ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Застосування методів машинного навчання в банківській сфері з метою підвищення ефективності прийняття рішення»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
(код, найменування спеціальності)
освітньо-професійної програми «Інженерія програмного забезпечення»
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Роман ФАЙРУШИН
(підпис)

Виконав: здобувач вищої освіти групи ПДМ-63

Роман ФАЙРУШИН

Керівник: Ірина ЗАМРІЙ
д.т.н., доцент

Рецензент: Ольга ЗІНЧЕНКО
д.т.н., доцент

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Файрушину Роману Віталійовичу

1. Тема кваліфікаційної роботи: Застосування методів машинного навчання в банківській сфері з метою підвищення ефективності прийняття рішення.

керівник кваліфікаційної роботи Ірина ЗАМРІЙ д.т.н., доцент,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» 10.2023р. №145.

2. Строк подання кваліфікаційної роботи «29» грудня 2023р.

3. Вихідні дані до кваліфікаційної роботи: демографічна статистика банку та узагальнена інформація за рахунками позичальників

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

1.Дослідження банківської сфери та її даних.

2.Підбір оптимальних методологій та моделювань для дослідження з метою підвищення прогнозування та класифікації з використанням вибраних технік.

3.Аналіз та оцінювання здобутих результатів.

4.Визначення найефективнішої моделі.

5. Перелік графічного матеріалу: *презентація*
1.Методи прийняття рішень в банку.
2.Концепції прийняття рішень.
3.Інтелектуальний аналіз даних моделі.
4.Розробка та підвищення точності моделей.
5.Метрики оцінювання та їх значення.

6. Дата видачі завдання«19» жовтня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз основних методів та підходів, пошук науково-технічної літератури. Планування структури роботи.	19.10-02.11.23	
2	Пошук даних для аналізу, огляд існуючих методів машинного навчання з метою прогнозування кредитних ризиків та прийняття рішення. Написання теоретичної частини роботи.	03.11-11.11.23	
3	Робота над практичною частиною, підготовка даних для роботи з ними, експерементальне моделювання.	12.11-15.11.23	
6	Дослідження технологій машинного навчання	16.11-25.11.23	
7	Написання практичної частини роботи	26.11-02.12.23	
8	Завершення роботи над основною частиною дипломної роботи. Аналіз результатів	03.12-19.12.23	
10	Розробка демонстраційних матеріалів	20.12-29.12.23	

Здобувач вищої освіти _____ Роман ФАЙРУШИН
(підпис)

Керівник
кваліфікаційної роботи _____ Ірина ЗАМРІЙ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 149 стор., 11 табл., 80 рис., 45 джерел.

Об'єкт дослідження – процес прийняття рішень у банківській сфері задачі кредитного скорингу.

Предмет дослідження – методи машинного навчання в процесі прийняття рішень при вирішенні задачі кредитного скорингу у банківській сфері.

У даній науковій роботі виконано детальний аналіз застосування методів машинного навчання у банківській діяльності. Розкрито основні стратегії та техніки машинного навчання, їх переваги та обмеження, а також методологію розробки цих методів, процедури обробки даних і вивчення інформації щодо кредитних операцій для подальшого створення прогностичних моделей. Презентовано огляд передових практик у сфері машинного навчання та проведено аналіз результатів моделювання з метою визначення найефективнішої моделі для розробки системи підтримки прийняття рішень при вирішенні задачі кредитного скорингу у банківській сфері.

КЛЮЧОВІ СЛОВА: МАШИННЕ НАВЧАННЯ, БАНКІВСЬКА СФЕРА, МОДЕЛІ ТА МЕТОДИ МАШИННОГО НАВЧАННЯ, ПРИЙНЯТТЯ РІШЕНЬ, АНАЛІЗ ЯКОСТІ МОДЕЛЕЙ, КРЕДИТНИЙ СКОРИНГ, УПРАВЛІННЯ РИЗИКАМИ, АНАЛІЗ ВЕЛИКИХ ДАНИХ.

ABSTRACT

The text part of the qualification work for obtaining a master's degree: 149 pages, 11 tables, 80 figures, 45 sources.

The purpose of the work enhance the precision of decision-making by applying machine learning techniques to address the challenges of credit scoring within the banking industry.

The object of research is the decision-making mechanism employed in the task of credit scoring by banks.

The subject of research the utilization of machine learning approaches within the decision-making cycle for credit scoring issues in the banking sector.

This research has conducted an in-depth scrutiny of how machine learning methods are applied to the processes of the banking sector. It uncovers the primary strategies and tactics of machine learning, elucidating their benefits and constraints, and discusses the procedural development of these methods. It also examines data handling techniques and the examination of information related to credit transactions to further the development of predictive modeling. The work provides a synopsis of the foremost practices in machine learning, evaluating the outcomes of various models to ascertain the most proficient model for the creation of decision support systems in the banking field.

KEYWORDS: MACHINE LEARNING, BANKING SECTOR, MODELS AND METHODS OF MACHINE LEARNING, DECISION MAKING, ANALYSIS OF MODEL QUALITY, CREDIT SCORING, RISK MANAGEMENT, BIG DATA ANALYSIS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 10
ВСТУП 11
РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ 15
1.1 Визначення процесу прийняття рішень у секторі банківських послуг 15
1.2 Огляд сучасних методів прийняття рішень в банківській сфері 16
1.3 Аналіз існуючих методів та концепцій оцінки 18
1.4 Висновки до розділу 19
РОЗДІЛ 2 МЕТОДИ МАШИННОГО НАВЧАННЯ ДЛЯ АНАЛІЗУ ЕФЕКТИВНОСТІ ПРИЙНЯТТЯ РІШЕНЬ 20
2.1. Аналіз існуючих методів прийняття рішень 20
2.2 Логістична регресія (Logistic Regression) 21
2.3 Дерево ухвалення рішень (Decision Tree) 23
2.4 Випадковий ліс (Random Forest) 26
2.5 K-Nearest Neighbors (KNN) 30
2.6 Опорні векторні машини (SVM) 32
2.7 Екстремальне посилення градієнта (XGBoost) 35
2.8 Нейронні мережі (Deep Neural Network) 38
2.9 Оцінка ефективності та обмеження методів 40
2.10 Метрики оцінювання якості моделей 41
2.11 Висновки до розділу 47
РОЗДІЛ 3 СИСТЕМА ПРИЙНЯТТЯ РІШЕНЬ ДЛЯ ПРОГНОЗУВАННЯ СХВАЛЕННЯ КРЕДИТІВ 48
3.1 Постановка проблеми 48
3.2 Перетворення бізнес-проблеми банку в постановку задачі в розрізі Data Science / Machine Learning problem 49
3.3 Генерація гіпотези 49
3.4 Збір даних 50
3.5 Дослідницький аналіз даних 52
3.6 Аналіз одиничних змінних у дослідженні 56
3.7 Попередня обробка даних 76

3.8 Розробка та оцінка моделі 80
3.9 Висновки до розділу 111
ВИСНОВКИ 113
ПЕРЕЛІК ПОСИЛАНЬ 114
ДОДАТКИ 116
ДОДАТОК А. Лістинг програми 116
ДОДАТОК Б. Навчальний набір змінних 130
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація) 131

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ML – Machine Learning

KNN – K-Nearest Neighbors

SVM – Support Vector Machine

LOOCV – Leave One Out Cross Validation

EMI – Equated Monthly Installment

TP – True Positive

TN – True Negative

FP – False Positive

FN – False Negative

ВСТУП

Актуальність теми дослідження. У сучасному глобальному фінансовому кліматі банківський сектор виступає стовпом стабільності та довіри. Одним із ключових аспектів, які керують успішною діяльністю цього сектору, є прийняття рішень. Прийняття рішень в банківській сфері охоплює як стратегічні рішення високого рівня, так і щоденні оперативні рішення, такі як затвердження кредитів, управління ризиками, виявлення шахрайства та сегментація клієнтів.

Традиційно ці рішення були керовані людським судженням, підкріпленим багаторічним досвідом та системами на основі правил. Однак цей метод прийняття рішень в банківському секторі має свої обмеження. Він може бути схильний до упередженості, невідповідностей та помилок, що потенційно може призвести до значних фінансових втрат або пропущених можливостей. Більше того, з урахуванням постійного зростання обсягів даних, які генеруються в банківському секторі, людське судження може бути недостатнім для ефективного аналізу та прийняття рішень.

З постійним збільшенням складності фінансових продуктів та постійним потоком даних, традиційні методи прийняття рішень у банківській сфері борються з точною та своєчасною обробкою інформації. Вони не можуть ефективно справлятися з великими наборами даних, ні вони не можуть ефективно адаптуватися до швидко змінюваного фінансового середовища.

Як і продовжують розвиватися та ставати все складнішими фінансові ринки, банківська сектор повинен модернізувати свої практики прийняття рішень, щоб забезпечити своє лідерство. Ця потреба ще більше посилюється зростаючим попитом на персоналізовані банківські послуги. Клієнти очікують послуг, що націлені на їх конкретні потреби, і банки знаходяться під тиском, щоб своєчасно приймати обґрунтовані рішення, щоб задовольнити ці потреби.

Банківський сектор також стикається з посиленням регуляторного контролю. Регулятори очікують, що в банках будуть створені надійні системи для управління ризиками та прийняття рішень. Традиційні системи прийняття рішень можуть бути

недостатніми для задоволення цих зростаючих регуляторних вимог, особливо в областях, таких як правила "Anti-Money Laundering" (AML) та "Know Your Customer" (KYC).

Тому є значна потреба в поліпшенні методів прийняття рішень в банківському секторі. Тут на сцену виходить застосування машинного навчання. Машинне навчання, з його здатністю обробляти великі обсяги даних, виявляти шаблони та робити прогнози, пропонує обіцяюче рішення для цих викликів. Застосовуючи техніки машинного навчання, банки можуть покращити ефективність прийняття рішень, зменшити помилки, краще управляти ризиком, і в кінцевому рахунку, покращити задоволеність клієнтів.

У складному та швидкому світі банківської справи прийняття рішень є критичним процесом, який значно впливає на оперативну ефективність, управління ризиками та задоволеність клієнтів. Однак сучасні практики прийняття рішень у банківському секторі все більше виявляються недостатніми в ряді областей.

Основна проблема полягає у залежності від традиційних методів прийняття рішень, які в значній мірі залежать від людського судження та систем, основаних на правилах. Хоча ці методи до певної міри були функціональними, вони мають обмеження у роботі з потоком даних, що генеруються, що призводить до проблем, таких як затримки в прийнятті рішень, неефективність, невідповідності та помилки. Цей центрований навколо людини підхід також має тенденцію вносити упередженість, суб'єктивність та потенціал для маніпуляцій, що впливає на справедливість та цілісність рішень.

Крім того, ці традиційні методи виявляються неспроможними встигати за динамічним фінансовим середовищем, враховуючи швидкі зміни умов ринку та регулятивного ландшафту. Нездатність швидко адаптуватися до цих змін та прогнозувати потенційні зміни призводить до пропущених можливостей та збільшення виставлення ризику.

Більше того, зі зростанням очікувань клієнтів щодо персоналізованих послуг, нездатність традиційних систем прийняття рішень надавати спеціалізовані рішення

та їхній недостаток чутливості до індивідуальних потреб клієнтів стає все більш проблематичною.

Отже, проблема, яку ця дисертація намагається вирішити, - це потреба в удосконаленому процесі прийняття рішень у банківському секторі.

Метою цієї магістерської роботи є дослідження можливостей застосування методів машинного навчання в банківській сфері для покращення процесів прийняття рішень, зменшення помилок, кращого управління ризиками та підвищення задоволеності клієнтів.

Для досягнення поставленої мети необхідно розв’язати наступні завдання:

1. Аналіз сучасного стану процесів прийняття рішень у банківському секторі.
2. Дослідження потенційних методів машинного навчання, які можуть бути застосовані в банківській сфері.
3. Оцінка ефективності впровадження цих методів на конкретних прикладах та кейсах.
4. Аналіз викликів та обмежень, пов'язаних з інтеграцією машинного навчання в банківські системи.

Об'єктом дослідження є процеси прийняття рішень у банківській сфері, зокрема в контексті кредитування, управління ризиками, виявлення шахрайства та персоналізації послуг для клієнтів.

Предметом дослідження є застосування методів машинного навчання в процесах прийняття рішень у банківській сфері.

Теоретична основа включає аналіз наукових праць, статей та досліджень, які стосуються машинного навчання, банківської аналітики, фінансових технологій та управління ризиками.

Методологія дослідження охоплює аналітичний підхід до вивчення літератури, кейс-стаді, порівняльний аналіз, а також емпіричні методи для оцінки впровадження та ефективності методів машинного навчання.

Роботу було апробовано на конференції «Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії» (01- 03 червня 2023 року в м. Київ, Україна).

Стаття:

Жебка В.В., Файрушин Р.В., EN: ”Application of machine learning methods in the banking sector to increase the efficiency of decision-making” UA: “Застосування методів машинного навчання в банківській сфері з метою підвищення ефективності прийняття рішення”

Науковий журнал “Телекомунікаційні та інформаційні технології” 2023. №2 (79)

Тези доповідей:

1. Жебка В.В., Файрушин Р.В., ”Comparison of existing loan decision-making systems based on machine learning methods” // Міжнародна науково-практична конференція «Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії» – Київ: ДУТ, 2023. С. 147-149.
2. Жебка В.В., Файрушин Р.В., “Disadvantages of existing decision-making systems in the banking sector based on machine learning methods” // Науково-практична конференція «Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії» – Київ: ДУТ, 2023. С. 150-152.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Визначення процесу прийняття рішень у секторі банківських послуг

Прийняття рішень в банківській сфері є ключовим процесом, який впливає на всі аспекти банківської діяльності. Цей процес охоплює ряд важливих функцій, від оцінки кредитоспроможності клієнтів до управління ризиками та інвестиційним портфелем. Ефективність прийняття рішень у банківській сфері безпосередньо впливає на фінансову стабільність та прибутковість банку.

З появою та розвитком технологій машинного навчання банківський сектор отримав нові інструменти для оптимізації процесів прийняття рішень. Методи машинного навчання дозволяють аналізувати великі обсяги даних, виявляти складні шаблони та прогнозувати результати, що раніше було недоступно для традиційних аналітичних методів.

Основною перевагою застосування машинного навчання в банківській сфері є його здатність до обробки та аналізу великих даних, які є критично важливими для рішень, що базуються на даних. Це включає оцінку кредитного ризику, фрод-моніторинг, оптимізацію інвестиційних стратегій та персоналізацію банківських послуг.

Використання машинного навчання також сприяє зниженню ризику помилок, які можуть виникати через суб'єктивність або обмеження людського фактору. Алгоритми машинного навчання можуть ефективно ідентифікувати візерунки та аномалії, які можуть вказувати на потенційний ризик або можливості для банку.

Однак, разом з перевагами, існують і певні виклики та обмеження, пов'язані з впровадженням машинного навчання. До них відносяться питання конфіденційності даних, потреба в великих наборах даних для тренування моделей, а також ризик упередженості в алгоритмах, що може призвести до неправильних рішень.

1.2 Огляд сучасних методів прийняття рішень в банківській сфері

Сучасний банківський сектор невпинно еволюціонує, зосереджуючи свою увагу на оптимізації та автоматизації процесів прийняття рішень, особливо у контексті кредитування. Це включає в себе застосування новітніх технологій та методів машинного навчання для аналізу великих даних. Вони в свою чергу поділяються на класифікації. Основні класифікації методів прийняття рішень у банківській сфері представлені в таблиці 1.

Таблиця 1.2.1

Класифікацій методів прийняття рішень та їх застосування

№	**Класифікація**	**Методи**	**Застосування**
1	**Антифродові системи**	Використовуються нейронні мережі, методи кластеризації, anomaly detection	Виявлення підозрілих транзакцій, запобігання шахрайству
2	**Управління активами та пасивами**	Регресійний аналіз, часові ряди, прогнозування	Оптимізація портфеля інвестицій, прогнозування ринкових тенденцій
3	**Персоналізація послуг**	Системи рекомендацій, кластеризація, глибинне навчання	Пропозиція персоналізованих фінансових продуктів та послуг
4	**Оптимізація внутрішніх процесів**	Автоматизація процесів за допомогою RPA (Robotic Process Automation), NLP для обробки текстових даних	Підвищення ефективності внутрішніх операцій.
5	**Ризик-менеджмент**	Статистичний аналіз, прогнозування, симуляції	Оцінка кредитних ризиків, управління ризиками портфеля.

Продовження таблиці 1.2.1

Класифікацій методів прийняття рішень та їх застосування

№	Класифікація	Методи	Застосування
6	**Виявлення відмивання грошей**	Графові алгоритми, аналіз поведінкових патернів	Виявлення підозрілих фінансових потоків і зв'язків
7	**Кредитний скоринг**	Використовуються алгоритми класифікації, такі як рішучі дерева, випадкові ліси, градієнтний бустинг	Автоматизація процесу оцінки кредитоспроможності клієнтів, мінімізація ризиків

Вданій робі буде розглянуто класифікацію кредитного скорингу який саме використовується в автоматизації процесу оцінки кредитоспроможності клієнтів, мінімізації ризиків. В основі цієї класифікації є використання алгоритмів для автоматизації оцінки кредитоспроможності. До інших класифікацій відносяться антифродові системи, управління активами та пасивами, персоналізація послуг, оптимізація внутрішніх процесів, ризик-менеджмент та виявлення відмивання грошей. Кожна з цих класифікацій включає в себе методи та інструменти, які застосовуються для конкретних задач. Наприклад, антифродові системи використовують нейронні мережі та **anomaly detection** для виявлення та запобігання шахрайству.

Кожна з цих класифікацій відіграє важливу роль у стратегічному плануванні та щоденному функціонуванні фінансових інститутів.

1.3 Аналіз існуючих методів та концепцій оцінки

В процесі прийняття рішень, особливо в контексті кредитування, критичним етапом є процес оцінювання, який відбувається завдяки існуючим підходам та концепціям які використовуються при прийнятті рішень у банківському секторі.

Оцінка включає систематичний процес ідентифікації ризикових факторів та їх кількісного виміру, поєднуючи кількісні та якісні методи аналізу.

Якісний аналіз спрямований на визначення причин ризиків і включає ідентифікацію потенційних ризикових зон, створення списку ризиків, властивих бізнесу, та прогнозування їх можливих наслідків. Кількісний аналіз, у свою чергу, використовується для оцінки ризиків, що можуть виникнути внаслідок прийняття рішень, і базується на інструментах таких дисциплін, як теорія ймовірностей, математична статистика та дослідження операцій представлена на рисунку 1.3.

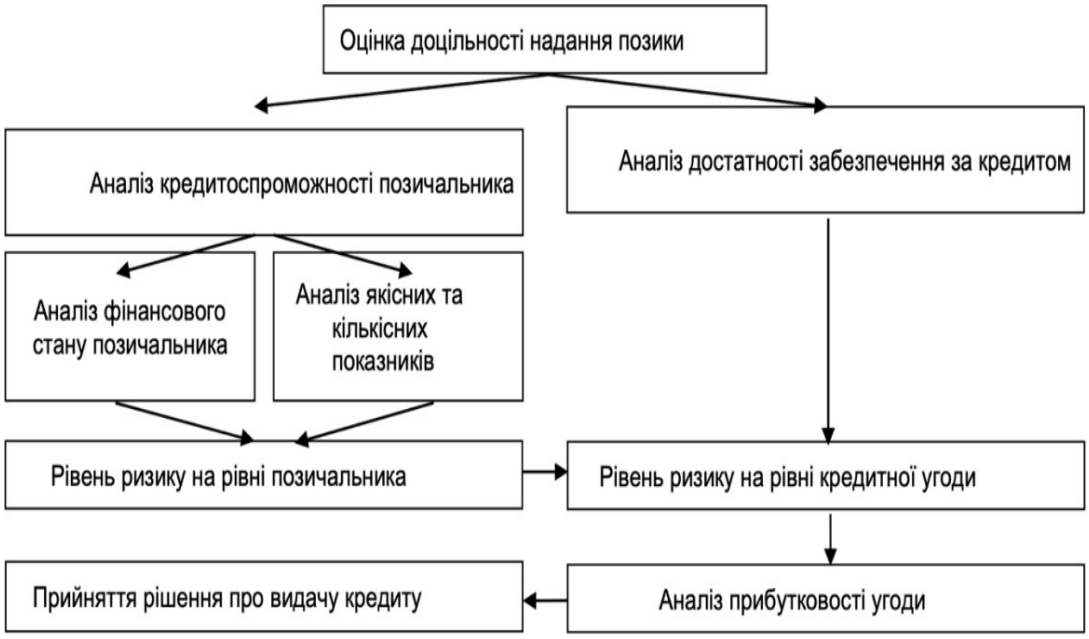


Рис. 1.3. Методологія оцінки обґрунтованості кредитних рішень

У контексті роздрібного кредитування існують два основні методи оцінки кредитоспроможності:

- Експертна оцінка, заснована на суб'єктивному судженні експерта, та кредитний скоринг, який є об'єктивною системою оцінки, базованою

на статистичній моделі, що аналізує історичні дані. Експертна оцінка включає аналіз опитувань та використовує інтуїцію та досвід експертів, в той час як кредитний скоринг класифікує клієнтів на "добрих" та "поганих" і використовує набір характеристик для оцінки рівня ризику.

- Скоринг кредитних запитів включає аплікаційний аналіз, який застосовується на етапі подачі заявки на кредит для прогнозування ризику неплатоспроможності, та поведінковий аналіз, який використовується після видачі кредиту для оцінювання ризику дефолту з боку клієнта, який уже користується кредитом. Ці методи дозволяють банкам прогнозувати ймовірність дефолту та застосовувати заходи до клієнтів з високим ризиком.

1.4 Висновки до розділу

Управління банківськими ризиками є складним завданням через специфіку кожної банківської операції та потенційні негативні наслідки накопичення ризиків для банків.

Перша частина дослідження була присвячена аналізу сфери прийняття рішень у банківському секторі. Вона включала розгляд таких понять, як 'прийняття рішення', а також ознайомлення з передовими методами прийняття рішень в банківській діяльності, систематизацією та підходами до оцінки процесів прийняття рішень. Зіткнувшись із складнощами в ідентифікації конкретних аспектів, що супроводжують банківські операції, виявилося, що вони мають тенденцію до акумуляції в портфелях банку, що може призвести до небажаних економічних наслідків у процесі прийняття рішень.

Основний акцент роботи зосереджено на покращенні показників методів машинного навчання для кредитному та методах його оцінки з метою запобігання збиткам.

2 МЕТОДИ МАШИННОГО НАВЧАННЯ ДЛЯ АНАЛІЗУ ЕФЕКТИВНОСТІ ПРИЙНЯТТЯ РІШЕНЬ

2.1. Аналіз існуючих методів прийняття рішень

Аналіз існуючих методів прийняття рішень та їх застосування в банківській сфері включає вивчення різних підходів та технологій, які використовуються для оцінки кредитоспроможності клієнтів. Ось декілька ключових аспектів цієї теми:

Традиційні методи скорингу: методи ґрунтуються на історичних даних про кредитну історію клієнта, його доходи, зобов'язання, вік, стаж роботи тощо. Вони використовують статистичні моделі, такі як логістична регресія, для прогнозування ймовірності невиплати кредиту.

Скоринг на основі машинного навчання: сучасні методи включають використання алгоритмів машинного навчання, таких як логічна регресія, дерево ухвалення рішень, випадковий ліс, к-найближчих сусідів, опорні векторні машини тощо . Ці методи здатні аналізувати більш складні та неструктуровані дані для точнішого прогнозування.

Альтернативні дані: Крім традиційних фінансових показників, банки все більше використовують альтернативні дані, такі як інформація з соціальних мереж, поведінкові дані, дані з мобільних додатків та інше для оцінки кредитоспроможності.

Регулятивні та етичні міркування: Важливим аспектом є дотримання нормативних вимог і етичних принципів. Банки повинні забезпечити, що їх методи скорингу не дискримінують клієнтів на основі раси, статі чи інших характеристик.

Виклики та перспективи: Одним з викликів є забезпечення точності та надійності скорингових систем. Також важливо враховувати зміни в економічному середовищі та адаптувати моделі скорингу до цих змін.

2.2 Логістична регресія (Logistic Regression)

Логістична регресія - це метод прогностичного аналізу, що описує дані та відносини між однією залежною бінарною змінною і однією чи кількома незалежними змінними. Вона оцінює ймовірність того, що дана вибірка потрапляє в одну з двох категорій, що є модифікацією моделі лінійної регресії, адаптованої для класифікаційних завдань. По суті, логістична регресія моделює шанси приналежності зразка до певної категорії. Хоча сама модель видає лише ймовірності, класифікація зразка до класу залежить від обраного порога ймовірності. Коефіцієнти цієї моделі, які можуть бути зваженими, вказують на значущість кожної ознаки у прогнозуванні результату.

Особливо логістична регресія підходить, коли класи у наборі даних лінійно віддільні, оскільки вона створює гладку, лінійну межу прийняття рішень між ними. Вона моделює зв'язок між незалежними змінними забезпечуючи, що прогнозовані ймовірності знаходяться в межах від 0 до 1. Вірогідність можливості на виникнення події y=1, що може представляти "хорошого" позичальника як кредитоспроможного, визначається так:

$$P_{r=}\{y = 1|X\} = g(z) \tag{2.1}$$

де g(z) є логістичною функцію

Оскільки Y може приймати тільки один з двох бінарних значень, 0 або 1, ймовірність події y = 0, що може представляти не кредитоспроможного позичальника, є (2.2):

$$P_{r=}\{y = 0|X\} = 1 - g(z) \tag{2.2}$$

Як наслідок отримуємо нову модель для подальших розрахунків (2.3):

$$log = \frac{P_r(y = 1|X)}{P_r(y = 0|X)} = \frac{g(z)}{1 - g(z)} = w_0 + w_1X_1 + \cdots + w_nX_n \quad (2.3)$$

де w – являється вектором регресивних параметрів, що розраховується шляхом максимізації функції правдоподібності.

Логістична регресія є фундаментальною технікою у статистиці та машинному навчанні, яка в основному використовується для завдань бінарної класифікації.

Основна концепція:

- Логістична регресія моделює ймовірність того, що двійкова цільова змінна набере одне з двох можливих значень (зазвичай 0 і 1).
- Він використовує логістичну функцію (сигмоїдну функцію) для моделювання зв'язку між змінними (ознаками) предиктора та ймовірністю приналежності до певного класу.

Ключові особливості:

- Сигмоїдна функція: логістична функція (сигмоїдна) відображає будь-яке дійсне число на значення від 0 до 1, яке представляє ймовірність позитивного класу.
- Лінійна границя прийняття рішень: логістична регресія створює лінійну межу прийняття рішень, яка розділяє два класи в просторі ознак. Ця межа визначається коефіцієнтами, вивченими під час навчання.

Плюси:

- **Інтерпретація:** логістична регресія забезпечує легко інтерпретовані результати. Коефіцієнти можна безпосередньо інтерпретувати в термінах впливу кожної ознаки на ймовірність позитивного класу.
- **Ефективність:** він ефективний з точки зору обчислень і підходить для великих наборів даних.
- **Регуляризація:** такі методи, як регулярізація L1 і L2, можуть бути застосовані для запобігання переобладнанню.
- **Імовірнісний вихід:** логістична регресія надає ймовірності, які можуть бути корисними під час ранжування або встановлення пріоритетів прогнозів.

Таким чином, логістична регресія є основоположним алгоритмом для бінарної класифікації, який цінується за його простоту, можливість інтерпретації та ефективність. Він забезпечує імовірнісну структуру для моделювання зв'язку між ознаками та класовими ймовірностями.

2.3 Дерево ухвалення рішень (Decision Tree)

Алгоритм дерево ухвалення рішень у машинному навчанні є дуже універсальним, оскільки він ефективно обробляє як завдання класифікації, так і регресії. Ця універсальність досягається шляхом побудови дерев рішень. Дерево ухвалення рішень широко застосовується в банківській галузі завдяки їх високій точності та здатності створювати статистичні моделі, які можна пояснити простою мовою.

У дереві рішень:

- Вузли представляють критерії або діагностику. Ці вузли є точками в дереві, де приймається рішення на основі значення певної функції.
- Ребра або гілки представляють рішення або правила. Кожна гілка означає шлях або маршрут, обраний на основі результату критерію або діагностики.

- Листя представляють результати, які можуть бути категоричними або постійними значеннями. Це кінцеві результати або прогнози моделі.

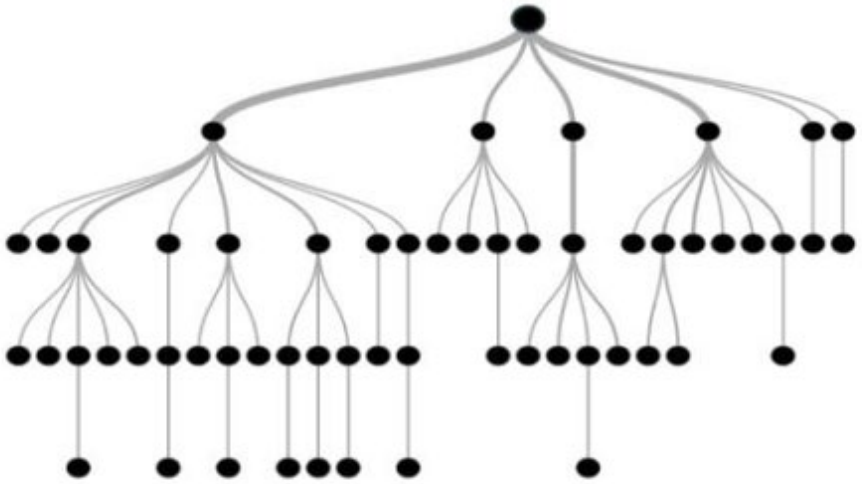


Рис. 2.1. Блок-схема для алгоритму дерева рішень

Сила дерев рішень полягає в їх здатності розбивати складні процеси прийняття рішень на ряд простих кроків, які можна інтерпретувати. Це робить їх особливо цінними в таких галузях, як банківська справа, де прозорість і можливість інтерпретації моделей мають вирішальне значення. Банківські спеціалісти та зацікавлені сторони можуть легко зрозуміти та довіряти рішенням, прийнятим за допомогою моделі дерева ухвалення рішень, оскільки їх можна виразити простою мовою та візуалізувати у вигляді деревовидної структури.

Таким чином, дерево ухвалення рішень є універсальним інструментом машинного навчання, особливо в банківському секторі, завдяки своїй точності, прозорості та здатності представляти складні процеси прийняття рішень у зрозумілій формі.

Дерево ухвалення рішень – це тип алгоритму машинного навчання, який працює шляхом створення моделі рішень на основі фактичних даних. У контексті схвалення кредиту дерево рішень може розглядати дохід заявника, його кредитний рейтинг, історію зайнятості та інші відповідні фактори, приймаючи рішення в

кожному вузлі дерева, доки не прийме остаточне рішення. Цей метод є прозорим і зрозумілим, але іноді може призвести до спрощення рішень.

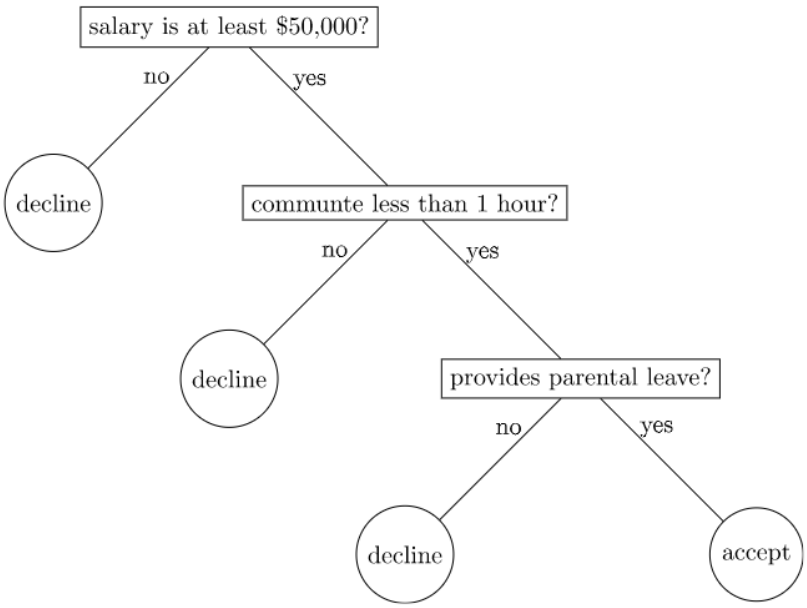


Рис. 2.2. Ілюстрація дерева рішень на основі фактичних даних

Дерево ухвалення рішень приймає рішення, розбиваючи дані на основі значень ознак. Зазвичай використовується набір правил, які можна візуалізувати в деревоподібній структурі для прийняття рішень.

Ось спрощений приклад того, як можна структурувати дерево рішень на основі нашого набору даних:

```
1. Is the Credit Score ⩾ 700?
   - Yes: Go to 2.
   - No: Go to 3.
2. Has the client defaulted before (Number of Defaults > 0)?
   - Yes: Loan is denied.
   - No: Loan is approved.
3. Is the Annual Income ⩾ $60,000?
   - Yes: Go to 4.
   - No: Loan is denied.
4. Are the Number of Open Accounts > 5?
   - Yes: Loan is denied.
   - No: Loan is approved.
```

Рис. 2.3. Структуризація дерева рішень на основі нашого набору даних.

2.4 Випадковий ліс (Random Forest)

Випадковий ліс — це потужний метод ансамблевого навчання, який використовується як для завдань класифікації, так і для регресії. Він заснований на ідеї поєднання кількох дерев рішень для підвищення точності прогнозування та зменшення надмірного оснащення.

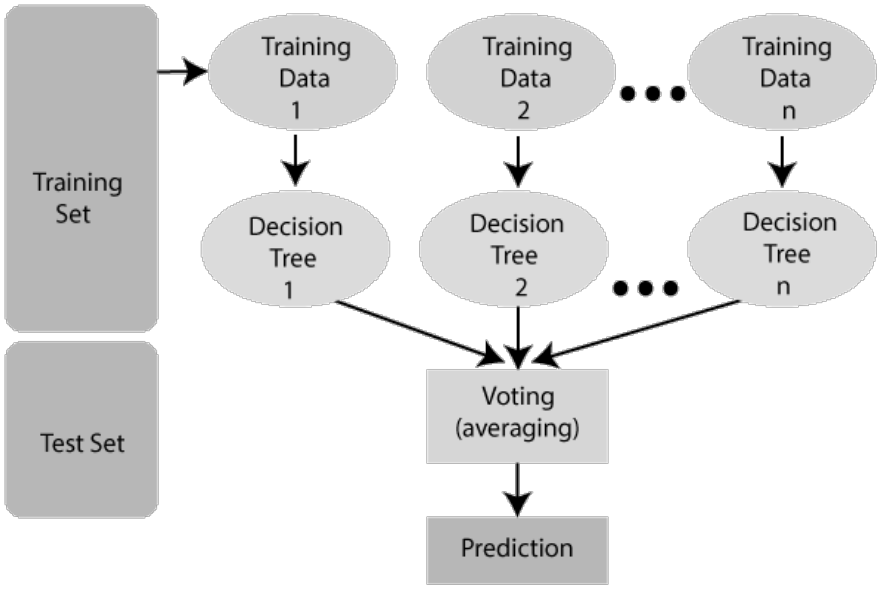


Рис. 2.4. Навчання алгоритму Random Forest

Навчання в ансамблі:

- **Випадковий ліс** (Random Forest) належить до сімейства методів ансамблевого навчання, що означає, що він поєднує прогнози кількох моделей машинного навчання для більш точних і надійних прогнозів.
- У випадку випадкового лісу він поєднує передбачення колекції дерев рішень.

Ключові особливості:

- **Дерево ухвалення рішень:** Випадковий ліс складається з великої кількості дерев рішень, як правило, сотень або навіть тисяч.

- **Випадкова вибірка:** під час процесу навчання кожне дерево навчається на випадковій підмножині навчальних даних. Це відоме як початкове завантаження.
- **Випадковість функцій:** під час створення кожного дерева розглядається випадкова підмножина ознак для розділення на кожному вузлі. Це допомагає прикрасити дерева та покращити узагальнення.
- **Голосування:** для класифікаційних завдань остаточний прогноз визначається більшістю голосів серед окремих прогнозів дерева. Для завдань регресії це зазвичай середнє значення прогнозів дерева.

Плюси:

- **Висока точність:** Випадковий ліс має тенденцію створювати високоточні прогнози, часто перевершуючи окремі Дерево ухвалення рішень.
- **Зменшене переобладнання:** поєднання кількох дерев і випадковість функцій зменшує переобладнання, роблячи модель більш стійкою до шуму в даних.
- **Обробляє великі набори даних:** Random Forest може обробляти великі набори даних із численними функціями.
- **Важливість функції:** забезпечує міру важливості функції, допомагаючи визначити, які функції мають найбільший вплив на прогнози.
- **Помилка Out-of-Bag (OOB) :** він має вбудований метод для оцінки продуктивності моделі без необхідності окремого набору перевірки, який називається помилкою OOB.

Таким чином, Random Forest — це надійна та універсальна методика ансамблевого навчання, яка відома своєю високою точністю прогнозування та здатністю обробляти складні набори даних. Це популярний вибір у машинному навчанні як для завдань класифікації, так і для регресії.

Випадковий ліс є продовженням дерева рішень. Він працює, створюючи безліч дерев рішень і виводячи клас, який є модою класів окремих дерев. Усереднюючи

рішення багатьох різних дерев, випадкові ліси, як правило, більш точні та менш схильні до переобладнання, ніж окремі дерева рішень.

