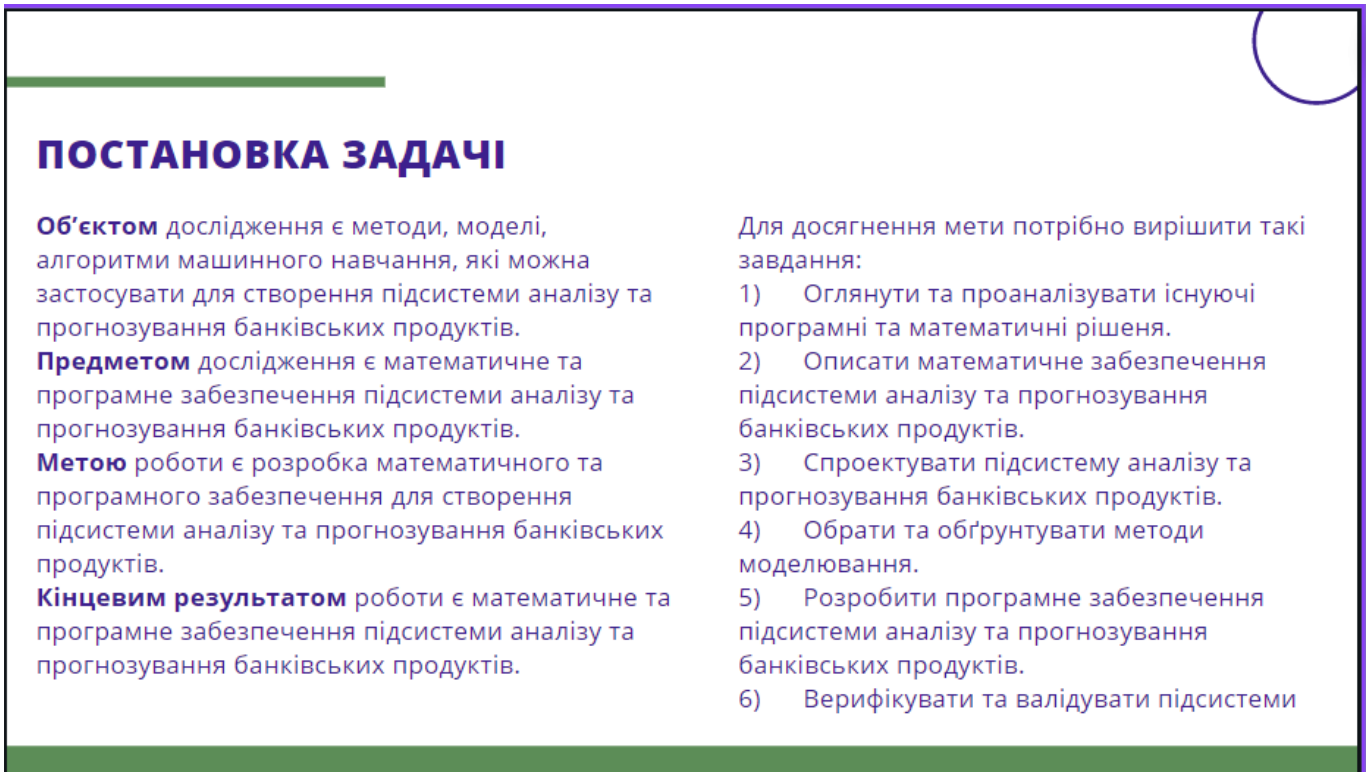


Рисунок Б.3 – Слайд 3

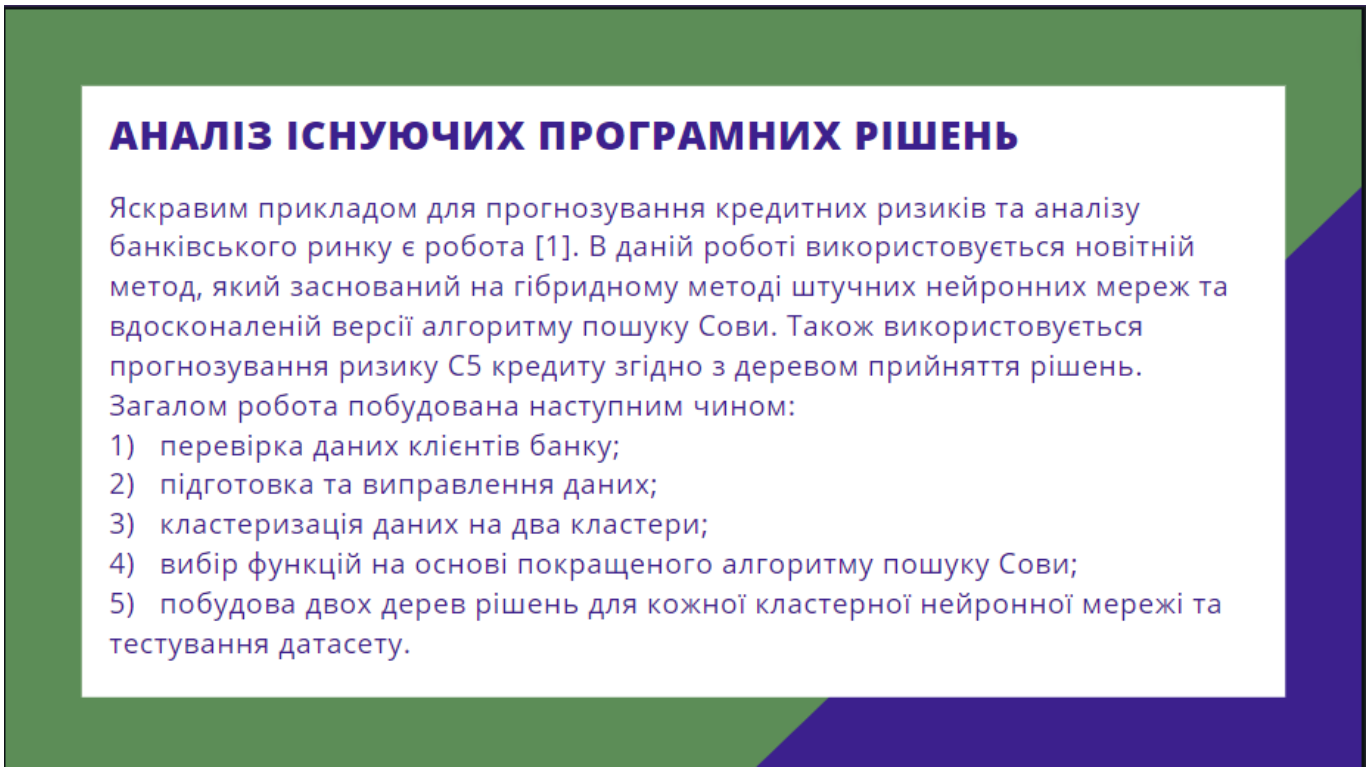


Рисунок Б.4 – Слайд 4

АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

Для розв'язання поставленої задачі більшість компанії застосовує емпіричний підхід, в якому розглядають створені моделі на основі нейронних мереж, а навчання та перевірка моделей використовується для представлення експериментальних результатах, як це зроблено в роботі [2]. В даній роботі розробники прийшли до того, що створена нейронна мережа є методом «чорної скриньки», через що отриманий результат складно обґрунтувати. Додатково автори виділили, що при використанні нейронних мереж для прогнозування кредитних ризиків користувачеві необхідно виконати додаткові кроки у вигляді нормалізації даних та перевірці атрибутів.

Рисунок Б.5 – Слайд 5

АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

