Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій

Бакалаврська кваліфікаційна робота на тему:

«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПРОГНОЗУВАННЯ РЕЗУЛЬТАТІВ КРЕДИТНОГО СХВАЛЕННЯ НА ОСНОВІ ФІНАНСОВИХ ТА СОЦІАЛЬНИХ ПОКАЗНИКІВ»

Виконав: студент 4 курсу, групи 2ІСТ-21б спеціальності 126 – «Інформаційні системи та технології»
_____ Максим СІЧКАР
Керівник: к.т.н., доц. каф. САІТ
_____ Олексій КОЗАЧКО
«04» 06 2025 р.
Рецензент: к.т.н., доц. каф.
_____ Володимир ОЗЕРАНСЬКИЙ
«08» 06 2025 р.

Допущено до захисту

Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«06» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність 126 – «Інформаційні системи та технології»
Освітньо-професійна програма – Прикладні інформаційні технології

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
"28" 03 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Січкару Максиму Анатолійовичу

1. Тема роботи: «Інформаційна технологія прогнозування результатів кредитного схвалення на основі фінансових та соціальних показників»
керівник роботи: Олексій КОЗАЧКО, к.т.н., доц. каф. САІТ
затверджені наказом ВНТУ від «10» 03 2025 року № 97
2. Термін подання студентом роботи 31.05.25
3. Вихідні дані до роботи:
Kaggle Dataset «Loan Approval Prediction Dataset»
https://www.kaggle.com/datasets/architsharma01/loan-approval-prediction-dataset/data
4. Зміст текстової частини:
1) Аналіз предметної області;
2) Вибір інструментів для розв'язання поставленої задачі та розвідувальний аналіз даних;
3) Розроблення інформаційної технології.
5. Перелік ілюстративного матеріалу:
1) Розвідувальний аналіз даних;
2) Діаграма архітектури системи;
3) Структурна схема інформаційної технології;
4) Блок-схема алгоритму роботи інформаційної технології;
5) Порівняння точності класифікаційних моделей;
6) Результати передбачення оптимальної моделі;
7) Важливість ознак оптимальної моделі.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 28.03.25

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	01.04	10.04	вик
2	Вибір інструментів	10.04	20.04	вик
3	Розвідувальний аналіз даних	20.04	30.04	вик
4	Розроблення інформаційної технології	1.05	15.05	вик
5	Оформлення матеріалів до захисту БКР	15.05	30.05	вик

Студент Максим СІЧКАР

Керівник роботи Олексій КОЗАЧКО

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 86 сторінок формату А4, на яких є 31 рисунка, список використаних джерел містить 38 найменувань.

В роботі розроблено інформаційну технологію прогнозування результатів кредитного схвалення на основі фінансових та соціальних показників.

Виконано аналіз предметної області, проаналізовано проблеми передбачення результатів кредитного схвалення та обґрунтовано актуальність дослідження. Проведено розвідувальний аналіз, відібрано оптимальний набір ознак для дослідження. Вибрано оптимальну ІТ-інфраструктуру інформаційної технології прогнозування результатів кредитного схвалення. Проведено аналіз методів машинного навчання для класифікації. Розроблено інформаційну технологію та протестовано моделі класифікації, проаналізовано результат їх виконання.

Ключові слова: кредитне схвалення, фінансові показники, соціальні показники, інформаційна технологія, ІТ-інфраструктура, розвідувальний аналіз даних, прогнозування, моделі машинного навчання.

ABSTRACT

The bachelor's qualification thesis consists of 86 A4 pages and includes 31 figures. The list of references contains 38 sources.

An information technology for predicting credit approval outcomes based on financial and social indicators was developed.

The study includes an analysis of the subject area, an examination of the challenges related to credit approval prediction, and a justification of the research relevance. Exploratory data analysis was conducted, and an optimal set of features was selected for the research. The optimal IT infrastructure for the credit approval prediction system was identified. Various machine learning classification methods were analyzed. An information technology solution was developed, classification models were tested, and their performance was analyzed.

Keywords: credit approval, financial indicators, social indicators, information technology, IT infrastructure, exploratory data analysis, prediction, machine learning models.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному фінансовому секторі, зокрема в банківській сфері та мікрофінансових організаціях, питання оцінки кредитоспроможності позичальників є критично важливим для мінімізації ризиків неплатежів та забезпечення стабільності фінансових установ. Традиційні методи оцінки, що базуються переважно на фінансовій історії та базових демографічних даних, часто виявляються недостатніми для формування повного та точного профілю позичальника. Це обумовлює необхідність інтеграції більш широкого спектру даних, включаючи соціальні показники, та застосування передових інформаційних технологій. Таким чином розроблення інформаційної технології, здатної точно прогнозувати результати кредитного схвалення, спираючись на всебічний аналіз фінансових та соціальних даних є актуальною задачею.

Мета і задачі дослідження. Метою дослідження є підвищення точності прогнозування результатів кредитного схвалення на основі фінансових та соціальних показників шляхом розроблення інформаційної технології з використанням методів машинного навчання.

Для досягнення поставленої мети необхідно розв’язати такі задачі:

– зробити аналіз об’єкта дослідження;

– здійснити вибір оптимальної IT-інфраструктури;

– провести розвідувальний аналіз даних та виконати обробку вхідних даних;

– розробити інформаційну технологію для прогнозування результатів кредитного схвалення, використовуючи методи машинного навчання.

Об’єктом дослідження є процес створення інформаційної технології для прогнозування результатів кредитного схвалення з використанням методів машинного навчання.

Предметом дослідження є методи і програмні засоби створення інформаційної технології для прогнозування результатів кредитного схвалення з використанням методів машинного навчання.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз об’єкта дослідження

Об'єктом дослідження даної бакалаврської кваліфікаційної роботи є процес прогнозування результатів кредитного схвалення на основі комплексного аналізу фінансових та соціальних показників. Цей процес розглядається як складна інформаційна система, що функціонує в умовах невизначеності та потребує застосування сучасних теоретичних, методологічних та інструментальних засобів для забезпечення її ефективності, надійності та якості.

У сучасному фінансовому секторі, зокрема в банківській сфері та мікрофінансових організаціях, питання оцінки кредитоспроможності позичальників є критично важливим для мінімізації ризиків неплатежів та забезпечення стабільності фінансових установ. Традиційні методи оцінки, що базуються переважно на фінансовій історії та базових демографічних даних, часто виявляються недостатніми для формування повного та точного профілю позичальника. Це обумовлює необхідність інтеграції більш широкого спектру даних, включаючи соціальні показники, та застосування передових інформаційних технологій.

З точки зору інформаційних систем та технологій, процес кредитного схвалення представляє собою складний об'єкт вивчення, що охоплює:

Теоретичні та методологічні основи: Включає принципи інформаційного менеджменту, що регулюють збір, зберігання, обробку та використання даних про позичальників. Системна інтеграція дозволяє об'єднувати дані з різних джерел (фінансові установи, бюро кредитних історій, соціальні мережі, відкриті дані).

Інструментальні засоби створення та використання інформаційних систем: Для реалізації системи прогнозування необхідно обрати та обґрунтувати відповідні мови програмування, фреймворки, бази даних, хмарні сервіси та спеціалізовані бібліотеки для машинного навчання, що відповідають вимогам до ефективності, масштабованості та безпеки [1,2].

Критерії оцінювання та методи забезпечення якості, надійності, відмовостійкості, живучості інформаційних систем: Розроблювана інформаційна технологія прогнозування має відповідати високим стандартам якості. Це передбачає не лише точність прогнозування (якість моделі), а й надійність функціонування системи в цілому, її здатність витримувати збої (відмовостійкість) та адаптуватися до змін (живучість). Важливо враховувати етичні аспекти та питання упередженості (bias) у даних та моделях.