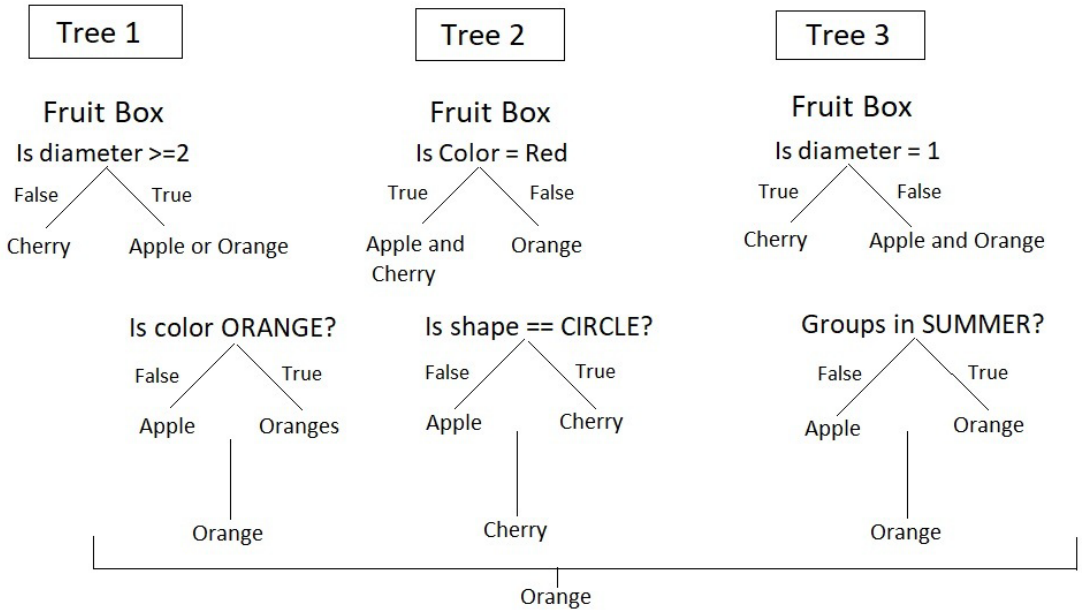


Рис. 2.4. Ілюстрація випадкових лісів

Випадковий ліс — це набір дерев рішень, які були навчені на різних підмножинах даних. Кожне дерево в лісі приймає своє рішення, а остаточне рішення приймається більшістю голосів за рішення, прийняті всіма деревами в лісі.

Створюючи кожне дерево рішень, алгоритм Random Forest враховує лише випадкову підмножину ознак на кожному поділі в процесі навчання, що допомагає збільшити різноманітність дерев і зменшує ймовірність перенавчання.

Ось спрощений приклад того, як Random Forest може прийняти рішення про позику на основі нашого набору даних. Припустимо, що ми працюємо з випадковим лісом, що складається з трьох дерев рішень. Зауважте, що реальні програми часто включають сотні або тисячі дерев і набагато складніші процеси прийняття рішень.

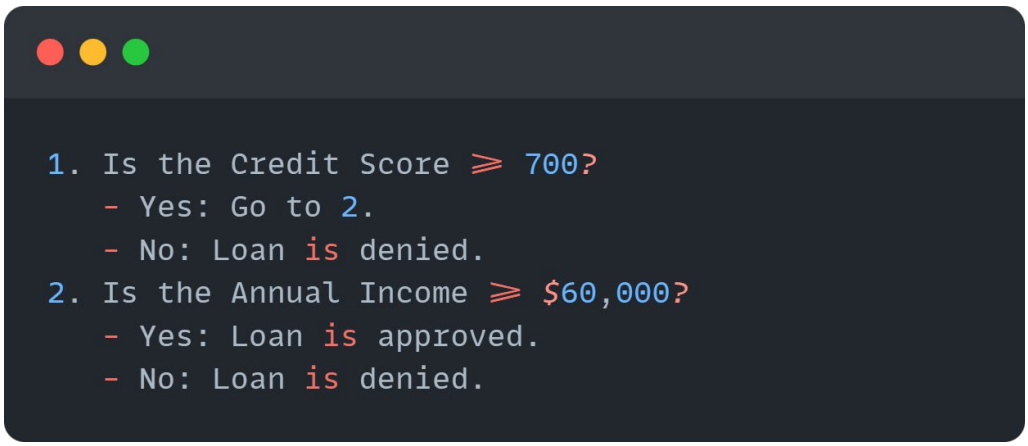


Рис. 2.5. Дерево рішень на основі кредитного рейтингу та річного доходу

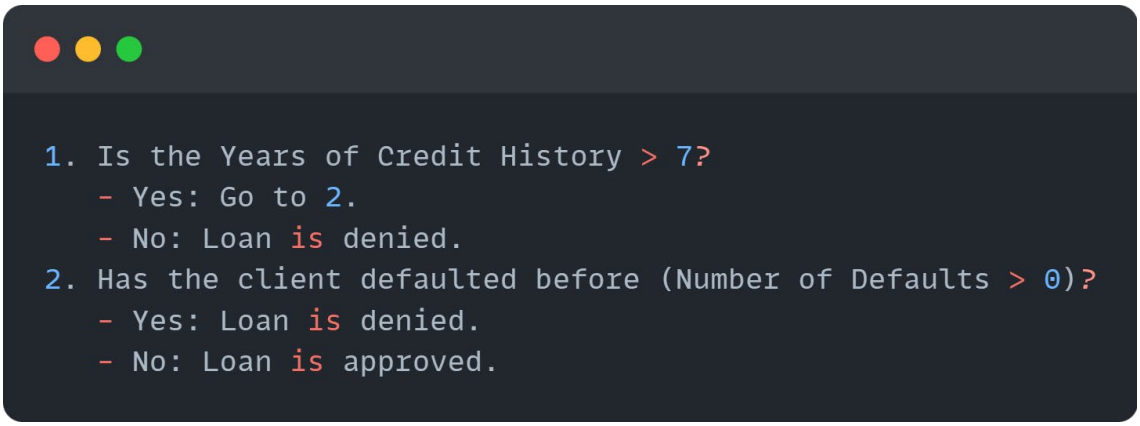


Рис. 2.6. Дерево рішень на основі років кредитної історії та кількості дефолтів

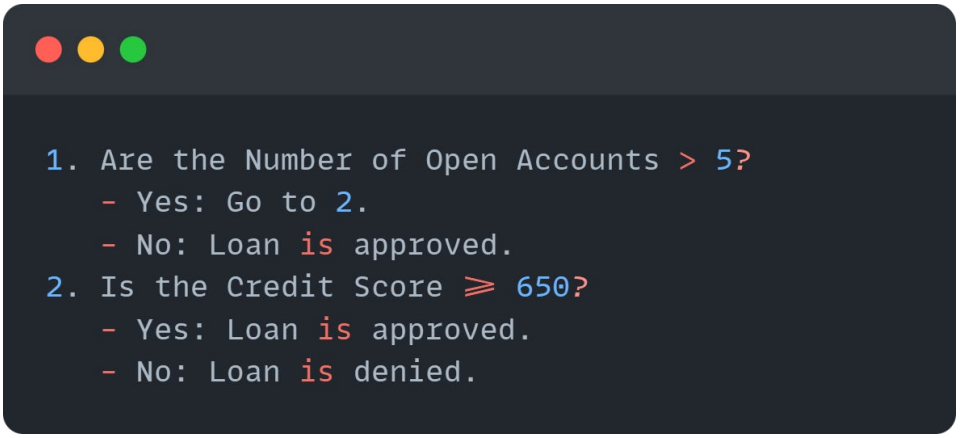


Рис. 2.7. Дерево рішень на основі кількості відкритих рахунків та кредитного рейтингу

Для клієнта з кредитним рейтингом 750, річним доходом $80 000, 10 роками кредитної історії, без дефолтів та 5 відкритими рахунками:

- Дерево рішень 1 схвалить кредит (кредитний рейтинг >= 700, дохід >= 60 000 доларів США).
- Дерево рішень 2 схвалить кредит (кредитна історія > 7 років, без дефолтів).
- Дерево рішень 3 схвалить кредит (кількість відкритих рахунків <= 5).

Оскільки всі дерева проголосували «за», остаточним рішенням Випадкового лісу є схвалення позики.

Для клієнта з кредитним рейтингом 640, річним доходом $45 000, 5 років кредитної історії, 1 дефолтом та 3 відкритими рахунками:

- Дерево рішень 1 відмовить у видачі кредиту (кредитний рейтинг становить < 700).
- Дерево рішень 2 відмовить у видачі кредиту (кредитна історія <= 7 років).
- Дерево рішень 3 схвалить кредит (кількість відкритих рахунків <= 5).

У цьому випадку більшість дерев голосують «проти», тому остаточним рішенням Випадкового лісу є відмова у видачі кредиту.

Це дуже спрощений приклад. Сила моделі Random Forest полягає в її здатності враховувати широкий спектр шляхів і результатів, тим самим забезпечуючи більш надійне і точне рішення в цілому. Він також більш стійкий до перенавчання, ніж єдине дерево рішень.

2.5 K-Nearest Neighbors (KNN)

Класифікатор **K-Nearest Neighbors** (KNN) — це простий і ефективний алгоритм машинного навчання, який використовується як для завдань класифікації, так і для регресії.

Принцип. KNN працює за принципом близькості. Припускається, що подібні точки даних знаходяться близько одна до одної в просторі ознак. Для класифікації, отримавши нову точку даних, KNN ідентифікує своїх K-найближчих

сусідів із навчального набору даних і призначає мітку класу на основі більшості голосів серед цих сусідів. Для регресії він обчислює середнє (або середньозважене) цільових значень K-найближчих сусідів, щоб передбачити значення для нової точки даних.

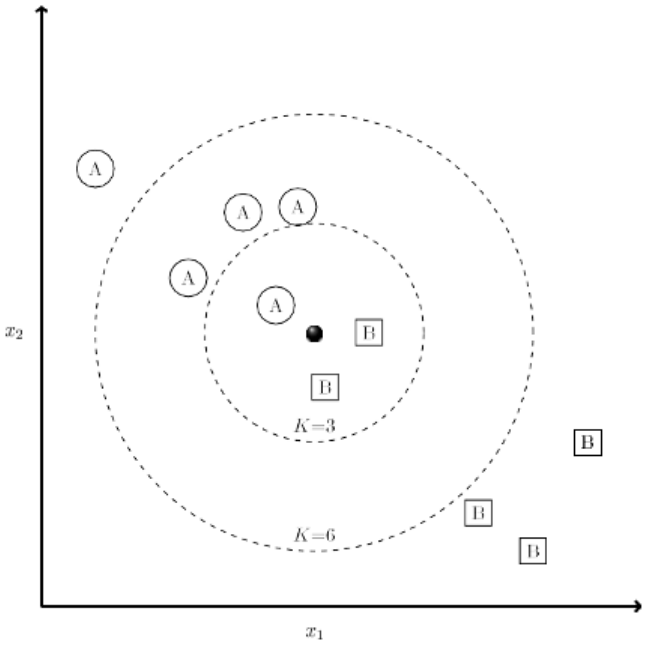


Рис. 2.8. Ілюстрація KNN

Ключові особливості:

- **Параметр K:** кількість сусідів (K) є ключовим параметром у KNN. Він визначає, скільки найближчих точок даних враховуються під час прогнозування. Вибір правильного значення K є важливим для продуктивності моделі.

Плюси:

- **Простота.** KNN легко зрозуміти та реалізувати, що робить його гарним вибором для новачків.
- **Відсутність фази навчання.** KNN ледачий учень, тобто не має фази явного навчання. Він зберігає весь навчальний набір даних і робить прогнози на льоту, що може бути корисним для динамічних наборів даних.
- **Непараметричний.** KNN не робить припущень щодо базового розподілу даних, що робить його придатним для широкого кола проблем.

Таким чином, KNN — це простий, але ефективний алгоритм, який покладається на подібність між точками даних, щоб робити прогнози. Це особливо корисно, коли важливі інтерпретабельність і простота, і може слугувати корисною точкою відліку для більш складних моделей машинного навчання.

2.6 Опорні векторні машини (SVM)

SVM є потужними моделями контрольованого навчання для класифікації та регресійного аналізу. Вони працюють, знаходячи гіперплощину в багатовимірному просторі, яка чітко класифікує точки даних. У контексті схвалення кредиту SVM може бути використаний для класифікації того, чи слід схвалити чи відхилити заявку на кредит на основі кількох факторів.

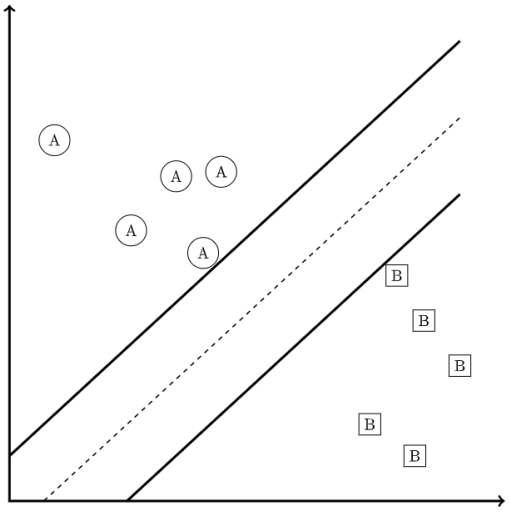


Рис. 2.9. Ілюстрація SVM

Опорні векторні машини (SVM) працюють принципово інакше, ніж дерево ухвалення рішень або випадкові ліси. SVM зазвичай використовуються для вирішення проблем класифікації, таких як процес схвалення кредиту. Вони працюють, знаходячи гіперплощину (у двовимірному просторі це просто лінія), яка найкраще ділить дані на відповідні класи, максимізуючи межу (відстань) між найближчими точками (опорними векторами) та гіперплощиною.

Перед запуском SVM попередня обробка даних має вирішальне значення, оскільки SVM чутливі до масштабу даних. Нам потрібно буде нормалізувати або стандартизувати наші числові дані, такі як кредитний рейтинг, річний дохід тощо.

Оскільки процес прийняття рішень SVM не так легко візуалізувати, як дерево ухвалення рішень, давайте проілюструємо цю концепцію на спрощеному двовимірному прикладі лише з двома характеристиками: кредитним рейтингом та річним доходом. Насправді SVM можуть обробляти багатовимірні дані, враховуючи всі функції нашого набору даних одночасно.

Припустимо, ми побудували графік «Кредитний рейтинг» на осі X і «Річний дохід» на осі Y. Наша мета полягала б у тому, щоб знайти лінію, яка відокремлює клієнтів, яким схвалили кредит, від тих, хто цього не зробив, таким чином, щоб лінія була якомога далі від найближчих точок кожного класу.

Потренувавши SVM з нашими даними, знаходимо такий рядок. Щоб зробити прогноз для нового клієнта, нам потрібно побудувати на нашому графіку характеристики цього клієнта (кредитний рейтинг і річний дохід) і подивитися, на яку сторону лінії він знаходиться.

Наприклад, новий клієнт з високим кредитним рейтингом і річним доходом, швидше за все, потрапить на сторону лінії, що відповідає «Схвалено кредит». І навпаки, клієнт з низьким кредитним рейтингом і низьким річним доходом потрапить на сторону «Відмовлено в кредиті».

Пам'ятайте, що це занадто спрощена версія того, як працюють SVM, особливо з огляду на потужну здатність SVM використовувати трюки ядра для обробки нелінійно відокремлюваних даних. Однак загальна ідея проведення кордону (у вищих вимірах це гіперплощина) і категоризації нових точок залежно від того, на яку сторону вони падають, залишається незмінною.

Support Vector Machine (SVM) — це потужний керований алгоритм машинного навчання, який використовується для завдань класифікації та регресії. Він відомий своєю здатністю знаходити оптимальні гіперплощини, які ефективно розділяють точки даних на різні класи.

принцип:

- SVM базується на концепції пошуку найкращої можливої межі рішення, яка називається гіперплощиною, яка максимізує запас між двома класами в просторі ознак.
- Термін «опорний вектор» стосується точок даних, найближчих до межі прийняття рішення, які мають найбільший вплив на визначення запасу.

Ключові особливості:

- **Лінійна та нелінійна класифікація:** SVM можна використовувати для завдань лінійної та нелінійної класифікації. У лінійній SVM використовується лінійна гіперплощина, тоді як у нелінійній SVM використовується трюк ядра для перетворення даних у простір з більшою вимірністю, де можна знайти лінійну гіперплощину.
- **Максимізація маржі:** SVM прагне максимізувати маржу, яка є відстанню між межею прийняття рішення та найближчими точками даних кожного класу. Це допомагає покращити узагальнення та стійкість моделі.
- **Опорні вектори:** опорні вектори – це точки даних, які лежать найближче до межі прийняття рішень і відіграють вирішальну роль у визначенні гіперплощини.

Хитрість ядра: SVM може використовувати різні функції ядра (наприклад, лінійну, поліноміальну, радіальну базисну функцію) для обробки нелінійно розділених даних шляхом неявного відображення їх у просторі вищих розмірів.

Плюси:

- **Ефективність для даних великої розмірності:** SVM добре працює навіть у просторі функцій великої розмірності, що робить його придатним для таких завдань, як класифікація тексту та розпізнавання зображень.
- **Стійкий до викидів:** SVM менш чутливий до викидів, оскільки в основному залежить від опорних векторів.

- **Глобальний оптимум:** SVM прагне знайти глобальний оптимум, що означає, що він загалом веде до кращого узагальнення порівняно з деякими іншими алгоритмами.
- **Універсальність:** SVM можна застосовувати як для завдань класифікації, так і для регресії.

Підсумовуючи, опорні векторні машини — це універсальні та ефективні алгоритми машинного навчання, особливо при роботі з великими даними та складними межами прийняття рішень. Вони є цінними інструментами як для завдань класифікації, так і для регресії.

2.7 Екстремальне посилення градієнта (XGBoost)

XGBoost означає Extreme Gradient Boosting. Це розпаралелена та ретельно оптимізована версія алгоритму посилення градієнта. Розпаралелювання всього процесу посилення значно покращує час навчання.

Замість навчання найкращої моделі на основі даних як у традиційних методах, ми навчаємо тисячі моделей на різних підмножинах навчального набору даних, а потім голосуємо за найкращу ефективна модель.

У багатьох випадках XGBoost є кращим за звичайні алгоритми посилення градієнта. Реалізація Python надає доступ до великої кількості внутрішніх параметрів, які можна налаштувати для кращої точності та точності.

Важливі функції XGBoost:

- **Розпаралелювання.** Модель реалізовано для навчання з кількома ядрами ЦП.
- **Регулярізація.** XGBoost включає різні покарання за регулярізацію, щоб уникнути пере налаштування. Регулярізація штрафів забезпечує успішне навчання, щоб модель могла адекватно узагальнюватися.

- **Нелінійність.** XGBoost може виявляти нелінійні шаблони даних і навчатися на них.
- **Перехресна перевірка.** вбудована та легка у використанні.

Алгоритм XGBoost

Розглянемо функцію або оцінку (2.10). Для початку ми будуємо послідовність, отриману з градієнтів функції. Наведене нижче рівняння моделює певну форму градієнтного спуску. F представляє функцію втрати для мінімізації, отже, вона вказує напрямок, у якому функція зменшується. Це швидкість зміни, яка відповідає функції втрат, вона еквівалентна швидкості навчання при градієнтному спуску. Очікується, що відповідним чином наблизить поведінку збитку.

$$F_{x_t} + 1 \;=\; F_{x_t} + \epsilon_{x_t} \frac{\partial F}{\partial x}(x_t) \qquad (2.10)$$

Щоб переглянути модель і знайти оптимальне визначення, нам потрібно виразити всю формулу як послідовність і знайти ефективну функцію, яка буде сходитися до мінімуму функції. Ця функція слугуватиме мірою помилки, щоб допомогти нам зменшити втрати та зберегти ефективність з часом.

Послідовність збігається до мінімуму функції. Ця конкретна нотація визначає функцію помилки, яка застосовується під час оцінювання регресії, що посилює градієнт.

$$f(x, \theta) \;=\; \sum l(F\big((X_i\ , \theta), y_i\big)) \qquad (2.11)$$

Переваги та недоліки XGBoost

Переваги:

- Градієнтний бустинг має алгоритм, який легко читати та інтерпретувати, що робить переважну більшість прогнозів легкими для обробки.
- Бустинг є міцним та надійним методом, який легко запобігає перенавчанню моделей.
- XGBoost дуже добре працює на середніх та малих наборах даних із підгрупами та структурованими наборами даних з невеликою кількістю особливостей.
- Це відмінний підхід, оскільки більшість проблем реального світу включають класифікацію та регресію — два завдання, де XGBoost виявляє себе краще за інших.

Недоліки:

- XGBoost не так добре працює з розрідженими та неструктурованими даними.
- Часто забувають, що градієнтний бустинг дуже чутливий до викидів, оскільки кожен класифікатор змушений виправляти помилки попередніх.
- Метод важко масштабувати через те, що оцінювачі базують свою правильність на попередніх прогностичних показниках, тому процедура вимагає значних зусиль для оптимізації.
- XGBoost може бути не найкращим варіантом для високо вимірних, розріджених даних, які часто зустрічаються в аналітиці текстів та розпізнаванні зображень, де краще можуть справлятися моделі глибокого навчання.
- Чутливість до викидів може вплинути на продуктивність моделі, що вимагає ретельної попередньої обробки даних для зменшення цього ризику.
- Масштабування може бути викликом для XGBoost, особливо при роботі з дуже великими наборами даних або в розподілених обчислювальних

середовищах, незважаючи на його ефективність для помірно великих наборів даних.

Підсумовуючи, XGBoost є потужним алгоритмом машинного навчання, який забезпечує хороший баланс швидкості та продуктивності для широкого спектру типів та розмірів даних, хоча має свої обмеження в певних сценаріях.

потужним алгоритмом машинного навчання, який забезпечує хороший баланс швидкості та продуктивності для широкого спектру типів та розмірів даних, хоча має свої обмеження в певних сценаріях.

2.8 Нейронні мережі (Deep Neural Network)

Нейронні мережі, особливо глибокі, здійснили революцію в галузі машинного навчання. Глибока нейронна мережа (DNN) складається з кількох шарів взаємопов'язаних вузлів або "нейронів", які до певної міри імітують біологічні процеси людського мозку, надаючи можливість визнавати складні шаблони та приймати рішення.

2.8.1 Принцип роботи глибоких нейронних мереж

Основна функція DNN полягає в отриманні вхідних даних та їх обробці через кілька шарів нейронів, кожен з яких сприяє вилученню ознак вищого рівня. Вхідний шар отримує сиру інформацію, яка потім обробляється одним або декількома прихованими шарами, що веде до кінцевого вихідного шару, який надає результат. Сила з'єднань між нейронами, відомих як ваги, регулюється під час процесу навчання за допомогою зворотного розповсюдження та алгоритму оптимізації, такого як градієнтний спуск, для мінімізації помилки в прогнозах.

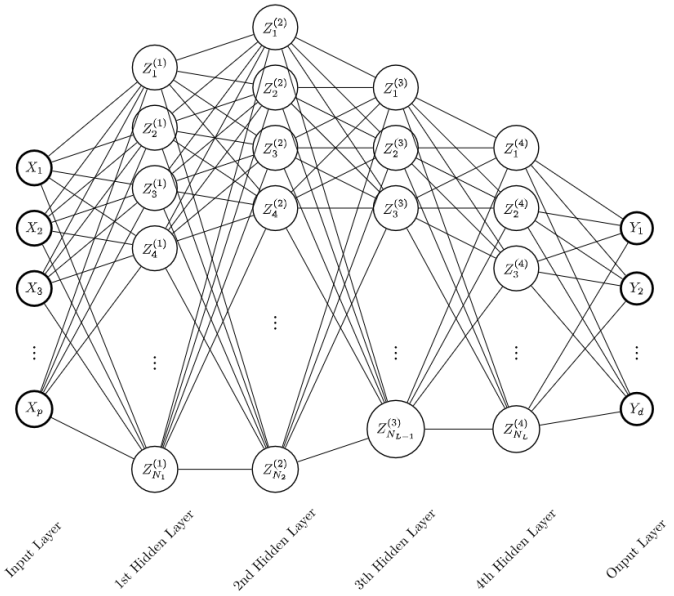


Рис. 2.10. Ілюстрація нейронної мережі з чотирма шарами.

Математичні формулювання, що лежать в основі нейронних мереж

Математично дія в нейроні може бути описана за формулою

$$f(x) = \emptyset\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{2.12}$$

де ϕ представляє активаційну функцію, w_i вагу, x_i вхід і b зсув. Процес навчання включає регулювання ваг і зсувів для мінімізації функції втрат, яка вимірює різницю між прогнозованим виходом і фактичним виходом.

2.8.2 Ключові особливості

DNN характеризуються своєю глибиною, яка дозволяє їм моделювати складні функції; їхньою здатністю автоматично навчати представлення ознак, що усуває необхідність ручного вилучення ознак; та їх адаптабельністю до різних типів даних та завдань, від розпізнавання зображень і мови до обробки природної мови.

2.8.3 Переваги та недоліки

Перевагами DNN є їх висока точність у завданнях класифікації та регресії, а також гнучкість у роботі з різними типами даних. Однак вони також мають кілька недоліків: для навчання потрібні великі обсяги даних, вони є обчислювально інтенсивними, а їх "чорний ящик" ускладнює інтерпретацію та розуміння процесів прийняття рішень.

2.8.4 Потенціал та виклики нейронних мереж в банківській сфері

У банківській сфері DNN пропонують потенціал для виявлення шахрайства, автоматизації обслуговування клієнтів та управління ризиками. Однак їх застосування не є широко поширеним через виклики, такі як необхідність пояснювальності в процесі прийняття рішень, дотримання регуляторних вимог, а також питання чутливості та конфіденційності фінансових даних.

2.9 Оцінка ефективності та обмеження методів

Як і у випадку з усіма моделями машинного навчання, важливо оцінити продуктивність нейронної мережі за допомогою окремого тестового набору або перехресної перевірки, щоб переконатися, що вона добре узагальнюється на невидимі дані, підтверджуючи свою корисність в практичних фінансових системах прийняття рішень.

Таблиця 2.1

Оцінка ефективності та обмеження методів

№	Метод	Переваги	Недоліки
1	**K-найближчих сусідів (K-Nearest Neighbors)**	Простий, не вимагає навчання моделі.	Повільний на великих наборах даних, чутливий до нормалізації даних.
2	**Метод опорних векторів (SVM)**	Ефективний у високо вимірних просторах, гнучкий за допомогою ядер.	Не найкращий вибір для великих наборів даних, чутливий до параметрів.

Продовження таблиці 2.1

Оцінка ефективності та обмеження методів

№	Метод	Переваги	Недоліки
3	**Нейронні мережі**	Висока точність, гнучкість у навчанні складних закономірностей.	Вимогливий до обчислювальних ресурсів, складний для інтерпретації.
4	**Логістична регресія (Logistic Regression)**	Проста у використанні, ефективна для бінарної класифікації.	Погано справляється з нелінійними відносинами, вразлива до перенавчання.
5	**Дерево ухвалення рішень (Decision Tree)**	Інтуїтивно зрозумілі, здатні до візуалізації, ефективні для великих даних.	Схильність до перенавчання, нестабільність при малих змінах у даних.
6	**Випадковий ліс**	Має хорошу продуктивність для багатьох проблем та має меншу схильність до перенавчання	Може бути повільним на великих даних, складний для інтерпретації.
7		Швидкість і продуктивність, хороші результати на багатьох задачах.	Вимогливий до ресурсів, може бути складним для налаштування.

2.10 Метрики оцінювання якості моделей

2.10.1 Інтерпретація матриці помилок

Розповсюджений метод оцінки продуктивності моделей з бінарними відповідями - це застосування матриці помилок. Позитивні випадки відносяться до спостережуваних дефолтів, а негативні - до несплати. Можливі результати включають істинно позитивні (TP), де дефолти правильно прогнозуються, або істинно негативні (TN), де не дефолтні ситуації правильно ідентифіковані.

Помилково позитивні (FP) - це випадки, де помилково зроблено позитивний прогноз, та помилково негативні (FN) - випадки, де дефолти неправильно прогнозуються як не дефолтні. Цю матрицю можна побачити на рисунку 13.

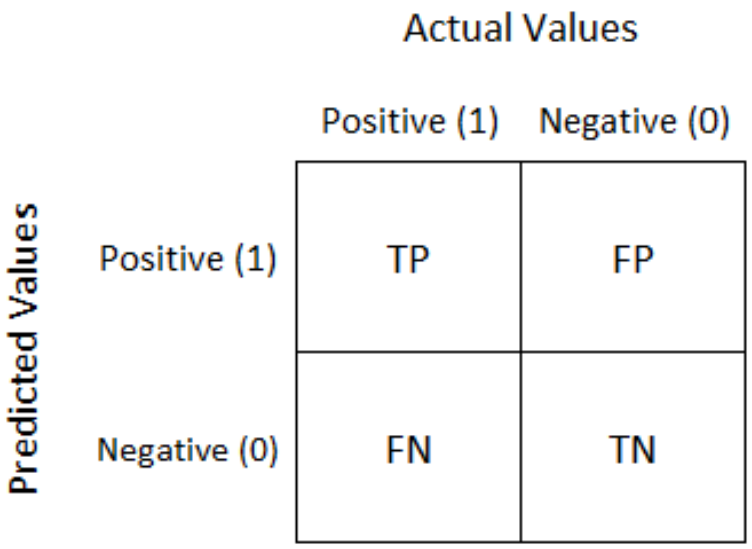


Рис. 2.11. Таблиця класифікаційних помилок

True Positive(TP) – цілі, які насправді відповідають дійсності (Y), і ми пер едбачили, що вони відповідають дійсності (Y)

True Negative(TN) – цілі, які насправді є хибними (N), і ми передбачили, щ о вони хибні (N)

False Positive(FP) – цілі, які насправді є хибними (N), але ми передбачили їх істинними (T)

False Negative(FN) – цілі, які насправді є істинними (T), але ми передбачили їх хибні (N)

З цієї матриці помилок можна отримати кілька показників продуктивності.

Точність (Accuracy) - це міра правильності, досягнутої в істинному передбаченні, тобто кількість спостережень, позначених як істинні, скільки насправді позначено як істинні. Точність визначається:

$$\textbf{Accuracy} = TP / (TP + FP) \qquad (2.13)$$

Однак точність може бути ненадійною для незбалансованих даних. Наприклад набір даних який має 97% одного класу може вводити в оману, показуючи 97% точності.

Чутливість (Precision) є досить актуальним показником який є мірою фактичних спостережень, які прогнозовано правильно, тобто скільки спостережень справжнього класу позначено правильно. Це відношення правильно передбачених позитивних спостережень до всіх спостережень в актуальному класі, відображає здатність моделі знаходити всі релевантні випадки. Чутливість визначається:

$$\textbf{Precision} = \boldsymbol{TP / (TP + FN)} \tag{2.14}$$

Специфічність (Specificity) є більш актуальним показником для цього проекту яка є показником того, наскільки ефективно модель ідентифікує негативні випадки nf необхідна для розуміння продуктивності неправильних прогнозів. Специфічність визначається:

$$\textbf{Specificity} = \boldsymbol{TN / (TN + FP)} \tag{2.15}$$

Специфічність і чутливість відіграють вирішальну роль у отриманні кривої ROC.

2.10.2 Загальна оцінка характеристик ROC-кривої

В прагненні покращити методи прийняття рішень в банківській сфері, наше дослідження було зосереджене на оцінці ефективності моделей прогнозу за допомогою кривої Receiver Operating Characteristic (ROC) та відповідної площі під кривою (AUC). ROC-крива є графічним відображенням, яке ілюструє діагностичну здатність бінарної класифікаційної системи при зміні порога дискримінації, тоді як AUC представляє ступінь відокремленості. Вона показує, наскільки модель здатна відрізняти класи.

Емпірична робота включає розрахунок чутливості та специфічності моделей.

Чутливість, або істинно позитивна частка, відображає пропорцію справжніх позитивів, які модель ідентифікувала правильно, що математично визначається як:

$$\textbf{Sensitivity} = \frac{\textbf{True Positives}}{\textbf{True Positives} + \textbf{False Negatives}} \quad (2.16)$$

Специфічність, або істинно негативна частка, вимірює пропорцію справжніх негативів, які модель ідентифікувала правильно, що виражено як (2.10.5):

$$\textbf{Specificity} = \frac{\textbf{True Negatives}}{\textbf{True Negatives} + \textbf{False Positives}} \quad (2.17)$$

AUC зазвичай варіюється від 0 до 1 та надає одне скалярне значення для узагальнення загальної продуктивності моделі по всіх порогах класифікації, де 1 означає ідеальне прогнозування, а 0.5 - марну модель. У контексті нашого дослідження, моделі, які демонстрували AUC, ближчі до 1, вважалися більш компетентними у прогнозуванні результатів по позиках.

Інтегруючи ці розрахунки в рамки оцінки ризиків банківського сектора, ми забезпечили міцну статистичну основу, на якій можна більш точно базувати кредитні рішення. Такий підхід дозволяє фінансовим інститутам мінімізувати ризик невиконання зобов'язань, тим самим забезпечуючи економічну стабільність і цілісність.

На завершення нашого дослідження представило всеосяжну модель оцінки, яка вправно балансує кількісні показники ROC і AUC з якісними уявленнями фінансових експертів. Такий збалансований підхід є вирішальним для прийняття обізнаних, обґрунтованих на даних рішень у складному середовищі управління фінансовими ризиками.

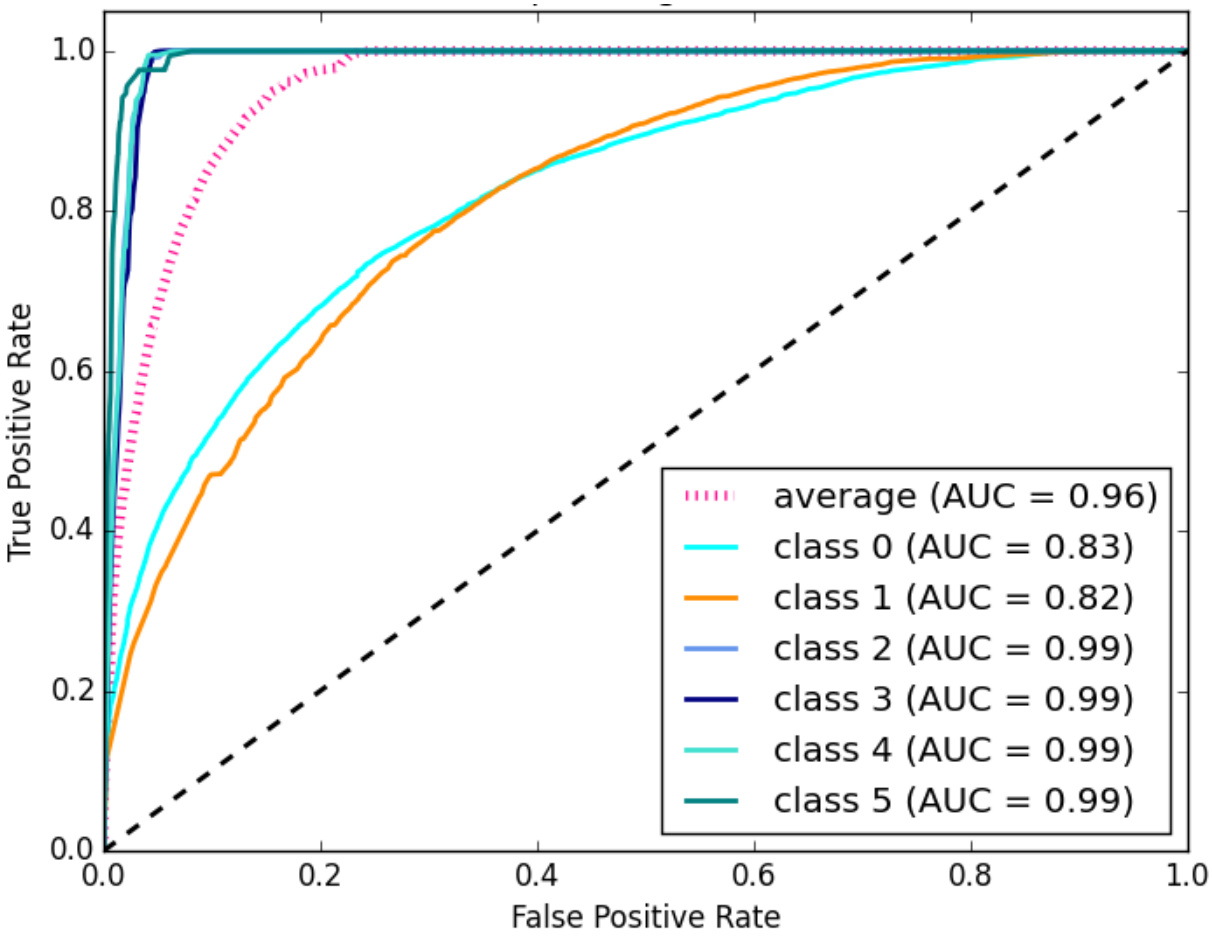


Рис. 2.12. Зразок розробленої кривої ROC

2.10.3 Індекс Gini

У сфері прогностичного аналізу коефіцієнт Джині виходить на передній план як фундаментальний показник, який надає безцінні відомості про дискримінаційну силу прогностичних моделей. Походячи з економіки, де традиційно використовується для вимірювання нерівності доходів, коефіцієнт Джині був вправно адаптований у статистичному моделюванні для оцінки точності та ефективності бінарних класифікаторів.

У своїй основі коефіцієнт Джині виводиться з кривої Робочої Характеристики Одержувача (ROC), графічного зображення, яке ілюструє продуктивність моделі класифікації за різних порогових налаштувань. Крива ROC зображує Справжню Позитивну Частоту (True Positive Rate) проти

Частоти Хибних Позитивів (False Positive Rate), надаючи візуальний інструмент оцінювання балансу між чутливістю та специфічністю.

Площа під Кривою (AUC) ROC, широко визнаний показник продуктивності моделі, служить основою для розрахунку коефіцієнта Джині. Коефіцієнт Джині - це просто вдвічі більше AUC мінус один:

$$\mathbf{Gini = 2 * AUC - 1} \quad (2.16)$$

Це перетворення масштабує метрику, розміщуючи її в діапазоні, де 0 вказує на модель без дискримінаційної сили (подібно до випадкового вгадування), а 1 - на ідеальну дискримінацію.

На практиці вищий коефіцієнт Джині свідчить про вищу здатність моделі розрізняти бінарні результати. Він особливо інформативний у таких сферах, як кредитний скоринг, де надзвичайно важливо відрізняти 'хороші' та 'погані' кредитні ризики.

Використання коефіцієнта Джині поряд з іншими показниками, такими як точність та прецизійність, розширює всебічність оцінки моделі. Це дозволяє досягти глибшого розуміння прогностичних можливостей моделі, виходячи за рамки простої точності, яка іноді може бути основною, особливо у випадках незбалансованих наборів даних.