Більшість IT-компаній вже створили програмні продукти для аналізу клієнтської бази, прогнозування кредитних ризиків тощо. Однією з них є компанія Moody'sAnalytics [3]. Програмний продукт, створений ними, використовує нейронні мережі для прогнозування фінансових показників. Створені моделі кредитних ризиків активно відстежуються та перевіряються на основі поточних економічних умов та останніх доступних наборів даних для забезпечення високої передбачуваності та точності.

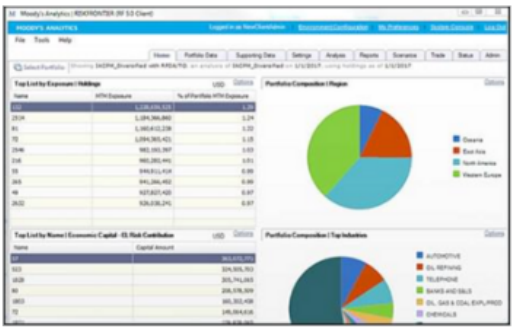


Рисунок Б.6 – Слайд 6

АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

Окрім нейронних мереж, існують програмні рішення, які застосовують методи машинного навчання. Платформа Alteryx Analytics Cloud Platform [4] є саме такою. Дана платформа має вбудовані інструменти для машинного навчання і дозволяє користувачам аналізувати банківські дані, будувати моделі прогнозування тощо