Моделі, методи та засоби оптимізації та прийняття рішень: Суть об'єкта дослідження полягає у розробці моделі прогнозування, яка буде оптимізована для прийняття рішення щодо кредитного схвалення. Це включає вибір оптимальних алгоритмів машинного навчання, налаштування їхніх гіперпараметрів, а також використання метрик оцінки, що відповідають бізнес-цілям (наприклад, баланс між точністю прогнозування дефолту та кількістю схвалених кредитів).

Теоретичний зміст предметної області охоплює:

– Поняття та принципи інформаційного менеджменту: Кредитне схвалення є процесом, що генерує та споживає значний обсяг інформації. Ефективний інформаційний менеджмент забезпечує релевантність, повноту та достовірність даних, необхідних для прогнозування.

– Системна інтеграція: Для поєднання фінансових та соціальних показників з різних джерел (внутрішні бази даних банку, публічні реєстри, агрегатори соціальних даних) виникає потреба у використанні принципів системної інтеграції.

– Архітектура IT-інфраструктури підприємств: Розроблена інформаційна технологія має бути інтегрована в існуючу або перспективну IT-інфраструктуру фінансової установи. Це передбачає врахування аспектів безпеки даних, ефективності та масштабованості рішення.

– Методи, методики, підходи та технології фундаментальних та прикладних наук, моделювання: Центральним елементом дослідження є застосування методів машинного навчання (класифікації) для побудови прогностичної моделі. Це включає методи статистичного аналізу, Data Mining, а

також підходи до Feature Engineering для ефективного використання як фінансових (дохід, витрати, кредитна історія, наявність заборгованостей), так і соціальних показників (освіта, сімейний стан, професійна діяльність, поведінкові патерни, дані з відкритих джерел, що можуть вказувати на соціальний статус або стабільність) [3,4].

Таким чином, об'єкт дослідження є багатогранною системою, що вимагає системного підходу до аналізу, розробки та впровадження інформаційної технології, здатної ефективно прогнозувати результати кредитного схвалення, спираючись на всебічний аналіз фінансових та соціальних даних.

1.2 Аналіз методів прогнозування

Задача прогнозування результатів кредитного схвалення (тобто прийняття рішення "схвалити" або "відмовити") є типовою задачею бінарної класифікації у сфері фінансового аналізу та кредитного скорингу. Ефективність таких систем має прямий вплив на прибутковість фінансових установ, їхню стійкість до ризиків та якість обслуговування клієнтів. Історично для вирішення цієї проблеми використовувалися різноманітні підходи, від традиційних статистичних методів до складних алгоритмів машинного навчання та штучного інтелекту.

На початкових етапах розвитку кредитного скорингу активно застосовувалися статистичні методи, що базуються на емпіричних правилах та математичних моделях:

– Логістична регресія (Logistic Regression): Один з найпоширеніших методів для задач бінарної класифікації. Він моделює ймовірність належності до певного класу (наприклад, ймовірність дефолту) за допомогою сигмоїдної функції (рис.1.1). Переваги: простота інтерпретації, відносна швидкість навчання. Недоліки: припущення про лінійну залежність між ознаками та лог-ймовірністю, що може бути недостатньо для складних взаємозв'язків у фінансових даних [5].

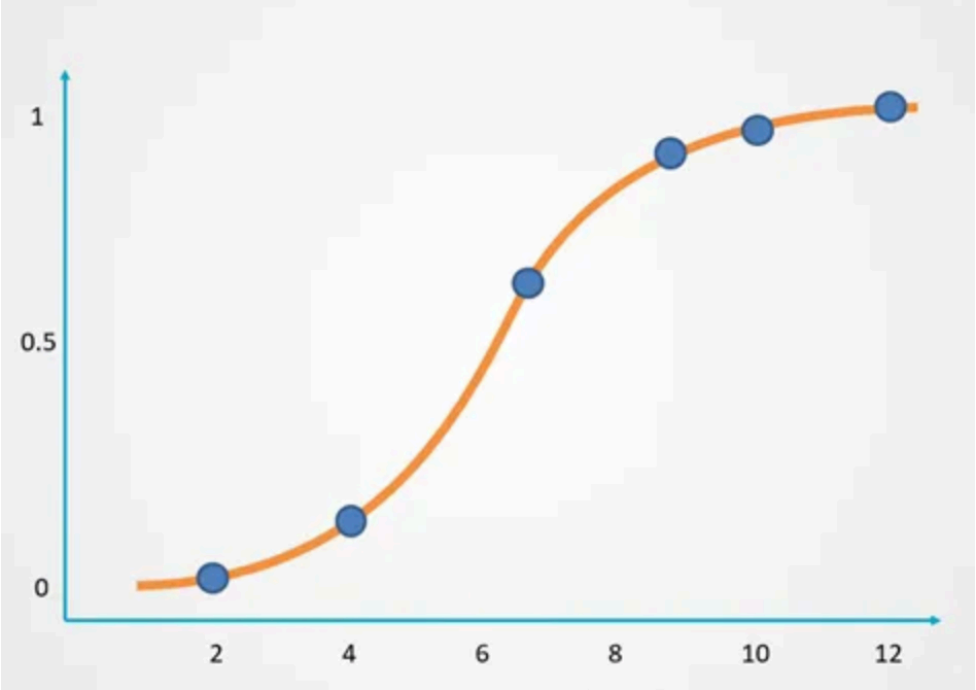

Рисунок 1.1 – Сигмоїдна функція Logistic Regression

Лінійний дискримінантний аналіз (Linear Discriminant Analysis, LDA): Метод, що шукає лінійну комбінацію ознак, яка найкраще розділяє класи [6]. Припускає нормальний розподіл даних та однакові коваріаційні матриці для класів (рис.1.2).

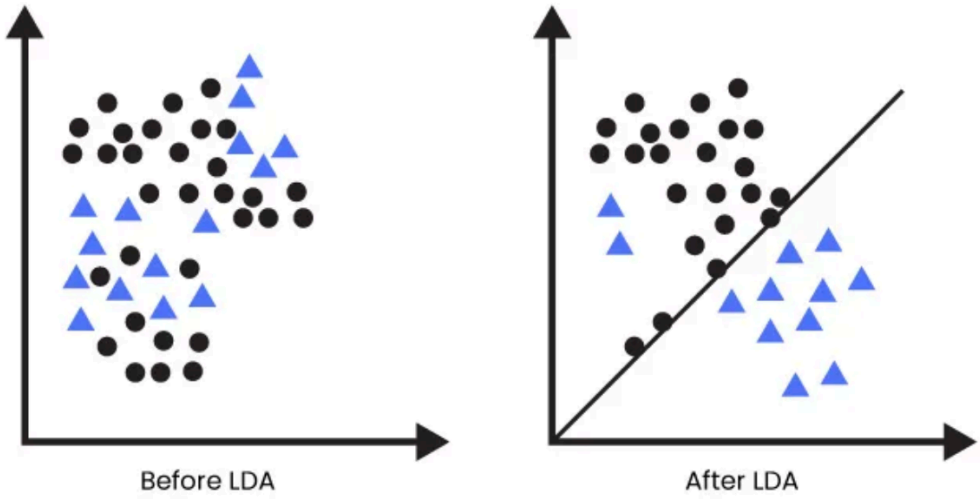

Рисунок 1.2 – Лінійний дискримінантний аналіз

Скорингові карти (Scorecards) [7]: Емпірично розроблені системи, де кожній характеристиці позичальника (дохід, вік, стаж роботи тощо)

присвоюється певна кількість балів. Сума балів визначає кредитний рейтинг. Ці підходи є прозорими, але вимагають глибокої експертизи та можуть бути менш гнучкими до нових даних.