Як результат, коефіцієнт Джині - це витончений, але доступний інструмент, що збагачує арсенал технік, доступних для дата-сайентистів та статистиків. Його інтеграція у робочі процеси прогностичного моделювання не тільки посилює строгість перевірки моделі, але й сприяє розробці більш надійних та ефективних прогностичних моделей.

2.11 Висновки до розділу

У контексті прийняття кредитних рішень було аналізовано передові методики аналітики даних для оцінювання платоспроможності клієнтів. Обговорено різноманіття моделей, включно з логістичною регресією, методом k-найближчих сусідів, деревами рішень, випадковими лісами, опорними векторними машинами, екстремального посилення градієнта та нейронними мережами.

В рамках даного розділу також було вивчено методи оцінки ефективності цих моделей: матрицю класифікаційних помилок та ROC-криву. Особлива увага приділялась оцінюванню помилок першого та другого роду, які дають змогу оцінити ймовірність неправильного ідентифікування клієнтів як потенційно ненадійних, тим самим прогнозуючи потенційні збитки, а також помилкове віднесення надійних клієнтів до ризикованих, що призводить до упущених можливостей для заробітку.

3 СИСТЕМА ПРИЙНЯТТЯ РІШЕНЬ ДЛЯ ПРОГНОЗУВАННЯ СХВАЛЕННЯ КРЕДИТІВ

3.1 Постановка проблеми

Бізнес-проблема. «Metro Bank» отримує значну частину своїх прибутків від позик. Кредити пропонуються для міських, напівміських і сільських районів. З великою кількістю заявників на позику точно визначити надійних позичальників, які будуть повертати їхні позики, є складним завданням. Процес оцінювання вручну сприйнятливий до непорозумінь, що призводить до відбору потенційно ненадійних кандидатів. Як рішення банком було створено систему прогнозування кредитів на основі машинного навчання, яка автономно визначає кваліфікованих кандидатів, що принесуть прибуток від позик банку та не будуть проблемними клієнтами. Крім того, це значно скорочує час, необхідний для погодження кредиту. Процес отримання кредиту проходить в реальному часі на основі даних про клієнта, наданих під час заповнення онлайн-форми заявки.

Ці дані: стать, сімейний стан, освіта, кількість утриманців, дохід, сума позики, кредитна історія та інші. Щоб автоматизувати цей процес, було поставлено задачу ідентифікувати клієнтів на основі даних заявок, які можуть потенційно отримати кредит, та націлитися на цих клієнтів».

Прогнозування кредитів — це дуже поширена проблема в реальному житті, з якою стикається кожен банк під час своїх кредитних операцій. Якщо процес затвердження кредиту автоматизований, це може заощадити багато робочих годин і підвищити швидкість обслуговування клієнтів. Підвищення задоволеності клієнтів і економія операційних витрат є значними. Однак такі переваги можна отримати, лише якщо банк має надійну модель для точного прогнозування кредиту: якого клієнта він має схвалити, а якого відхилити, щоб мінімізувати ризик неповернення кредиту.

3.2 Перетворення бізнес-проблеми банку в постановку задачі в розрізі Data Science / Machine Learning problem

Проблема класифікації, коли банк повинен передбачити, чи буде схвалена позика чи ні. Зокрема, це скоріше проблема бінарної класифікації, де має бути передбачено одне з наданих класів, тобто схвалено (Y) або не схвалено (N). Іншими словами постановка задачі полягає в тому, щоб передбачити, чи буде ймовірність дефолту за кредитом чи ні. Якщо є ймовірність дефолту, то позику не буде схвалено, і навпаки. Залежною змінною або цільовою змінною є `Loan_Status`, а решта є незалежними змінними або функціями. Задача полягає в тому щоб розробити модель, використовуючи функції для прогнозування цільової змінної.

3.3 Генерація гіпотези

Важливою частиною в академічному дослідженні є фаза генерації гіпотез, яка є ключовим кроком у дослідницькому процесі, де потрібно ідентифікувати всі потенційні фактори, що можуть впливати на рішення про схвалення кредиту. Цей систематичний підхід є основоположним для розуміння, які характеристики можуть призвести до затвердження або відхилення заявки на позику.

Для дослідження запропоновано та критично оцінено кілька гіпотез:

- **Освітній фон.** припускаючи, що заявники з вищою освітою мають більше шансів на отримання кредиту.
- **Рівень доходу.** Гіпотеза, що заявники з вищим доходом мають кращі шанси на отримання кредиту.
- **Сума позики.** Припущення, що менші суми позик збільшують імовірність схвалення.
- **Тривалість позики.** Коротші терміни позики можуть корелювати з вищими рівнями схвалення.

- **Попередня кредитна історія.** Припущення, що заявники, які погасили попередні борги, мають більшу ймовірність отримати нові позики.
- **Розмір щомісячного внеску.** Розгляд того, що менші щомісячні платежі можуть підвищити ймовірність надання кредиту.

Хоча деякі з цих гіпотез можуть здаватися інтуїтивно правдоподібними, інші вимагають емпіричного підтвердження. Тому потрібно ретельно перевірити кожну гіпотезу на основі набору даних, щоб підтвердити або спростувати ці первинні припущення.

Результати нашого дослідження зроблять значний внесок у фінансову сферу, надаючи нюансоване розуміння змінних, що впливають на схвалення кредиту. Це знання не тільки допоможе фінансовим установам у процесах прийняття рішень, але й сприятиме розвитку більш справедливих практик кредитування.

Підсумовуючи, дане наукове дослідження дотримується найвищих академічних стандартів, збагачуючи галузь оцінки фінансових ризиків надійними аналітичними методами та вносячи вклад в базу знань банківських установ з метою подальшого використання у прийнятті рішень.

3.4 Збір даних

У науковому дослідженні моделей скорингу ми підкреслили значимість володіння високоякісними даними про позичальників банку. Точність передбачень майбутньої моделі безпосередньо пропорційна якості використаних даних. Методологія розробки скорингових моделей кредитоспроможності базується на історичних даних, що підкреслює важливість значного обсягу даних, включаючи як характеристики позичальника, так і їх внутрішні (зовнішні) кредитні історії.

Особливо важливим було використання історичних даних, які відповідають конкретній сфері, для якої будується скорингова модель. Наприклад, слід використовувати дані про споживчі кредити для розробки моделі у тій самій сфері кредитування.

Для побудови цих моделей було використано тестовий набір даних кредитним відділом Метро Банку в Лондоні, який складався з анонімізованої соціо-демографічної інформації клієнтів та деталей про надані їм кредити. Як результат, було створено навчальний та тестовий набори з метою досягнення міцної та прогностичної системи, яка буде здатна точно оцінювати кредитоспроможність потенційних позичальників.

В процесі розробки прогностичної моделі було ретельно підготовано навчальний набір даних, який включав усі незалежні змінні разом із залежною змінною. Цей всебічний набір даних слугує основою для навчання моделі розпізнавати та розуміти властиві даним закономірності.

Крім того, було створено тестовий набір, що містить усі незалежні змінні, але не включає залежну змінну. Цей набір даних призначений для оцінки здатності моделі прогнозувати залежну змінну при її застосуванні до нових, раніше не бачених даних. Такий структурований підхід дозволяє систематично та академічно оцінювати здібності моделі до прогнозування

Змінні в наборі даних були синтезовані та представлені нижче в деталізованій формі (Таблиця 3.4.1):

Таблиця 3.1

Атрибути набору та їх опис

№	Змінна	Тип	Опис
1	**Loan_ID**	Числовий – Дискретний	Унікальний ID
2	**Gender**	Категоричний - Номінальний	Чоловічий / Жіночий
3	**Married**	Категоричний - Номінальний	
4	**Dependents**	Категоріальний – Порядковий	Кількість утриманців (0, 1,

Продовження таблиці 3.1

Атрибути набору та їх опис

№	Змінна	Тип	Опис
5	**Education**	Категоричний - Номінальний	
6	**Self_Employed**	Категоричний - Номінальний	Самозайнятий (Так/Ні)
7	**ApplicantIncome**	Числовий - Неперервний	Дохід заявника
8	**CoapplicantIncome**	Числовий - Неперервний	Дохід співзаявника
9	**LoanAmount**	Числовий - Неперервний	Сума кредиту в тисячах
10	**Loan_Amount_Term**	Категоричний – Порядковий	Термін позики в місяцях
11	**Credit_History**	Категоричний - Номінальний	Кредитна історія відповідає вимогам (0, 1)
12	**Property_Area**	Категоричний – Порядковий	Міський / Напівміський / Сільський
13	**Loan_Status**	Категоричний - Номінальний	Кредит затверджено (Так/Ні)

3.5 Дослідницький аналіз даних

Завданням цієї роботи було дослідження існуючих алгоритмів машинного навчання та практична реалізація з метою підвищення ефективності вибраних методів для подальшого використання при прийнятті рішень в банківській сфері.

Набір програмний продуктів, які були використані для реалізації процесів автоматизації у вирішенні задачі інтелектуального аналізу даних і машинного навчання були представлені в таблиці.

Таблиця 3.2

Набір програмних продуктів для задач інтелектуального аналізу даних та

машинного навчання

	Стандарт для розробки та аналізу в машинному навчанні.
	Містить в собі безліч алгоритмів таких як: логістична регресія, дерево ухвалення рішень, випадковий ліс, k-folds та інші.
	Бібліотека для вилучення і підготовки даних. Працює з високорівневими структурами даних. Достатньо інтуїтивно зрозуміла та досить проста у використанні. Має багато вбудованих методів для роботи з даними.
	Швидка в роботі бібліотека для роботи з математичними та числовими функціями.
	Стандартна Python бібліотека яка була спеціально розроблена для візуалізації 2D-графіків і діаграм.
	Допоміжна бібліотека яка має додаткові внутрішні функції, необхідні для семантичного маппинга і статистичної агрегації перетворення даних на інформативні графіки.

В рамках дослідницького аналізу було здійснено вивчення великих обсягів даних, які згодо були розділені на дві фундаментальні категорії – тренувальні та тестові набори.

Для виконання роботи було використано низку програмних бібліотек – комплекс інструментів для обробки та аналізу даних. Серед вижче представлених

інструментів слід виділити: NumPy, Pandas, Matplotlib та Seaborn, які були інтегровані для структурування, візуалізації та статистичного аналізу. Для забезпечення неперервності аналітичного процесу була активована функція ігнорування попереджень для того щоб можна було сфокусуватися на головному без відволікаючих факторів. Дані було надано та завантажено у форматі CSV.

Передбачається, що тренувальний набір містить цільову змінну "Статус позики", що є ключовим елементом у процесі прогнозування. Тестовий набір даних, в свою чергу, не містить цієї змінної, і завдання полягає у визначенні її значень на основі моделі, розробленої за тренувальним набором.

Таким чином було створено копії оригінальних датасетів, що є стандартною практикою для забезпечення можливості повернення до первісних даних у разі необхідності коригування чи подальшого аналізу.

Таблиця 3.3

Тренувальний набір зі змінною "Статус позики"

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

Таблиця 3.4

Тестовий набір без змінної "Статус позики"

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	360.0	1.0	Urban
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	360.0	1.0	Urban
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	360.0	1.0	Urban
3	LP001035	Male	Yes	2	Graduate	No	2340	2546	100.0	360.0	NaN	Urban
4	LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	360.0	1.0	Urban

В рамках підготовки аналітичної моделі важливою процедурою є вивчення структурних характеристик наборів даних. Було виконано ідентифікацію кількісних параметрів навчального та тестового датасетів, які включають кількість рядків та стовпців. Визначено, що навчальний набір містить 614 записів, поділених

на 13 функціональних стовпців, тоді як тестовий набір включає 367 записів із 12 стовпцями функцій. Загальна кількість записів, що об'єднують обидва набори, становить 981, з пропорційним розподілом приблизно 63% до 37% відповідно до навчального та тестового датасетів. Відзначається, що стовпець 'Loan_Status' відсутній у тестовому наборі, і його значення буде прогнозуватися в подальшому.

```
(Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
        'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
        'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
       dtype='object'),
 Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
        'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
        'Loan_Amount_Term', 'Credit_History', 'Property_Area'],
       dtype='object'))
```

Визначено наявність трьох різних типів даних: об'єктні (категоріальні змінні), цілі числа (int64), та числа з плаваючою крапкою (float64).

```
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
```

Рис. 3.1. Типи даних датасетів

Категоріальні змінні включають, зокрема, ідентифікатор позики, стать, сімейний стан, кількість залежних, рівень освіти, самозайнятість, район нерухомості та статус позики. Змінні типу int64 та float64 представляють дохід заявника, дохід співзаявника, суму позики, термін позики та кредитну історію.

Важливість розуміння типів даних полягає в можливості застосування відповідних статистичних методів. Кожен тип даних вимагає специфічного підходу до аналізу. Наприклад, категоріальні дані аналізуються інакше ніж числові, тому що невідповідний аналіз може призвести до помилкових висновків. Таким чином, знання про типи даних, що аналізуються, є фундаментальним для вибору коректної аналітичної стратегії.

3.6 Аналіз одиничних змінних у дослідженні

У сфері статистичних методологій одновимірний аналіз виступає як базова техніка, в рамках якої кожна змінна розглядається окремо. Для аналізу категоріальних атрибутів можуть застосовуватися частотні таблиці чи стовпчасті діаграми, які дозволяють визначити кількість випадків кожної категорії в рамках окремо взятої змінної. У свою чергу, для чисельних характеристик використовуються гістограми або бокс-плоти для візуалізації розподілу змінної.

Гістограми сприяють аналізу центральної тенденції, мінливості, модальності та ексцесу розподілу. Проте слід зазначити, що гістограми не виявляють аномальних значень. Це обмеження зумовлює додаткове використання бокс-плотів, які забезпечують більш детальне відображення дисперсії даних, включно з виявленням можливих викидів. Такий всеосяжний підхід гарантує ретельний аналіз характеристик розподілу змінних.

3.6.1 Цільова змінна (категорійна)

На початковому етапі нашого кількісного аналізу ми досліджуємо цільову змінну, яка позначена як 'Loan_Status'. Враховуючи її категорійний характер, необхідно вивчити її частотний розподіл, відсоткове представлення та графічне зображення через стовпчасту діаграму.

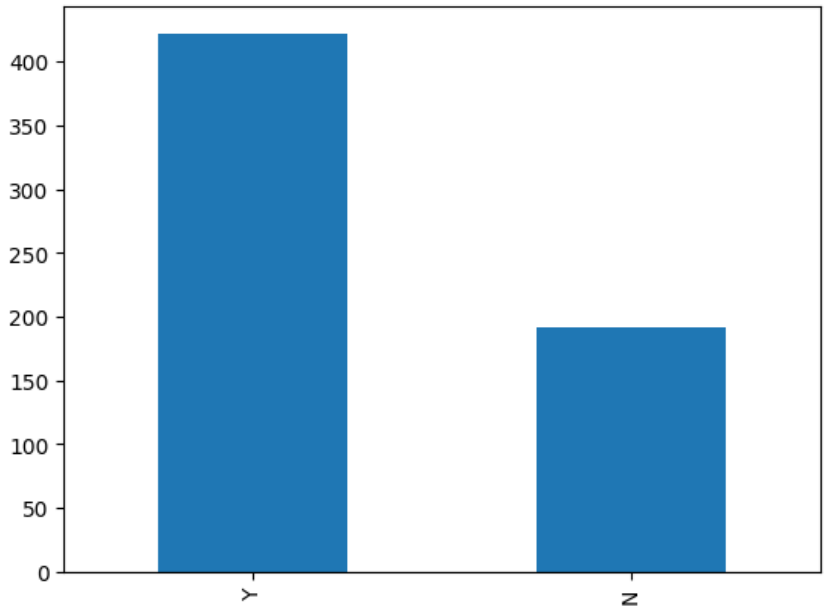


Рис. 3.2. Графічне відсоткове представлення отриманих кредитів

Частотна таблиця змінної надає нам кількість кожної категорії в цій змінній. Команда `**train['Loan_Status'].value_counts()**` виконує цю функцію, надаючи емпіричний підрахунок.

Для отримання відсоткового розподілу використовується процедура нормалізації, яку можна реалізувати, встановивши `**normalize=True**`. Ця корекція параметрів дозволяє представити дані у пропорційних термінах замість абсолютних величин. Команда **train['Loan_Status'].value_counts(normalize=True)** здійснює цей розрахунок.

Стовпчаста діаграма служить візуальною допомогою для пояснення частотного розподілу.

Використання `**train['Loan_Status'].value_counts().plot.bar()**` створює візуальну стовпчасту діаграму, що покращує інтерпретацію даних.

Примітка. У нашому наборі даних 422 з 614 суб'єктів, приблизно 69%, отримали кредит. Відсутність дисбалансу класів у даному контексті легітимізує використання точності як придатного показника оцінки. Навпаки, у разі наявності дисбалансу чи перекосу в класах, точність та повнота виступають як важливі показники оцінки для забезпечення точного аналізу продуктивності прогнозної моделі.

3.6.2 Незалежні змінні (Категорійні)

У досліджуваному наборі даних існує п'ять атрибутів, які класифіковані як категорійні або бінарні, зокрема стосуються статі, сімейного стану, самозайнятості, кредитної історії та рівня освіти.

Візуалізація категорійних ознак: Процес візуального представлення категорійних змінних у наборі даних був здійснений через серію стовпчастих діаграм, кожна з яких підтверджує пропорційний розподіл даних. Ці візуалізації були створені за допомогою конфігурації субплотів для забезпечення всебічного порівняльного аналізу.

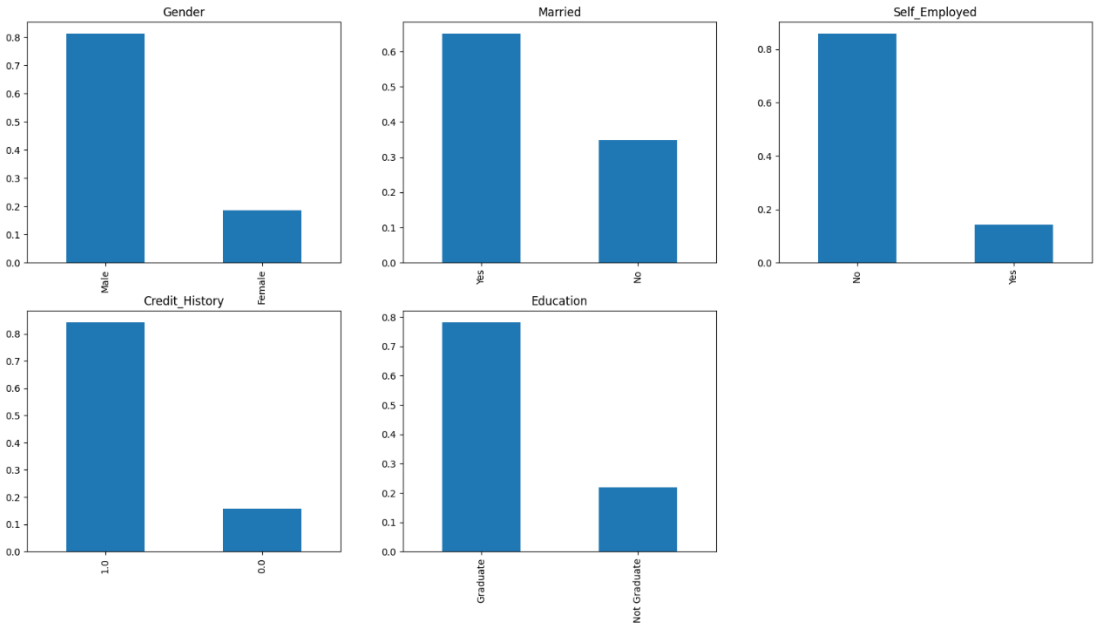


Рис. 3.3. Графічне відсоткове представлення категорійних ознак

На основі вищезазначених стовпчастих діаграм було зроблено висновок, що:

- Приблизно 80% заявників у наборі даних - чоловіки.
- Близько 65% заявників у наборі даних є одруженими.
- Орієнтовно 15% заявників є самозайнятими.
- Приблизно 85% заявників мають кредитну історію з погашеними боргами.
- Майже 80% осіб у наборі даних мають вищу освіту.

Ця графічна експозиція категорійних змінних надає ключові інсайти в інтринсичні характеристики набору даних, дозволяючи глибоке розуміння демографічного ландшафту, представленого в зразку популяції дослідження.

3.6.3 Незалежні Змінні (Порядкові)

У даному дослідженні ідентифіковано дві порядкові змінні в рамках категоріальних атрибутів: 'Dependents' (кількість утриманців) та 'Property_Area' (територіальний район), які мають ієрархічну структуру.

Для візуалізації залишкових категоріальних ознак були використані стовпчасті діаграми, які нормалізовано відображають відсотковий розподіл кожної категорії.

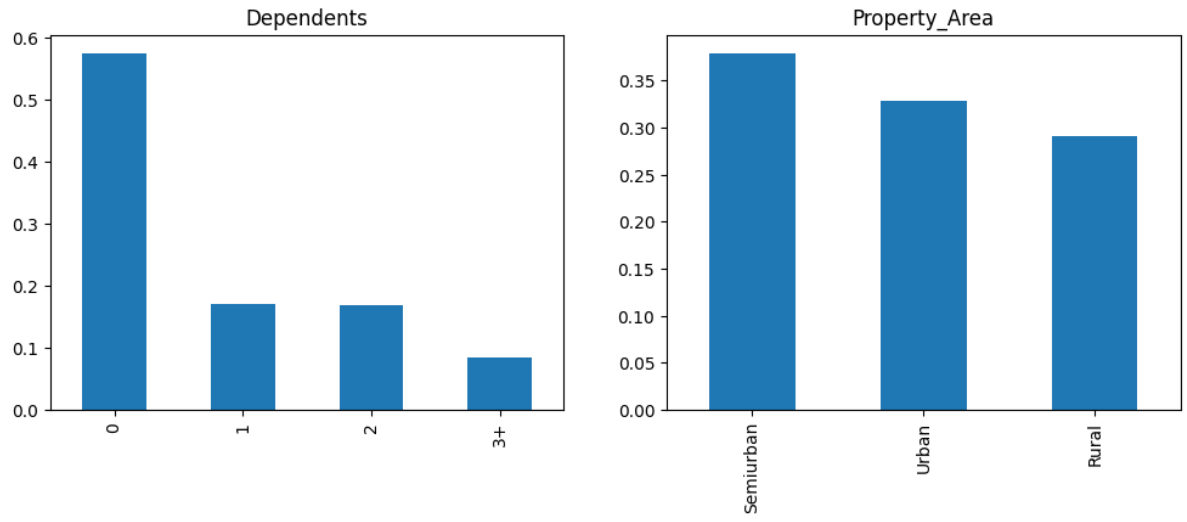


Рис. 3.4. Графічне відсоткове представлення категорійних ознак

Висновки з візуалізації:

- Переважна більшість заявників на кредит не мають утриманців.
- Значна частина заявників на кредит проживає у напівміських районах.

Ці аналітичні спостереження відіграють важливу роль у подальшому аналізі даних, надаючи змогу глибше зрозуміти соціально-демографічну структуру популяції заявників.

3.6.4 Незалежні Змінні (Числові)

У нашому аналізі ми виділяємо чотири числові змінні: '**ApplicantIncome**' (дохід заявника), '**CoapplicantIncome**' (дохід співзаявника), 'LoanAmount' (сума позики), '**Loan_Amount_Term**' (термін позики). Ці змінні представляють кількісні показники у нашій моделі.

Візуалізація доходу заявника: Ми починаємо з розгляду розподілу '**ApplicantIncome**'. За допомогою бібліотеки Seaborn ми генеруємо гістограму, яка візуалізує розподіл доходів заявників. Далі, з використанням Matplotlib, ми створюємо коробкову діаграму, що дозволяє виявити наявність екстремальних значень, які можуть свідчити про значні коливання у розподілі доході

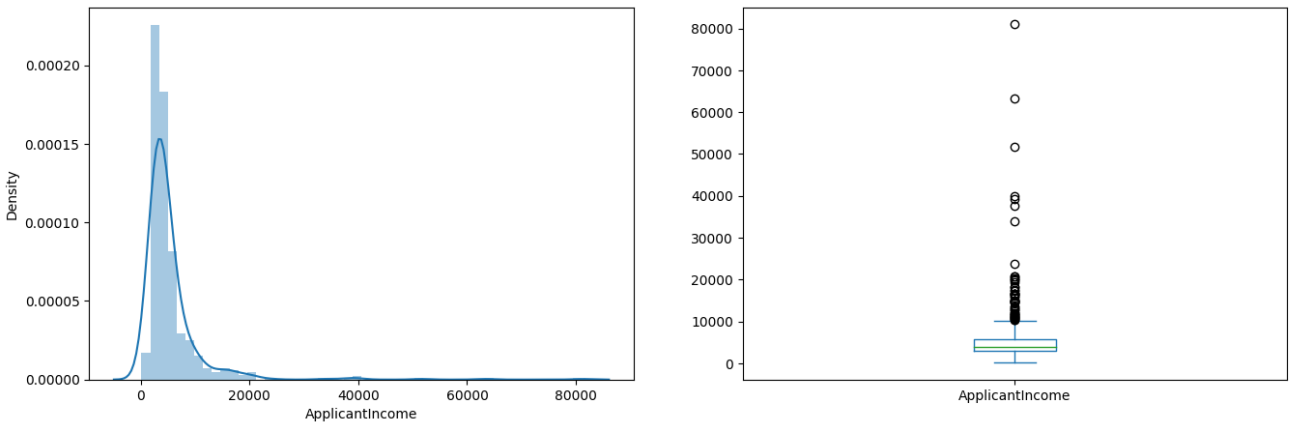


Рис. 3.5. Графічне представлення доходу заявника

Аналіз цих візуалізацій вказує на лівосторонню концентрацію більшості доходів, що натякає на ненормальний розподіл із позитивною асиметрією. В наступних розділах ми прагнемо перетворити цей розподіл на нормальний, адже моделі машинного навчання найкраще працюють з нормально розподіленими даними.

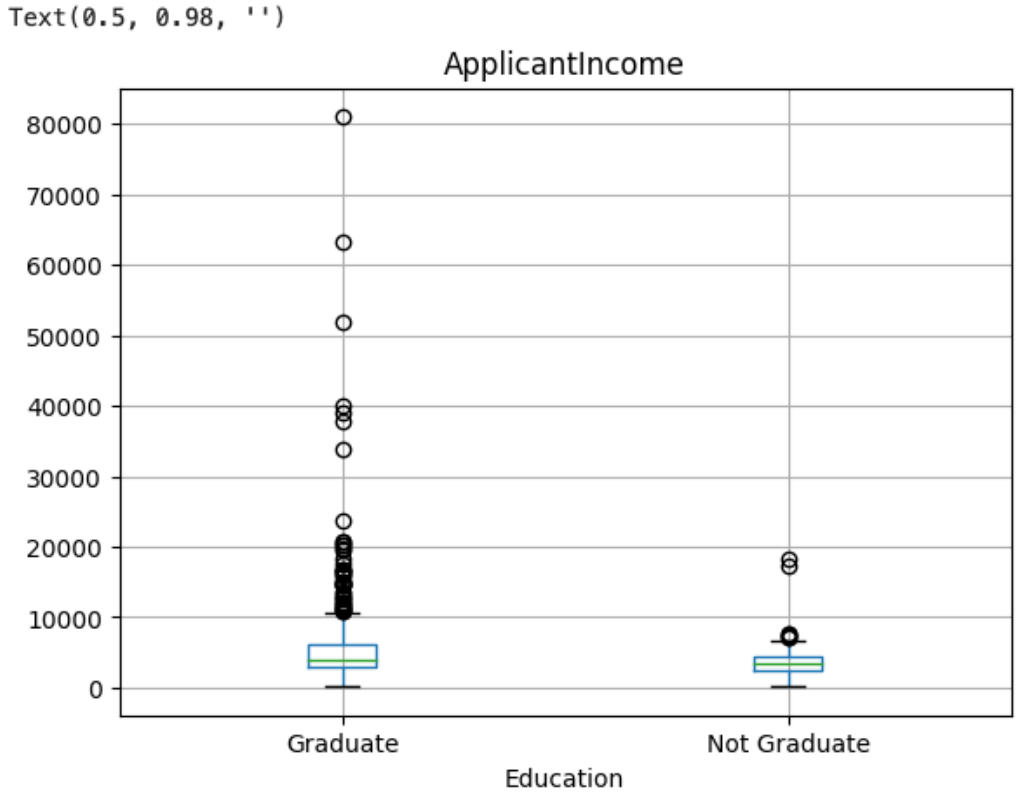


Рис. 3.6. Графічне представлення доходу заявника

Візуалізація доходу співзаявника: Схожий розподіл спостерігається і для 'CoapplicantIncome', де більшість значень концентруються в діапазоні від 0 до 5000, з численними викидами, що також свідчить про ненормальний розподіл.

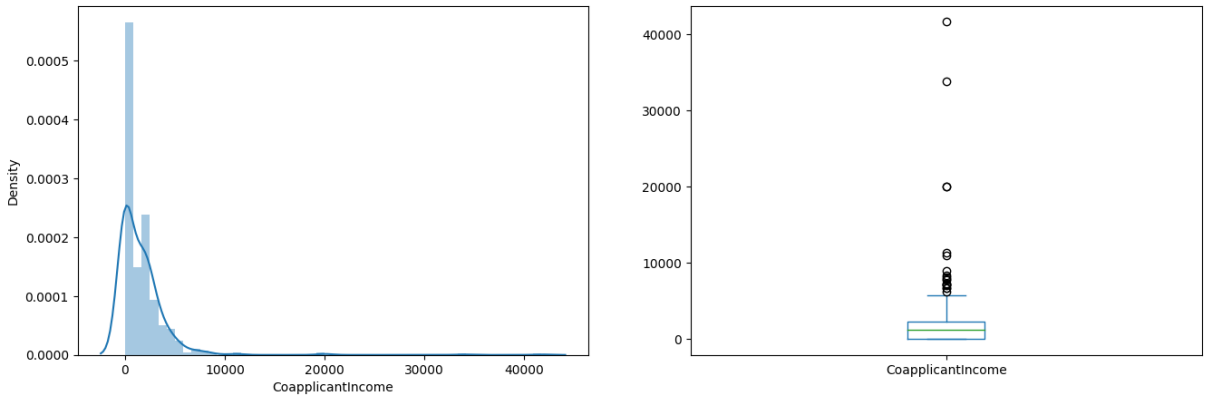


Рис. 3.7. Графічне представлення доходу заявника

Візуалізація 'LoanAmount': Гістограма для 'LoanAmount' показує більш нормалізований розподіл з деяким правостороннім нахилом, однак також з великою кількістю викидів, які будуть детальніше досліджені в наступних секціях.

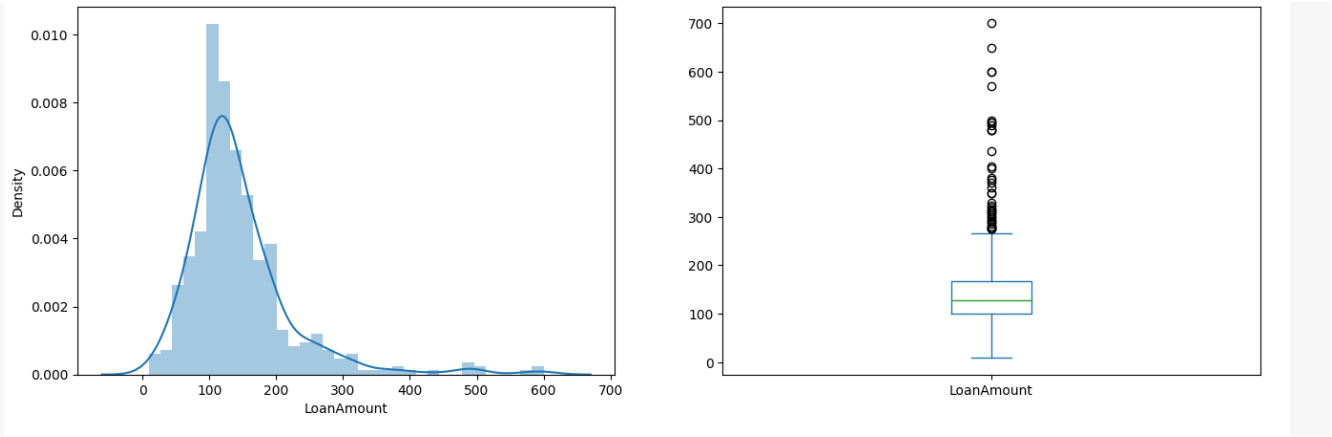


Рис. 3.8. Графічне представлення доходу заявника LoanAmount

Візуалізація 'Loan_Amount_Term': Оскільки 'Loan_Amount_Term' є дискретною змінною, ми використовуємо частотну таблицю та стовпчасті діаграми для відображення розподілу кредитних термінів. З цих діаграм ми можемо зробити висновок, що приблизно 85% позик мають термін у 360 місяців.

```
360.0    512
180.0     44
480.0     15
300.0     13
240.0      4
84.0       4
120.0      3
60.0       2
36.0       2
12.0       1
Name: Loan_Amount_Term, dtype: int64
```

Рис. 3.9. Частотна таблиця розподілу кредитних термінів

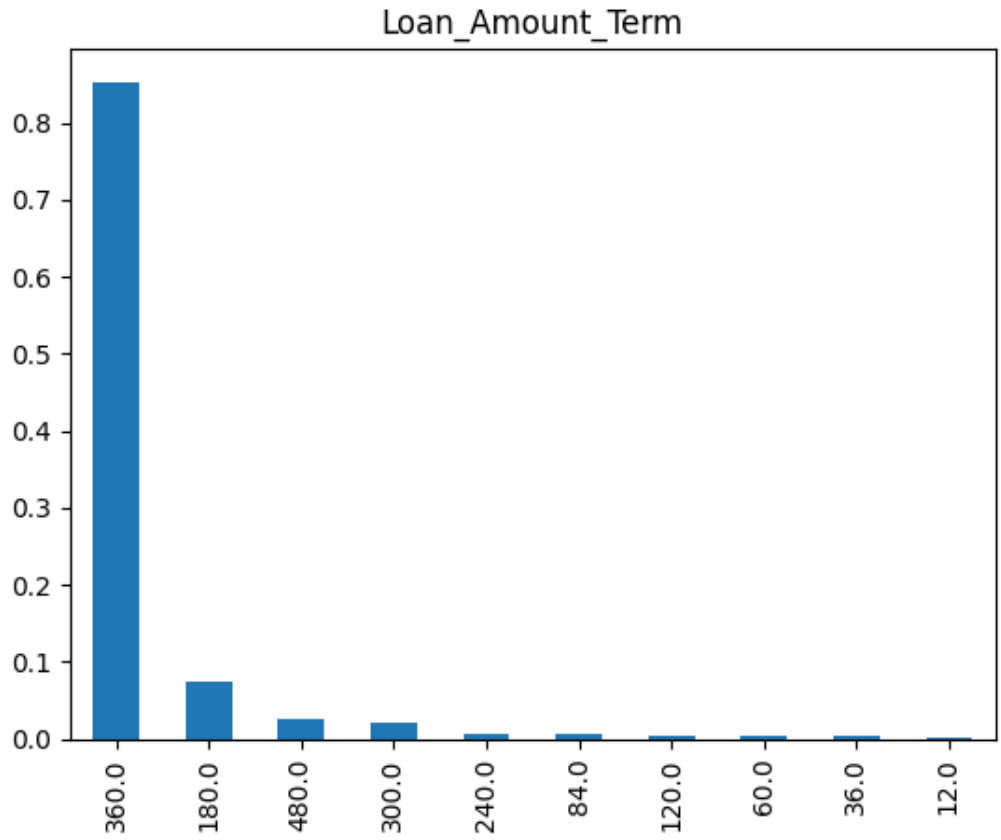


Рис. 3.10. Графічне представлення розподілу кредитних термінів

3.6.5 Біфакторний Аналіз

Після аналізу окремих змінних у рамках однофакторного дослідження, переходимо до більш комплексного біфакторного аналізу, де кожна змінна розглядається у взаємозв'язку з цільовою змінною. Це дозволяє перевірити попередньо сформульовані гіпотези і зробити висновки про взаємозв'язок між змінними.

Для ілюстрації цього зв'язку ми використовуємо таблицю спряженості (**crosstab**) і метод .plot.bar для створення стовпчастих діаграм, які нормалізують і показують відсоткове співвідношення між змінними **'Gender' (стать), 'Married' (сімейний стан), 'Dependents' (кількість утриманців), 'Education' (освіта), 'Self_Employed' (самозайнятість), 'Credit_History' (кредитна історія), 'Property_Area' (територіальна зона) та статусом кредиту ('Loan_Status')**.