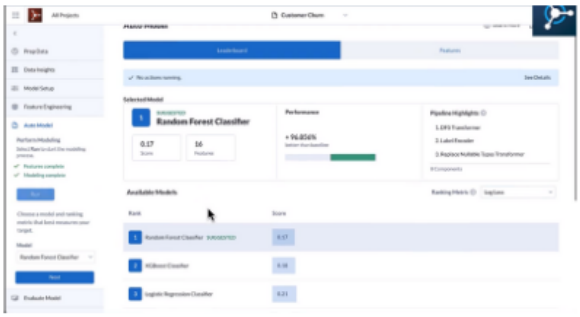


Рисунок Б.7 – Слайд 7

ОБРАНІ МЕТОДИ МАШИННОГО НАВЧАННЯ

- Логістична регресія
- Класифікатор випадкових лісів
- Класифікатор XGBoost
- Класифікатор дерева рішень

Рисунок Б.8 – Слайд 8

ЛОГІСТИЧНА РЕГРЕСІЯ

Логістична регресія - це метод аналізу даних, який використовує математику для пошуку взаємозв'язку між двома факторами даних. Потім вона використовує цей зв'язок для прогнозування значення одного з цих факторів на основі іншого. Прогноз зазвичай має обмежену кількість результатів, наприклад, "так" або "ні".

Переваги:

1. Простота
2. Швидкість
3. Гнучкість
4. Наочність

Рисунок Б.9 – Слайд 9

ЛОГІСТИЧНА РЕГРЕСІЯ

Метод логістичної регресії має два класи:

1. бінарний;
2. мультиноміальний.

У бінарному випадку припускається, що цільова функція y_i приймає значення з множини $\{0, 1\}$ для точки даних i. Після підбору прогнозується ймовірність позитивного класу $P(y_i = 1|X_i)$ як

$$\hat{p}(X_i) = expit(X_i w + w_0) = \frac{1}{1+exp(-X_i w - w_0)} \quad (3.9)$$

Логістична регресія бінарного класу з регуляризаційним членом $r(w)$ мінімізує наступну функцію витрат:

$$\min_w C \sum_{i=1}^{n} \left(-y_i \log\left(\hat{p}(X_i)\right) - (1 - y_i) \log\left(1 - \hat{p}(X_i)\right)\right) + r(w) \quad (3.10)$$

Рисунок Б.10 – Слайд 10

ЛОГІСТИЧНА РЕГРЕСІЯ

Якщо ж бінарний випадок розширити до класів, то отримаємо мультиноміальний випадок. Отже передбачувана ймовірність класів розраховується наступним чином

$$\hat{p}(X_i) = \frac{\exp(X_i W_k + W_{0,k})}{\sum_{l=0}^{K-1} \exp(X_i W_l + W_{0,l})}$$

Логістична регресія мультиноміального класу з регуляризаційним членом r(w) мінімізує наступну функцію витрат:

$$\min_w -C \sum_{i=1}^{n} \sum_{k=0}^{K-1} [y_i = k] \log(\hat{p}(X_i)) + r(w)$$

Рисунок Б.11 – Слайд 11

КЛАСИФІКАТОР ВИПАДКОВИХ ЛІСІВ

Кроки алгоритму випадкового лісу

1) Спочатку необхідно створити підмножини вихідних даних. Для цього зробимо вибірку рядків і вибірку ознак, тобто виберемо рядки і стовпці із заміною і створимо підмножини навчального набору даних;
2) створимо індивідуальне дерево рішень для кожної підмножини, яку ми візьмемо;
3) кожне дерево рішень дасть результат;
4) остаточний результат розглядається на основі голосування більшості, якщо це задача класифікації, і середнього значення, якщо це задача регресії.

Рисунок Б.12 – Слайд 12

КЛАСИФІКАТОР XGBOOST

XGBoost - це алгоритм градієнтного бустінгу, який широко використовується в науці про дані. Це реалізація градієнтного бустінгу, яка розроблена для того, щоб бути високоефективною, гнучкою та портативною.

Алгоритм був розроблений з наступними цілями:

- Бути високоефективним
- бути гнучким
- Бути портативним
- Було доведено, що XGBoost перевершує інші алгоритми машинного навчання в різних завданнях, включаючи класифікацію, регресію та ранжування.

Рисунок Б.13 – Слайд 13

КЛАСИФІКАТОР XGBOOST

XGBoost працює, об'єднуючи кілька слабких учнів, щоб сформувати сильного учня. Слабкий учень - це модель машинного навчання, яка лише трохи краща за випадкове вгадування. Однак, коли слабкі учні об'єднуються, вони можуть сформувати сильного учня, який буде набагато точнішим.

XGBoost працює шляхом навчання декількох дерев рішень. Кожне дерево тренується на підмножині даних, а прогнози з кожного дерева об'єднуються для формування остаточного прогнозу.