З розвитком обчислювальних потужностей та появою великих обсягів даних, методи машинного навчання стали домінуючими у прогнозуванні кредитного схвалення, пропонуючи вищу точність та здатність виявляти нелінійні залежності:

Логістична регресія (Logistic Regression).

Хоча назва містить "регресія", це фундаментальний лінійний алгоритм для бінарної класифікації. Він моделює ймовірність належності об'єкта до позитивного класу (наприклад, схвалення кредиту) за допомогою сигмоїдної функції, яка відображає будь-яке дійсне число в діапазон від 0 до 1 (див. рис.1.1).

Принцип роботи: алгоритм знаходить оптимальні коефіцієнти для кожної вхідної ознаки, які максимізують правдоподібність спостережуваних даних. Прогноз формується шляхом застосування сигмоїди до лінійної комбінації ознак з цими коефіцієнтами [1-4, 8].

Переваги:

– Простота та інтерпретованість: Коефіцієнти моделі чітко показують напрямок і силу впливу кожної ознаки на ймовірність. Це вкрай важливо у фінансовій сфері, де потрібна прозорість рішень.

– Швидкість: Відносно швидке навчання та прогнозування, що важливо для великих наборів даних або систем реального часу.

– Ймовірнісні прогнози: Надає не просто клас, а ймовірність належності до класу, що дозволяє встановити порогові значення для прийняття рішень.

– Невисокі обчислювальні вимоги: Не потребує значних обчислювальних ресурсів.

Недоліки:

– Припущення про лінійність: Припускає лінійний зв'язок між логарифмом шансів та ознаками. Це обмежує її здатність моделювати складні нелінійні залежності у даних.

– Чутливість до викидів: Може бути чутливою до аномальних значень в даних.

– Проблема мультиколінеарності: Висока кореляція між ознаками може призвести до нестабільності коефіцієнтів.

Умови застосування: Ідеально підходить як базова модель (baseline), для швидкої оцінки даних, коли потрібна висока інтерпретованість, або коли дані мають лінійні взаємозв'язки. Часто використовується як частина більш складних систем.

Метод опорних векторів (Support Vector Machines, SVM) [1-4].

SVM — це потужний алгоритм, який шукає оптимальну розділяючу гіперплощину (або набір гіперплощин) у просторі ознак, що максимально розділяє об'єкти різних класів (рис.1.3).

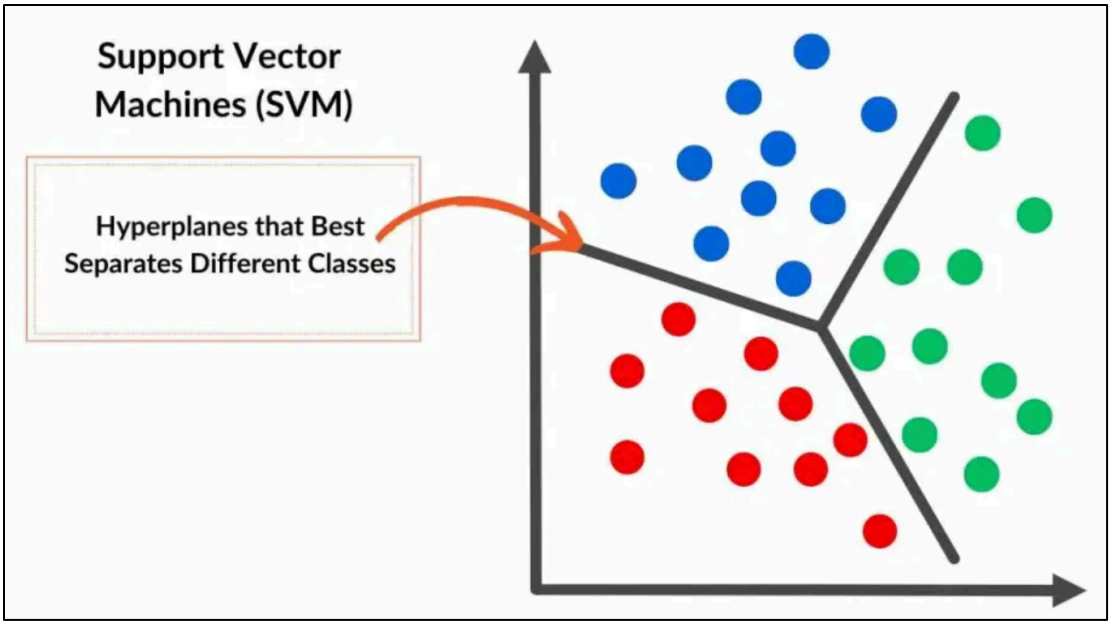

Рисунок 1.3 – Support Vector Machines

Принцип роботи: основна ідея SVM полягає у знаходженні гіперплощини з максимальним "відступом" (margin) між найближчими точками різних класів (опорними векторами). Для нелінійно розділених даних SVM використовує "ядерний трюк" (kernel trick), який дозволяє відображати дані у простір вищої розмірності, де вони стають лінійно розділеними [9].

Переваги:

– Висока ефективність: Добре працює у просторах високої розмірності та з великою кількістю ознак.

– Гнучкість: Завдяки ядерним функціям (лінійне, поліноміальне, радіально-базисне (RBF) ядро тощо) може моделювати складні нелінійні взаємозв'язки.

– Надійність: Менш чутливий до перенавчання порівняно з деякими іншими алгоритмами, особливо при використанні регуляризації.

Недоліки:

– Чутливість до вибору ядерної функції та гіперпараметрів: точність сильно залежить від коректного вибору ядра та його параметрів (наприклад, C, gamma для RBF).

– Обчислювальна складність: Може бути повільним для дуже великих наборів даних, оскільки складність зростає квадратично або кубічно від кількості об'єктів.

– Відсутність ймовірнісних прогнозів: У базовому вигляді надає лише класифікацію, не ймовірність. Додаткові методи (наприклад, калібрування Платта) потрібні для ймовірнісних оцінок.

– Складність інтерпретації: Особливо з нелінійними ядрами, важко зрозуміти, як саме модель приймає рішення.

Умови застосування: ефективний, коли кількість ознак перевищує кількість зразків, або коли дані мають складні нелінійні взаємозв'язки. Може бути доцільним для задач з помірним обсягом даних.

Дерева рішень (Decision Trees) [1-4, 10].

Це непараметричний алгоритм, який розділяє простір ознак на прямокутні регіони. Кожен внутрішній вузол дерева представляє перевірку умови на певну ознаку, кожна гілка — результат перевірки, а кожен листовий вузол — клас (рис.1.4).