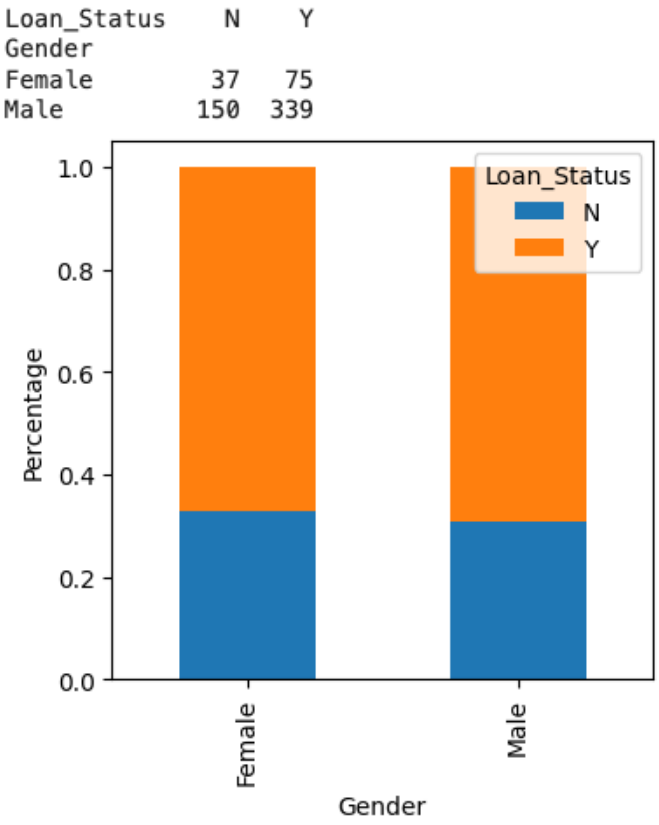


Рис. 3.11. Частка заявників чоловічої та жіночої статі

Loan_Status	N	Y
Married		
No	79	134
Yes	113	285

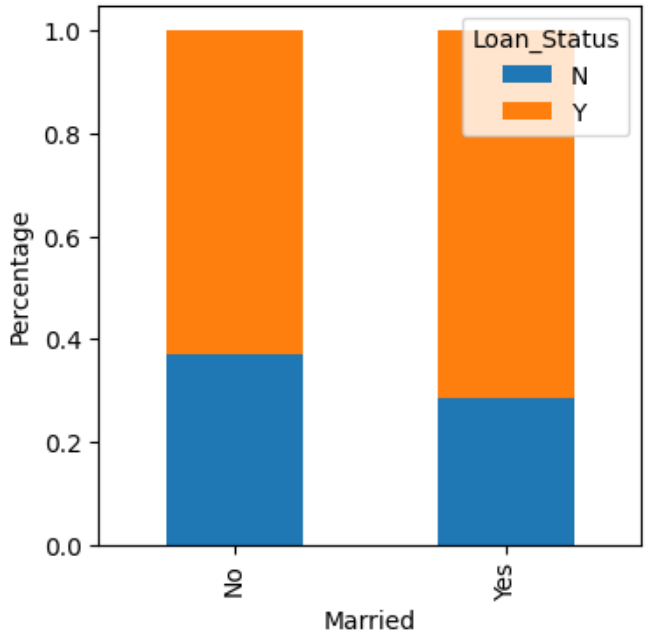


Рис. 3.12. Частка одружених та неодружених заявників

Loan_Status	N	Y
Dependents		
0	107	238
1	36	66
2	25	76
3+	18	33

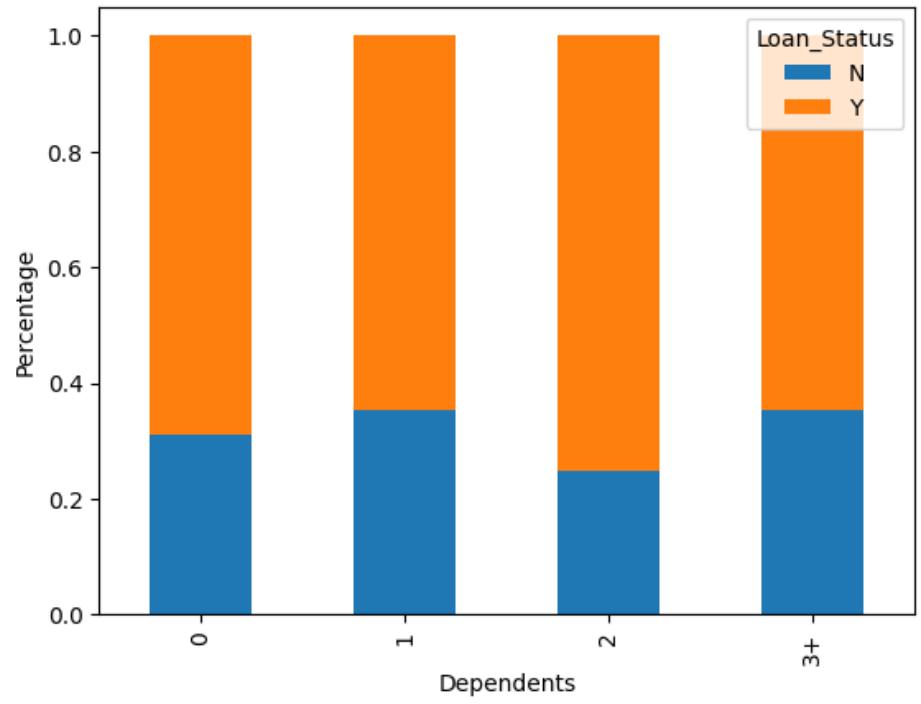


Рис. 3.13. Заявники з одним або трьома і більше утриманцями

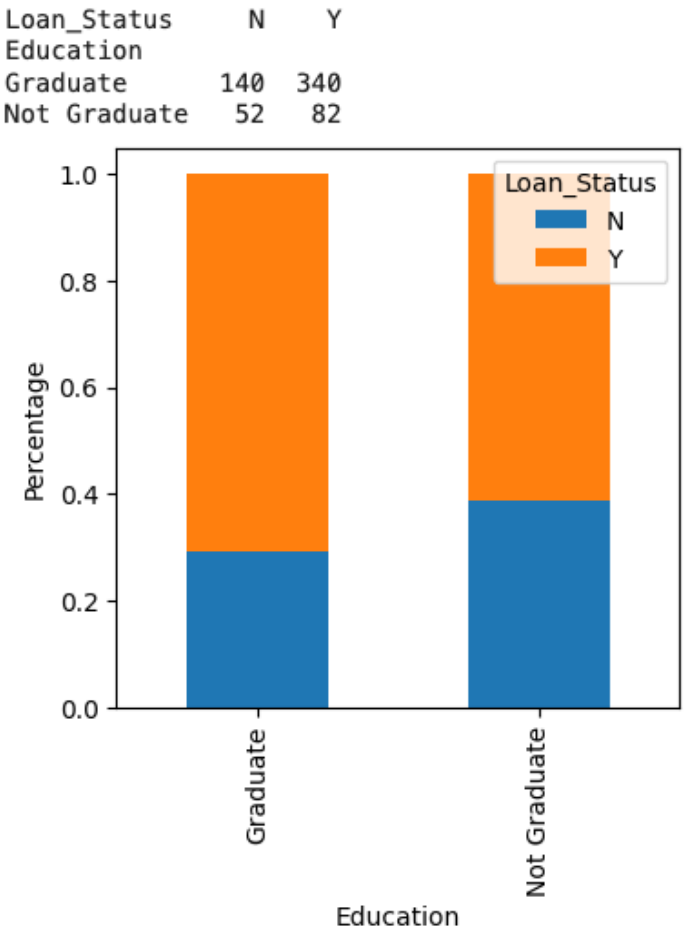


Рис. 3.14. Заявники з вищою та без вищої освіти

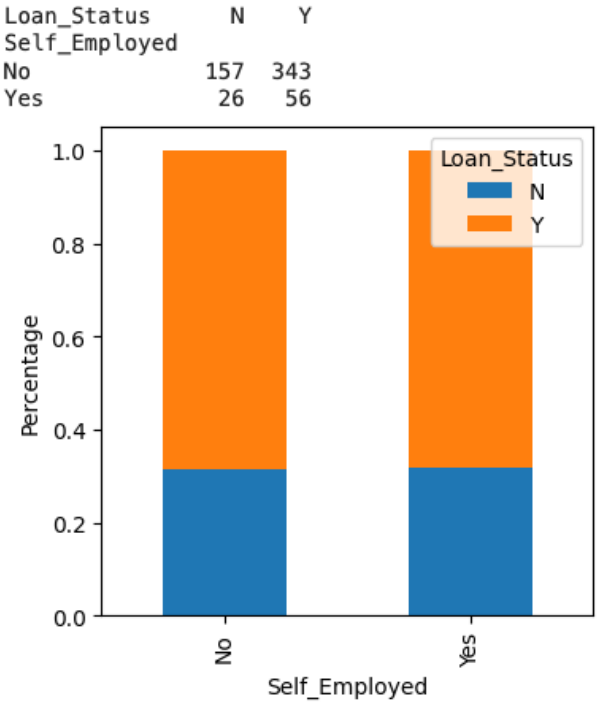


Рис. 3.15. Заявники **'Self_Employed'** проти **'Loan_Status'**

Loan_Status	N	Y
Credit_History		
0.0	82	7
1.0	97	378

Рис. 3.16. Заявники з позитивною кредитною історією

Loan_Status	N	Y
Property_Area		
Rural	69	110
Semiurban	54	179
Urban	69	133

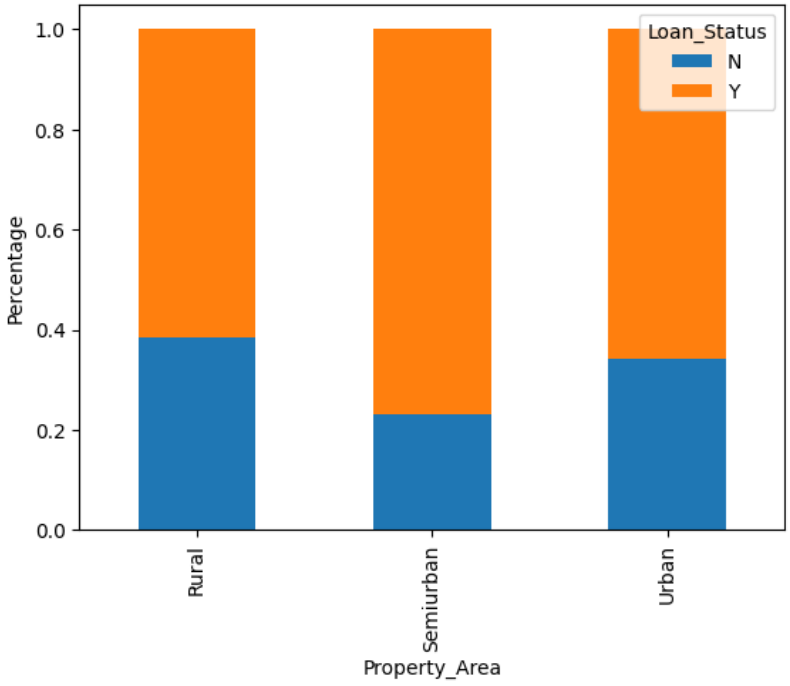


Рис. 3.17. Заявники за локалізацією

З аналізу даних можна зробити такі висновки:

- Частка заявників чоловічої та жіночої статі є приблизно однаковою серед тих, хто отримав і не отримав схвалення на позику.
- Одружені претенденти мають більшу ймовірність отримання схвалення на кредит.
- Заявники з одним або трьома і більше утриманцями мають схожий розподіл за статусом позики.
- Із діаграми 'Self_Employed' проти 'Loan_Status' не можна зробити значимих висновків.
- Випускники мають більшу частку схвалених кредитів порівняно з особами без вищої освіти.
- Особи з позитивною кредитною історією (тип 1) мають більшу ймовірність схвалення кредитів.
- У напівміських районах частка схвалених кредитів є вищою, ніж у сільських чи міських.

Також, проведено аналіз середнього доходу заявників за статусом позики, використовуючи групування **(.groupby**) і стовпчасту діаграму **(.plot.bar**), де було виявлено, що середній дохід заявників не має істотного впливу на схвалення позики.

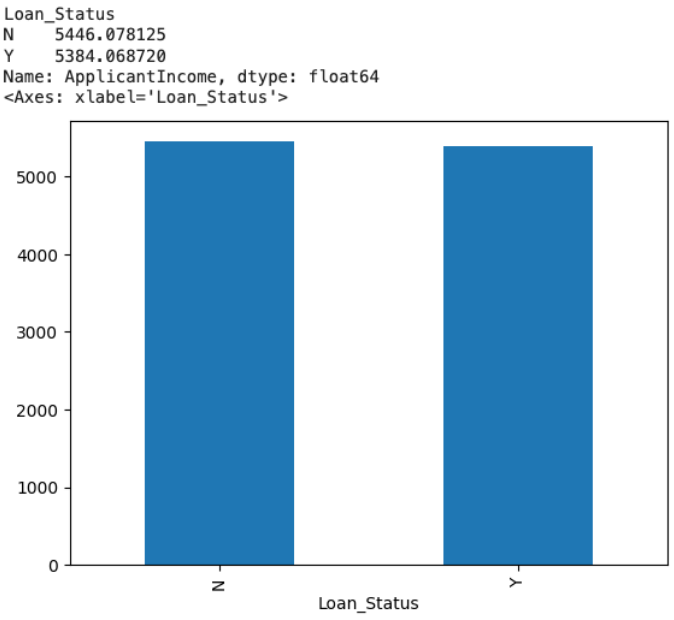


Рис. 3.18. Статус схвалення кредиту за доходом заявника

Подальша візуалізація включає створення контейнерів для змінної 'ApplicantIncome' та аналіз відповідного статусу позики для кожного контейнера. Аналіз показав, що дохід заявника не впливає на ймовірність схвалення позики, що суперечить попередній гіпотезі.

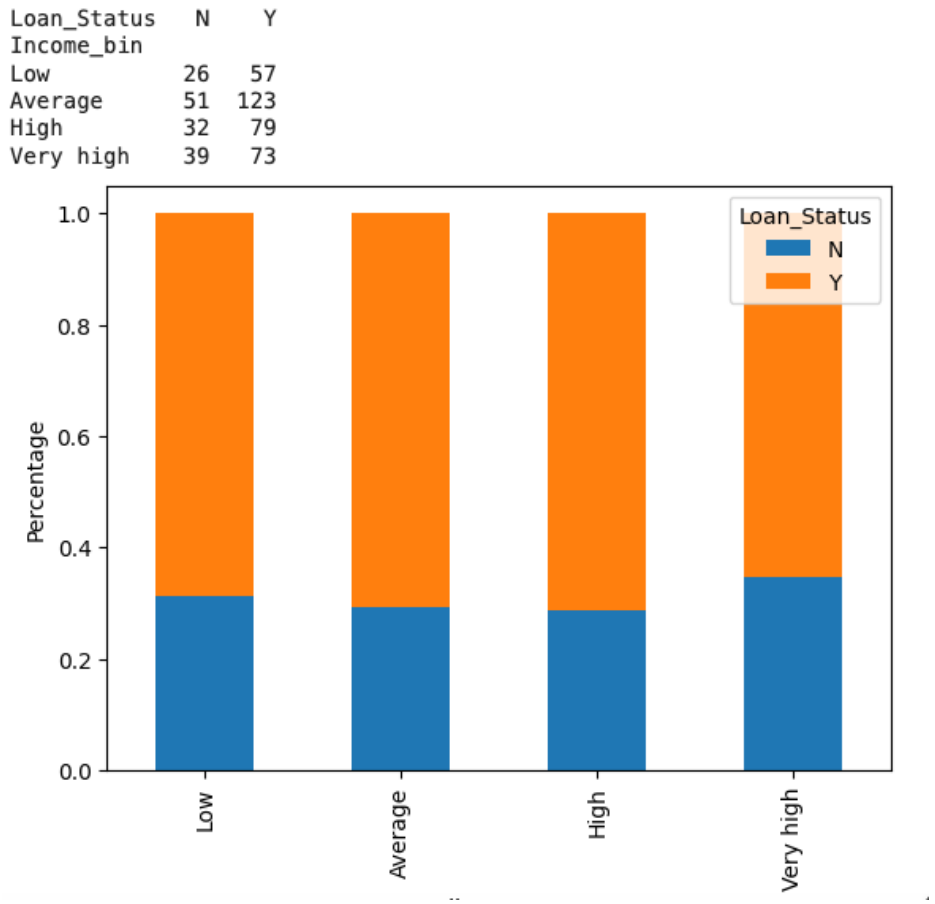


Рис. 3.19. Частка схвалення кредиту за доходом заявника

Такий самий підхід було застосовано до змінної **'CoapplicantIncome'** і суми позики **'LoanAmount'**, де було зроблено висновок, що менша сума позики корелює з більшою ймовірністю її схвалення.

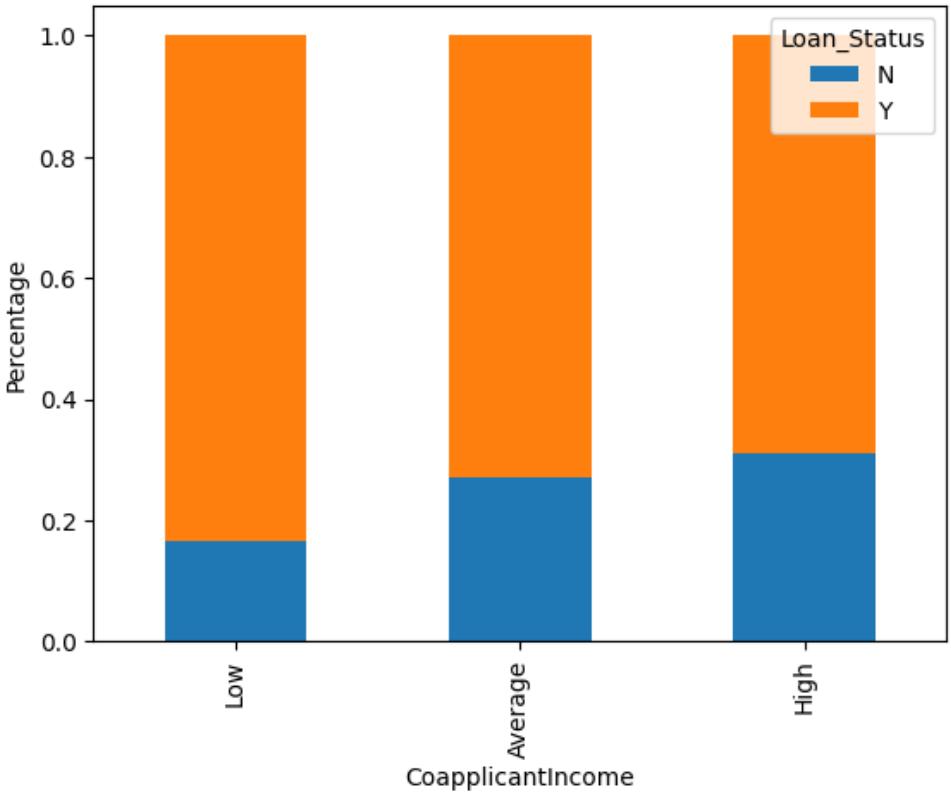


Рис. 3.20. Частка схвалення кредиту за доходом співзаявника

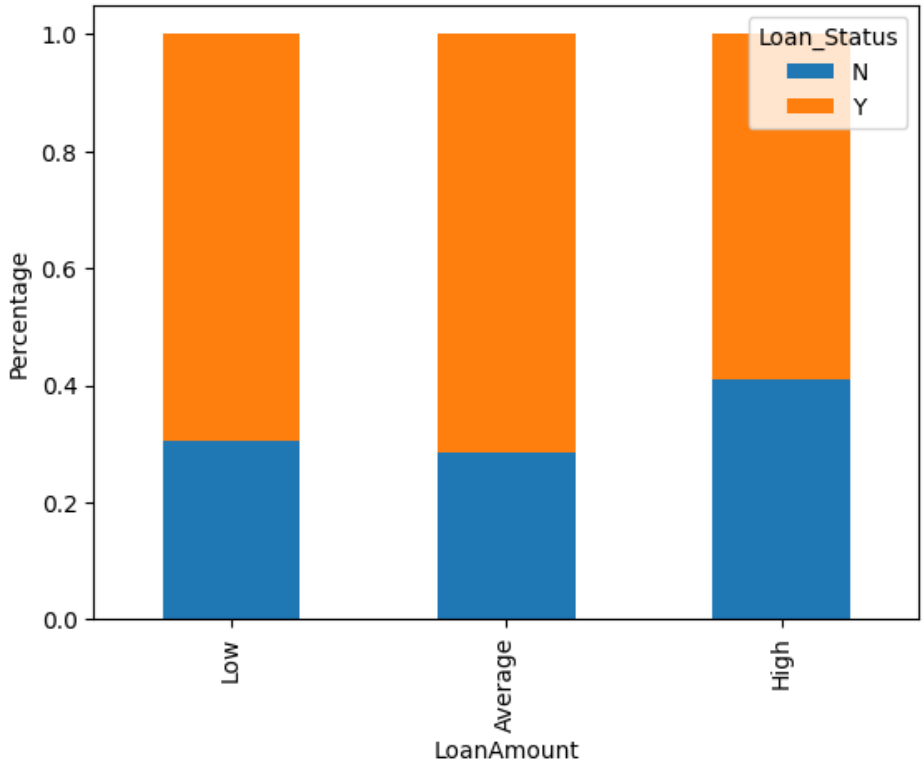


Рис. 3.21. Частка схвалення кредиту залежно від суми позики

На закінчення, були видалені створені контейнери для спрощення датасету, після чого було проведено кореляційний аналіз між всіма числовими змінними, який візуалізовано за допомогою теплової карти. Кореляція між змінними **'ApplicantIncome'** і **'LoanAmount'** а також **'Credit_History'** і **'Loan_Status'** була найвищою.

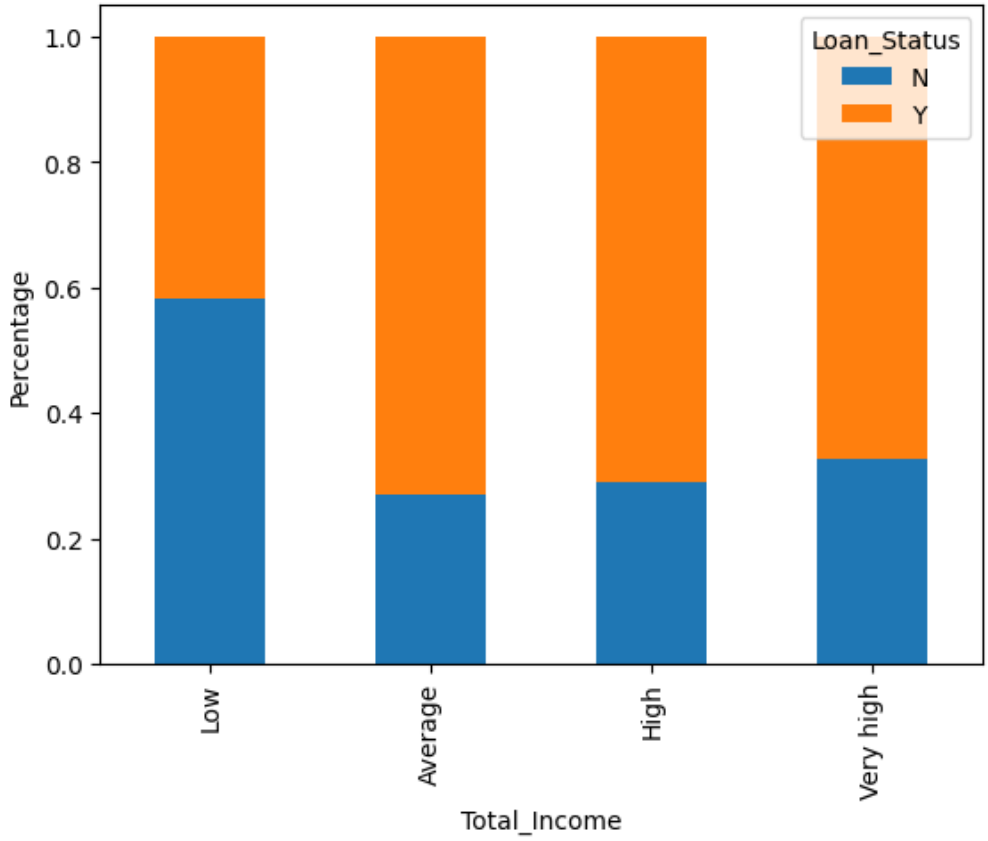


Рис. 3.21. Частка схвалення кредиту за доходом співзаявника

В рамках біфакторного аналізу ми переходимо від розгляду індивідуальних змінних до оцінювання їх взаємозв'язку з цільовою змінною — статусом позики. Такий підхід дозволяє нам перевірити наші первісні гіпотези щодо залежностей між змінними.

1. Створюємо інтервали для змінної **'LoanAmount'** (Сума позики) для класифікації позик за їх розмірами.

Таблиця 3.5

Крос-таблиця для відображення кількості схвалених та несхвалених позик

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
0	LP001002	Male	No	0	Graduate	No	5849
1	LP001003	Male	Yes	1	Graduate	No	4583
2	LP001005	Male	Yes	0	Graduate	Yes	3000
3	LP001006	Male	Yes	0	Not Graduate	No	2583
4	LP001008	Male	No	0	Graduate	No	6000

Продовження таблиці 3.5

CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0.0	NaN	360.0	1.0	Urban	Y
1508.0	128.0	360.0	1.0	Rural	N
0.0	66.0	360.0	1.0	Urban	Y
2358.0	120.0	360.0	1.0	Urban	Y
0.0	141.0	360.0	1.0	Urban	Y

Продовження таблиці 3.5

Income_bin	Coapplicant_Income_bin	Total_Income	Total_Income_bin	LoanAmount_bin
NaN	NaN	5849.0	High	NaN
High	Average	6091.0	Very high	Average
Average	NaN	3000.0	Average	Low
Average	Average	4941.0	High	Average
High	NaN	6000.0	High	Average

2. Використовуємо крос-таблицю для відображення кількості схвалених та несхвалених позик за цими категоріями суми позики, після чого візуалізуємо дані за допомогою стовпчастих діаграм.

Таблиця 3.6

Порівняльна таблиця відображення кількості схвалених та несхвалених позик

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
0	LP001002	Male	No	0	Graduate	No	5849
1	LP001003	Male	Yes	1	Graduate	No	4583
2	LP001005	Male	Yes	0	Graduate	Yes	3000
3	LP001006	Male	Yes	0	Not Graduate	No	2583
4	LP001008	Male	No	0	Graduate	No	6000

Продовження таблиці 3.6

CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0.0	NaN	360.0	1.0	Urban	Y
1508.0	128.0	360.0	1.0	Rural	N
0.0	66.0	360.0	1.0	Urban	Y
2358.0	120.0	360.0	1.0	Urban	Y
0.0	141.0	360.0	1.0	Urban	Y

Висновки з аналізу: Спостерігається, що заявники з низьким та середнім загальним доходом мають меншу частку схвалених позик порівняно з заявниками з високим доходом.

Проте, цікаво, що схвалення позик є вищим для заявників з меншими сумами позики, що підтверджує гіпотезу про вищі шанси на схвалення позики при її менших розмірах.

Після візуалізації та аналізу, ми очистили дані від тимчасово створених змінних, таких як інтервали доходів і сум позики. Далі змінили категоріальну змінну **'Dependents'** з **"3+"** на **"3"** для використання у числовому аналізі та перекодували цільову змінну **'Loan_Status'** з категорій **"N"** та **"Y"** на числові значення **"0"** та **"1"**.

Таблиця 3.7

Результат перекодування цільової змінної **'Loan_Status'** з категорій **"N"** та **"Y"** на числові значення **"0"** та **"1"**.

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	1
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	1
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	1
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	1

Завершальним етапом є аналіз кореляцій між усіма числовими змінними. Ми використовуємо кореляційну матрицю та теплові карти для відображення ступеню взаємозв'язку між змінними.

Таблиця 3.8

Вхідні параметри

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Loan_Status
ApplicantIncome	1.000000	-0.116605	0.570909	-0.045306	-0.014715	-0.004710
CoapplicantIncome	-0.116605	1.000000	0.188619	-0.059878	-0.002056	-0.059187
LoanAmount	0.570909	0.188619	1.000000	0.039447	-0.008433	-0.037318
Loan_Amount_Term	-0.045306	-0.059878	0.039447	1.000000	0.001470	-0.021268
Credit_History	-0.014715	-0.002056	-0.008433	0.001470	1.000000	0.561678
Loan_Status	-0.004710	-0.059187	-0.037318	-0.021268	0.561678	1.000000

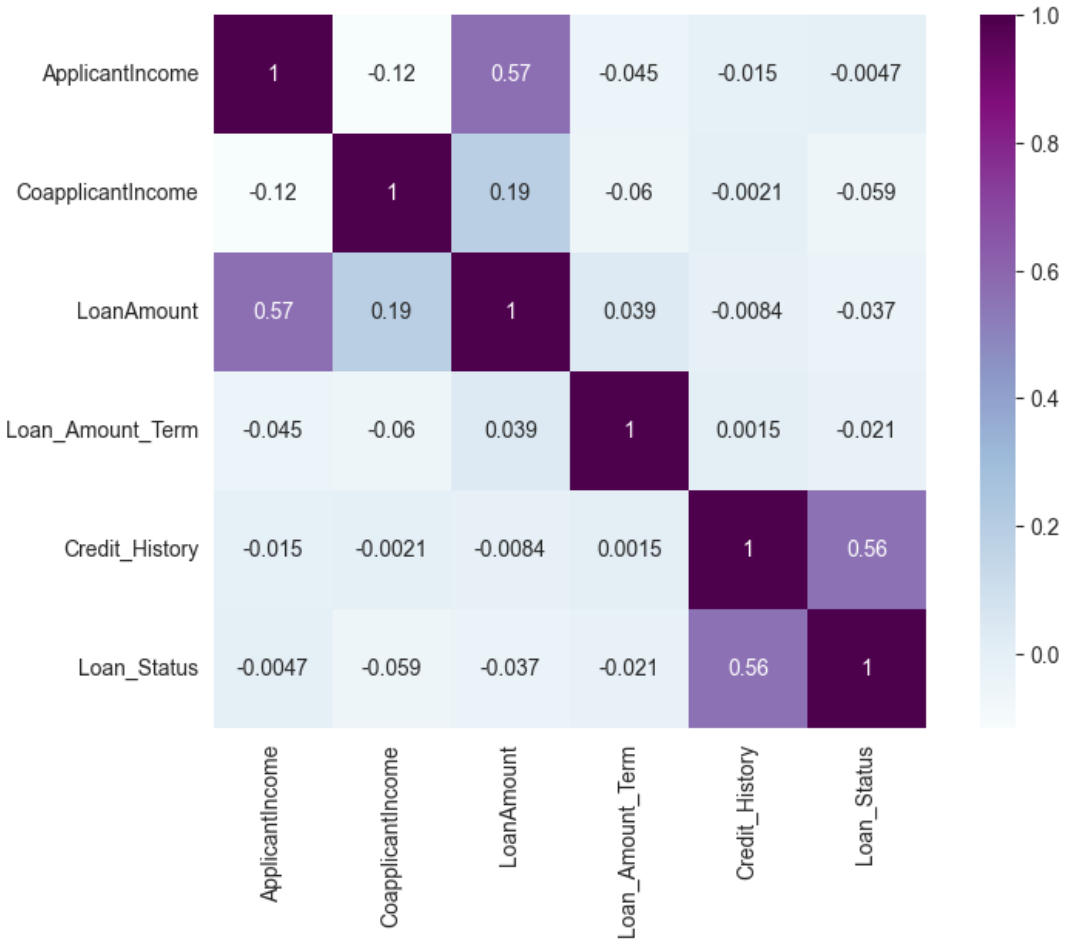


Рис. 3.22. Результати роботи кореляційного аналізу

Результати кореляційного аналізу: Найвища кореляція спостерігається між **'ApplicantIncome'** та **'LoanAmount'**, а також між **'Credit_History'** та **'Loan_Status'**, що підкреслює значення цих змінних у процесі схвалення позики. Сума позики має помірну кореляцію з доходом співзаявника, що може вказувати на важливість загального доходу сім'ї при розгляді позик.

Цей аналіз дає змогу глибше зрозуміти динаміку взаємозв'язків у даних та може слугувати важливим критерієм для прийняття обґрунтованих фінансових рішень.

3.7 Попередня обробка даних

У контексті підготовки даних до аналітичної обробки, ключовим етапом є попередня обробка, що передбачає перетворення сировини інформації в структурований та зрозумілий формат. Часто даних, з якими ми маємо справу, є неповними, містять протиріччя або не відображають чітких тенденцій, і нерідко в них виявляється значна кількість помилок. Як методологічний підхід до вирішення цих проблем пропонується використання інструментів попередньої обробки даних.

3.7.1 Обробка відсутніх значень

Вивчивши розподіл відсутніх значень у всіх змінних дослідження, ми здійснюємо їхнє призначення та коригування. Цей крок є важливим, оскільки прогалини в даних та їх викиди можуть негативно вплинути на точність та ефективність моделювання.

Підрахунок відсутніх значень: Для здійснення перевірки наявності пропущених даних ми використовуємо кодову конструкцію `**train.isnull().sum()**`, яка дозволяє виявити та кількісно оцінити відсутні дані по кожній змінній у наборі даних.

```
Loan_ID              0
Gender              13
Married              3
Dependents          15
Education            0
Self_Employed       32
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount          22
Loan_Amount_Term    14
Credit_History      50
Property_Area        0
Loan_Status          0
dtype: int64
```

Рис. 3.23. Підрахунок відсутніх значень

Підходи до заповнення відсутніх значень: Для числових змінних ми можемо використовувати імпутацію за допомогою середнього або медіани.
Для категоріальних змінних зазвичай використовується імпутація на основі моди (найчастіше зустрічного значення).

У випадках, коли пропущених даних незначна кількість, ми вдаємося до імпутації за допомогою моди. Проте, якщо незалежна змінна має значну кількість відсутніх значень, наприклад, понад 80%, то таку змінну може бути виправдано вилучити з набору даних.

Заповнення відсутніх значень: Ми заповнюємо відсутні значення в категоріальних змінних, таких як **'Gender'**, **'Married'**, **'Dependents'**, **'Self_Employed'**, **'Credit_History'**, використовуючи найчастіше зустрічне значення у цих змінних. Для числової змінної **'LoanAmount'**, ми використовуємо медіану, оскільки наявність викидів робить використання середнього значення недоречним.

Після внесення коригувань перевіряємо, що усі відсутні значення було заповнено, як у навчальному, так і в тестовому наборі даних.

Примітка: Важливо підкреслити, що при обробці тестового набору даних ми повинні використовувати показники, отримані з навчального набору, щоб уникнути витоку інформації та забезпечити коректність оцінювання моделі.

```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

Рис. 3.24. Відсоток пропусків навчальному наборі даних

```
Loan_ID               0
Gender                0
Married               0
Dependents            0
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term      0
Credit_History        0
Property_Area         0
dtype: int64
```

Рис. 3.25. Відсоток пропусків тестовому наборі даних

3.7.2 Виправлення викидів

Як ми вже зазначали раніше під час однофакторного аналізу, змінна LoanAmount містить викиди. Надзвичайно важливо врахувати їх, оскільки присутність викидів може суттєво спотворити розподіл даних. Викиди часто впливають на середнє значення та стандартне відхилення, внаслідок чого зміщується загальний розподіл. Необхідно вжити заходів для мінімізації викидів у наших наборах даних.

Через ці викиди основна маса даних за сумами кредитів зосереджена зліва, а правий край розподілу є довшим, що відомо як позитивна асиметрія. Один із способів вирішення цієї асиметрії — застосування логарифмічного перетворення. Враховуючи, що ми застосовуємо перетворення за логарифмом, воно не істотно впливає на менші значення, але значно зменшує більші значення, тож ми отримуємо розподіл, близький до нормального.

Давайте візуалізуємо ефект логарифмічного перетворення, і одночасно ми внесемо відповідні зміни у тестовий файл.

До логарифмічного перетворення: Гістограми змінної 'LoanAmount' для навчального та тестового наборів даних відображають початковий розподіл сум кредитів.

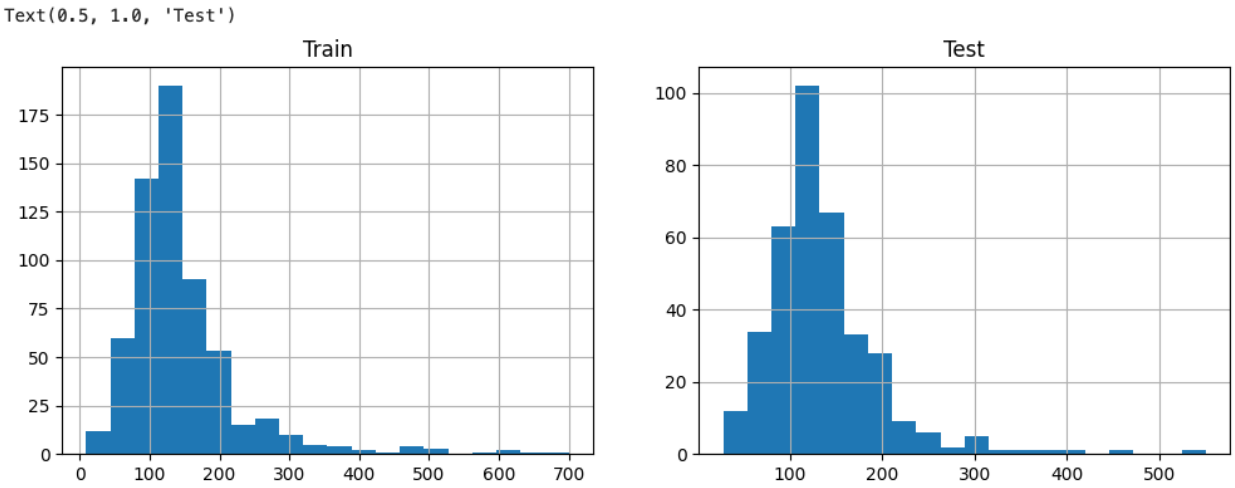


Рис. 3.26. Гістограми змінної 'LoanAmount' для навчального та тестового наборів даних до логарифмічного перетворення

Застосування логарифмічного перетворення для виправлення викидів у змінній LoanAmount: Змінна **'LoanAmount'** трансформується за допомогою природнього логарифму та зберігається у новій змінній **'LoanAmount_log'** для обох наборів даних — навчального та тестового.

Після логарифмічного перетворення: Гістограми трансформованої змінної **'LoanAmount_log'** для обох наборів даних показують розподіл, який значно ближчий до нормального і зменшує вплив екстремальних значень.

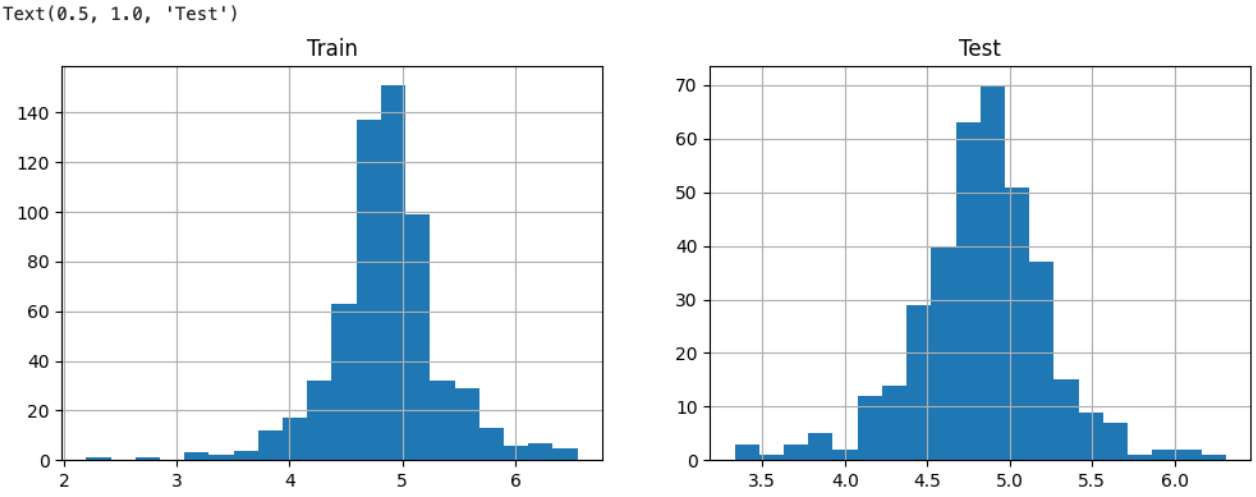


Рис. 3.27. Гістограми змінної '**LoanAmount**' для навчального та тестового наборів даних після логарифмічного перетворення

Тепер розподіл виглядає значно більш нормальним, і вплив екстремальних значень істотно зменшено. Давайте побудуємо модель логістичної регресії та здійснимо прогнозування для тестового набору даних.

3.8 Розробка та оцінка моделі

3.8.1 Побудова моделі логістичної регресії

Дане дослідження було розпочате зі створення моделі логістичної регресії, яка була то призначена для передбачення дихотомічних результатів. Як алгоритм класифікації, логістична регресія використовує асортимент незалежних змінних для оцінки ймовірності бінарних результатів, таких як **'Так/Ні'**, **'Правда/Неправда'** або **'1/0'**.

Кутовим каменем цієї передбачуваної моделі є функція **Логіт** (**Logit**), яка математично відображає логарифм шансів на користь події. Ця функція створює S-подібну криву, відому як сигмоїдна функція, яка є суттєвою для моделювання бінарних результатів, оскільки вона відноситься до ймовірності певного класу чи події.

Як прелюдію до тренування моделі, ми вилучили атрибут **'Loan_ID'**, визнаючи його як несуттєвий для передбачення статусу позики. Цей крок забезпечує, що наша модель не буде змінюватись зайвими змінними. Щоб пристосувати наші алгоритми машинного навчання, які потребують числового вводу, ми застосували метод двійкового кодування.