XGBoost є вдосконаленням алгоритму GBM. Основна відмінність полягає в тому, що XGBoost використовує більш регуляризовану модель, яка допомагає запобігти надмірному пристосуванню.

Рисунок Б.14 – Слайд 14

КЛАСИФІКАТОР XGBOOST

XGBoost має ряд параметрів, які можна налаштовувати для покращення роботи алгоритму. Найбільш важливими параметрами є

- max_depth: Максимальна глибина дерев рішень.
- eta: Швидкість навчання.
- gamma: Мінімальне зменшення втрат, необхідне для розбиття.
- Підвибірка: Частка навчальних даних, яка використовується для навчання кожного дерева.

Рисунок Б.15 – Слайд 15

КЛАСИФІКАТОР XGBOOST

XGBoost має низку переваг над іншими алгоритмами машинного навчання:

- Він високоефективний.
- Він гнучкий.
- Він портативний.
- Він точний.

Рисунок Б.16 – Слайд 16

КЛАСИФІКАТОР ДЕРЕВА РІШЕНЬ

Класифікатор дерева рішень (Decision Tree Classifier) є алгоритмом, який розбиває дані на менші підмножини засновані на різних критеріях, при цьому кожна підмножина має свою категорію сортування. Кількість об'єктів, що відповідають певному критерію, зменшується з кожним поділом. Процес класифікації закінчується, коли мережа доходить до підмножини з лише одним об'єктом.

Рисунок Б.17 – Слайд 17

КЛАСИФІКАТОР ДЕРЕВА РІШЕНЬ

Переваги Decision Tree:

1) Інтерпретованість моделі, що дозволяє легко розуміти та пояснювати рішення;
2) Легка візуалізація дерева рішень, яка дозволяє як представити саму модель, так і зробити прогноз для окремих тестових об'єктів;
3) Швидкі процеси навчання та прогнозування;
4) Мінімальна кількість параметрів моделі, що сприяє простоті та ефективності;
5) Підтримка як числових, так і категоріальних ознак.

Недоліки Decision Tree:

1) Чутливість дерев до шумів у вхідних даних;
2) Обмеження розділяючої межі, яку побудоване дерево рішень може мати, що на практиці призводить до меншої якості класифікації порівняно з іншими методами.