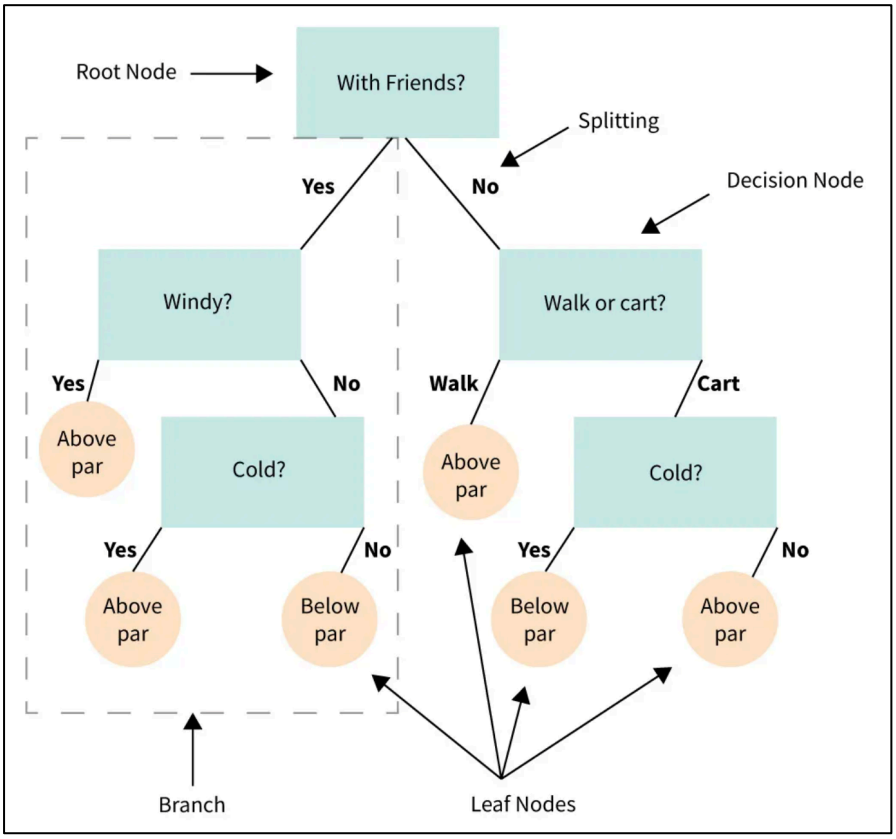

Рисунок 1.4 – Decision Trees

Принцип роботи: дерево будується шляхом рекурсивного розбиття набору даних на підмножини на основі критерію чистоти (наприклад, ентропія, індекс Джині). Метою є створення "чистих" листових вузлів, де більшість об'єктів належать до одного класу [11].

Переваги:

– Простота інтерпретації: Дерева легко зрозуміти та візуалізувати, що є великою перевагою у фінансовій сфері.

– Не вимагає попередньої обробки: Не потребує масштабування ознак або обробки викидів.

– Може обробляти категоріальні та числові дані: Гнучкий у роботі з різними типами даних.

Недоліки:

– Схильність до перенавчання (Overfitting): Одиничні дерева дуже схильні до перенавчання, особливо якщо їм дозволено рости занадто глибоко, що призводить до поганої узагальнюючої здатності на нових даних.

– Нестабільність: Невеликі зміни у даних можуть призвести до значних змін у структурі дерева.

– Неоптимальні локальні рішення: Жадібний підхід до побудови дерева може не знайти глобально оптимального рішення.

Умови застосування: можуть використовуватися для швидкого прототипування, або як базові класифікатори у складі ансамблевих методів. Для самостійного використання вимагають ретельної обрізки (pruning) або обмеження глибини для уникнення перенавчання.

Ансамблеві методи (Ensemble Methods) [12].

Ці методи об'єднують прогнози декількох базових класифікаторів (як правило, дерев рішень) для отримання більш точного та стійкого прогнозу.

Випадковий ліс (Random Forest) [13-15].

Випадковий ліс будує велику кількість (ліс) дерев рішень під час навчання. Для класифікації, він видає моду класів (найпоширеніший клас) окремих дерев (рис.1.6) [16].

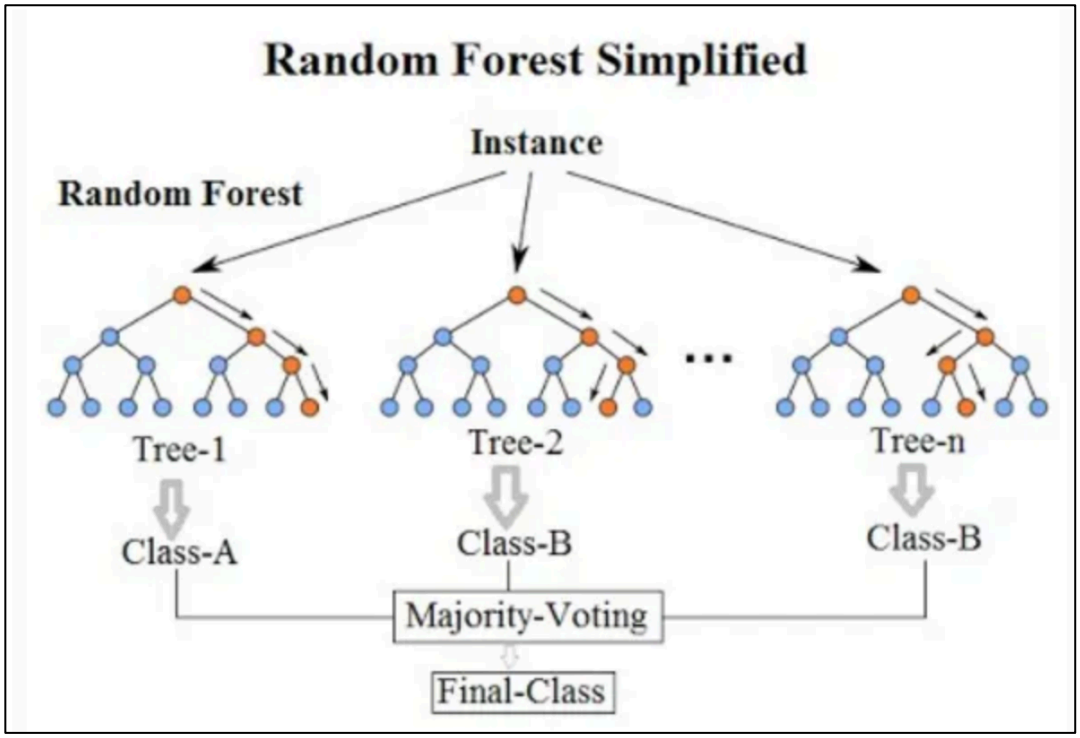

Рисунок 1.5 – Decision Trees

Принцип роботи: Кожне дерево в лісі будується на випадковій підмножині тренувальних даних (bootstrapping) та на випадковій підмножині ознак на кожному розгалуженні. Це зменшує кореляцію між деревами та робить ансамбль більш стійким до перенавчання.

Переваги:

– Висока точність: Зазвичай демонструє дуже високу точність класифікації.

– Стійкість до перенавчання: Завдяки рандомізації та усередненню, значно менш схильний до перенавчання, ніж окремі дерева.

– Обробка великої кількості ознак: Добре працює з великим числом ознак, автоматично визначаючи їхню важливість.

– Нечутливість до масштабування: Не вимагає масштабування ознак.

– Обробка пропущених значень: Може ефективно обробляти пропущені значення.

– Оцінка важливості ознак: Надає вбудовану метрику важливості ознак.

Недоліки:

– Складність інтерпретації: Хоча окремі дерева можна візуалізувати, весь ліс є "чорним ящиком".

– Обчислювальні витрати: Навчання великої кількості дерев може бути ресурсомістким і вимагати більше часу.

Умови застосування: один з найкращих "універсальних" класифікаторів. Чудово підходить для задач кредитного скорингу завдяки високій точності та стійкості до шуму. Використовується, коли важлива висока ефективність, а інтерпретованість не є абсолютним пріоритетом.

Градієнтний бустинг (Gradient Boosting Machines, GBM) [18-21].

GBM — це потужний метод, що послідовно будує ансамбль "слабких" моделей (зазвичай неглибоких дерев рішень), де кожна наступна модель навчається виправляти помилки попередніх.