```
array([1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1], dtype=int64)
```

Ця техніка є інструментальною у перетворенні категоріальних даних у двійковий числовий формат, таким чином дозволяючи оцінювати передбачувальні зв'язки всередині моделі.

Використовуючи відкриту бібліотеку **scikit-learn**, універсальний набір інструментів для програмування на Python, ми розробили різні моделі. Бібліотека scikit-learn шанована за свій всебічний набір вбудованих функціональностей, які сприяють ефективним процесам моделювання.

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

Щоб підвищити здатність моделі навчатися з даних, ми розділили набір даних на тренувальний набір, використаний для калібрування моделі, та набір для перевірки, використаний для оцінки її передбачуваної валідності. Тренувальний набір дозволяє моделі навчитися зв'язку між змінними та результатом, тоді як набір для перевірки служить для оцінки передбачень моделі проти відомих результатів.

Після тренування моделі, ми перейшли до передбачення статусу позики набору для перевірки, досягнувши похвальної точності понад 82%. Це вказує на високий рівень точності моделі у передбаченні статусу затвердження позики.

Accuracy_score = 0.802

Щоб забезпечити відтінений огляд продуктивності моделі, була згенерована матриця плутанини, яка надає інсайти щодо точності передбачень.

```
[[ 23  28]
 [  4 130]]

Text(50.722222222222214, 0.5, 'True')
```

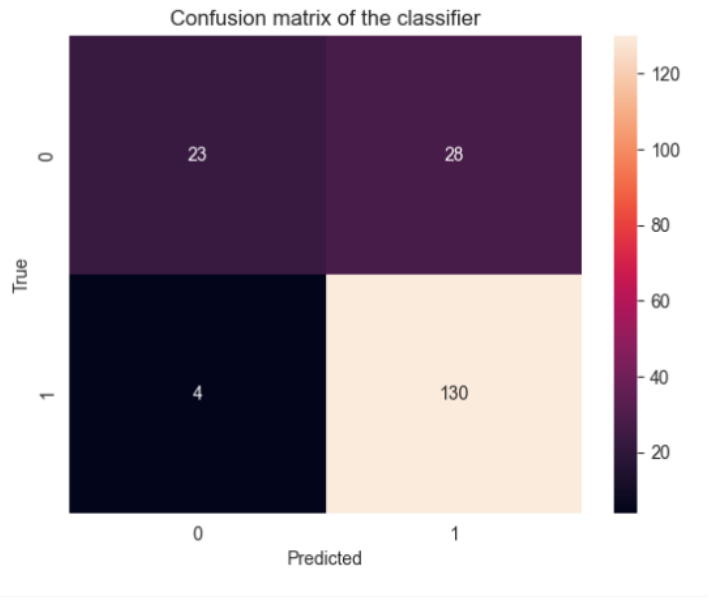


Рис. 3.28. Матриця плутанини

Крім того, був виготовлений класифікаційний звіт, що деталізує ключові метрики, такі як точність, відгук та f1-бал. Ці метрики підкреслюють ефективність моделі, з точністю 83%, точністю 82% та дивовижним відгуком 97%.

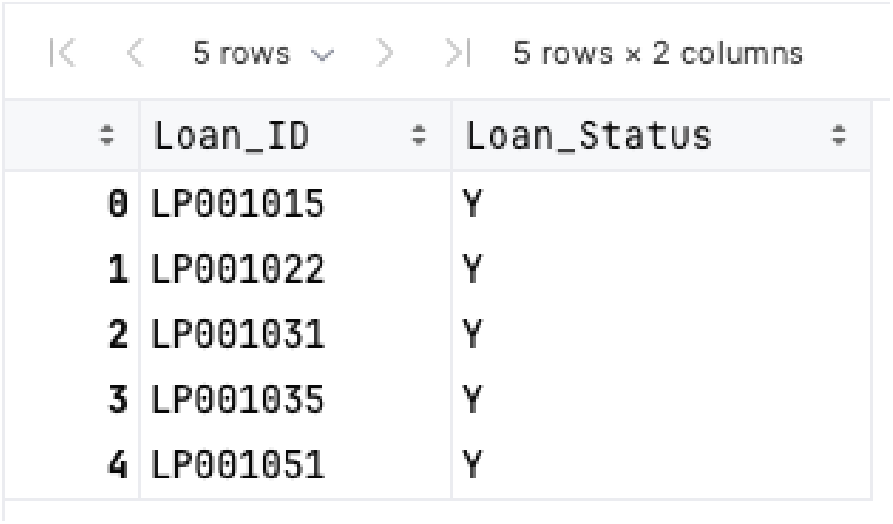
	precision	recall	f1-score	support
0	0.85	0.45	0.59	51
1	0.82	0.97	0.89	134
accuracy			0.83	185
macro avg	0.84	0.71	0.74	185
weighted avg	0.83	0.83	0.81	185

Рис. 3.29. Класифікаційний звіт

Міцність моделі була додатково підтверджена застосуванням її до незалежного тестового набору даних. Було зроблено передбачення, які мали бути подані для

зовнішньої оцінки, потребуючи лише **'Loan_ID'** та відповідного передбаченого **'Loan_Status'**.

В очікуванні подання, передбачення були перетворені у передбачений двійковий формат—конвертуючи числові передбачення в **'Y'** для так і **'N'** для ні.



	Loan_ID	Loan_Status
0	LP001015	Y
1	LP001022	Y
2	LP001031	Y
3	LP001035	Y
4	LP001051	Y

Рис. 3.30. Результати для зовнішньої оцінки

Останнім кроком було перетворення поданого набору даних у формат файлу CSV, позбавленого індексів рядків, для відповідності стандартам подання оцінювальної платформи.

З цього подання ми отримали точність 78.47% на зовнішньому лідерборді, демонструючи ефективність моделі та підтверджуючи її передбачувальні здібності.

В подальшому досліджені є сенс зробити перехресну перевірку як альтернативну стратегію для додаткової перевірки передбачень моделі, щоб забезпечити її узагальнюваність та зменшуючи потенційний ризик перенавчання.

3.8.2 Логістична регресія з використанням стратифікованої k-кратної перехресної перевірки

Щоб визначити стійкість нашої прогностичної моделі до невідомих даних, необхідно провести валідаційний тест. Ця техніка передбачає відокремлення

певного зразка даних, який не використовувався раніше для навчання моделі. Після цього модель оцінюватиметься на цьому зарезервованому зразку перед її остаточним завершенням.

- Нижче наведено декілька поширених методів валідації:
- K-кратна перехресна перевірка
- Перехресна перевірка залишити-один-за-межами (LOOCV)
- Стратифікована K-кратна перехресна перевірка

Ці методи є невід'ємною частиною процесу валідації, надаючи оцінку продуктивності моделі на незалежному наборі даних. Стратифікація гарантує, що кожна частина даних містить приблизно однаковий відсоток зразків кожного цільового класу, як і повний набір, тим самим підтримуючи розподіл класів і знижуючи упередженість у процесі валідації. Цей всебічний підхід до валідації є вирішальним у розробці надійної та узагальнювальної прогностичної моделі.

Стратифікація – це процес реорганізації даних так, що кожна підгрупа (складка) адекватно представляє всю вибірку.

Коли кількість складок K дорівнює загальній кількості спостережень **N**, метод називається **"Leave One Out Cross Validation"** перевіркою.

Представлено ілюстрацію стратифікованої k-кратної перевірки з k=5.

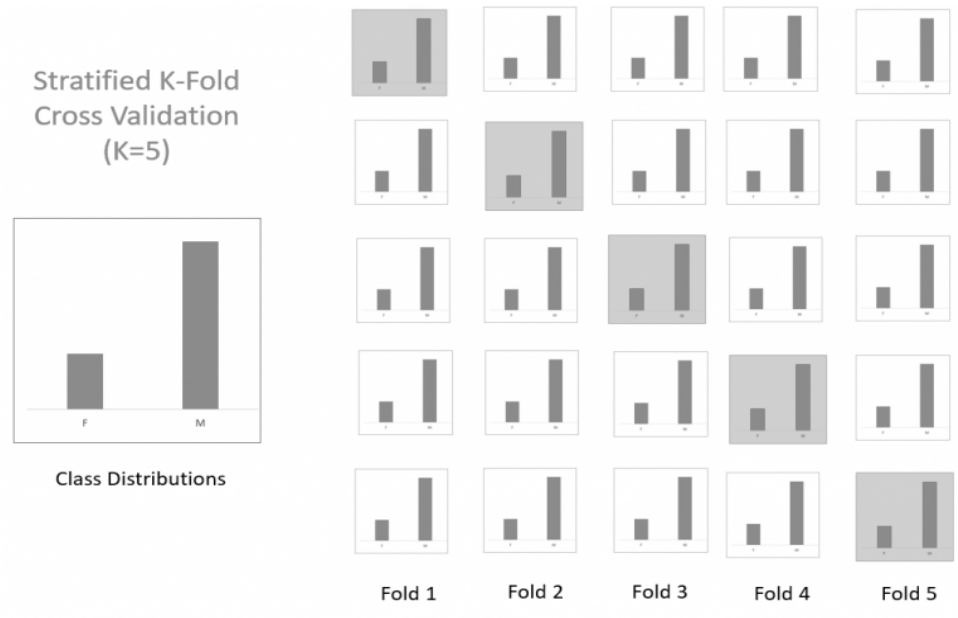


Рис. 3.31. Ілюстрація стратифікованої k-кратної перевірки з k=5

Щоб провести стратифіковану перехресну перевірку, ми використовуємо клас **StratifiedKFold** із бібліотеки scikit-learn, який дозволяє тренувати модель логістичної регресії з використанням 5-кратної стратифікації і робити прогнози для тестового набору даних. Складки формуються з урахуванням збереження пропорцій класів у кожній підгрупі.

```
1 of kfold 5
accuracy_score 0.8048780487804879

2 of kfold 5
accuracy_score 0.8373983739837398

3 of kfold 5
accuracy_score 0.7804878048780488

4 of kfold 5
accuracy_score 0.7967479674796748

5 of kfold 5
accuracy_score 0.7950819672131147
```

Рис. 3.32. Результати стратифікованої k-кратної перевірки з k=5.

Середня точність валідації цієї моделі становить 0.802

Далі відбувається візуалізація ROC-кривої, що демонструє залежність між часткою істинно позитивних і хибно позитивних результатів.

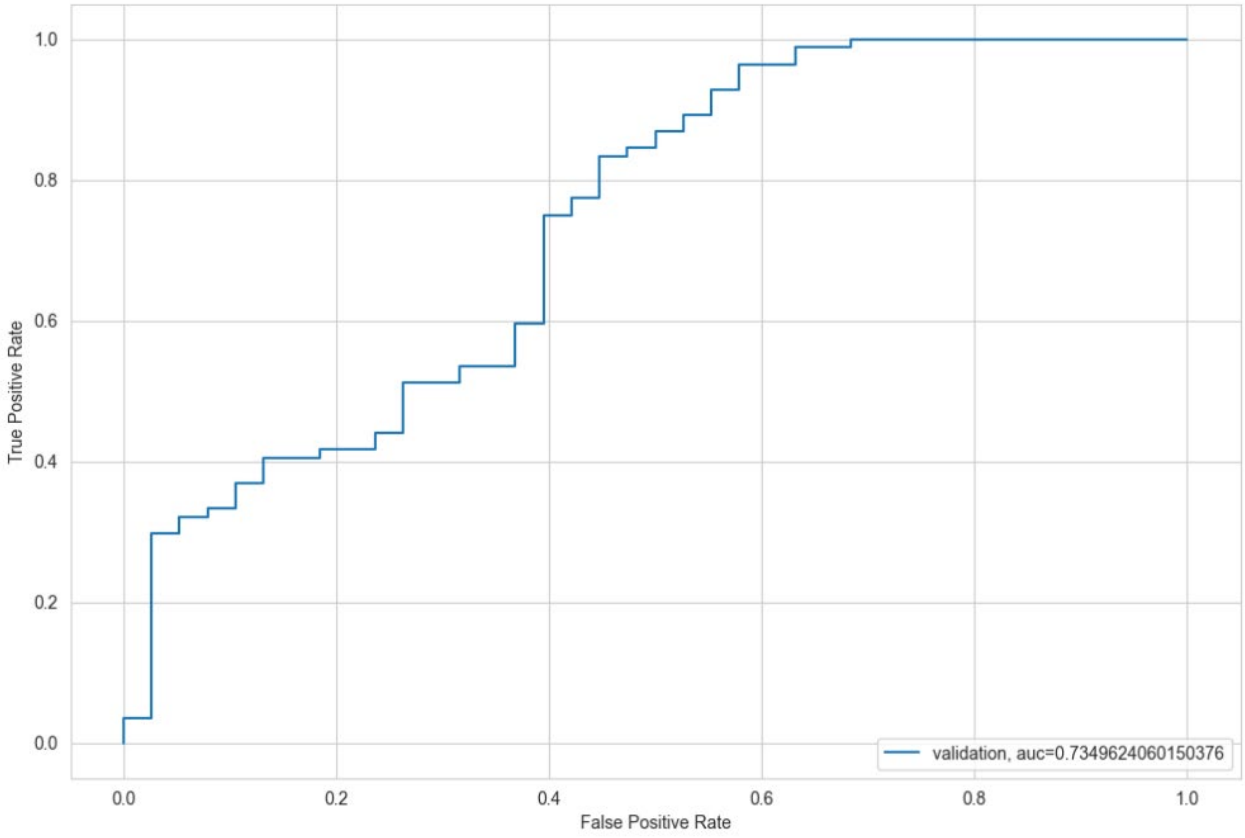


Рис. 3.33. Візуалізація ROC-кривої (AUC)

Значення площі під кривою (AUC) дорівнює 0.734, що свідчить про достатню здатність моделі розрізняти класи.

Після отримання прогнозів, виконується їх конвертація з числового формату в категорійний (Y і N), а потім – експорт у формат CSV для подальшої оцінки точності моделі.

Використовуючи цю оцінку, ми прагнемо до підвищення точності моделі за допомогою різних методів вдосконалення.

3.8.3 Розробка функцій

На основі знань у відповідній предметній області, нами були розроблені нові функції, які можуть вплинути на цільову змінну.

Вводяться три нові характеристики:

1. **Загальний дохід:** Це об'єднання доходів заявника та співзаявника, яке обговорювалося під час біваріативного аналізу. Виходить з припущення, що якщо загальний дохід високий, ймовірність схвалення кредиту також може бути високою.
2. **Рівноцінний щомісячний внесок**: що представляє собою місячну суму, яку заявник зобов'язаний сплачувати для погашення кредиту. Логіка створення цієї змінної полягає в тому, що особам із високим рівноцінним щомісячним внеском може бути складно повернути позику. Рівноцінний щомісячний внесок можна обчислити, взявши відношення суми позики до терміну позики.
3. **Балансовий дохід:** Це дохід, що залишається після сплати рівноцінного щомісячного внеску. Ідея створення цієї змінної ґрунтується на розумінні, що якщо ця величина висока, існує більша ймовірність повернення позики, отже, зростають шанси на схвалення кредиту.

Практичне втілення включало об'єднання доходів заявника та співзаявника в нову змінну **'Total_Income'** та перевірку її розподілу, який був зміщений вліво, що вказує на правостороннє спотворення. Було застосовано логарифмічне перетворення для нормалізації розподілу.

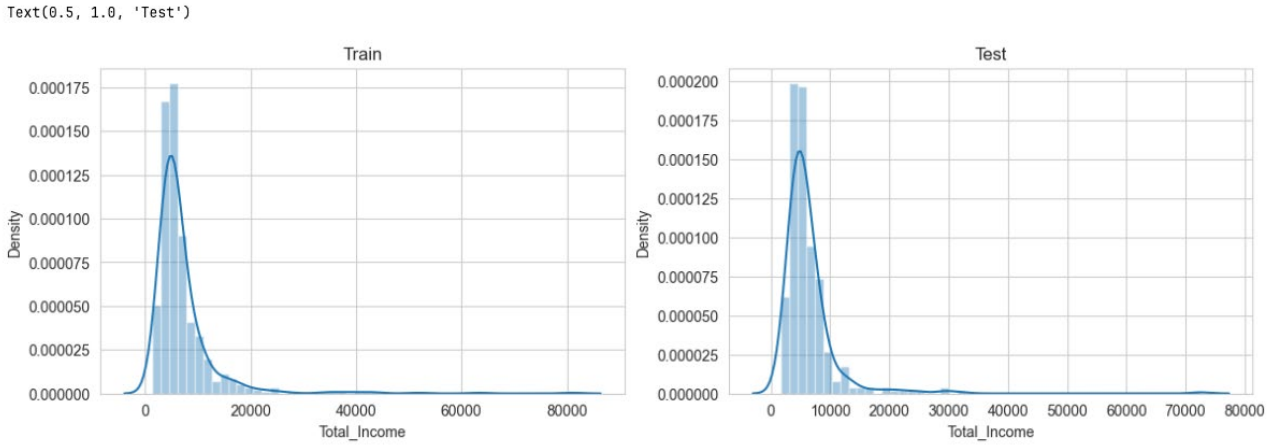


Рис. 3.34. Застосування логарифмічного перетворення для нормалізації розподілу

Далі була розрахована характеристика рівноцінного щомісячного внеску шляхом ділення 'LoanAmount' на 'Loan_Amount_Term'. Це вважається наближенням до фактичного рівноцінного щомісячного внеску. Було досліджено розподіл рівноцінного щомісячного внеску неперервної числової змінної.

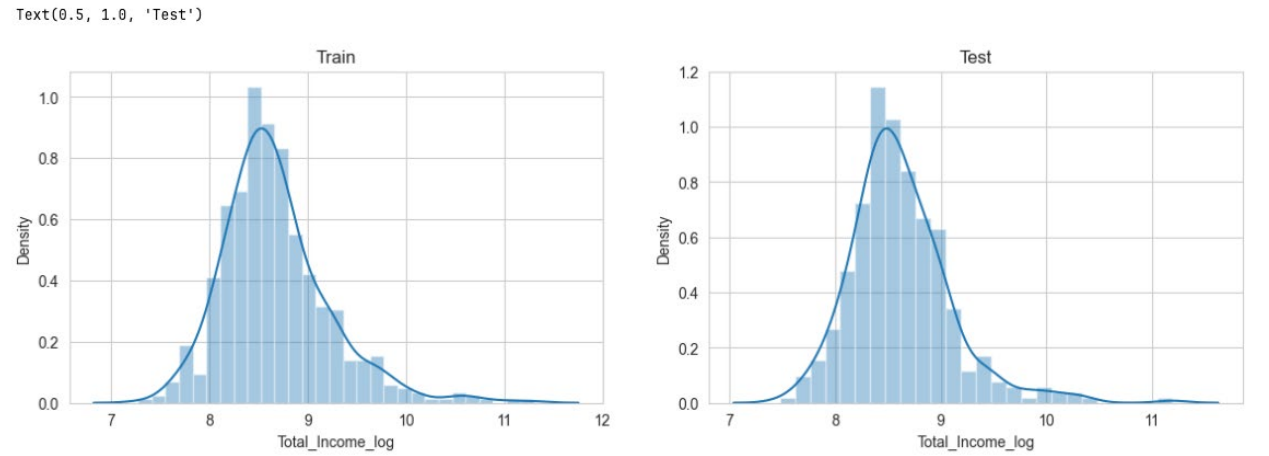


Рис. 3.35. Розрахування характеристики рівноцінного щомісячного внеску

Після цього була створена змінна **'Balance_Income'**, яка представляє дохід, що залишається після сплати рівноцінного щомісячного внеску, та було проаналізовано її розподіл.

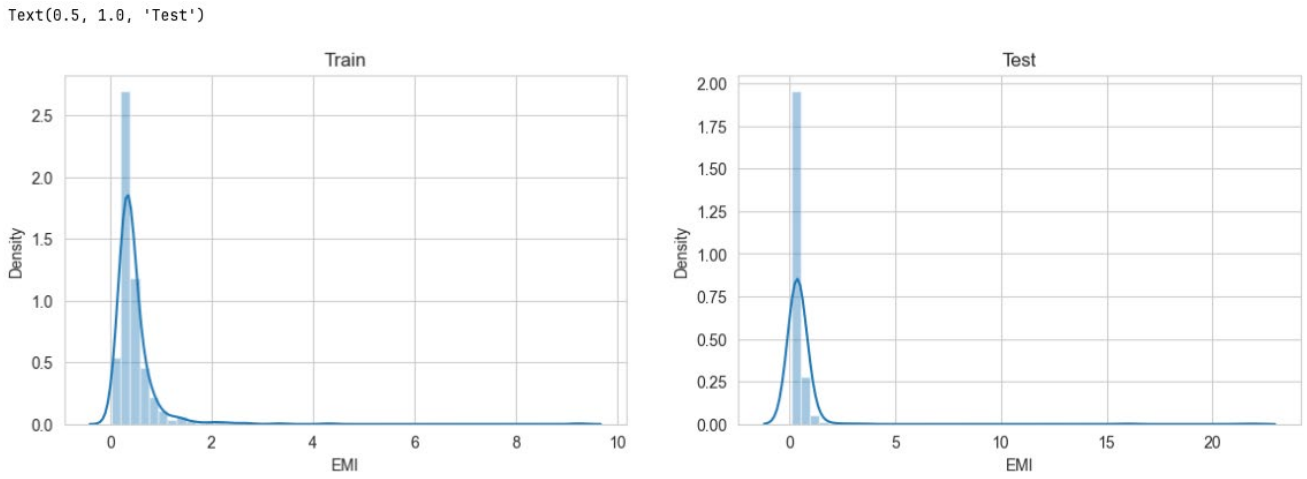


Рис. 3.36. Результат розподілу функції Balance Income після сплати рівноцінного щомісячного внеску **(Equated Monthly Installment (EMI))**

Нарешті, змінні, які використовувались для створення цих нових характеристик, були відкинуті для зменшення шуму в наборі даних та уникнення проблеми високої кореляції між цими старими та новими характеристиками, оскільки логістична регресія передбачає, що змінні не мають бути сильно корельованими.

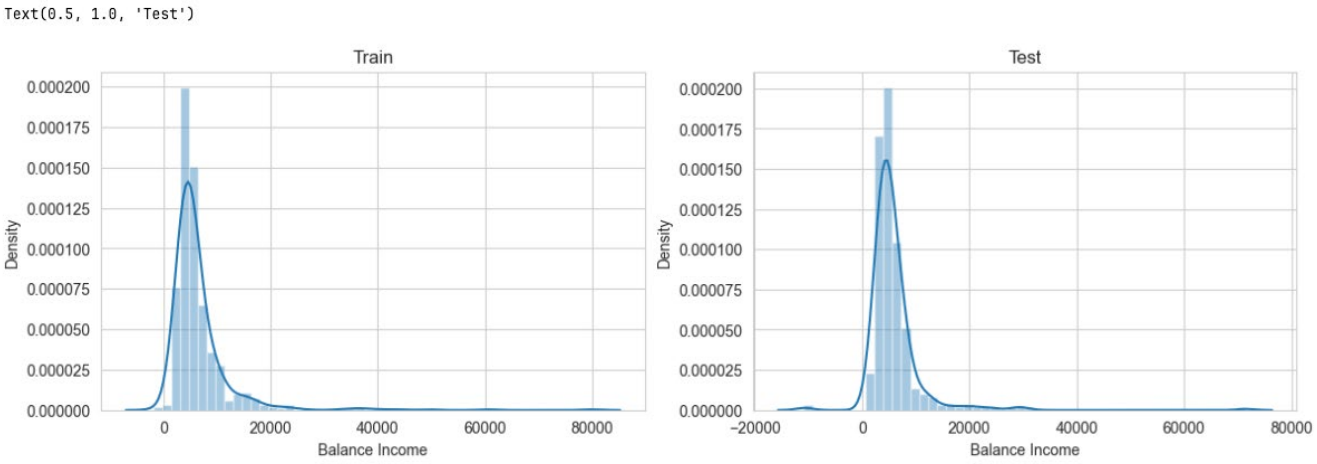


Рис. 3.37. Результат розподілу після відкидання змінних для зменшення шуму

Процес включав:

1. Об'єднання **'ApplicantIncome'** та **'CoapplicantIncome'** у нову змінну **'Total_Income'**.
2. Перевірка розподілу **'Total_Income'** до і після логарифмічного перетворення.
3. Обчислення рівноцінного щомісячного внеску шляхом ділення **'LoanAmount'** на **'Loan_Amount_Term'**.
4. Перевірка розподілу рівноцінного щомісячного внеску.
5. Створення змінної **'Balance Income'** шляхом віднімання рівноцінного щомісячного внеску (помноженого на 1000 для узгодження одиниць) від **'Total_Income'**.
6. Перевірка розподілу **'Balance Income'**.

Ця методологія має на меті удосконалити прогностичну модель шляхом інтеграції всебічного аналізу доходів та здатності до погашення боргу, що призводить до більш точного та ефективного прогнозування схвалення кредитів.

3.8.4 Удосконалення логістичної регресії за допомогою розроблених функцій

У контексті кваліфікаційної роботи, була зроблена спроба удосконалення логістичної регресії за допомогою розроблених функцій та проведено оцінювання продуктивності моделі за допомогою стратифікованої k-кратної перехресної перевірки.

Первісний крок включає підготовку даних для введення у модель шляхом відокремлення прогностичної цільової змінної **'Loan_Status'** від тренувального набору даних і збереження її окремо. Цей етап підготовки є важливим для забезпечення цілісності та відповідності даних перед тренуванням моделі. Після підготовки даних було застосовано модель логістичної регресії.

В процесі було використано стратифіковану k-кратну перехресну перевірку з п'ятьма складками та механізмом перемішування перед розподілом, щоб визначити стійкість моделі. Ця техніка є інструментальною в підтриманні послідовної

пропорції класових міток у кожній складці, забезпечуючи, щоб кожна партія даних була репрезентативною для загального набору даних.

У процесі ітеративної перехресної перевірки модель тренується на тренувальній підмножині, і робляться прогнози на підмножині перевірки. Точність моделі обчислюється для кожної складки, і розраховується кумулятивна середня точність перевірки, яка служить як всебічний показник узагальненості моделі.

Після завершення перехресної перевірки робляться прогнози на тестовій підмножині, і обчислюються ймовірнісні оцінки схвалення позики. Ці ймовірності надають уявлення про ймовірність невиконання позичальником своїх зобов'язань, що є ключовим аспектом у рішеннях про схвалення позик.

```
1 of kfold 5
accuracy_score 0.7560975609756098

2 of kfold 5
accuracy_score 0.7235772357723578

3 of kfold 5
accuracy_score 0.6666666666666666

4 of kfold 5
accuracy_score 0.7804878048780488

5 of kfold 5
accuracy_score 0.7540983606557377
```

Рис. 3.38. Результати перехресної перевірки удосконаленої логістичної регресії

classification_report:

	precision	recall	f1-score	support
0	0.83	0.26	0.40	38
1	0.75	0.98	0.85	84
accuracy			0.75	122
macro avg	0.79	0.62	0.62	122
weighted avg	0.77	0.75	0.71	122

Рис. 3.39. Результати класифікацій удосконаленої логістичної регресії

Confusion matrix of the classifier:

[[10 28]
[2 82]]

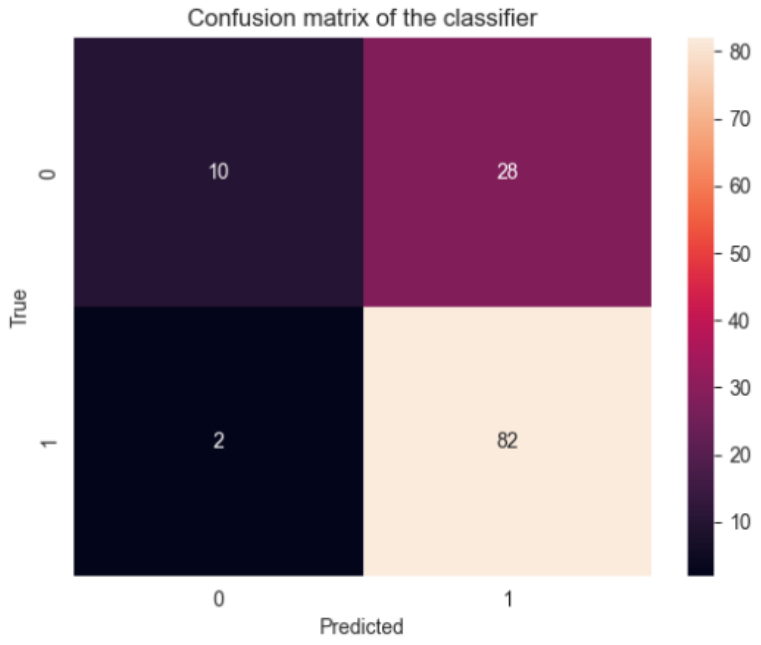


Рис. 3.40. Результати матриці плутанини удосконаленої логістичної регресії

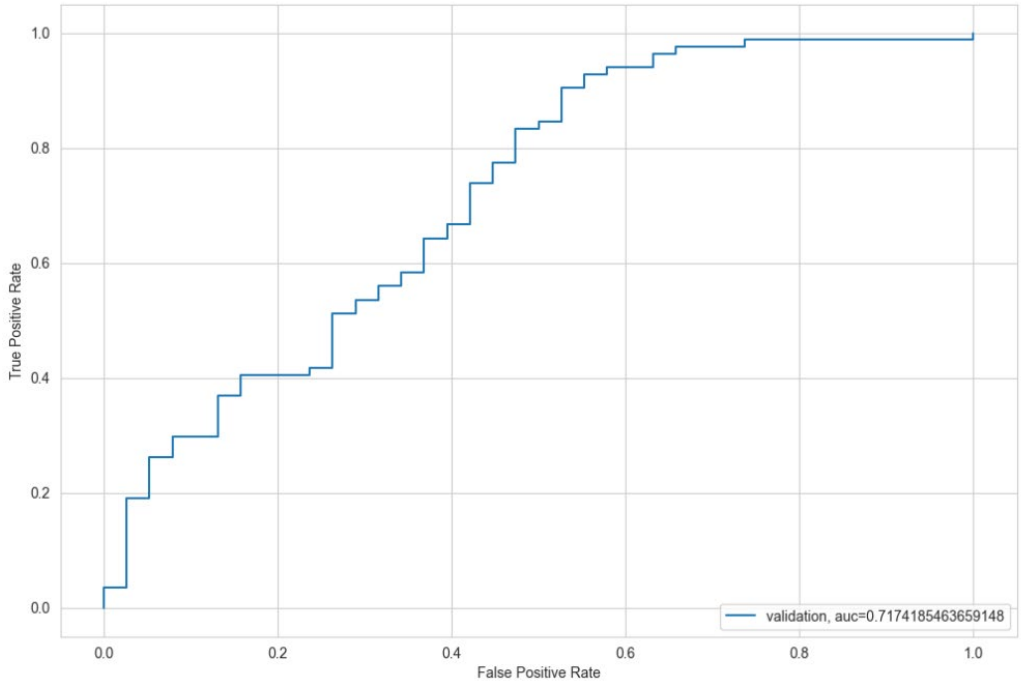


Рис. 3.41. Результати ROC-кривої (AUC) удосконаленої логістичної регресії

Середня точність перевірки, досягнута на складках, вказує на середню продуктивність моделі, яка потім порівнюється з метрикою лідерборду після подання. Однак досягнута точність свідчить про те, що інженерія функцій не значно підвищила прогностичні можливості моделі. Тому це викликає необхідність дослідження альтернативних алгоритмів для покращення продуктивності моделі.

Нарешті, процедура подання включає заміну чисельних прогнозів категорійними результатами (**'N'** для 0, **'Y'** для 1) та форматування подання у файл CSV, який потім оцінюється за точністю на лідерборді.

Незважаючи на те, що модель не показала покращення з новими функціями, такий методичний підхід до удосконалення та перевірки моделі є свідченням ретельних академічних стандартів дослідження.

3.8.5 Розробка моделі дерева ухвалення рішень (Decision Tree Model Creation)

Дерево ухвалення рішень (Decision Tree) є методом навчання під наглядом який заздалегідь має визначену цільову змінну. Даний метод переважно застосовується для завдань класифікації. Ця техніка передбачає розділення батьківського набору даних на два чи більше підмножин (підпопуляцій) на основі найважливішого атрибуту для розбиття серед вхідних змінних.

Алгоритм дерева рішень застосовує кілька алгоритмів для визначення оптимального способу розбиття вузла на кілька підвузлів, тим самим підвищуючи однорідність результуючих підвузлів. Іншими словами, цей процес ефективно збільшує чистоту вузла у відношенні до цільової змінної.

Як результат дерево рішень є потужним інструментом для класифікації, який надає чітке розуміння набору даних і дозволяє прогнозувати результати з похвальною точністю.

Нижче наведено результати практичного застосування методу в процесі дослідження наукової роботи:

```
1 of kfold 5
accuracy_score 0.6991869918699187

2 of kfold 5
accuracy_score 0.7479674796747967

3 of kfold 5
accuracy_score 0.6666666666666666

4 of kfold 5
accuracy_score 0.6910569105691057

5 of kfold 5
accuracy_score 0.6885245901639344
```

Рис. 3.42. Результати перехресної перевірки дерева рішень

classification_report:

	precision	recall	f1-score	support
0	0.50	0.55	0.53	38
1	0.79	0.75	0.77	84
accuracy			0.69	122
macro avg	0.64	0.65	0.65	122
weighted avg	0.70	0.69	0.69	122

Рис. 3.43. Результати класифікацій дерева рішень

```
[[21 17]
[21 63]]
```

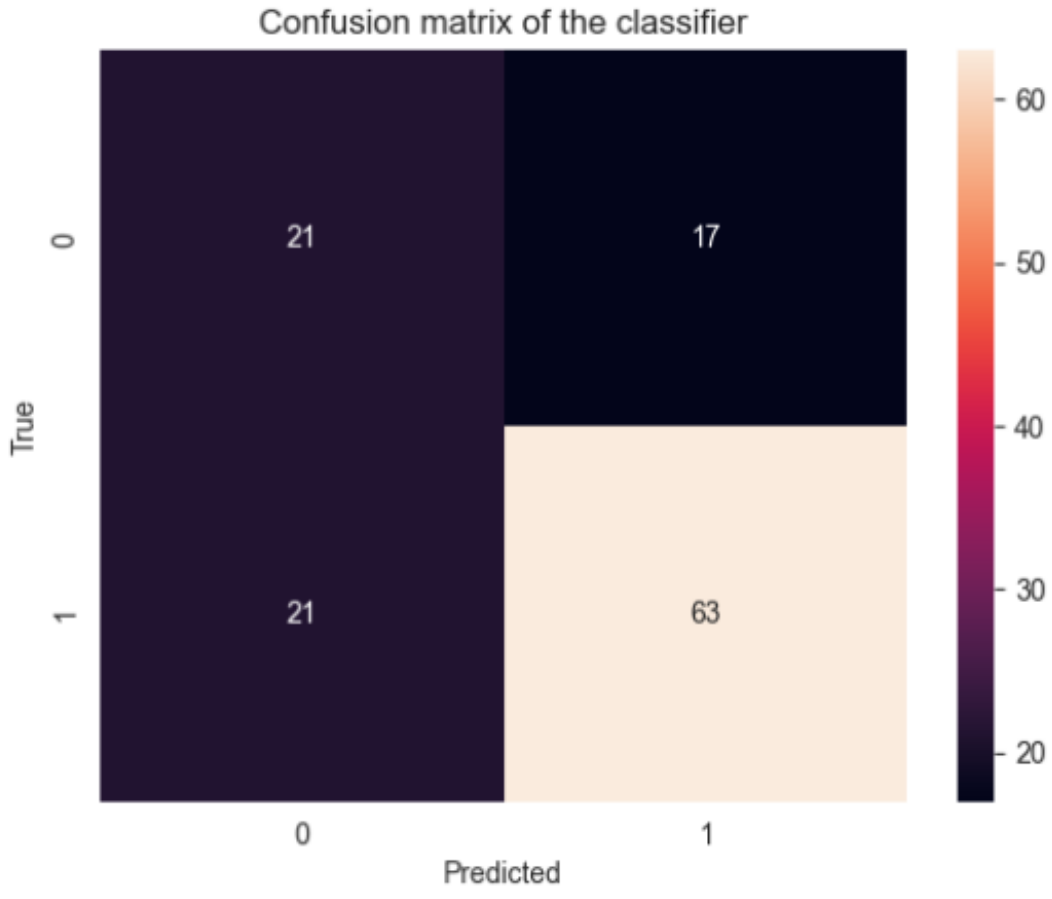


Рис. 3.44. Результати матриці плутанини дерева рішень

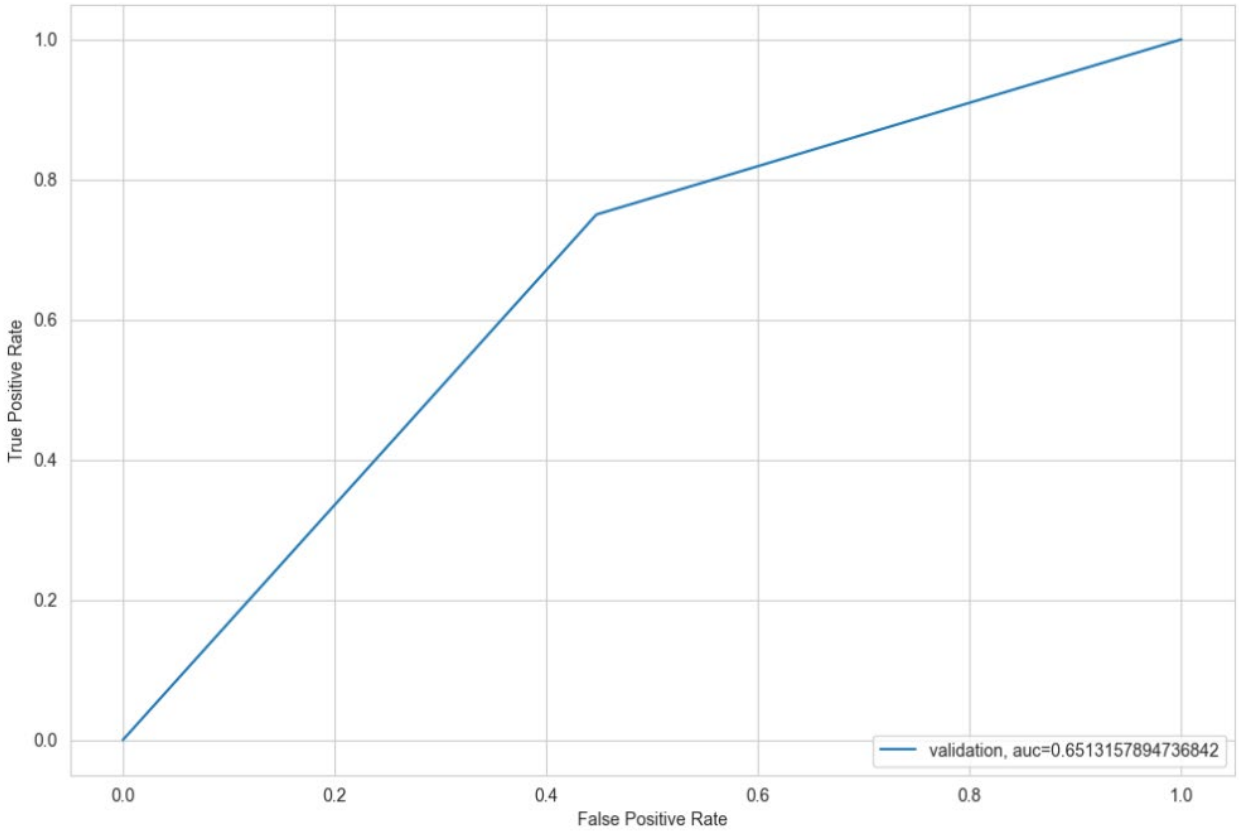


Рис. 3.45. Результати ROC-кривої (AUC) дерева рішень

Середня точність валідації для цієї моделі становить 0.68, що є нижче, ніж у моделі логістичної регресії.

Оскільки алгоритми Дерево рішень та Випадковий ліс побудовані на основі дерева, в даному випадку є доцільним застосувати Випадковий ліс в наступному дослідженні та спробувати покращити нашу модель, збільшивши точність.

3.8.6 Використання моделі Випадковий ліс

Випадковий ліс — це ансамблевий алгоритм машинного навчання, який включає множину дерев рішень, кожне з яких тренується на підмножині даних.

За допомогою вибірки з відновленням створюються численні Дерево ухвалення рішень, і кожне дерево дає свій прогноз. Кінцевий прогноз отримується шляхом усереднення прогнозів від усіх дерев у випадку регресії або вибору найпопулярнішого класу в прогнозах у випадку класифікації.

Для налаштування моделі Випадковий ліс у **sklearn** необхідно звернути увагу на такі параметри:

- **n_estimators** - кількість дерев у лісі. Зазвичай чим більше дерев, тим кращий прогноз, але також зростає ризик перенавчання. Стандартне значення — 10.
- **max_depth** - максимальна глибина кожного дерева. Обмеження глибини дерева може допомогти запобігти перенавчанню.