Рисунок Б.18 – Слайд 18

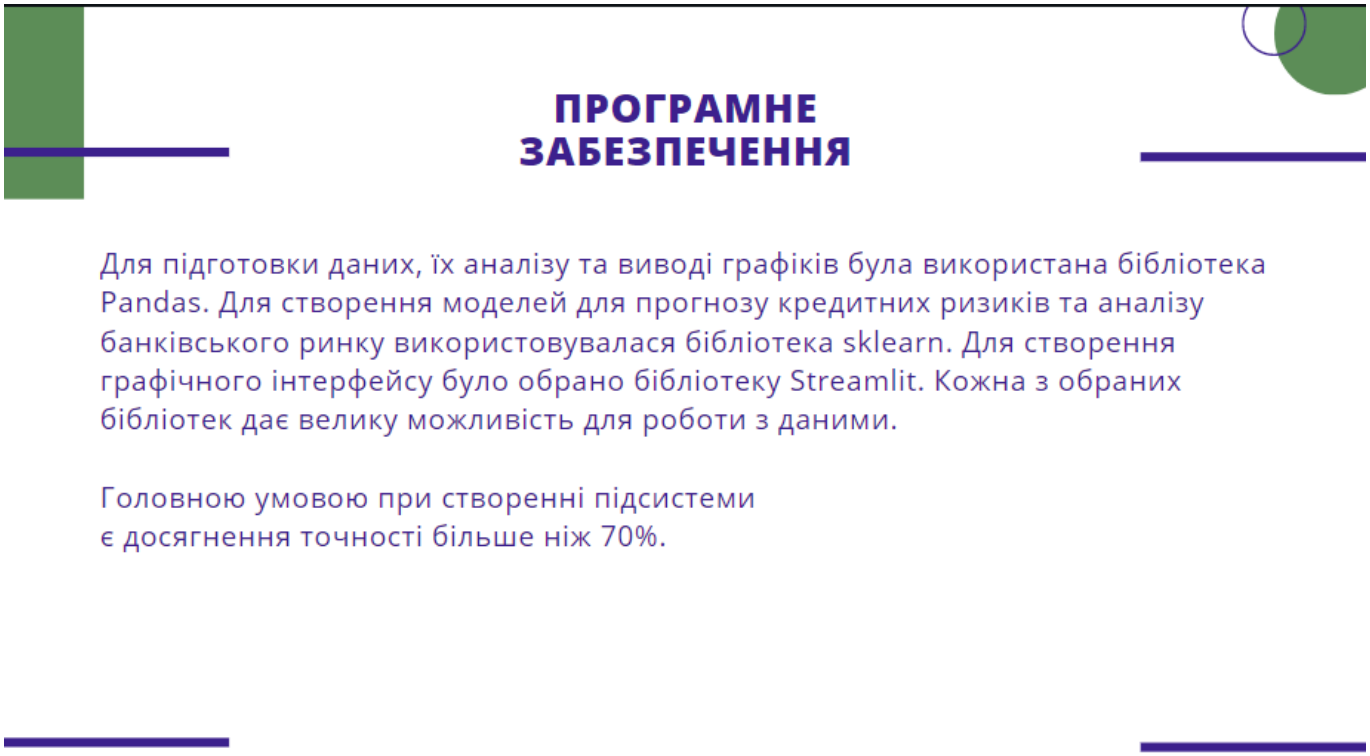


Рисунок Б.19 – Слайд 19

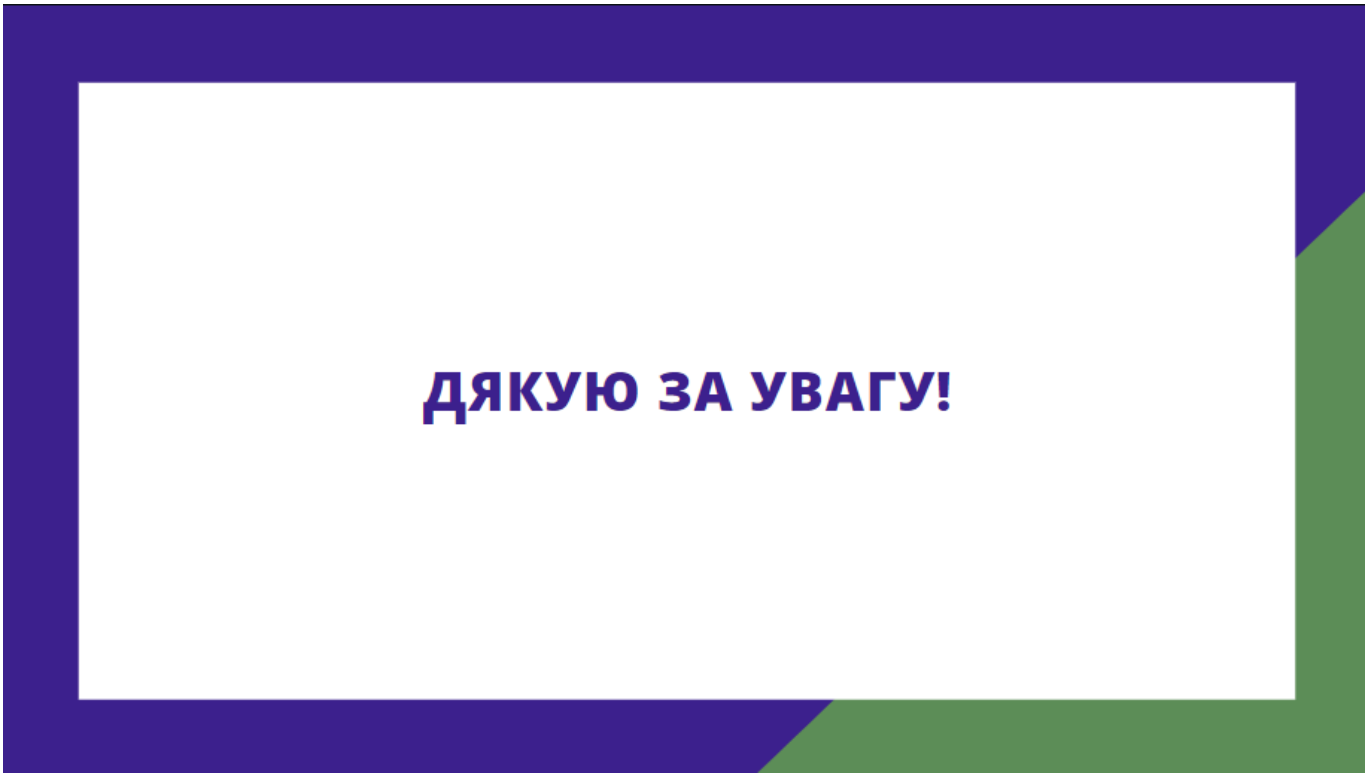


Рисунок Б.20 – Слайд 20