Принцип роботи: ідея полягає в тому, що кожне нове дерево навчається на "залишках" (residuas) або "градієнтах" помилки, що були зроблені попереднім

ансамблем. Це дозволяє поступово покращувати точність моделі. Популярні реалізації: XGBoost, LightGBM, CatBoost.

Переваги:

– Надзвичайно висока точність: Часто показує найкращі результати на змаганнях з машинного навчання.

– Гнучкість: Може використовувати різні функції втрат, оптимізуючись під конкретну задачу.

– Обробка різнотипних даних: Ефективно працює з числовими та категоріальними ознаками.

– Регуляризація: Сучасні реалізації (XGBoost, LightGBM) включають механізми регуляризації для запобігання перенавчанню.

Недоліки:

– Чутливість до гіперпараметрів: Вимагає ретельного тюнінгу великої кількості гіперпараметрів для досягнення оптимальних результатів.

– Обчислювальні витрати: Навчання може бути повільним, оскільки моделі будуються послідовно.

– Схильність до перенавчання: Якщо параметри не налаштовані правильно, може легко перенавчитися, особливо на невеликих або шумних даних.

– Складність інтерпретації: Аналогічно Random Forest, є "чорним ящиком".

Умови застосування: вибір №1 для багатьох задач класифікації, коли потрібна максимальна точність і є достатньо обчислювальних ресурсів для тюнінгу. Дуже ефективний для кредитного скорингу.

Штучні нейронні мережі (Artificial Neural Networks, ANN) [22].

Опис: моделі, натхненні структурою біологічного мозку, що складаються з взаємопов'язаних "нейронів", організованих у шари.

Принцип роботи: Нейронні мережі навчаються, підлаштовуючи ваги зв'язків між нейронами через процес зворотного поширення помилки

(backpropagation). Це дозволяє їм виявляти складні нелінійні патерни та ієрархічні ознаки в даних.

Переваги:

– Здатність до самонавчання: Можуть вивчати складні взаємозв'язки без явного програмування правил.

– Висока точність: За наявності достатнього обсягу даних, можуть досягати дуже високої точності на складних задачах.

– Гнучкість архітектури: Можливість створювати різноманітні архітектури (MLP, CNN, RNN) для різних типів даних.

Недоліки:

– Велика потреба у даних: Вимагають дуже великих обсягів даних для ефективного навчання та уникнення перенавчання.

– Високі обчислювальні вимоги: Навчання може бути дуже повільним і потребувати потужного обладнання (GPU).

– Складність інтерпретації ("чорний ящик"): Практично неможливо зрозуміти, як саме нейронна мережа приймає рішення, що є серйозним обмеженням у фінансових системах.

– Чутливість до гіперпараметрів: Вимагає ретельного вибору архітектури, функцій активації, оптимізатора, швидкості навчання та інших параметрів.

Умови застосування: рекомендовані, коли доступні дуже великі обсяги різнотипних даних (особливо неструктурованих, як текст або зображення), і коли інтерпретованість моделі не є головним пріоритетом. У фінансовій сфері часто використовуються в поєднанні з іншими моделями або для специфічних типів даних.

Останнім часом зростає інтерес до використання нетрадиційних джерел даних та соціальних показників для покращення якості прогнозування кредитоспроможності. Це може включати:

– Дані з соціальних мереж: Аналіз поведінки, інтересів, соціальних зв'язків (з урахуванням етичних норм та законодавства про персональні дані).

– Дані мобільних операторів: Моделі поведінки, використання послуг, топологія зв'язків.

– Транзакційні дані: Детальний аналіз витрат, що може дати уявлення про фінансову дисципліну та спосіб життя.

– Геопросторові дані: Інформація про місце проживання, роботу, типові маршрути.

Інтеграція таких даних вимагає використання методів обробки природної мови (NLP) для текстових даних та спеціалізованих технік Feature Engineering для вилучення значущих ознак [23].

Вибір оптимального підходу для прогнозування результатів кредитного схвалення залежить від низки факторів: обсягу та якості доступних даних, їхньої природи (структуровані/неструктуровані), обчислювальних ресурсів, вимог до швидкості прогнозування та, що найважливіше, необхідності інтерпретації моделі. Для даної роботи, враховуючи наявність фінансових та соціальних показників, доцільним є застосування методів машинного навчання, здатних обробляти різнотипні дані та виявляти складні взаємозв'язки, такі як ансамблеві методи (Random Forest, Gradient Boosting). Для задачі прогнозування результатів кредитного схвалення, яка характеризується необхідністю високої точності та часто вимагає певної інтерпретованості (хоча б на рівні важливості ознак), ансамблеві методи, такі як Random Forest та Gradient Boosting (зокрема, XGBoost, LightGBM, CatBoost), є найбільш перспективними. Вони поєднують високу прогностичну силу з відносною стійкістю до перенавчання та здатністю обробляти різні типи фінансових та соціальних показників.

Random Forest є чудовим вибором для первинної реалізації та як надійна, робастна модель, що добре працює "з коробки" і менш чутлива до точного налаштування гіперпараметрів [23].

Gradient Boosting може забезпечити ще вищу точність, але потребуватиме більш ретельного тюнінгу та більших обчислювальних ресурсів.

Логістична регресія буде корисною як базова модель для порівняння та для розуміння лінійних взаємозв'язків.

1.3 Висновки

У першому розділі бакалаврської кваліфікаційної роботи було здійснено комплексний аналіз предметної області, пов’язаної з прогнозуванням результатів кредитного схвалення на основі фінансових і соціальних показників. Дослідження охопило теоретичні, методологічні та практичні аспекти побудови інформаційної системи в умовах невизначеності та високих вимог до якості рішень у сфері фінансів.

Обґрунтовано актуальність використання сучасних інформаційних технологій і системної інтеграції для підвищення точності кредитного скорингу, з урахуванням як традиційних фінансових, так і нетрадиційних соціальних ознак. Розглянуто вимоги до IT-інфраструктури, інструментальні засоби та критерії забезпечення якості, відмовостійкості, адаптивності та безпеки майбутньої системи.

Проведено ґрунтовний аналіз методів прогнозування результатів кредитного схвалення — від класичних статистичних моделей до сучасних підходів машинного навчання. Особливу увагу приділено алгоритмам логістичної регресії, SVM, дерев рішень, Random Forest та Gradient Boosting. Визначено переваги та недоліки кожного підходу в контексті задачі бінарної класифікації, що дозволило сформувати обґрунтовану базу для подальшої розробки інформаційної технології.

У підсумку, об’єкт дослідження – процес прогнозування результатів кредитного схвалення – охарактеризовано як складну інформаційну систему, що потребує системного підходу, інтеграції різнотипних джерел даних і застосування ансамблевих моделей машинного навчання. Найбільш доцільним для подальшої реалізації визначено використання методів Random Forest і Gradient Boosting, які забезпечують високу точність, стійкість до перенавчання та здатність працювати з великою кількістю фінансових і соціальних ознак.

.

2 ВИБІР ІТ-ІНФРАСТРУКТУРИ ТА РОЗВІДУВАЛЬНИЙ АНАЛІЗ ДАНИХ

2.1 Вибір мови програмування та середовища розробки

Успішна реалізація інформаційної технології прогнозування результатів кредитного схвалення вимагає ретельного вибору відповідних інструментальних засобів, які забезпечать ефективність розробки, гнучкість у роботі з даними та потужні можливості для застосування алгоритмів машинного навчання. Вибір мови програмування та середовища розробки є ключовим етапом, оскільки він визначає архітектурні можливості системи, її ефективність та зручність подальшої підтримки.

Для задач, пов'язаних з аналізом даних, машинним навчанням та побудовою прогностичних моделей, найчастіше розглядаються такі мови програмування:

– Python;

– R;

– Java.