```
1 of kfold 5
accuracy_score 0.7886178861788617

2 of kfold 5
accuracy_score 0.8455284552845529

3 of kfold 5
accuracy_score 0.7479674796747967

4 of kfold 5
accuracy_score 0.7642276422764228

5 of kfold 5
accuracy_score 0.7786885245901639
```

Рис. 3.46. Результати перевірки середньої точності моделі дерева ухвалення рішень

```
classification_report:

              precision    recall  f1-score   support

           0       0.79      0.39      0.53        38
           1       0.78      0.95      0.86        84

    accuracy                           0.78       122
   macro avg       0.78      0.67      0.69       122
weighted avg       0.78      0.78      0.75       122
```

Рис. 3.47. Результати класифікацій дерева ухвалення рішень

```
[[15 23]
[ 4 80]]
```

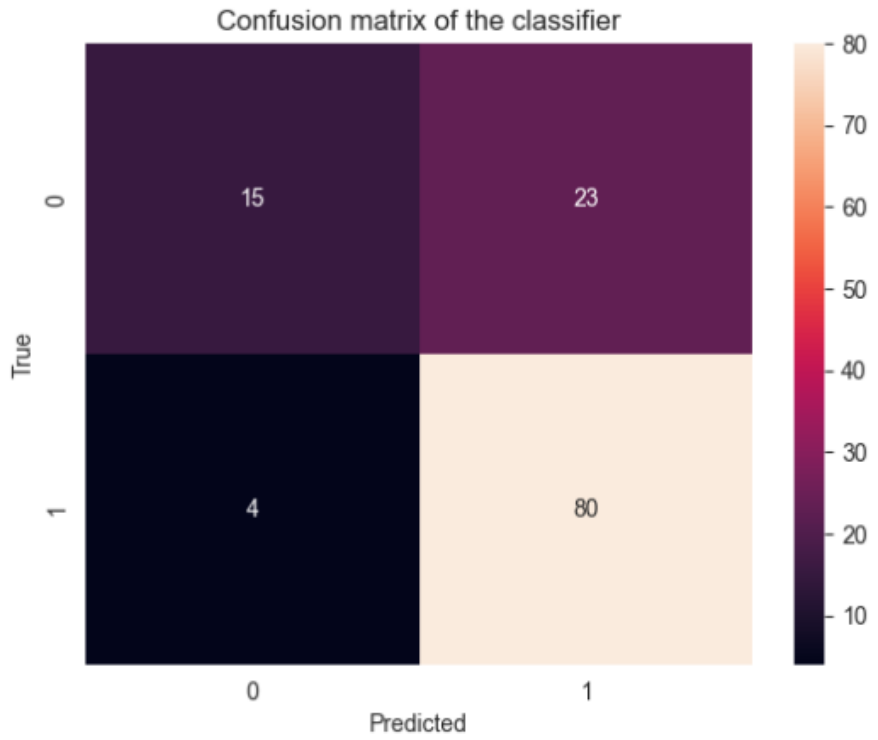


Рис. 3.48. Результати матриці плутанини дерева ухвалення рішень

За результатами перевірки середня точність моделі виявилася приблизно 0.785, що є досить високим показником

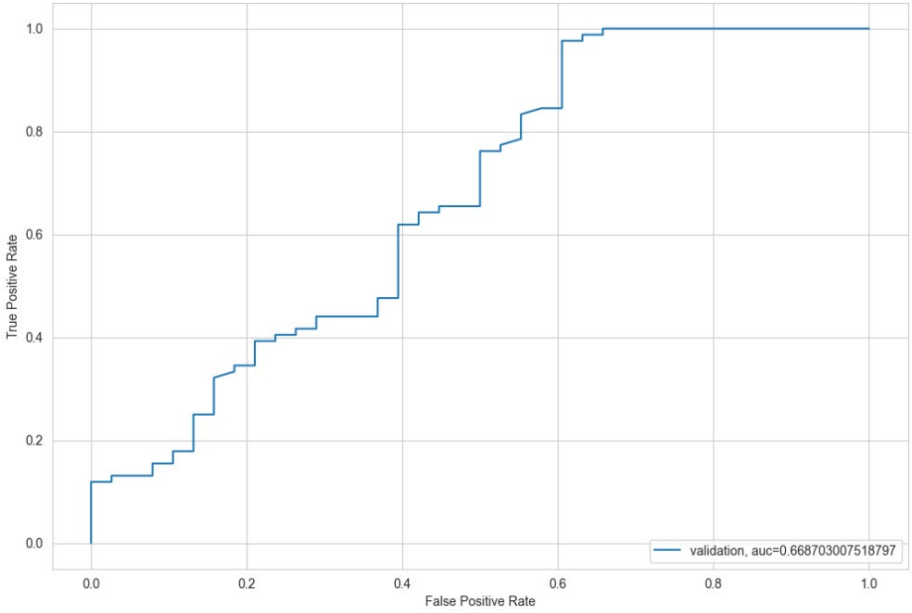


Рис. 3.49. Результати ROC-кривої (AUC) дерева ухвалення рішень

Після тренування моделі прогнози були використані для заповнення стовпця 'Loan_Status' у таблиці подання, де числові прогнози замінюються на 'Y' або 'N'. Результати експортовані у файл CSV для подальшого аналізу точності моделі.

3.8.7 Підвищення точності алгоритму Випадковий ліс через використання GridSearchCV

У контексті кваліфікаційної роботи було проведено ретельне дослідження з метою підвищення передбачувальної точності алгоритму Випадкового Лісу — ця модель належить до парадигми ансамблевого навчання. Було проведено спробу підвищення передбачувальної точності алгоритму шляхом точної корекції його гіперпараметрів за допомогою процедури GridSearchCV. Ця техніка є ключовим компонентом бібліотеки **`sklearn.model_selection`**, яка вирізняється своєю здатністю систематично проходити через переустановлений простір гіперпараметрів, щоб визначити найбільш ефективну конфігурацію моделі.

Алгоритм Випадкового Лісу функціонує, створюючи згусток дерев рішень та інтегруючи їхні індивідуальні прогнози для отримання результату, який є одночасно точним та послідовним. Зусилля щодо оптимізації зосередились на критичних гіперпараметрах: **`max_depth`**, який встановлює кінцеву глибину дерев у складі, та **`n_estimators`**, який визначає загальну кількість дерев, що будуть створені у ансамблі.

```
GridSearchCV(cv=None, error_score='raise',
       estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=1, verbose=0, warm_start=False),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'max_depth': [1, 3, 5, 7, 9, 11, 13, 15, 17, 19], 'n_estimators': [1, 21,
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

Рис. 3.50. Гіперпараметри процедури GridSearchCV

Зусилля з калібрування були організовані шляхом створення сітки параметрів з конкретними інтервалами для `**max_depth**` та `**n_estimators**`. За цим послідувало застосування тричастотної стратегії перехресної валідації, щоб з ретельністю оцінити ефективність кожної комбінації параметрів. Процес оптимізації після фази тренування вказав, що найсприятливіші гіперпараметри для моделі Випадкового Лісу — це `**max_depth**` 3 та `**n_estimators**`.

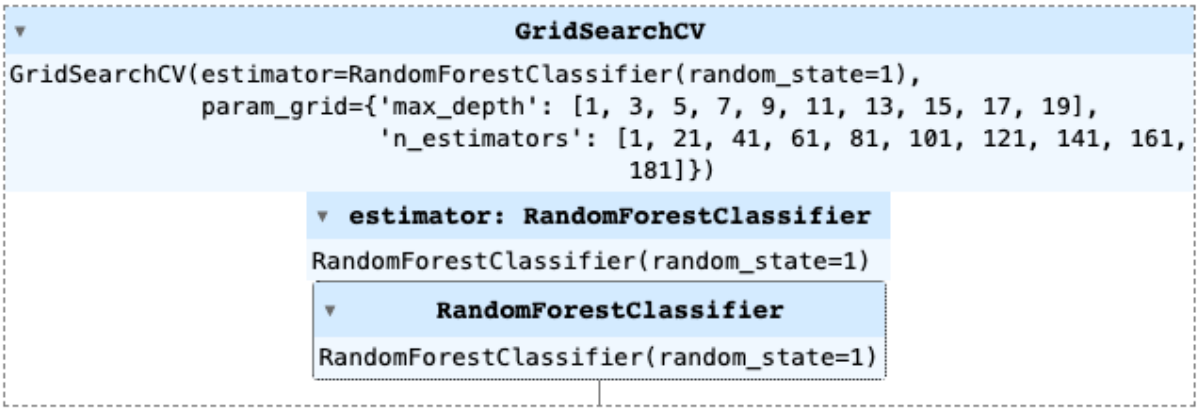


Рис. 3.51. Приклад моделі Випадкового Лісу з процедурою GridSearchCV

Використовуючи ці ретельно налаштовані параметри, модель Випадкового Лісу була піддана режиму стратифікованої п'ятикратної перехресної валідації, що завершилося значним підвищенням середньої точності валідації, що просунулася з попереднього значення 0.784 до 0.813.

```
1 of kfold 5
accuracy_score 0.8130081300813008

2 of kfold 5
accuracy_score 0.8373983739837398

3 of kfold 5
accuracy_score 0.7967479674796748

4 of kfold 5
accuracy_score 0.7967479674796748

5 of kfold 5
accuracy_score 0.7950819672131147
```

Рис. 3.52. Результати перевірки середньої точності моделі удосконаленого випадкового лісу з процедурою GridSearchCV

classification_report:

	precision	recall	f1-score	support
0	0.93	0.37	0.53	38
1	0.78	0.99	0.87	84
accuracy			0.80	122
macro avg	0.85	0.68	0.70	122
weighted avg	0.82	0.80	0.76	122

Рис. 3.53. Результати класифікацій удосконаленого випадкового лісу з процедурою GridSearchCV

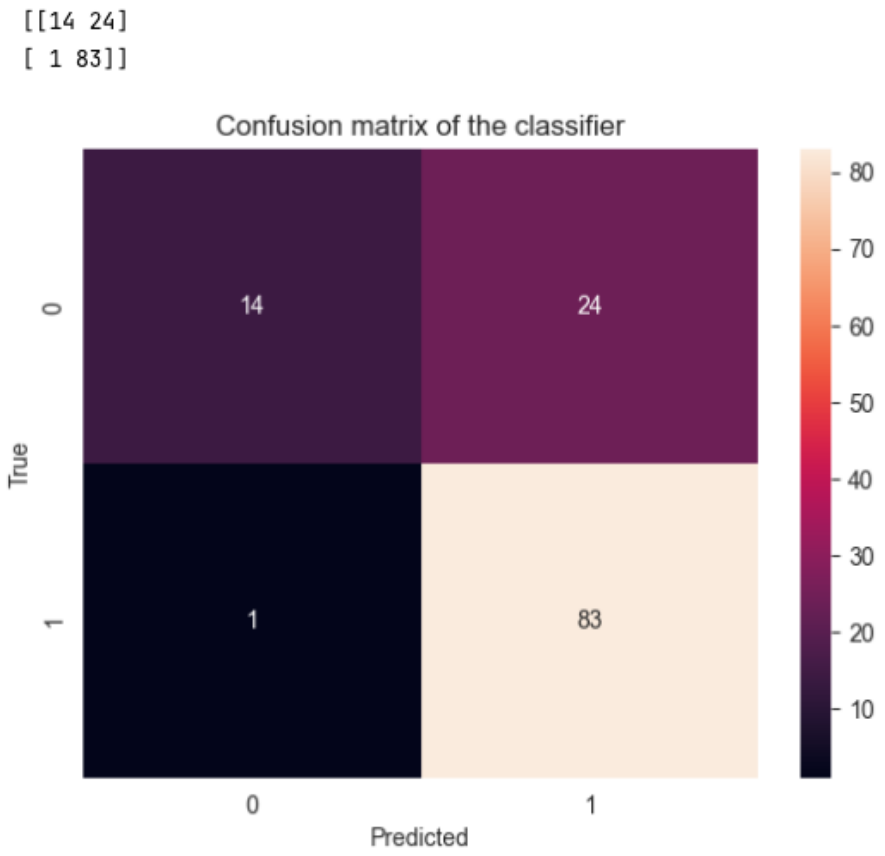


Рис. 3.54. Результати матриці плутанини удосконаленого випадкового лісу з процедурою GridSearchCV

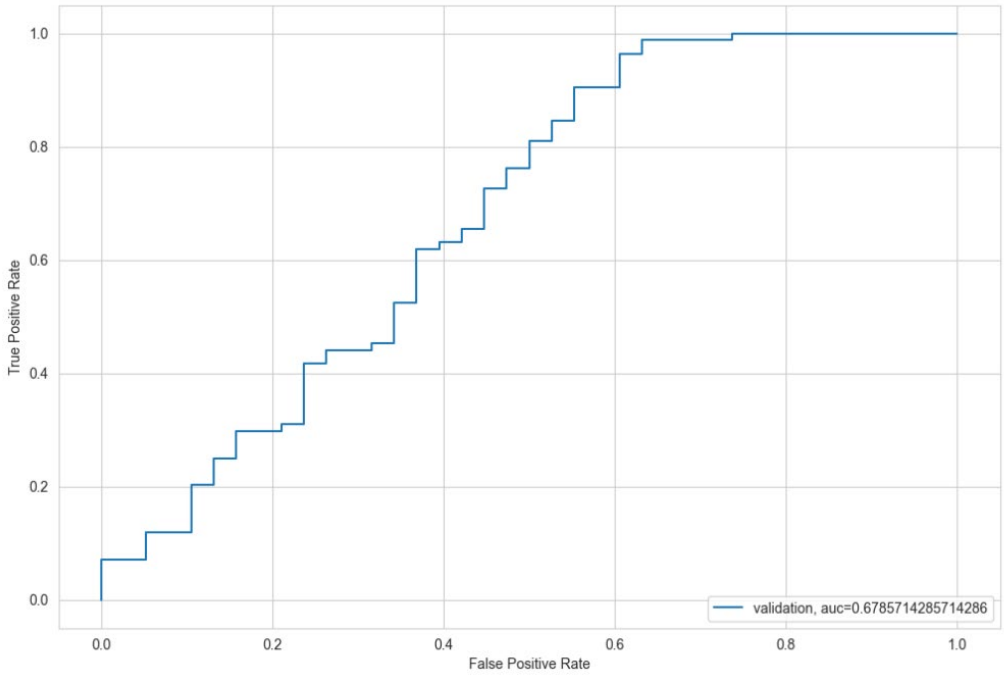


Рис. 3.55. Результати ROC-кривої (AUC) удосконаленого випадкового лісу з процедурою GridSearchCV

По завершенню фази тренування моделі, вдосконалена точність моделі була використана для визначення рішень щодо кредитних заявок, і прогнози, отримані в результаті, були ретельно зібрані у файл .**csv** для подання.

Майстерність вдосконаленої моделі Випадкового Лісу, як це було продемонстровано на лідерборді, проявилася у рівні точності 0.807, що свідчить про значне покращення у порівнянні з початковою базовою моделлю.

Це наукове дослідження окреслює трансформаційний потенціал, який міститься в практиках налаштування гіперпараметрів, особливо через Grid Search, у сфері підвищення передбачуваних здібностей алгоритмів машинного навчання. Дослідження, описане тут, вносить вклад у існуючий науковий корпус, надаючи емпіричну перевірку сили таких методів оптимізації в рамках прикладного машинного навчання.

3.8.8 Важливість функції

В основі методики оцінки методів машинного навчання важливими складовими є характеристики які використовуються у контексті прогностичних моделей. Для отримання результатів потрібно значення характеристик з моделі, та трансформувати у структуровану форму серії даних для подальшої обробки Представленні дані візуалізуються через горизонтальну стовпчасту діаграму.

Зазначена візуалізація є інструментом аналітичної оцінки, що дозволяє визначити релятивну значущість кожної з характеристик з точки зору їх впливу на точність прогнозу моделі, причому вища метрика відображає більшу важливість.

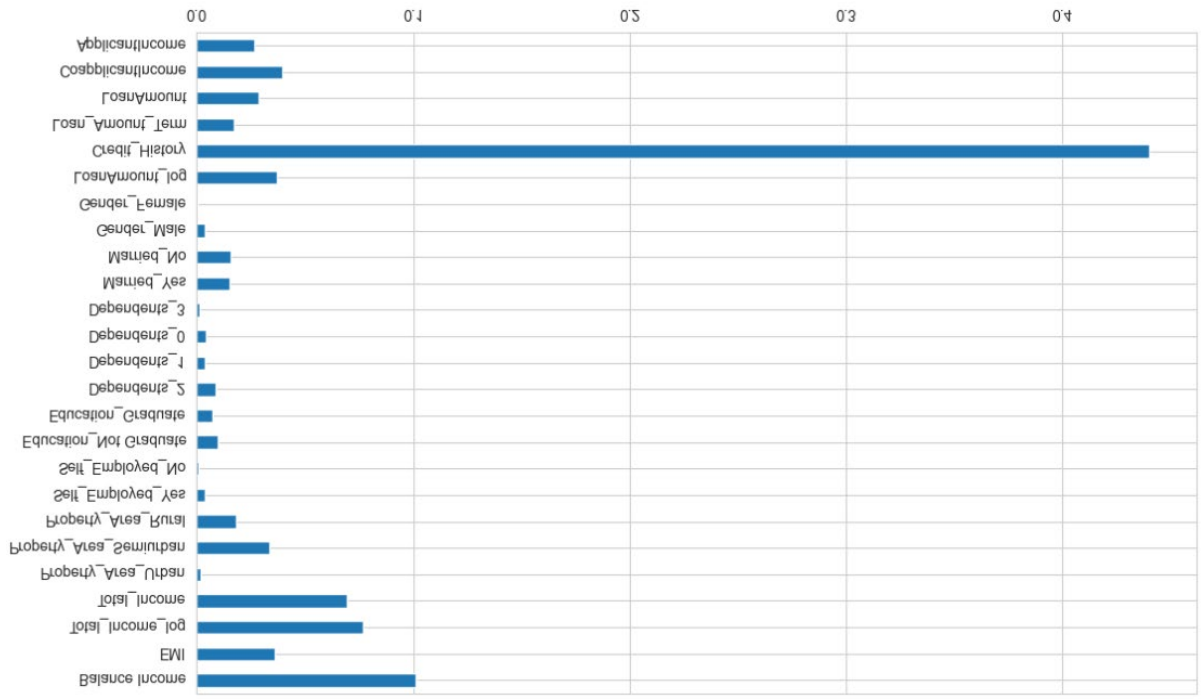


Рис. 3.56. Метрики показників за розміром важливості

Згідно з проведеним аналізом, 'Credit_History' виявляється найбільш вагомою характеристикою, за якою слідують такі показники, як загальний дохід, дохід заявника, дохід співзаявника, територія житла та величина щомісячного платежу. Встановлено, що набір даних характеристик сприяв підвищенню здатності моделі ефективно прогнозувати цільову змінну.

Ця частина дослідження ілюструє важливість відповідного вибору та інтерпретації характеристик у рамках статистичного моделювання та підкреслює значення аналітичної оцінки для підвищення продуктивності прогностичних моделей.

3.8.9 XGBoost (eXtreme Gradient Boosting)

Причиною вибору використання XGBoost те що даний алгоритм зарекомендував себе своєю швидкістю та потужними можливостями, та також здобув визнання в ряді змагань з науки про дані завдяки своїй здатності значно підвищувати ефективність моделей.

Для процесу емпіричного аналізу з XGBoost потребує перетворення категоріальних змінних у числовий формат, щоб забезпечити виконання

оперативних передумов алгоритму. Дослідження було зосереджене на оптимізації двох основних параметрів: **`n_estimator`**, який визначає розмір ансамблю дерев рішень, та **`max_depth`**, який регулює глибину, на яку ці дерева можуть розростатися.

Використовуючи класифікатор XGBoost, модель була піддана методології стратифікованої кратної перехресної валідації, розділяючи набір даних на п'ять підмножин для ретельної оцінки її передбачувальної точності.

```
1 of kfold 5
accuracy_score 0.7723577235772358

2 of kfold 5
accuracy_score 0.7642276422764228

3 of kfold 5
accuracy_score 0.8130081300813008

4 of kfold 5
accuracy_score 0.7723577235772358

5 of kfold 5
accuracy_score 0.7622950819672131
```

Рис. 3.57. Результати перевірки середньої точності моделі XGBoost

classification_report:

	precision	recall	f1-score	support
0	0.68	0.45	0.54	38
1	0.78	0.90	0.84	84
accuracy			0.76	122
macro avg	0.73	0.68	0.69	122
weighted avg	0.75	0.76	0.75	122

Рис. 3.58. Результати класифікацій XGBoost

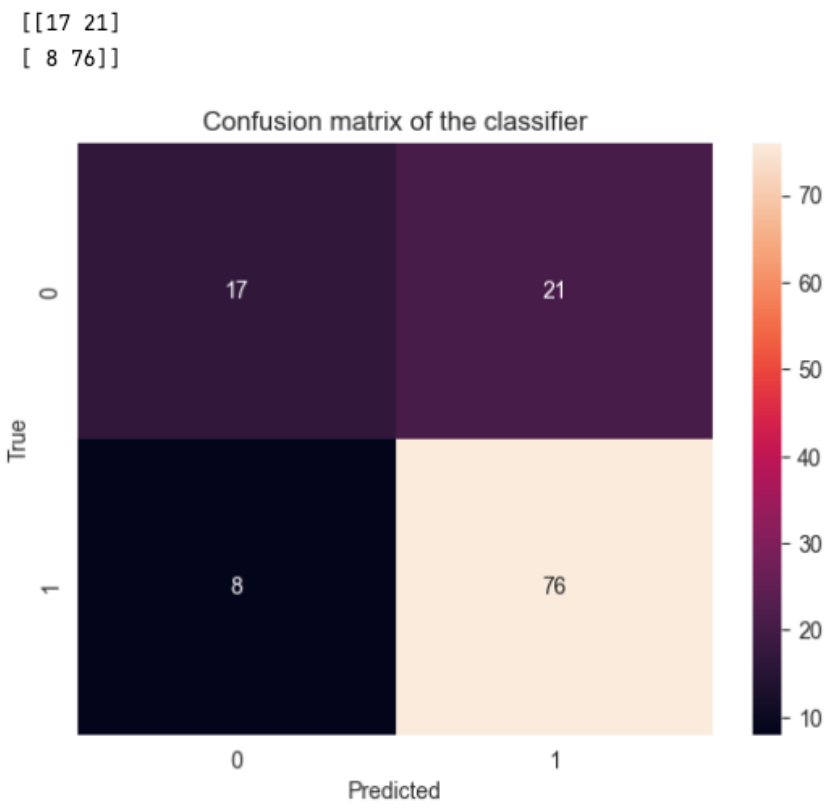


Рис. 3.59. Результати матриці плутанини XGBoost

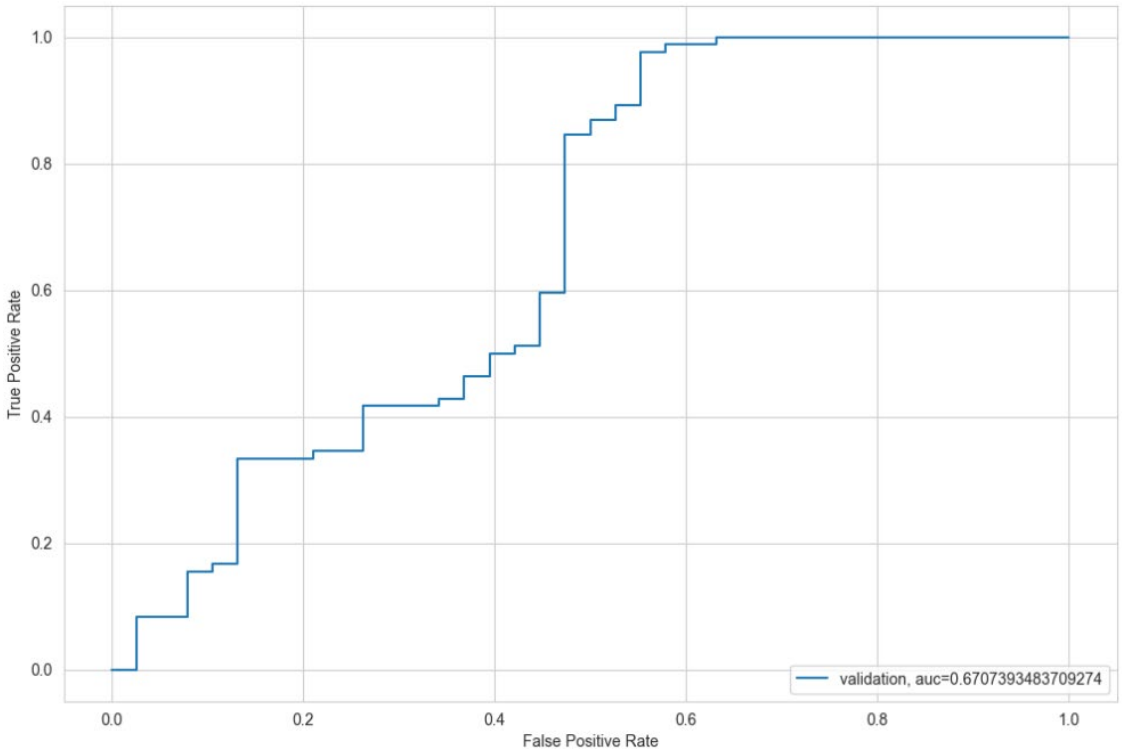


Рис. 3.60. Результати ROC-кривої (AUC) XGBoost

Емпіричні результати після валідації вказували на середній показник точності 0.76. Хоча це представляє похвальний базовий рівень продуктивності, було виявлено потенціал для удосконалення.

З метою досягнення вищих стандартів, дослідження запропоновано провести процес оптимізації гіперпараметрів, використовуючи техніки пошуку по сітці для ретельного калібрування алгоритму. ’

Такий методологічний підхід подібний до точності, необхідної в кулінарному мистецтві, де пошук оптимального поєднання інгредієнтів є найважливішим. Передбачається, що таке тонке налаштування підніме точність моделі на вищий рівень продуктивності, тим самим утілюючи суть оптимізації машинного навчання у сфері аналізу даних. Передбачається що це зміцнить передбачувальну точність, надійність та стійкість моделі.

3.8.10 Підвищення точності XGBoost за допомогою пошуку по сітці (GridSearchCV)

Для оптимізації моделі EXtreme Gradient Boosting (XGBoost) потрібно зробити калібрування гіперпараметрів — критичних детермінант, які модулюють ефективність навчання моделі.

Для сприяння цьому калібруванню було застосовано алгоритм GridSearchCV, знаний своєю систематичною експлорацією різноманітних конфігурацій параметрів разом із перехресною валідацією, щоб виявити конфігурацію, яка оптимізує продуктивність.

Було визначено параметричну сітку, що окреслює спектр для **'max_depth'** та **'n_estimators'** моделі, дозволяючи тим самим вичерпне дослідження потенційних конфігурацій моделі. Метою було виявити оптимальне поєднання параметрів, яке веде до піку точності моделі.

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=1, gamma=0, learning_rate=0.1, max_delta_step=0,
       max_depth=1, min_child_weight=1, missing=None, n_estimators=81,
       n_jobs=1, nthread=None, objective='binary:logistic', random_state=1,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=True, subsample=1)
```

Рис. 3.61. Приклад класифікатора XGBoost з процедурою GridSearchCV

Після поділу набору даних та застосування GridSearchCV було виявлено оптимальні **'max_depth'** та **'n_estimators'** для моделі. Потім модель була піддана стратифікованій п'ятикратній перехресній валідації — вимогливій та надійній методології оцінки стабільності та передбачувальної точності моделі.

Емпіричні результати цього процесу були свідченням сприятливого тренду: зростання середньої точності валідації моделі, що натякає на ефективність обраних гіперпараметрів.