Переваги Python [23-25]:

– Широка екосистема та бібліотеки: Python є де-факто стандартом у галузі Data Science та машинного навчання завдяки величезній кількості високоякісних бібліотек, таких як NumPy (для числових обчислень), Pandas (для маніпуляцій даними), Scikit-learn (для класичних алгоритмів машинного навчання), TensorFlow та PyTorch (для глибокого навчання). Ця розвинена екосистема значно прискорює розробку та дозволяє реалізувати будь-які етапи проекту – від збору даних до розгортання моделей.

– Простота та читабельність коду: Синтаксис Python є інтуїтивно зрозумілим та схожим на природну мову, що робить його легким для вивчення та підтримки. Це сприяє швидкому прототипуванню та ітераційній розробці.

– Крос-платформність: Python є крос-платформною мовою, що дозволяє розробляти рішення, які можуть працювати на різних операційних системах (Windows, macOS, Linux).

– Активна спільнота: Велика та активна спільнота розробників постійно створює нові інструменти, бібліотеки та надає підтримку.

– Інтеграція: Легко інтегрується з іншими системами та мовами програмування (наприклад, C++, Java) через відповідні інтерфейси.

Недоліки Python:

– Ефективність: У деяких випадках Python може бути повільнішим порівняно з компільованими мовами (наприклад, C++, Java) для обчислювально інтенсивних операцій. Однак більшість критичних за швидкістю обчислень у Data Science виконуються оптимізованими бібліотеками, написаними на C/C++, що нівелює цей недолік.

– Високе споживання пам'яті: Для деяких операцій Python може споживати більше пам'яті, ніж інші мови, хоча це рідко є критичною проблемою для більшості задач машинного навчання.

Переваги R [26-28]:

– Спеціалізація на статистиці: R є мовою, спочатку розробленою для статистичних обчислень та графіки. Має дуже багатий набір пакетів для статистичного моделювання, біоінформатики та візуалізації даних.

– Високоякісна візуалізація: Відомий своїми потужними можливостями для створення високоякісної статистичної графіки (наприклад, за допомогою `ggplot2`).

– Ідеальний для академічних досліджень: Дуже популярний у наукових колах для статистичного аналізу та публікації результатів.

Недоліки R:

– Крива навчання: Синтаксис R може бути менш інтуїтивним для програмістів без досвіду в статистиці.

– Ефективність: аналогічно Python, може бути повільним для дуже великих об'ємів даних або обчислювально інтенсивних завдань, якщо не використовувати оптимізовані пакети.

– Складність розгортання: розгортання моделей R у промислових системах може бути складнішим, ніж для Python.

Переваги Java [29,30]:

– Висока ефективність: компільована мова, що забезпечує високу швидкість виконання.

– Надійність та масштабованість: широко використовується у великих корпоративних системах та системах з високим навантаженням (наприклад, Big Data платформи, такі як Apache Spark).

– Типізація: сильна типізація допомагає уникнути багатьох помилок на етапі компіляції.

Недоліки Java:

– Менша кількість спеціалізованих ML-бібліотек: хоча існують бібліотеки для ML (DL4J, Weka), їхня екосистема не така широка та розвинена, як у Python.

– Багатослівність коду: код на Java зазвичай є більш об'ємним, що уповільнює прототипування.

– Складність розробки: високий поріг входу для новачків у ML, що не мають досвіду в Java.

Вибір середовища розробки (IDE/Notebook Environment) також має значний вплив на ефективність та продуктивність роботи розробника.

Переваги Jupyter Notebook / JupyterLab [31,32]:

– Інтерактивність: дозволяє виконувати код у комірках, що ідеально підходить для розвідувального аналізу даних, покрокової побудови моделей, візуалізації та документування процесу.

– Комбінація коду, тексту та візуалізації: можливість поєднувати код, markdown-текст, формули та вихідні дані (графіки, таблиці) в одному документі.

– Спільна робота: легко ділитися ноутбуками з іншими розробниками.

– Підтримка різних ядер: може працювати з різними мовами програмування (Python, R, Julia та інші).

Недоліки Jupyter Notebook / JupyterLab:

– Управління версіями: складніше інтегрувати з системами контролю версій (Git) порівняно з файлами .py.

– Масштабування: для дуже великих проектів або розгортання у промисловому середовищі можуть знадобитися більш спеціалізовані інструменти.

Переваги PyCharm / VS Code [33]:

– Повноцінне IDE: надають широкий спектр функцій для розробки: автодоповнення коду, налагодження (debugging), інтеграція з системами контролю версій, управління проектами, рефакторинг.

– Зручність для великих проектів: ідеально підходять для розробки складних, багаторазових програмних рішень.

– Ефективність: дозволяють підтримувати високу якість коду та ефективність розробника.

Недоліки PyCharm / VS Code:

– Менша інтерактивність: не настільки зручні для покрокового розвідувального аналізу, як Jupyter-подібні середовища.

– Потреба в налаштуванні: можуть вимагати більшого початкового налаштування.

Переваги Google Colaboratory (Colab) [34,35]:

– Безкоштовний доступ до GPU/TPU: надає безкоштовний доступ до потужних обчислювальних ресурсів, що є критично важливим для навчання великих моделей глибокого навчання.

– Хмарне середовище: не вимагає локального встановлення, доступ через браузер.

– Інтеграція з Google Drive: зручна робота з даними, що зберігаються у хмарному сховищі.

Недоліки Google Colaboratory (Colab):

– Обмеження по часу/ресурсам: Безкоштовна версія має обмеження на час сесії та обсяг доступних ресурсів.

– Залежність від інтернету: Потребує стабільного інтернет-з'єднання.

Kaggle Notebooks (Kaggle Kernels) – це інтерактивне хмарне середовище для розробки та аналізу даних, що надається платформою Kaggle. Воно базується на технології Jupyter Notebook [36,37].

Переваги Kaggle:

– Безкоштовний доступ до GPU/TPU: Kaggle Notebooks також надає безкоштовний доступ до потужних обчислювальних ресурсів, що є значною перевагою для інтенсивних обчислень машинного навчання.

– Інтеграція з даними Kaggle: Максимально зручна інтеграція з численними публічними наборами даних, що доступні на платформі Kaggle, а також можливість завантажувати власні дані.

– Готові бібліотеки: Всі необхідні бібліотеки для Data Science та ML вже встановлені та налаштовані, що дозволяє одразу приступити до роботи.

– Спільна робота та публікація: Зручні інструменти для спільної роботи над проектами та публікації результатів.

– Відтворюваність: Забезпечує високу відтворюваність досліджень, оскільки середовище та версії бібліотек фіксовані.

Недоліки Kaggle:

– Обмеження ресурсів: Існують ліміти на час роботи, обсяг пам'яті та доступні GPU/TPU у безкоштовній версії.

– Залежність від інтернету: Потребує постійного інтернет-з'єднання.

– Менш гнучке для розгортання: Не призначене для розгортання повноцінних промислових додатків.

Враховуючи специфіку задачі прогнозування результатів кредитного схвалення на основі фінансових та соціальних показників, а також вимоги до розробки та аналізу інформаційних систем, було зроблено вибір на користь мови програмування Python та середовища розробки Kaggle Notebooks.

Python обрано з наступних причин:

– Домінування в Data Science: Python є визнаним стандартом для розробки в галузі Data Science та машинного навчання завдяки своїй багатій екосистемі бібліотек (Pandas, NumPy, Scikit-learn, XGBoost тощо), які дозволяють ефективно виконувати всі етапи роботи – від збору та попередньої обробки даних до навчання, оцінки та інтерпретації моделей.

– Простота та швидкість прототипування: Читабельний синтаксис Python дозволяє швидко реалізовувати та тестувати різні гіпотези та моделі, що є критично важливим для ітераційного процесу системного аналізу та моделювання.

– Підтримка та спільнота: Активна спільнота забезпечує постійний розвиток нових інструментів та легкий доступ до рішень типових проблем.

Kaggle Notebooks (Kaggle Kernels) обрано як основне середовище розробки з наступних причин:

– Доступ до обчислювальних ресурсів: Kaggle Notebooks надає безкоштовний доступ до GPU та TPU, що є незамінним для прискорення навчання складних моделей машинного навчання, таких як ансамблеві алгоритми або, за потреби, нейронні мережі. Це усуває потребу у придбанні та налаштуванні потужного локального обладнання.

– Готовність середовища: всі необхідні бібліотеки для аналізу даних та машинного навчання вже попередньо встановлені та налаштовані, що дозволяє миттєво приступити до розробки без витрат часу на конфігурацію оточення.

– Зручність роботи з даними: інтеграція з платформою Kaggle значно спрощує завантаження та використання наборів даних, що може бути актуальним для даної роботи (навіть якщо використовується приватний набір даних, його легко інтегрувати).

– Відтворюваність та спільна робота: хмарна природа та функціонал ноутбуків сприяють легкій відтворюваності експериментів та можливості спільної роботи, що відповідає сучасним підходам у системному аналізі.

Таким чином, комбінація Python як потужної та гнучкої мови програмування з Kaggle Notebooks як ефективним, хмарним середовищем

розробки забезпечує оптимальні умови для успішної реалізації інформаційної технології прогнозування результатів кредитного схвалення.

2.2 ІТ-інфраструктура для розв'язання поставленої задачі

Розв'язання задачі прогнозування результатів кредитного схвалення на основі фінансових та соціальних показників вимагає створення та належного функціонування відповідної інформаційної технології, що, в свою чергу, спирається на продуману ІТ-інфраструктуру. Цей підрозділ детально описує архітектуру та компоненти інфраструктури, які були задіяні для реалізації поставленого завдання, з урахуванням об'єктів вивчення та теоретичного змісту освітньої програми спеціальності "Інформаційні системи та технології".

2.2.1 Архітектура ІТ-інфраструктури для задачі прогнозування кредитів

При створенні інформаційних систем та технологій для прогнозування кредитів, ключовим є забезпечення їхньої якості, надійності, відмовостійкості та живучості. Архітектура ІТ-інфраструктури для даної задачі ґрунтується на концепції масштабованої та модульної системи, здатної ефективно обробляти дані, навчати моделі та надавати прогнози. І вона буде складатись з наступних рівнів:

1. Рівень збирання та зберігання даних (Data Ingestion & Storage Layer). Забезпечення якості та надійності інформаційних систем вимагає вибору відповідних технологій зберігання даних, які гарантують доступність та збереження інформації.

2. Рівень попередньої обробки та підготовки даних (Data Preprocessing & Preparation Layer). Методи, методики та підходи фундаментальних та прикладних наук, зокрема статистики та лінійної алгебри, застосовуються для оптимізації даних для подальшого аналізу та моделювання.

3. Рівень моделювання та навчання (Modeling & Training Layer). Тут застосовуються моделі, методи та засоби оптимізації та прийняття рішень, що є

фундаментальними для розробки інформаційних систем, здатних генерувати обґрунтовані прогнози.

4. Рівень оцінки та моніторингу (Evaluation & Monitoring Layer). Безперервне оцінювання та моніторинг є ключовими для забезпечення якості, надійності та живучості інформаційних систем, дозволяючи своєчасно виявляти та усувати проблеми.

Розглянемо більш детальні всі зазначені рівні архітектура IT-інфраструктури.

Рівень збирання та зберігання даних (Data Ingestion & Storage Layer):

– Об'єкти вивчення: Теоретичні та методологічні основи створення інформаційних систем, критерії оцінювання надійності.

– Реалізація: Для ефективного збирання та зберігання фінансових та соціальних показників, які є основою для прогнозування, використовується система управління базами даних. У контексті розробки цієї БКР, дані представлені у файловому форматі (наприклад, CSV), що є типовим для початкових етапів розробки та валідації. У реальних виробничих умовах, це міг би бути реляційна СУБД (наприклад, PostgreSQL, MySQL) для структурованих даних клієнтів та транзакцій, або NoSQL бази даних (наприклад, MongoDB) для менш структурованих соціальних показників чи великих обсягів логів. Вибір СУБД визначається вимогами до обсягу даних, їхньої структури, швидкості доступу та масштабованості. Надійність системи зберігання даних є критично важливою для забезпечення цілісності інформації.

Рівень попередньої обробки та підготовки даних (Data Preprocessing & Preparation Layer):

– Об'єкти вивчення: Інструментальні засоби використання інформаційних систем та технологій, методи оптимізації.

– Реалізація: На цьому рівні здійснюється очищення даних від пропусків, трансформація категоріальних ознак у числові (наприклад, One-Hot Encoding), масштабування числових ознак (наприклад, StandardScaler) та генерація нових ознак (Feature Engineering), таких як співвідношення активів до доходу чи суми кредиту. Використання бібліотек Python, таких як pandas та

scikit-learn, є типовим підходом на цьому етапі. Ефективна попередня обробка даних є ключовим етапом для

Рівень моделювання та навчання (Modeling & Training Layer):

– Об'єкти вивчення: інструментальні засоби створення інформаційних систем та технологій, моделі, методи та засоби прийняття рішень.

– Реалізація: цей рівень є серцем системи прогнозування. Він включає вибір та навчання моделей машинного навчання (Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, SVM, K-Nearest Neighbors), а також їхню валідацію. Для оптимізації точності моделей використовується тюнінг гіперпараметрів (наприклад, за допомогою GridSearchCV або RandomizedSearchCV). Моделі розробляються з використанням бібліотек scikit-learn та xgboost. Вибір моделей визначається необхідністю досягнення високої точності прогнозування та уникнення перенавчання.

Рівень оцінки та моніторингу (Evaluation & Monitoring Layer):

– Об'єкти вивчення: критерії оцінювання і методи забезпечення якості, надійності, відмовостійкості, живучості інформаційних систем та технологій.

– Реалізація: після навчання моделі, її точність ретельно оцінюється за допомогою таких метрик, як Accuracy, Precision, Recall, F1-Score та ROC AUC. Це дозволяє визначити найкращу модель для розгортання. У реальних системах, після розгортання, передбачаються механізми постійного моніторингу точності моделі на реальних даних, щоб виявляти деградацію (model drift) та забезпечувати живучість системи.

Для наочного представлення описаної IT-інфраструктури, можна використати діаграму архітектури системи (рис.2.1).