```
1 of kfold 5
accuracy_score 0.8130081300813008

2 of kfold 5
accuracy_score 0.8292682926829268

3 of kfold 5
accuracy_score 0.8048780487804879

4 of kfold 5
accuracy_score 0.7967479674796748

5 of kfold 5
accuracy_score 0.7868852459016393
```

Рис. 3.62. Результати перевірки середньої точності моделі XGBoost з процедурою GridSearchCV

classification_report:

	precision	recall	f1-score	support
0	0.83	0.39	0.54	38
1	0.78	0.96	0.86	84
accuracy			0.79	122
macro avg	0.81	0.68	0.70	122
weighted avg	0.80	0.79	0.76	122

Рис. 3.63. Результати класифікацій удосконаленої моделі XGBoost з процедурою GridSearchCV

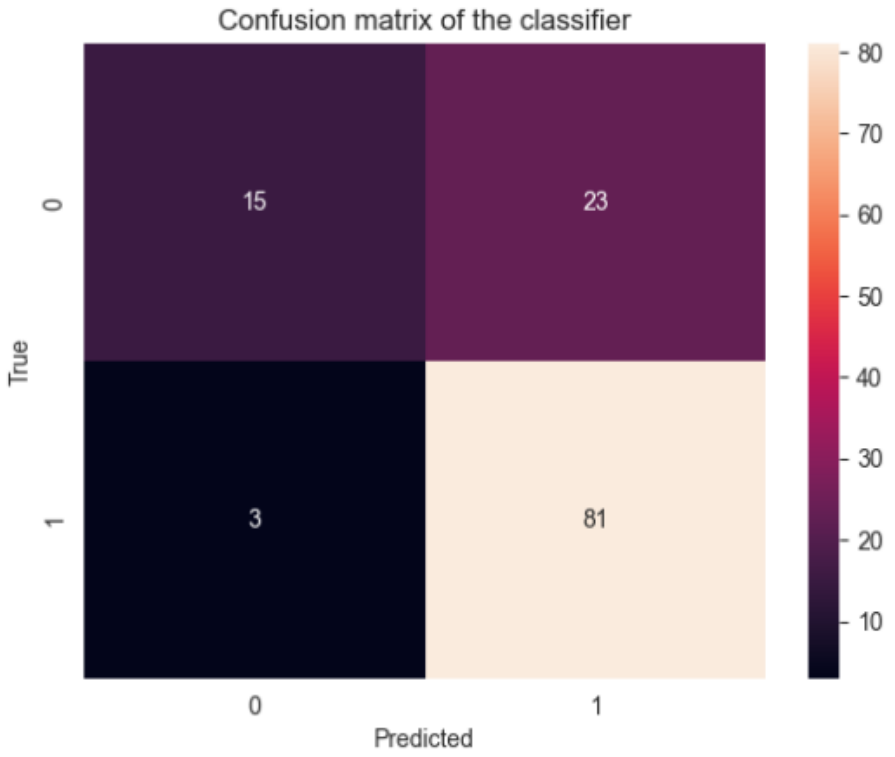


Рис. 3.64. Результати матриці плутанини моделі XGBoost з процедурою GridSearchCV

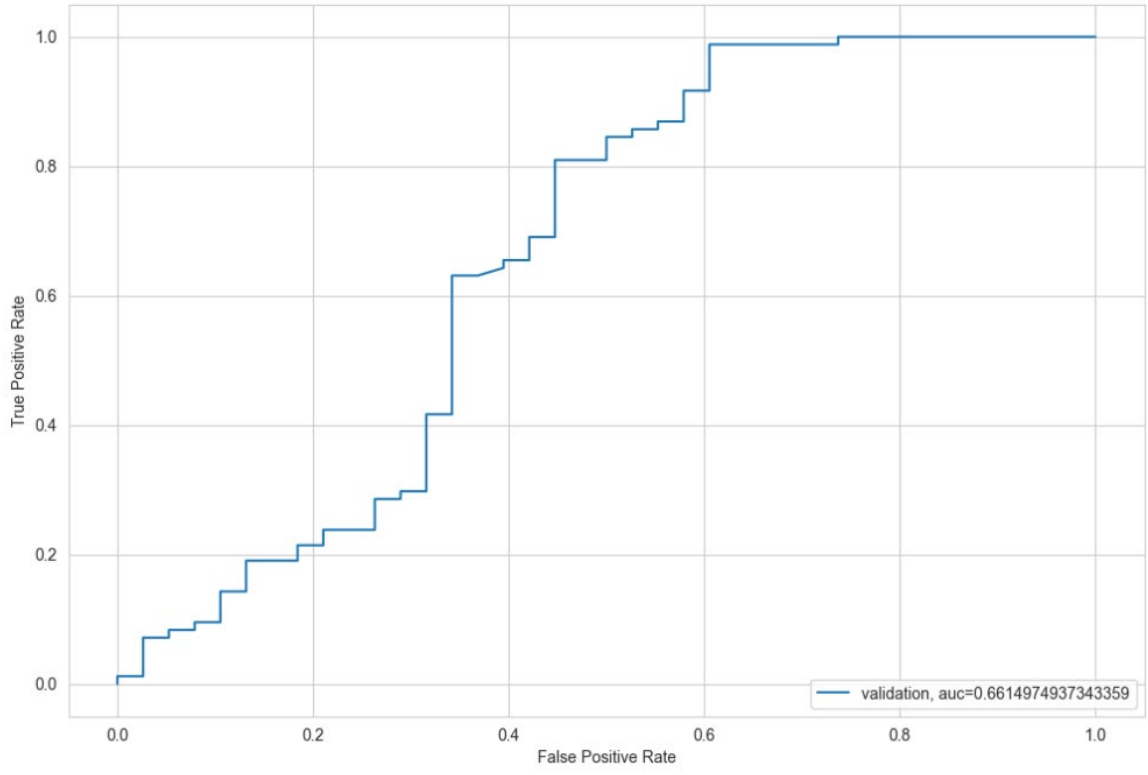


Рис. 3.65. Результати ROC-кривої (AUC) удосконаленої моделі XGBoost з процедурою GridSearchCV

Після цього оптимізовану модель було використано для прогнозування статусу кредитів, із ретельним перетворенням числових прогнозів у їх категорійні еквіваленти. Останнім актом було форматування прогнозів у структурований документ **.csv**, готовий до подання.

Результатом застосування GridSearchCV стало значне збільшення точності що свідчить про користь налаштування гіперпараметрів при застосуванні методів машинного навчання. Завдяки налаштуванням гіперпараметрів, продуктивність моделі XGBoost не тільки була покращена, але й піднялася на рівень точності, який перевищив початкові оцінки.

3.9 Висновки до розділу

Після емпіричної оцінки чотирьох різних алгоритмів, найвищу точність на публічному лідерборді показав алгоритм Логістична регресія із результатом 0.812, за нею слідують Random Forest з результатом 0.807, та XGBoost з результатом 0.806, тоді як алгоритм Дерево ухвалення рішень відстав з оцінкою 0.736.

Незважаючи на те, що нові характеристики, створені завдяки інженерії ознак, сприяли прогнозуванню цільової змінної, вони не істотно підвищили загальну точність моделі. У порівнянні з використанням стандартних значень параметрів, застосування GridSearchCV покращило середню точність валідації моделі, надаючи оптимізовані значення для гіперпараметрів моделі.

Загалом, класифікатор логістичної регресії забезпечує досить непоганий результат з точки зору точності для даного набору даних, не вимагаючи жодної розробки ознак. Завдяки своїй простоті та відносній легкості та швидкості застосування, логістична регресія часто виступає як міцна вихідна точка, з якої аналітики даних можуть вимірювати ефективність більш складних алгоритмів. Однак, у даному випадку, базова логістична регресія не перевершила інші більш складні алгоритми, такі як Random Forest та XGBoost, для цього набору даних.

Таблиця 3.9

Якісні характеристики моделей ухвалення рішень

Назва	Складність	Точність	AUC	GINI	Час роботи, с
Logistic Regression	Початкова	0.802	0.734	0.469	0.993
Logistic Regression	**З додаванням функцій**	**0.812**	**0.701**	**0.538**	**0.663**
Logistic Regression	З додаванням функцій та GridSearchCV	0.811	0.671	0.342	0.744
Decision Tree	З додаванням функцій	0.698	0.698	0.302	0.390
Decision Tree	**З додаванням функцій та GridSearchCV**	**0.736**	**0.692**	**0.384**	**0.334**
Random Forest	З додаванням функцій	0.785	0.668	0.337	0.354
Random Forest	**З додаванням функцій та GridSearchCV**	**0.807**	**0.678**	**0.357**	**0.532**
XGBoost	З додаванням функцій	0.776	0.670	0.341	0.397
XGBoost	**З додаванням функцій та GridSearchCV**	**0.806**	**0.661**	**0.322**	**0.299**

ВИСНОВКИ

У ході дослідження було вивчено систему прийняття рішень в банківській сфері, проаналізовано існуючі методи та підходи, що використовуються банками для створення скорингових моделей з метою оцінювання платоспроможності. Також було проаналізовано найпопулярніші підходи які використовуються при оцінці якості моделей. Встановлено, що класифікація клієнтів у сфері кредитного скорингу може бути ефективно виконана за допомогою моделей, побудованих на основі методів машинного навчання. Використання таких інструментів Python, як NumPy та Pandas, та бібліотеки Scikit-learn дозволило реалізувати ці моделі.

Отримані результати дозволяють зробити висновок, що класифікація позичальників за кредитоспроможністю може бути ефективно здійснена за допомогою моделей машинного навчання. Для порівняння ефективності з метою подальшого удосконалення були розроблені такі моделі, як логістична регресія, дерева ухвалення рішень, випадковий ліс, та XGBoost.

Слід також зазначити що істотність підвищення загальної точності моделі залежить від кількісних та якісних показників характеристик, які створюється завдяки інженерії та якісній обробці даних. Вони значно сприяють прогнозуванню цільової змінної та можуть істотно підвищити загальну точність моделі. У порівнянні з використанням стандартних значень параметрів, застосування GridSearchCV покращило середню точність валідації моделі, надаючи оптимізовані значення для гіперпараметрів моделі.

Причому модель випадкового лісу показала найвищу точність шляхом точної корекції його гіперпараметрів за допомогою процедури GridSearchCV.

Автоматизація процесів скорингу дозволяє банкам знижувати витрати і відводити ресурси на інші завдання, підкреслюючи практичну вигоду від застосування сучасних алгоритмів машинного навчання у вирішенні реальних бізнес-викликів.

Перспективи майбутніх досліджень включають удосконалення архітектури системи, вдосконалення моделей для прогнозування, застосування інших методів і розширення системи новими модулями для оцінювання ризиків та звітності. Модель випадкового лісу продемонструвала чудову точність, відрізняючи "поганих" позичальників з точністю 93% та "хороших" – 81%.

ПЕРЕЛІК ПОСИЛАНЬ

1. Гасті, Т., Тібшірані, Р., & Фрідман, Дж. (2009). "Елементи статистичного навчання: Дата-майнінг, Висновок та Прогнозування". Серія Springer зі Статистики.

2. Андерсон, Р. (2007). "Інструментарій кредитного скорингу: Теорія та Практика управління ризиками роздрібного кредитування та Автоматизації Рішень". Oxford University Press.

3. Бесенс, Б., Ван Гестел, Т., Віане, С., Степанова, М., & Суйкенс, Дж. (2003). "Бенчмаркінг передових алгоритмів класифікації для кредитного скорингу". Журнал товариства оперативних досліджень.

4. Чаудхурі, А.К., & Свайн, Р.К. (2018). "Машинне навчання у управлінні ризиками банківської сфери: Огляд літератури". Управління ризиками.

5. Ценг, М., Ле, Т., Лю, А., Чжан, Г., Хуссейн, Х., Чен, К., Тянь, К., & Лю, З. (2019). "Порівняльне дослідження алгоритмів дерев рішень для прогнозування непогашення банківських позик". Procedia Computer Science.

6. Брейман, Л. (2001). "Випадкові ліси". Машинне навчання.

7. Аданкон, М., & Шеріє, М. (2009). "Метод опорних векторів". У С.З. Лі & А. Джайн (Ред.), Енциклопедія біометрії. Springer, Boston, MA.

8. Jupyter Notebook. [Онлайн]. Доступно: https://jupyter.org (дата звернення 15 грудня 2023 р.).

9. Kaggle. [Онлайн]. Доступно: https://www.kaggle.com/ (дата звернення 8 грудня 2023 р.).

10. Документація TPOT. [Онлайн]. Доступно: http://epistasislab.github.io/tpot/ (дата звернення 10 грудня 2023 р.).

11. Томас, Л.К. (2000). "Огляд кредитного та поведінкового скорингу: прогнозування фінансового ризику позик споживачам". Міжнародний журнал прогнозування.

12. Естевес, П.А., Тесмер, М., Перес, К.А., & Зурада, Дж.М. (2009). "Нормалізована взаємна інформація для відбору ознак". IEEE Transactions on Neural Networks.

13. Мерфі, К.П. (2012). "Машинне навчання: Ймовірнісна перспектива". MIT Press.

14. Гудфеллоу, І., Бенгіо, Й., & Курвіль, А. (2016). "Глибоке навчання". MIT Press.

15. Шапір, Р.Е., & Фройнд, Й. (2012). "Підсилення: Основи та Алгоритми". MIT Press.

16. Джеймс, Г., Віттен, Д., Гасті, Т., & Тібшірані, Р. (2013). "Введення в статистичне навчання: з застосуваннями в R". Springer Texts in Statistics.

17. Крухі, М., Галай, Д., & Марк, Р. (2000). "Управління ризиками". McGraw-Hill Education.

18. Халл, Дж.С. (2018). "Управління ризиками та фінансовими установами". Wiley.

19. ЛеКун, Й., Бенгіо, Й., & Хінтон, Г. (2015). "Глибоке навчання". Nature, 521(7553), 436–444.

20. Хандані, А.Е., Кім, А.Дж., & Ло, А.В. (2010). "Моделі кредитного ризику споживачів через алгоритми машинного навчання". Журнал банківських та фінансових.

21. О'Ніл, К. (2016). "Зброя математичного знищення: Як Великі дані збільшують нерівність та загрожують демократії". Crown.

22. Барокас, С., Хардт, М., & Нараянан, А. (2019). "Справедливість та абстракція в соціотехнічних системах". ACM Конференція з справедливості, відповідальності та прозорості (FAT*).

23. МакКінні, В. (2012). "Python для аналізу даних". O'Reilly Media, Inc.

24. Рашка, С., & Мірджалілі, В. (2017). "Машинне навчання Python". Packt Publishing.

ДОДАТКИ

ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ

```
# import libraries
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

warnings.filterwarnings("ignore")

# load the train and test dataset
train = pd.read_csv("train_dataset.csv")
test = pd.read_csv("test_dataset.csv")

# make a copy of train and test dataset
train_original = train.copy()
test_original = test.copy()

# take a look at the train dataset
train.head()
test.head()

# check the features present in our data
train.shape, test.shape
train.shape[0] / (train.shape[0] + test.shape[0]), test.shape[0] / (train.shape[0] +
test.shape[0])
train.dtypes

# check the distribution of target variable
train.info()

# check the distribution of target variable
train['Loan_Status'].value_counts()
train['Loan_Status'].value_counts(normalize=True)
train['Loan_Status'].value_counts().plot.bar()

# visualizing categorical features
plt.subplot(231)
train['Gender'].value_counts(normalize=True).plot.bar(figsize=(20, 10), title='Gender')
plt.subplot(232)
train['Married'].value_counts(normalize=True).plot.bar(title='Married')
plt.subplot(233)
train['Self_Employed'].value_counts(normalize=True).plot.bar(title='Self_Employed')
plt.subplot(234)
train['Credit_History'].value_counts(normalize=True).plot.bar(title='Credit_History')
plt.subplot(235)
train['Education'].value_counts(normalize=True).plot.bar(title='Education')
plt.show()

# Visualizing remaining categorical features
plt.subplot(121)
train['Dependents'].value_counts(normalize=True).plot.bar(figsize=(12, 4),
title='Dependents')
plt.subplot(122)
train['Property_Area'].value_counts(normalize=True).plot.bar(title='Property_Area')
plt.show()

# Visualizing ApplicantIncome
plt.subplot(121)
sns.distplot(train['ApplicantIncome']);
plt.subplot(122)
```

```
train['ApplicantIncome'].plot.box(figsize=(16, 5))

plt.show()

train.boxplot(column='ApplicantIncome', by='Education')
plt.suptitle("")
plt.subplot(121)
sns.distplot(train['CoapplicantIncome']);
plt.subplot(122)
train['CoapplicantIncome'].plot.box(figsize=(16, 5))
plt.show()

plt.subplot(121)
df = train.dropna()
sns.distplot(df['LoanAmount']);
plt.subplot(122)
train['LoanAmount'].plot.box(figsize=(16, 5))
plt.show()

# Bivariate Analysis
train['Loan_Amount_Term'].value_counts()
train['Loan_Amount_Term'].value_counts(normalize=True).plot.bar(title='Loan_Amount_Term')

print(pd.crosstab(train['Gender'], train['Loan_Status']))

Gender = pd.crosstab(train['Gender'], train['Loan_Status'])
Gender.div(Gender.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True, figsize=(4,
4))

plt.xlabel('Gender')
p = plt.ylabel('Percentage')

print(pd.crosstab(train['Married'], train['Loan_Status']))

Married = pd.crosstab(train['Married'], train['Loan_Status'])
Married.div(Married.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True,
figsize=(4, 4))

plt.xlabel('Married')
p = plt.ylabel('Percentage')

print(pd.crosstab(train['Dependents'], train['Loan_Status']))

Dependents = pd.crosstab(train['Dependents'], train['Loan_Status'])
Dependents.div(Dependents.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True)
plt.xlabel('Dependents')
p = plt.ylabel('Percentage')

print(pd.crosstab(train['Education'], train['Loan_Status']))

Education = pd.crosstab(train['Education'], train['Loan_Status'])
Education.div(Education.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True,
figsize=(4, 4))
plt.xlabel('Education')
p = plt.ylabel('Percentage')

print(pd.crosstab(train['Self_Employed'], train['Loan_Status']))

Self_Employed = pd.crosstab(train['Self_Employed'], train['Loan_Status'])
Self_Employed.div(Self_Employed.sum(1).astype(float), axis=0).plot(kind="bar",
stacked=True, figsize=(4, 4))
plt.xlabel('Self_Employed')
p = plt.ylabel('Percentage')

print(pd.crosstab(train['Credit_History'], train['Loan_Status']))

Credit_History = pd.crosstab(train['Credit_History'], train['Loan_Status'])
Credit_History.div(Credit_History.sum(1).astype(float), axis=0).plot(kind="bar",
stacked=True, figsize=(4, 4))
plt.xlabel('Credit_History')
p = plt.ylabel('Percentage')
```

```
print(pd.crosstab(train['Property_Area'], train['Loan_Status']))

Property_Area = pd.crosstab(train['Property_Area'], train['Loan_Status'])
Property_Area.div(Property_Area.sum(1).astype(float), axis=0).plot(kind="bar",
stacked=True)
plt.xlabel('Property_Area')
P = plt.ylabel('Percentage')

print(train.groupby('Loan_Status')['ApplicantIncome'].mean())

# ApplicantIncome
train.groupby('Loan_Status')['ApplicantIncome'].mean().plot.bar()
bins = [0, 2500, 4000, 6000, 81000]
group = ['Low', 'Average', 'High', 'Very high']
train['Income_bin'] = pd.cut(df['ApplicantIncome'], bins, labels=group)
train.head(8)

print(pd.crosstab(train['Income_bin'], train['Loan_Status']))

Income_bin = pd.crosstab(train['Income_bin'], train['Loan_Status'])
Income_bin.div(Income_bin.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True)
plt.xlabel('ApplicantIncome')
P = plt.ylabel('Percentage')
# CoapplicantIncome
bins = [0, 1000, 3000, 42000]
group = ['Low', 'Average', 'High']
train['Coapplicant_Income_bin'] = pd.cut(df['CoapplicantIncome'], bins, labels=group)

Coapplicant_Income_bin = pd.crosstab(train['Coapplicant_Income_bin'], train['Loan_Status'])
Coapplicant_Income_bin.div(Coapplicant_Income_bin.sum(1).astype(float),
axis=0).plot(kind="bar", stacked=True)
plt.xlabel('CoapplicantIncome')
P = plt.ylabel('Percentage')

print(len(train[train["CoapplicantIncome"] == 0]))
"Percentage of CoapplicantIncome = 0 is:", len(train[train["CoapplicantIncome"] == 0]) /
len(train["CoapplicantIncome"])

# Total Income
train['Total_Income'] = train['ApplicantIncome'] + train['CoapplicantIncome']
bins = [0, 2500, 4000, 6000, 81000]
group = ['Low', 'Average', 'High', 'Very high']
train['Total_Income_bin'] = pd.cut(train['Total_Income'], bins, labels=group)

Total_Income_bin = pd.crosstab(train['Total_Income_bin'], train['Loan_Status'])
Total_Income_bin.div(Total_Income_bin.sum(1).astype(float), axis=0).plot(kind="bar",
stacked=True)

plt.xlabel('Total_Income')
P = plt.ylabel('Percentage')
bins = [0, 100, 200, 700]
group = ['Low', 'Average', 'High']
train['LoanAmount_bin'] = pd.cut(df['LoanAmount'], bins, labels=group)

LoanAmount_bin = pd.crosstab(train['LoanAmount_bin'], train['Loan_Status'])
LoanAmount_bin.div(LoanAmount_bin.sum(1).astype(float), axis=0).plot(kind="bar",
stacked=True)
plt.xlabel('LoanAmount')
P = plt.ylabel('Percentage')
train.head()
train = train.drop(['Income_bin', 'Coapplicant_Income_bin', 'LoanAmount_bin',
'Total_Income_bin', 'Total_Income'], axis=1)
train.head()
train['Dependents'].replace('3+', 3, inplace=True)
test['Dependents'].replace('3+', 3, inplace=True)
train['Loan_Status'].replace('N', 0, inplace=True)
train['Loan_Status'].replace('Y', 1, inplace=True)
train.head()

# find the correlation between all the numerical variables
```

```
matrix = train.corr()
f, ax = plt.subplots(figsize=(9, 6))
sns.heatmap(matrix, vmax=1, square=True, cmap="BuPu", annot=True)
matrix

# Missing value imputation
train.isnull().sum()
train['Gender'].fillna(train['Gender'].mode()[0], inplace=True)
train['Married'].fillna(train['Married'].mode()[0], inplace=True)
train['Dependents'].fillna(train['Dependents'].mode()[0], inplace=True)
train['Self_Employed'].fillna(train['Self_Employed'].mode()[0], inplace=True)
train['Credit_History'].fillna(train['Credit_History'].mode()[0], inplace=True)
train['Loan_Amount_Term'].value_counts()
train['Loan_Amount_Term'].fillna(train['Loan_Amount_Term'].mode()[0], inplace=True)
train['LoanAmount'].fillna(train['LoanAmount'].median(), inplace=True)
train.isnull().sum()

test['Gender'].fillna(train['Gender'].mode()[0], inplace=True)
test['Dependents'].fillna(train['Dependents'].mode()[0], inplace=True)
test['Self_Employed'].fillna(train['Self_Employed'].mode()[0], inplace=True)
test['Credit_History'].fillna(train['Credit_History'].mode()[0], inplace=True)
test['Loan_Amount_Term'].fillna(train['Loan_Amount_Term'].mode()[0], inplace=True)
test['LoanAmount'].fillna(train['LoanAmount'].median(), inplace=True)
test.isnull().sum()

# Outlier Treatment
ax1 = plt.subplot(121)
train['LoanAmount'].hist(bins=20, figsize=(12, 4))
ax1.set_title("Train")

ax2 = plt.subplot(122)
test['LoanAmount'].hist(bins=20)
ax2.set_title("Test")

# log transformation
train['LoanAmount_log'] = np.log(train['LoanAmount'])
test['LoanAmount_log'] = np.log(test['LoanAmount'])

# after log transformation
ax1 = plt.subplot(121)
train['LoanAmount_log'].hist(bins=20, figsize=(12, 4))
ax1.set_title("Train")

ax2 = plt.subplot(122)
test['LoanAmount_log'].hist(bins=20)
ax2.set_title("Test")

# drop Loan_ID
train = train.drop('Loan_ID', axis=1)
test = test.drop('Loan_ID', axis=1)

X = train.drop('Loan_Status', 1)
y = train.Loan_Status

# make dummy variables for categorical variables
X = pd.get_dummies(X)
train = pd.get_dummies(train)
test = pd.get_dummies(test)
X.shape, train.shape, test.shape
X.head()

# import libraries
from sklearn.model_selection import train_test_split

# split the data
x_train, x_cv, y_train, y_cv = train_test_split(X, y, test_size=0.3, random_state=0)
x_train.shape, x_cv.shape, y_train.shape, y_cv.shape

# import libraries
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

```
# instantiate the model
model = LogisticRegression()
model.fit(x_train, y_train)

# make prediction on cv
pred_cv = model.predict(x_cv)
accuracy_score(y_cv, pred_cv)

# import confusion_matrix
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_cv, pred_cv)
print(cm)

# visualize confusion matrix using heatmap
sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion matrix of the classifier')
plt.xlabel('Predicted')
plt.ylabel('True')

# import classification_report
from sklearn.metrics import classification_report
print(classification_report(y_cv, pred_cv))

# make prediction on test set
pred_test = model.predict(test)
pred_test[:50]

# save the solution in a csv file
submission = pd.read_csv("sample_submission.csv")
submission['Loan_Status'] = pred_test
submission['Loan_ID'] = test_original['Loan_ID']
submission['Loan_Status'].replace(0, 'N', inplace=True)
submission['Loan_Status'].replace(1, 'Y', inplace=True)
submission.head()
submission.to_csv('logistic.csv', index=False)

## LogisticRegression using stratified k-folds cross validation

# import libraries
from sklearn.model_selection import StratifiedKFold

mean_accuracy = []
i = 1
kf = StratifiedKFold(n_splits=5, random_state=1, shuffle=True)

for train_index, test_index in kf.split(X, y):
    print('\n{} of kfold {}'.format(i, kf.n_splits))
    xtr, xvl = X.loc[train_index], X.loc[test_index]
    ytr, yvl = y[train_index], y[test_index]

    model = LogisticRegression(random_state=1)
    model.fit(xtr, ytr)
    pred_test = model.predict(xvl)
    score = accuracy_score(yvl, pred_test)
    mean_accuracy.append(score)
    print('accuracy_score', score)
    i += 1

# import classification_report
from sklearn.metrics import classification_report

print('\nclassification_report: \n\n', classification_report(yvl, pred_test))

# import confusion_matrix
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(yvl, pred_test)
print('\nConfusion matrix of the classifier: \n\n', cm)
```

```
sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion matrix of the classifier')
plt.xlabel('Predicted')
plt.ylabel('True')

# make prediction on test set
pred_test = model.predict(test)
# calculate probability estimates of loan approval
pred = model.predict_proba(xvl)[:, 1]
# visualize ROC curve
from sklearn import metrics

fpr, tpr, _ = metrics.roc_curve(yvl, pred)
auc = metrics.roc_auc_score(yvl, pred)
gini = 2 * auc - 1
plt.figure(figsize=(12, 8))
plt.plot(fpr, tpr, label="validation, auc=" + str(auc))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.show()

print("\nGini coefficient: ", gini)
print("\nMean validation accuracy: ", sum(mean_accuracy) / len(mean_accuracy))
print("\nAUC score: ", auc)

# filling Loan_Status with predictions
submission['Loan_Status'] = pred_test
submission['Loan_ID'] = test_original['Loan_ID']
submission['Loan_Status'].replace(0, 'N', inplace=True)
submission['Loan_Status'].replace(1, 'Y', inplace=True)
submission.to_csv('logistic.csv', index=False)

# train a decision tree model
train['Total_Income'] = train['ApplicantIncome'] + train['CoapplicantIncome']
test['Total_Income'] = test['ApplicantIncome'] + test['CoapplicantIncome']

fig = plt.figure(figsize=(14, 4))
ax1 = plt.subplot(121)
sns.distplot(train['Total_Income'])
ax1.set_title("Train")

ax1 = plt.subplot(122)
sns.distplot(test['Total_Income'])
ax1.set_title("Test")

# before log transformation
train['Total_Income_log'] = np.log(train['Total_Income'])
test['Total_Income_log'] = np.log(test['Total_Income'])

# after log transformation
fig = plt.figure(figsize=(14, 4))
ax1 = plt.subplot(121)
sns.distplot(train['Total_Income_log'])
ax1.set_title("Train")

ax1 = plt.subplot(122)
sns.distplot(test['Total_Income_log'])
ax1.set_title("Test")

# create new "EMI" variable
train['EMI'] = train['LoanAmount'] / train['Loan_Amount_Term']
test['EMI'] = test['LoanAmount'] / test['Loan_Amount_Term']

fig = plt.figure(figsize=(14, 4))
ax1 = plt.subplot(121)
sns.distplot(train['EMI'])
ax1.set_title("Train")

ax1 = plt.subplot(122)
sns.distplot(test['EMI'])
```

```
ax1.set_title("Test")

# create new "Balance Income" variable
train['Balance Income'] = train['Total_Income'] - (train['EMI'] * 1000)
test['Balance Income'] = test['Total_Income'] - (test['EMI'] * 1000)

fig = plt.figure(figsize=(14, 4))
ax1 = plt.subplot(121)
sns.distplot(train['Balance Income'])
ax1.set_title("Train")

ax1 = plt.subplot(122)
sns.distplot(test['Balance Income'])
ax1.set_title("Test")
train.head()

# Logisitic Regression

# drop the variables which we used to create these new features
X = train.drop('Loan_Status', axis=1)
# saving target variable in separate dataset
y = train.Loan_Status

# make dummy variables for categorical variables
mean_accuracy = []
i = 1
kf = StratifiedKFold(n_splits=5, random_state=1, shuffle=True)

for train_index, test_index in kf.split(X, y):
    print('\n{} of kfold {}'.format(i, kf.n_splits))
    xtr, xvl = X.loc[train_index], X.loc[test_index]
    ytr, yvl = y[train_index], y[test_index]

    model = LogisticRegression(random_state=1)
    model.fit(xtr, ytr)
    pred_test = model.predict(xvl)
    score = accuracy_score(yvl, pred_test)
    mean_accuracy.append(score)
    print('accuracy_score', score)
    i += 1

# import classification_report
from sklearn.metrics import classification_report

print('\nclassification_report: \n\n', classification_report(yvl, pred_test))

# import confusion_matrix
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(yvl, pred_test)
print('\nConfusion matrix of the classifier: \n\n', cm)

sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion matrix of the classifier')
plt.xlabel('Predicted')
plt.ylabel('True')

# make prediction on test set
pred_test = model.predict(test)
pred = model.predict_proba(xvl)[:, 1]

# visualize ROC curve
from sklearn import metrics

fpr, tpr, _ = metrics.roc_curve(yvl, pred)
auc = metrics.roc_auc_score(yvl, pred)
gini = 2 * auc - 1
plt.figure(figsize=(12, 8))
plt.plot(fpr, tpr, label="validation, auc=" + str(auc))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

```
plt.legend(loc=4)
plt.show()

print("\nGini coefficient: ", gini)
print("\nMean validation accuracy: ", sum(mean_accuracy) / len(mean_accuracy))
print("\nAUC score: ", auc)

## Adding GridSearchCV

# import library
from sklearn.model_selection import GridSearchCV

# Provide range for max_depth from 1 to 20 with an interval of 2 and from 1 to 200 with an
interval of 20 for n_estimators

paramgrid = {
    'C': [1.0],
    'class_weight': [None],
    'dual': [False],
    'fit_intercept': [True],
    'intercept_scaling': [1],
    'max_iter': [100],
    'multi_class': ['ovr'],
    'n_jobs': [1],
    'penalty': ['l2'],
    'random_state': [None],
    'solver': ['liblinear'],
    'tol': [0.0001],
    'verbose': [0],
    'warm_start': [False]
}

grid_search = GridSearchCV(LogisticRegression(random_state=1), paramgrid)

from sklearn.model_selection import train_test_split

x_train, x_cv, y_train, y_cv = train_test_split(X, y, test_size=0.3, random_state=1)
grid_search.fit(x_train, y_train)
grid_search.best_estimator_


mean_accuracy = []
i = 1
kf = StratifiedKFold(n_splits=5, random_state=1, shuffle=True)

for train_index, test_index in kf.split(X, y):
    print('\n{} of kfold {}'.format(i, kf.n_splits))
    xtr, xvl = X.loc[train_index], X.loc[test_index]
    ytr, yvl = y[train_index], y[test_index]

    model = LogisticRegression(random_state=1)
    model.fit(xtr, ytr)
    pred_test = model.predict(xvl)
    score = accuracy_score(yvl, pred_test)
    mean_accuracy.append(score)
    print('accuracy_score', score)
    i += 1

# import classification_report
from sklearn.metrics import classification_report

print('\nclassification_report: \n\n', classification_report(yvl, pred_test))

# import confusion_matrix
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(yvl, pred_test)
print('\nConfusion matrix of the classifier: \n\n', cm)

sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion matrix of the classifier')
```

```
plt.xlabel('Predicted')
plt.ylabel('True')

# make prediction on test set
pred_test = model.predict(test)
pred_lg_gs_cv = model.predict_proba(xvl)[:, 1]

# visualize ROC curve
from sklearn import metrics

fpr, tpr, _ = metrics.roc_curve(yvl, pred_lg_gs_cv)
auc = metrics.roc_auc_score(yvl, pred_lg_gs_cv)
gini = 2 * auc - 1
plt.figure(figsize=(12, 8))
plt.plot(fpr, tpr, label="validation, auc=" + str(auc))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.show()

print("\nGini coefficient: ", gini)
print("\nMean validation accuracy: ", sum(mean_accuracy) / len(mean_accuracy))
print("\nAUC score: ", auc)

# filling Loan_Status with predictions
submission['Loan_Status'] = pred_test
submission['Loan_ID'] = test_original['Loan_ID']
submission['Loan_Status'].replace(0, 'N', inplace=True)
submission['Loan_Status'].replace(1, 'Y', inplace=True)
submission.to_csv('Log2.csv', index=False)

# import library
from sklearn import tree

mean_accuracy = []
i = 1
kf = StratifiedKFold(n_splits=5, random_state=1, shuffle=True)
for train_index, test_index in kf.split(X, y):
    print('\n{} of kfold {}'.format(i, kf.n_splits))
    xtr, xvl = X.loc[train_index], X.loc[test_index]
    ytr, yvl = y[train_index], y[test_index]

    model = tree.DecisionTreeClassifier(random_state=1)
    model.fit(xtr, ytr)
    pred_test = model.predict(xvl)
    score = accuracy_score(yvl, pred_test)
    mean_accuracy.append(score)
    print('accuracy_score', score)
    i += 1

# import classification_report
from sklearn.metrics import classification_report

print('\nclassification_report: \n\n', classification_report(yvl, pred_test))

# import confusion_matrix
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(yvl, pred_test)
print('\nConfusion matrix of the classifier: \n\n', cm)

sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion matrix of the classifier')
plt.xlabel('Predicted')
plt.ylabel('True')

pred_test = model.predict(test)
pred_dt = model.predict_proba(xvl)[:, 1]

# visualize ROC curve
from sklearn import metrics
```

```
fpr, tpr, _ = metrics.roc_curve(yvl, pred_dt)
auc = metrics.roc_auc_score(yvl, pred_dt)
gini = 2 * auc - 1
plt.figure(figsize=(12, 8))
plt.plot(fpr, tpr, label="validation, auc=" + str(auc))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.show()

print("\nGini coefficient: ", gini)
print("\nMean validation accuracy: ", sum(mean_accuracy) / len(mean_accuracy))
print("\nAUC score: ", auc)

# filling Loan_Status with predictions
submission['Loan_Status'] = pred_test
submission['Loan_ID'] = test_original['Loan_ID']
submission['Loan_Status'].replace(0, 'N', inplace=True)
submission['Loan_Status'].replace(1, 'Y', inplace=True)
submission.to_csv('Decision Tree.csv', index=False)

# import library
from sklearn.model_selection import GridSearchCV

# Provide range for max_depth from 1 to 20 with an interval of 2
paramgrid = {'max_depth': list(range(1, 20, 2))}
grid_search = GridSearchCV(tree.DecisionTreeClassifier(random_state=1), paramgrid)

from sklearn.model_selection import train_test_split

x_train, x_cv, y_train, y_cv = train_test_split(X, y, test_size=0.3, random_state=1)
grid_search.fit(x_train, y_train)
grid_search.best_estimator_

mean_accuracy = []
i = 1
kf = StratifiedKFold(n_splits=5, random_state=1, shuffle=True)
for train_index, test_index in kf.split(X, y):
    print('\n{} of kfold {}'.format(i, kf.n_splits))
    xtr, xvl = X.loc[train_index], X.loc[test_index]
    ytr, yvl = y[train_index], y[test_index]

    model = tree.DecisionTreeClassifier(random_state=1)
    model.fit(xtr, ytr)
    pred_test = model.predict(xvl)
    score = accuracy_score(yvl, pred_test)
    mean_accuracy.append(score)
    print('accuracy_score', score)
    i += 1

# import classification_report
from sklearn.metrics import classification_report

print('\nclassification_report: \n\n', classification_report(yvl, pred_test))

# import confusion_matrix
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(yvl, pred_test)
print('\nConfusion matrix of the classifier: \n\n', cm)

sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion matrix of the classifier')
plt.xlabel('Predicted')
plt.ylabel('True')

pred_test = model.predict(test)
pred_dt_gs_cv = model.predict_proba(xvl)[:, 1]

# visualize ROC curve
```

```
from sklearn import metrics

fpr, tpr, _ = metrics.roc_curve(yvl, pred_dt_gs_cv)
auc = metrics.roc_auc_score(yvl, pred_dt_gs_cv)
gini = 2 * auc - 1
plt.figure(figsize=(12, 8))
plt.plot(fpr, tpr, label="validation, auc=" + str(auc))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.show()

print("\nGini coefficient: ", gini)
print("\nMean validation accuracy: ", sum(mean_accuracy) / len(mean_accuracy))
print("\nAUC score: ", auc)

# import library
from sklearn.ensemble import RandomForestClassifier

mean_accuracy = []
i = 1
kf = StratifiedKFold(n_splits=5, random_state=1, shuffle=True)
for train_index, test_index in kf.split(X, y):
    print('\n{} of kfold {}'.format(i, kf.n_splits))
    xtr, xvl = X.loc[train_index], X.loc[test_index]
    ytr, yvl = y[train_index], y[test_index]

    model = RandomForestClassifier(random_state=1, max_depth=10, n_estimators=10)
    model.fit(xtr, ytr)
    pred_test = model.predict(xvl)
    score = accuracy_score(yvl, pred_test)
    mean_accuracy.append(score)
    print('accuracy_score', score)
    i += 1

# import classification_report
from sklearn.metrics import classification_report

print('\nclassification_report: \n\n', classification_report(yvl, pred_test))

# import confusion_matrix
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(yvl, pred_test)
print('\nConfusion matrix of the classifier: \n\n', cm)

sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion matrix of the classifier')
plt.xlabel('Predicted')
plt.ylabel('True')

pred_test = model.predict(test)
pred_rf = model.predict_proba(xvl)[:, 1]

# visualize ROC curve
from sklearn import metrics

fpr, tpr, _ = metrics.roc_curve(yvl, pred_rf)
auc = metrics.roc_auc_score(yvl, pred_rf)
gini = 2 * auc - 1
plt.figure(figsize=(12, 8))
plt.plot(fpr, tpr, label="validation, auc=" + str(auc))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.show()

print("\nGini coefficient: ", gini)
print("\nMean validation accuracy: ", sum(mean_accuracy) / len(mean_accuracy))
print("\nAUC score: ", auc)
```

```
# import library
from sklearn.model_selection import GridSearchCV
# Provide range for max_depth from 1 to 20 with an interval of 2 and from 1 to 200 with an
interval of 20 for n_estimators
paramgrid = {'max_depth': list(range(1, 20, 2)), 'n_estimators': list(range(1, 200, 20))}

grid_search = GridSearchCV(RandomForestClassifier(random_state=1), paramgrid)
from sklearn.model_selection import train_test_split

x_train, x_cv, y_train, y_cv = train_test_split(X, y, test_size=0.3, random_state=1)
grid_search.fit(x_train, y_train)
grid_search.best_estimator_

mean_accuracy = []
i = 1
kf = StratifiedKFold(n_splits=5, random_state=1, shuffle=True)
for train_index, test_index in kf.split(X, y):
    print('\n{} of kfold {}'.format(i, kf.n_splits))
    xtr, xvl = X.loc[train_index], X.loc[test_index]
    ytr, yvl = y[train_index], y[test_index]

    model = RandomForestClassifier(random_state=1, max_depth=3, n_estimators=141)
    model.fit(xtr, ytr)
    pred_test = model.predict(xvl)
    score = accuracy_score(yvl, pred_test)
    mean_accuracy.append(score)
    print('accuracy_score', score)
    i += 1

# import classification_report
from sklearn.metrics import classification_report

print('\nclassification_report: \n\n', classification_report(yvl, pred_test))

# import confusion_matrix
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(yvl, pred_test)
print('\nConfusion matrix of the classifier: \n\n', cm)

sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion matrix of the classifier')
plt.xlabel('Predicted')
plt.ylabel('True')

pred_test = model.predict(test)
pred2 = model.predict_proba(xvl)[:, 1]

# visualize ROC curve
from sklearn import metrics

fpr, tpr, _ = metrics.roc_curve(yvl, pred2)
auc = metrics.roc_auc_score(yvl, pred2)
gini = 2 * auc - 1
plt.figure(figsize=(12, 8))
plt.plot(fpr, tpr, label="validation, auc=" + str(auc))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.show()

print("\nGini coefficient: ", gini)
print("\nMean validation accuracy: ", sum(mean_accuracy) / len(mean_accuracy))
print("\nAUC score: ", auc)

# filling Loan_Status with predictions
submission['Loan_Status'] = pred_test
submission['Loan_ID'] = test_original['Loan_ID']
submission['Loan_Status'].replace(0, 'N', inplace=True)
submission['Loan_Status'].replace(1, 'Y', inplace=True)
submission.to_csv('Random Forest.csv', index=False)
```

```
importances = pd.Series(model.feature_importances_, index=X.columns)
importances.plot(kind='barh', figsize=(12, 8))

# import library
from xgboost import XGBClassifier

mean_accuracy = []
i = 1
kf = StratifiedKFold(n_splits=5, random_state=1, shuffle=True)
for train_index, test_index in kf.split(X, y):
    print('\n{} of kfold {}'.format(i, kf.n_splits))
    xtr, xvl = X.loc[train_index], X.loc[test_index]
    ytr, yvl = y[train_index], y[test_index]

    model = XGBClassifier(random_state=1, n_estimators=50, max_depth=4)
    model.fit(xtr, ytr)
    pred_test = model.predict(xvl)
    score = accuracy_score(yvl, pred_test)
    mean_accuracy.append(score)
    print('accuracy_score', score)
    i += 1

# import classification_report
from sklearn.metrics import classification_report

print('\nclassification_report: \n\n', classification_report(yvl, pred_test))

# import confusion_matrix
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(yvl, pred_test)
print('\nConfusion matrix of the classifier: \n\n', cm)

sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion matrix of the classifier')
plt.xlabel('Predicted')
plt.ylabel('True')

pred_test = model.predict(test)
pred_xb = model.predict_proba(xvl)[:, 1]

# visualize ROC curve
from sklearn import metrics

fpr, tpr, _ = metrics.roc_curve(yvl, pred_xb)
auc = metrics.roc_auc_score(yvl, pred_xb)
gini = 2 * auc - 1
plt.figure(figsize=(12, 8))
plt.plot(fpr, tpr, label="validation, auc=" + str(auc))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.show()

print("\nGini coefficient: ", gini)
print("\nMean validation accuracy: ", sum(mean_accuracy) / len(mean_accuracy))
print("\nAUC score: ", auc)

# import library
from sklearn.model_selection import GridSearchCV

# Provide range for max_depth from 1 to 20 with an interval of 2 and from 1 to 200 with an
interval of 20 for n_estimators
paramgrid = {'max_depth': list(range(1, 20, 2)), 'n_estimators': list(range(1, 200, 20))}
# default 3-fold cross validation, cv=3
grid_search = GridSearchCV(XGBClassifier(random_state=1), paramgrid)
# split the data
from sklearn.model_selection import train_test_split

x_train, x_cv, y_train, y_cv = train_test_split(X, y, test_size=0.3, random_state=1)
```

```
grid_search.fit(x_train, y_train)

mean_accuracy = []
i = 1
kf = StratifiedKFold(n_splits=5, random_state=1, shuffle=True)
for train_index, test_index in kf.split(X, y):
    print('\n{} of kfold {}'.format(i, kf.n_splits))
    xtr, xvl = X.loc[train_index], X.loc[test_index]
    ytr, yvl = y[train_index], y[test_index]

    model = XGBClassifier(random_state=1, n_estimators=81, max_depth=1)
    model.fit(xtr, ytr)
    pred_test = model.predict(xvl)
    score = accuracy_score(yvl, pred_test)
    mean_accuracy.append(score)
    print('accuracy_score', score)
    i += 1

# import classification_report
from sklearn.metrics import classification_report

print('\nclassification_report: \n\n', classification_report(yvl, pred_test))

# import confusion_matrix
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(yvl, pred_test)
print('\nConfusion matrix of the classifier: \n\n', cm)

sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion matrix of the classifier')
plt.xlabel('Predicted')
plt.ylabel('True')

pred_test = model.predict(test)
pred_xb_gs_cv = model.predict_proba(xvl)[:, 1]

# visualize ROC curve
from sklearn import metrics

fpr, tpr, _ = metrics.roc_curve(yvl, pred_xb_gs_cv)
auc = metrics.roc_auc_score(yvl, pred_xb_gs_cv)
gini = 2 * auc - 1
plt.figure(figsize=(12, 8))
plt.plot(fpr, tpr, label="validation, auc=" + str(auc))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.show()

print("\nGini coefficient: ", gini)
print("\nMean validation accuracy: ", sum(mean_accuracy) / len(mean_accuracy))
print("\nAUC score: ", auc)

# filling Loan_Status with predictions
submission['Loan_Status'] = pred_test
submission['Loan_ID'] = test_original['Loan_ID']
submission['Loan_Status'].replace(0, 'N', inplace=True)
submission['Loan_Status'].replace(1, 'Y', inplace=True)
submission.to_csv('XGBoost.csv', index=False)
```

ДОДАТОК Б. НАВЧАЛЬНИЙ НАБІР ЗМІННИХ

НАВЧАЛЬНИЙ НАБІР ЗМІННИХ ДЛЯ НАВЧАННЯ МОДЕЛІ

	Loan_ID	Стать	Одружений	Утриманці	Освіта	Самозайнятий	Дохід заявника	Дохід співзаявника	Розмір позики	Сума_позики_термін	Кредитна_історія
2	LP001002	Чоловік	Немає	0	Випускник	Немає	5849	0		360	1
3	LP001003	Чоловік	Так	1	Випускник	Немає	4583	1508 рік	128	360	1
4	LP001005	Чоловік	Так	0	Випускник	Так	3000	0	66	360	1
5	LP001006	Чоловік	Так	0	Не випускник	Немає	2583	2358	120	360	1
6	LP001008	Чоловік	Немає	0	Випускник	Немає	6000	0	141	360	1
7	LP001011	Чоловік	Так	2	Випускник	Так	5417	4196	267	360	1
8	LP001013	Чоловік	Так	0	Не випускник	Немає	2333	1516	95	360	1
9	LP001014	Чоловік	Так	3+	Випускник	Немає	3036	2504	158	360	0
10	LP001018	Чоловік	Так	2	Випускник	Немає	4006	1526	168	360	1
11	LP001020	Чоловік	Так	1	Випускник	Немає	12841	10968	349	360	1
12	LP001024	Чоловік	Так	2	Випускник	Немає	3200	700	70	360	1
13	LP001027	Чоловік	Так	2	Випускник		2500	1840 рік	109	360	1
14	LP001028	Чоловік	Так	2	Випускник	Немає	3073	8106	200	360	1
15	LP001029	Чоловік	Немає	0	Випускник	Немає	1853 рік	2840	114	360	1
16	LP001030	Чоловік	Так	2	Випускник	Немає	1299	1086	17	120	1
17	LP001032	Чоловік	Немає	0	Випускник	Немає	4950	0	125	360	1
18	LP001034	Чоловік	Немає	1	Не випускник	Немає	3596	0	100	240	
19	LP001036	Жінка	Немає	0	Випускник	Немає	3510	0	76	360	0

ТЕСТОВИЙ НАБІР ЗМІННИХ ДЛЯ НАВЧАННЯ МОДЕЛІ

	Loan_ID	Стать	Одружений	Утриманці	Освіта	Самозайнятий	Дохід заявника	Дохід співзаявника	Розмір позики	Сума_позики_термін
2	LP001015	Чоловік	Так	0	Випускник	Немає	5720	0	110	360
3	LP001022	Чоловік	Так	1	Випускник	Немає	3076	1500	126	360
4	LP001031	Чоловік	Так	2	Випускник	Немає	5000	1800 рік	208	360
5	LP001035	Чоловік	Так	2	Випускник	Немає	2340	2546	100	360
6	LP001051	Чоловік	Немає	0	Не випускник	Немає	3276	0	78	360
7	LP001054	Чоловік	Так	0	Не випускник	Так	2165	3422	152	360
8	LP001055	Жінка	Немає	1	Не випускник	Немає	2226	0	59	360
9	LP001056	Чоловік	Так	2	Не випускник	Немає	3881	0	147	360
10	LP001059	Чоловік	Так	2	Випускник		13633	0	280	240
11	LP001067	Чоловік	Немає	0	Не випускник	Немає	2400	2400	123	360
12	LP001078	Чоловік	Немає	0	Не випускник	Немає	3091	0	90	360
13	LP001082	Чоловік	Так	1	Випускник		2185	1516	162	360
14	LP001083	Чоловік	Немає	3+	Випускник	Немає	4166	0	40	180
15	LP001094	Чоловік	Так	2	Випускник		12173	0	166	360
16	LP001096	Жінка	Немає	0	Випускник	Немає	4666	0	124	360
17	LP001099	Чоловік	Немає	1	Випускник	Немає	5667	0	131	360
18	LP001105	Чоловік	Так	2	Випускник	Немає	4583	2916	200	360
19	LP001107	Чоловік	Так	3+	Випускник	Немає	3786	333	126	360

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ
(Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

МАГІСТЕРСЬКА РОБОТА
ЗАСТОСУВАННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ В БАНКІВСЬКІЙ СФЕРІ З МЕТОЮ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ПРИЙНЯТТЯ РІШЕННЯ

Виконав: Студент групи ПДМ-63 Файрушин Роман Віталійович

Керівник: д.т.н., доцент, зав. кафедрою ІПЗ Замрій Ірина Вікторівна

Київ - 2024

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ 2

Мета роботи: підвищення точності прийняття рішень методами машинного навчання при вирішенні задачі кредитного скорингу у банківській сфері.

Об'єкт дослідження: процес прийняття рішень у банківській сфері задачі кредитного скорингу.

Предмет дослідження: методи машинного навчання в процесі прийняття рішень при вирішенні задачі кредитного скорингу у банківській сфері.

ПОСТАНОВКА ЗАДАЧІ

3

- Аналіз існуючих методів побудови скорингових моделей;
- Вибір методу оцінювання якості моделей;
- Побудова моделей машинного навчання;
- Аналіз результатів виконаних обчислювальних експериментів;
- Розробка засобів підвищення точності та якості моделей
- Вибір оптимальної моделі для визначення ймовірності дефолту позичальника.

ОСНОВНІ МЕТОДИ ОЦІНКИ КРЕДИТОСПРОМОЖНОСТІ

4

У світовій практиці роздрібного кредитування існує два основних методи оцінки кредитоспроможності нового клієнта:

1. Експертна оцінка - традиційний процес надання позики, заснований на суб'єктивній оцінці експерта.

2. Кредитний скоринг - це об'єктивна система оцінки, заснована на статистичній моделі, що базується на аналізі історичних даних.

ОБРАНІ МЕТОДИ ДЛЯ ЗАДАЧІ КЛАСИФІКАЦІЇ ПОЗИЧАЛЬНИКІВ 5

№	Метод	Переваги	Недоліки
1	**K-найближчих сусідів (K-Nearest Neighbors)**	Простий, не вимагає навчання моделі.	Повільний на великих наборах даних, чутливий до нормалізації даних.
2	**Нейронні мережі (Deep Neural Network)**	Висока точність, гнучкість у навчанні складних закономірностей.	Вимогливий до обчислювальних ресурсів, складний для інтерпретації.
3	**Метод опорних векторів (SVM)**	Ефективний у високо вимірних просторах, гнучкий за допомогою ядер.	Не найкращий вибір для великих наборів даних, чутливий до параметрів.
4	**Логістична регресія (Logistic Regression)**	Проста у використанні, ефективна для бінарної класифікації.	Погано справляється з нелінійними відносинами, вразлива до перенавчання.
5	**Дерева рішень (Decision Tree)**	Інтуїтивно зрозумілі, здатні до візуалізації, ефективні для великих даних.	Схильність до перенавчання, нестабільність при малих змінах у даних.
6	**Випадковий ліс (Random Forest)**	Менш схильний до перенавчання, хороша продуктивність для багатьох проблем.	Може бути повільним на великих даних, складний для інтерпретації.
7	**XGBoost**	Швидкість і продуктивність, хороші результати на багатьох задачах.	Вимогливий до ресурсів, може бути складним для налаштування.

МЕТОДИ ОЦІНКИ ЯКОСТІ МОДЕЛІ 6

Один з поширених способів оцінки ефективності моделі з бінарними відповідями є використання матриці помилок.

	Дійсна приналежність	
Результат прогнозування	Негативний	Позитивний
Негативний	TN (Істинно негативний)	FN (Хибно негативний)
Позитивний	FP (Хибно позитивний)	TP (Істинно позитивний)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$TPR = \frac{TP}{TP+FN} \qquad FPR = \frac{FP}{FP+TN}$$

ROC-крива-відображає залежність кількості вірно класифікованих позитивних прикладів від кількості невірно класифікованих негативних прикладів.
AUC — площа, обмежена ROC-кривою і віссю частки помилкових позитивних класифікацій. Чим вище показник AUC, тим якісніше діє класифікатор.

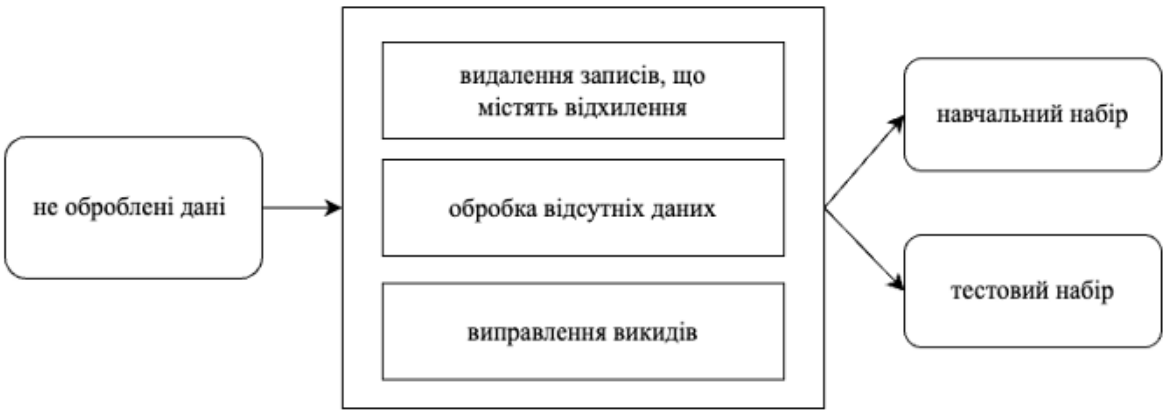
ВХІДНІ ДАНІ 7

Отримані дані були надані кредитним відділом Метро Банку в Лондоні.

Навчальний набір – створено для навчання моделі, який містить усі незалежні змінні та цільову змінну. Містить 13 стовпців ознак і 614 рядків записів.
Тестовий набір – створено для застосування моделей з метою подальшого прогнозування цільової змінної для тестових даних, який містить усі незалежні змінні, без цільової змінної. Набір Містить 12 стовпців ознак і 367 рядків записів.

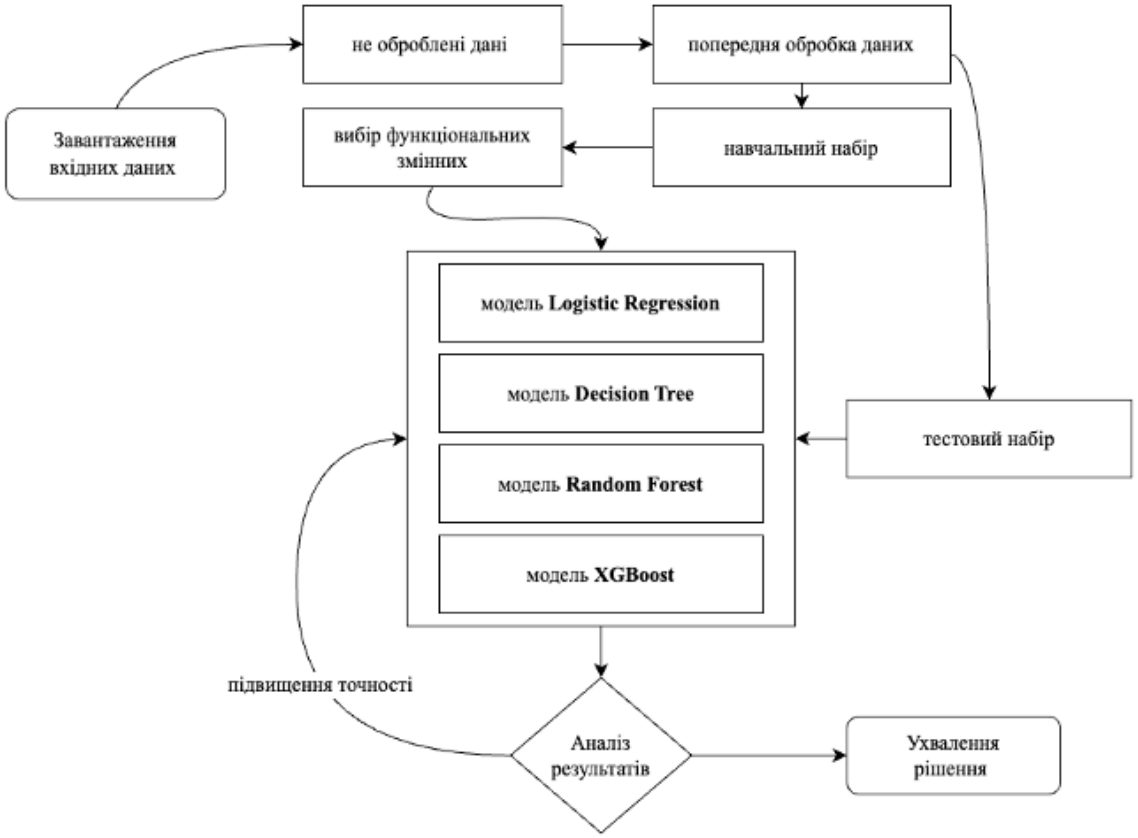
- Стать
- Сімейний стан
- Кількість утриманців (0, 1, 2, 3+)
- Тип зайнятості
- Дохід заявника
- Дохід співзаявника
- Сума кредиту
- Термін позики в місяцях
- Кредитна історія
- Місцевість

ПОПЕРЕДНЯ ОБРОБКА ДАНИХ 8



БЛОК-СХЕМА РОЗРОБЛЕННОЇ ПРОГРАМИ

9

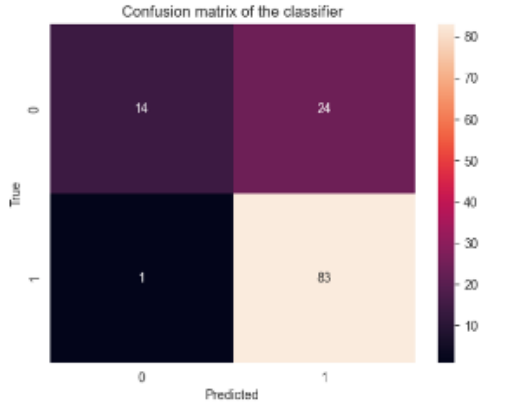


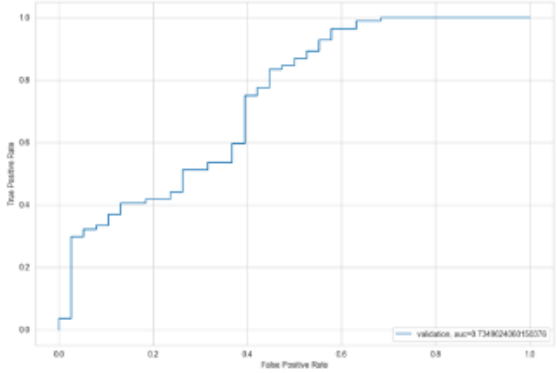
РЕЗУЛЬТАТИ ПОЧАТКОВОЇ МОДЕЛІ ЛОГІЧНОЇ РЕГРЕСІЇ

10

```
1 of kfold 5
accuracy_score 0.8048780487804879

2 of kfold 5
accuracy_score 0.8373983739837398

3 of kfold 5
accuracy_score 0.7804878048780488

4 of kfold 5
accuracy_score 0.7967479674796748

5 of kfold 5
accuracy_score 0.7950819672131147
```

	precision	recall	f1-score	support
0	0.93	0.37	0.53	38
1	0.78	0.99	0.87	84
accuracy			0.80	122
macro avg	0.85	0.68	0.70	122
weighted avg	0.82	0.80	0.76	122

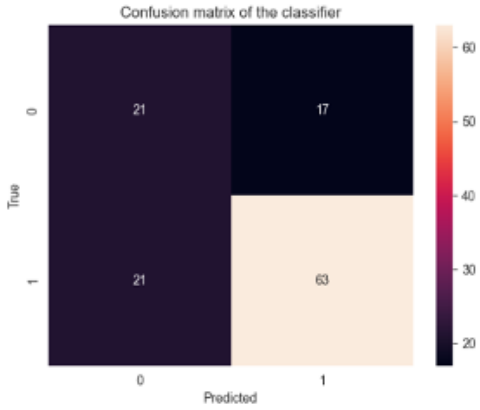




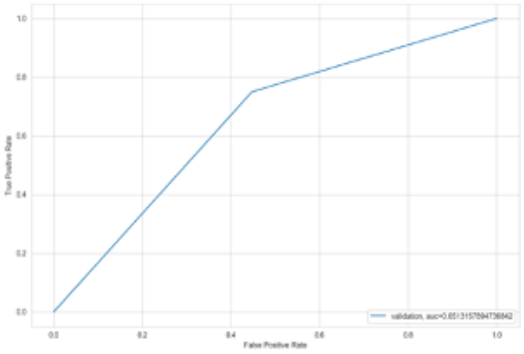
РЕЗУЛЬТАТИ МОДЕЛІ ДЕРЕВА РІШЕНЬ

11

```
1 of kfold 5
accuracy_score 0.6991869918699187

2 of kfold 5
accuracy_score 0.7479674796747967

3 of kfold 5
accuracy_score 0.6666666666666666

4 of kfold 5
accuracy_score 0.6910569105691057

5 of kfold 5
accuracy_score 0.6885245901639344
```



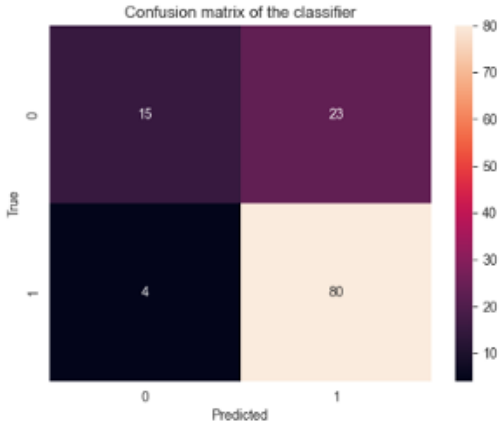
	precision	recall	f1-score	support
0	0.50	0.55	0.53	38
1	0.79	0.75	0.77	84
accuracy			0.69	122
macro avg	0.64	0.65	0.65	122
weighted avg	0.70	0.69	0.69	122



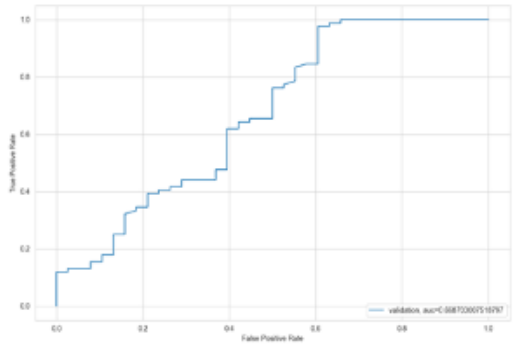
РЕЗУЛЬТАТИ МОДЕЛІ ВИПАДКОВИЙ ЛІС

12

```
1 of kfold 5
accuracy_score 0.7886178861788617

2 of kfold 5
accuracy_score 0.8455284552845529

3 of kfold 5
accuracy_score 0.7479674796747967

4 of kfold 5
accuracy_score 0.7642276422764228

5 of kfold 5
accuracy_score 0.7786885245901639
```



	precision	recall	f1-score	support
0	0.79	0.39	0.53	38
1	0.78	0.95	0.86	84
accuracy			0.78	122
macro avg	0.78	0.67	0.69	122
weighted avg	0.78	0.78	0.75	122

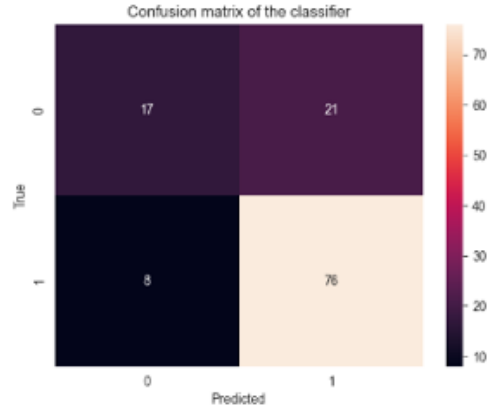


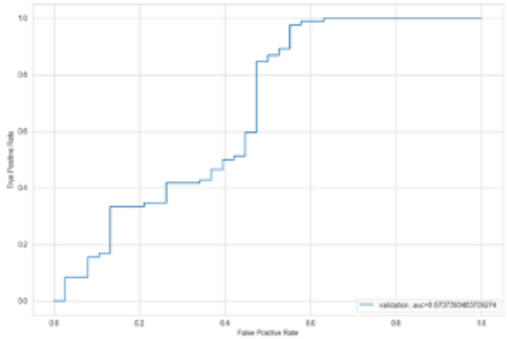
РЕЗУЛЬТАТИ МОДЕЛІ XGBOOST

13

```
1 of kfold 5
accuracy_score 0.7723577235772358

2 of kfold 5
accuracy_score 0.7642276422764228

3 of kfold 5
accuracy_score 0.8130081300813008

4 of kfold 5
accuracy_score 0.7723577235772358

5 of kfold 5
accuracy_score 0.7622950819672131
```

	precision	recall	f1-score	support
0	0.68	0.45	0.54	38
1	0.78	0.90	0.84	84
accuracy			0.76	122
macro avg	0.73	0.68	0.69	122
weighted avg	0.75	0.76	0.75	122





ЗАСОБИ ПІДВИЩЕННЯ ТОЧНОСТІ ПОБУДОВАНИХ МОДЕЛЕЙ

14

1. Створення нових функціональних змінних: Загальний дохід, Прирівняний місячний внесок, Балансовий дохід.

2. Додавання стратифікованої k-кратної перехресної перевірки для визначення стійкості прогностичної моделі.

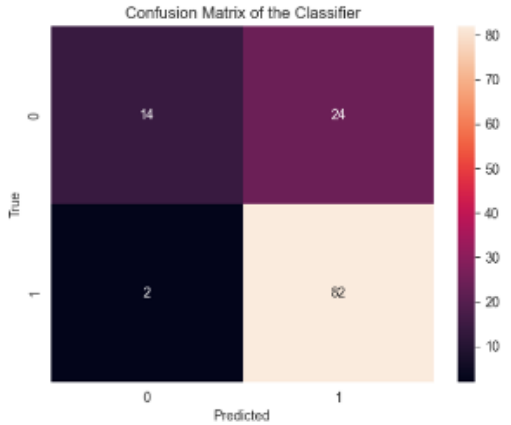
3. Підвищення передбачувальної точності алгоритму **за допомогою процедури GridSearchCV.**

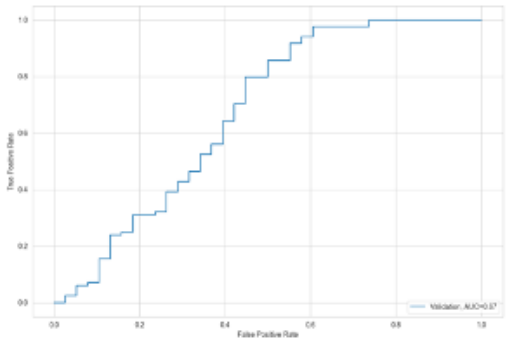
4. Підвищення передбачувальної точності алгоритму шляхом точної корекції його гіперпараметрів

РЕЗУЛЬТАТИ МОДЕЛІ ЛОГІЧНОЇ РЕГРЕСІЇ З ДОДАВАННЯМ ФУНКЦІЙ 15

```
1 of kfold 5
accuracy_score 0.8130081300813008

2 of kfold 5
accuracy_score 0.8373983739837398

3 of kfold 5
accuracy_score 0.7967479674796748

4 of kfold 5
accuracy_score 0.8211382113821138

5 of kfold 5
accuracy_score 0.7868852459016393
```

	precision	recall	f1-score	support
0	0.88	0.37	0.52	38
1	0.77	0.98	0.86	84
accuracy			0.79	122
macro avg	0.82	0.67	0.69	122
weighted avg	0.81	0.79	0.76	122

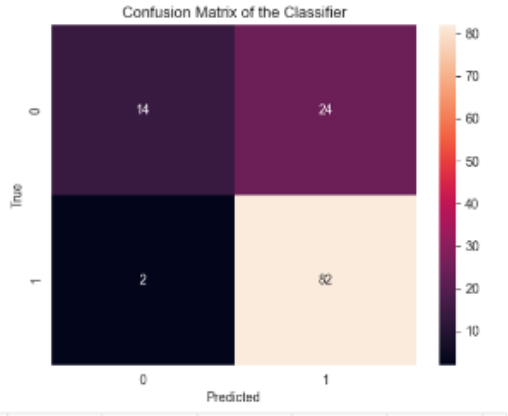


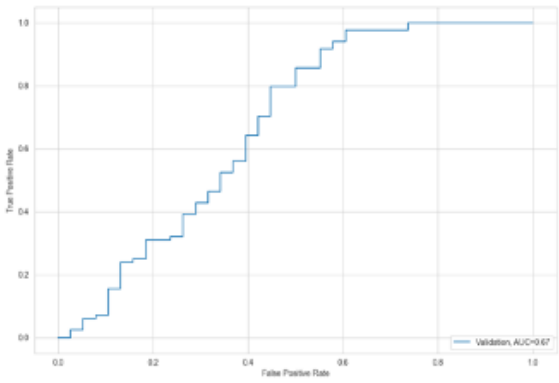


РЕЗУЛЬТАТИ МОДЕЛІ ЛОГІЧНОЇ РЕГРЕСІЇ З ДОДАВАННЯМ ФУНКЦІЙ ТА GRIDSEARCHCV 16

```
1 of kfold 5
accuracy_score 0.8130081300813008

2 of kfold 5
accuracy_score 0.8373983739837398

3 of kfold 5
accuracy_score 0.7967479674796748

4 of kfold 5
accuracy_score 0.8211382113821138

5 of kfold 5
accuracy_score 0.7868852459016393
```

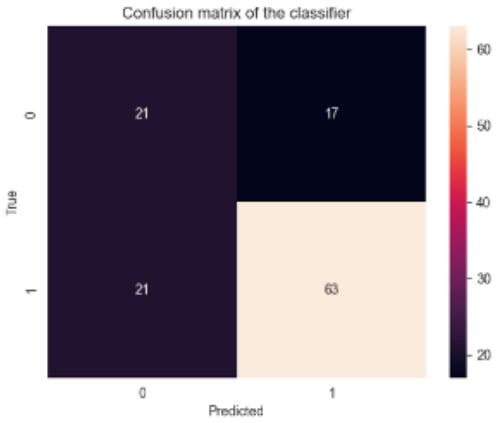
	precision	recall	f1-score	support
0	0.88	0.37	0.52	38
1	0.77	0.98	0.86	84
accuracy			0.79	122
macro avg	0.82	0.67	0.69	122
weighted avg	0.81	0.79	0.76	122



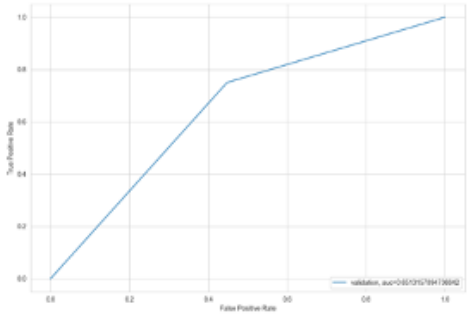


РЕЗУЛЬТАТИ МОДЕЛІ ДЕРЕВА РІШЕНЬ З ДОДАВАННЯМ ФУНКЦІЙ ТА GRIDSEARCHCV 17

```
1 of kfold 5
accuracy_score 0.6991869918699187

2 of kfold 5
accuracy_score 0.7479674796747967

3 of kfold 5
accuracy_score 0.6666666666666666

4 of kfold 5
accuracy_score 0.6910569105691057

5 of kfold 5
accuracy_score 0.6885245901639344
```

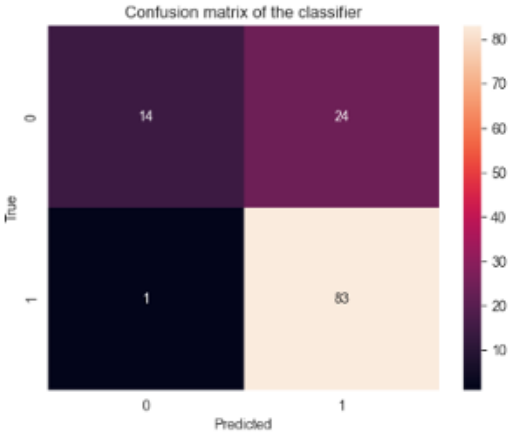


	precision	recall	f1-score	support
0	0.50	0.55	0.53	38
1	0.79	0.75	0.77	84
accuracy			0.69	122
macro avg	0.64	0.65	0.65	122
weighted avg	0.70	0.69	0.69	122

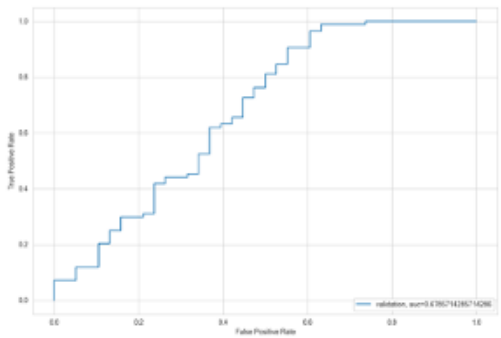


РЕЗУЛЬТАТИ МОДЕЛІ ВИПАДКОВИЙ ЛІС З ДОДАВАННЯМ ФУНКЦІЙ ТА GRIDSEARCHCV 18

```
1 of kfold 5
accuracy_score 0.8130081300813008

2 of kfold 5
accuracy_score 0.8373983739837398

3 of kfold 5
accuracy_score 0.7967479674796748

4 of kfold 5
accuracy_score 0.7967479674796748

5 of kfold 5
accuracy_score 0.7950819672131147
```



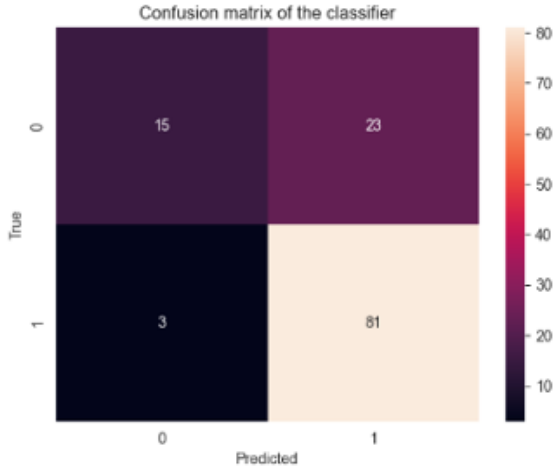
	precision	recall	f1-score	support
0	0.93	0.37	0.53	38
1	0.78	0.99	0.87	84
accuracy			0.80	122
macro avg	0.85	0.68	0.70	122
weighted avg	0.82	0.80	0.76	122

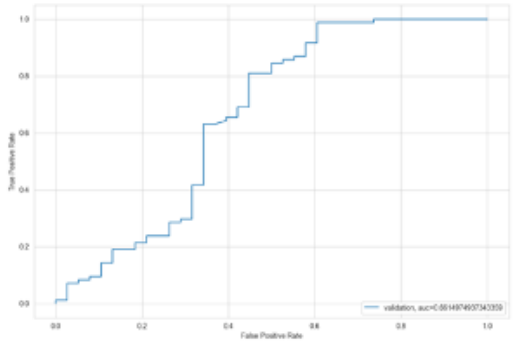


РЕЗУЛЬТАТИ МОДЕЛІ XGBOOST З ДОДАВАННЯМ ФУНКЦІЙ ТА GRIDSEARCHCV 19

```
1 of kfold 5
accuracy_score 0.8130081300813008

2 of kfold 5
accuracy_score 0.8292682926829268

3 of kfold 5
accuracy_score 0.8048780487804879

4 of kfold 5
accuracy_score 0.7967479674796748

5 of kfold 5
accuracy_score 0.7868852459016393
```

	precision	recall	f1-score	support
0	0.83	0.39	0.54	38
1	0.78	0.96	0.86	84
accuracy			0.79	122
macro avg	0.81	0.68	0.70	122
weighted avg	0.80	0.79	0.76	122





ПОРІВНЯННЯ ПОБУДОВАНИХ МОДЕЛЕЙ ТА АНАЛІЗ РЕЗУЛЬТАТІВ 20

Назва	Складність	Точність	AUC	GINI	Час роботи, с	Приріст
Logistic Regression	Початкова	0.802	0.734	0.469	0.506	-
Logistic Regression	З додаванням функцій	**0.812**	**0.701**	**0.402**	**0.538**	**1%**
Logistic Regression	З додаванням функцій та GridSearchCV	0.811	0.671	0.342	**0.744**	-
Decision Tree	З додаванням функцій	0.698	0.698	0.302	0.390	-
Decision Tree	З додаванням функцій та GridSearchCV	**0.736**	**0.692**	**0.384**	**0.334**	**3.8%**
Random Forest	З додаванням функцій	0.785	0.668	0.337	0.354	-
Random Forest	З додаванням функцій та GridSearchCV	**0.807**	**0.678**	**0.357**	**0.532**	**2.2%**
XGBoost	З додаванням функцій	0.776	0.670	0.341	0.397	-
XGBoost	З додаванням функцій та GridSearchCV	**0.806**	**0.661**	**0.322**	**0.299**	**3.0%**

ВИСНОВКИ

21

1. Виконано аналіз найбільш популярних методів побудови скорингових моделей таких як логістична регресія, дерево рішень, випадковий ліс та XGBoost.

2. Обрано метрики оцінювання якості моделі: точність, чутливість, специфічність, характеристика ROC-кривої, та індекс Gini.

3. Виконано аналіз і обробку вхідних даних для побудови моделі.

4. Розроблено програмний продукт, за допомогою якого реалізована класифікація та прогноз ймовірності дефолту позичальника банку.

5. Розроблено методи підвищення точності побудованих моделей: створення нових функціональних змінних, додавання стратифікованої k-кратної перехресної перевірки для визначення стійкості прогностичної моделіб підвищення передбачувальної точності алгоритму за допомогою процедури GridSearchCV та шляхом точної корекції його гіперпараметрів.

6. Виконано аналіз отриманих результатів та визначено найкращу модель за метриками для оцінки кредитоспроможності позичальника.

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

22

Стаття:

Жебка В.В., Файрушин Р.В., EN: ”Application of machine learning methods in the banking sector to increase the efficiency of decision-making” UA: “Застосування методів машинного навчання в банківській сфері з метою підвищення ефективності прийняття рішення”
// Надруковано в “Телекомунікаційні та інформаційні технології” 2023. №2 (79) https://tit.dut.edu.ua/index.php/telecommunication/article/view/2470/2352

Тези доповідей:

1. Жебка В.В., Файрушин Р.В., ”Comparison of existing loan decision-making systems based on machine learning methods” // Міжнародна науково-практична конференція «Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії» – Київ: ДУТ, 2023. С. 147-149.
2. Жебка В.В., Файрушин Р.В., “Disadvantages of existing decision-making systems in the banking sector based on machine learning methods” // Науково-практична конференція «Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії» – Київ: ДУТ, 2023. С. 150-152. https://duikt.edu.ua/uploads/n_11337_64054605.pdf