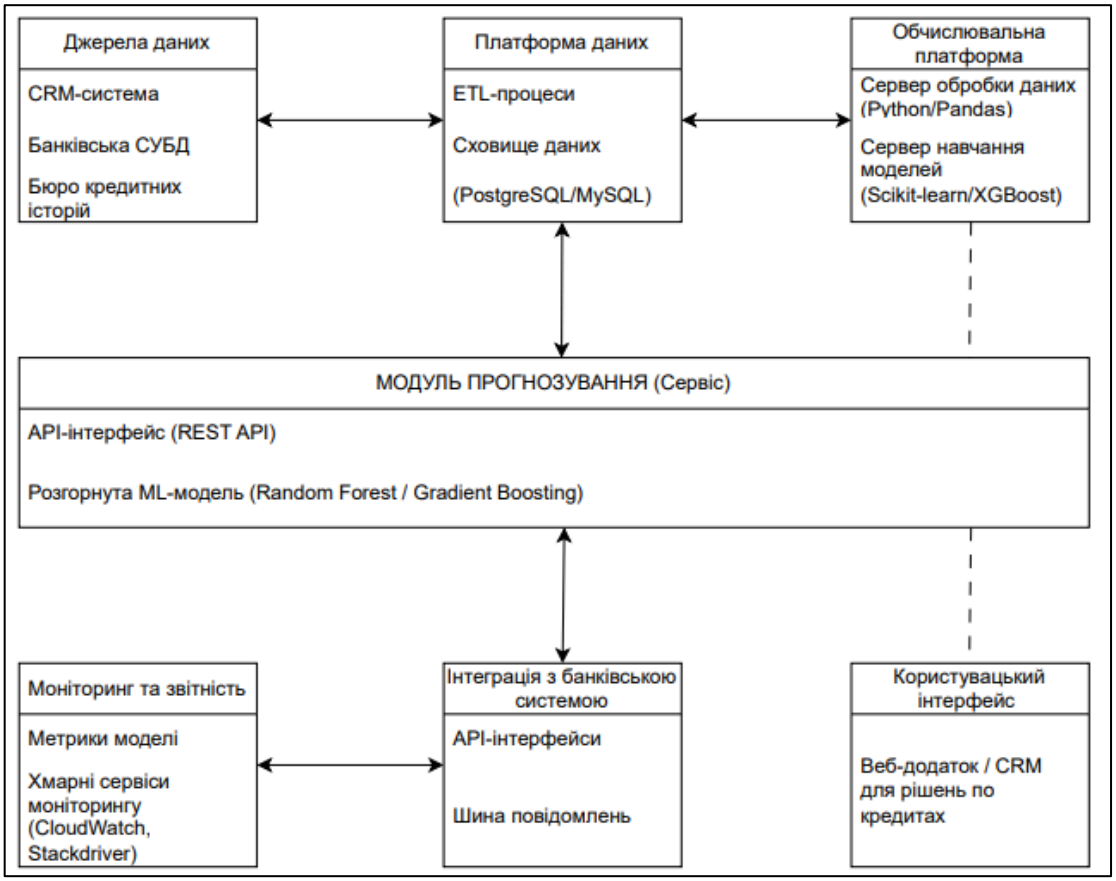

Рисунок 2.1 – Діаграма архітектури системи

Елементи діаграми:

– Джерела даних: CRM-система, Банківська система, Зовнішні джерела (наприклад, бюро кредитних історій).

– Платформа даних:

1) ETL-процеси: для вилучення, трансформації та завантаження даних.
2) Сховище даних: Data Warehouse або Data Lake (для більших систем).
3) База даних: конкретна СУБД (наприклад, PostgreSQL).

– Обчислювальна платформа:

1) Сервер обробки даних: (наприклад, віртуальна машина, контейнер Docker) з встановленим Python, Pandas, NumPy.
2) Сервер для навчання моделей: (можливо, з GPU для великих моделей) з Scikit-learn, XGBoost.
3) Середовище для Jupyter/IDE: для розробки та прототипування.

– Модуль прогнозування (сервіс):
    1) API-інтерфейс: для отримання запитів та повернення прогнозів.
    2) Завантажена модель: збережена та готова до використання.
– Моніторинг та звітність:
    1) Система моніторингу: (наприклад, Prometheus, Grafana) для відстеження продуктивності моделі та інфраструктури.
    2) Хмарні сервіси моніторингу (CloudWatch, Stackdriver, Azure Monitor): надають інструменти для збору метрик, логів, відстеження стану інфраструктури та продуктивності моделей у реальному часі.
– Користувацький інтерфейс / Інтеграція:
    1) Банківська система / Фронт-енд: де відображаються результати прогнозу.

2.2.2 Інструментальні засоби та технології

Для розв'язання поставленої задачі були використані наступні інструментальні засоби та технології, що є частиною типової IT-інфраструктури для завдань аналізу даних та машинного навчання:

– Мова програмування Python – обраний через свою гнучкість, велику кількість бібліотек для аналізу даних та машинного навчання, а також широке застосування в IT-індустрії.
– Бібліотеки для аналізу даних:
    1) pandas – для маніпуляцій з даними, очищення, трансформації та агрегації.
    2) numpy – для числових обчислень.
    3) Бібліотеки для машинного навчання:
    4) scikit-learn – для реалізації класифікаційних алгоритмів, попередньої обробки даних (масштабування, кодування), розділення даних та оцінки моделей.

5) xgboost – для реалізації Gradient Boosting, що забезпечує високу точність.

– Бібліотеки для візуалізації даних:

1) matplotlib та seaborn – для створення інформативних графіків, таких як гістограми, боксплоти, кореляційні матриці та confusion matrix, що є важливим для EDA та представлення результатів.

– Середовище розробки Kaggle Notebooks – інтерактивне середовище, що дозволяє поєднувати код, текст та візуалізації, ідеально підходить для розвідувального аналізу даних та прототипування моделей.

– Обчислювальні ресурси – проєкт розроблявся на локальній обчислювальній машині, що є початковим етапом для подальшого розгортання на більш потужних серверах або хмарних платформах у промислових масштабах. У випадку реального розгортання, IT-інфраструктура могла б включати хмарні обчислювальні платформи (наприклад, AWS, Google Cloud, Azure) для масштабованості, або ж локальні сервери з GPU для прискорення навчання складних моделей.

2.2.3 Принципи системної інтеграції та адміністрування

Хоча ця робота зосереджена на моделюванні, реалізація подібної інформаційної технології в банківській системі передбачає:

– Системна інтеграція: необхідність інтеграції моделі прогнозування з існуючими банківськими інформаційними системами (наприклад, системою обліку клієнтів, системою видачі кредитів). Це вимагає використання API, стандартизованих протоколів обміну даними та забезпечення сумісності.

– Адміністрування інформаційних систем: для підтримки такої системи в робочому стані потрібні механізми адміністрування, включаючи моніторинг ресурсів, управління версіями моделей, резервне копіювання даних та журналювання подій.

– Управління IT-проектами: розробка та впровадження подібної системи є повноцінним IT-проектом, який вимагає застосування методологій управління (наприклад, Agile) для ефективного планування, виконання та контролю.

Таким чином, IT-інфраструктура для розв'язання задачі прогнозування кредитного схвалення є комплексним рішенням, що охоплює різні рівні обробки даних та моделювання. Вона ґрунтується на фундаментальних принципах створення інформаційних систем та технологій, забезпечуючи їхню функціональність, надійність та ефективність у реальному бізнес-середовищі.

2.3 Розвідувальний аналіз даних

Для проведення аналізу було обрано набір даних у Kaggle, що має назву «Loan-Approval-Prediction-Dataset». Даний датасет має відкритий доступ для загального використання на платформі [38]. Цей набір даних про схвалення позики – це сукупність фінансових записів та пов’язаної з ними інформації, що використовується для визначення права фізичних осіб або організацій на отримання позик від кредитної установи. Він включає різні фактори, такі як громадянський рейтинг, дохід, статус зайнятості, термін позики, сума позики, вартість активів та статус позики. Всього в датасеті 13 ознак та 4269 записів (рис. 2.2).

	loan_id	no_of_dependents	education	self_employed	income_annum	loan_amount	loan_term	cibil_score	residential_assets_value
0	1	2	Graduate	No	9600000	29900000	12	778	2400000
1	2	0	Not Graduate	Yes	4100000	12200000	8	417	2700000
2	3	3	Graduate	No	9100000	29700000	20	506	7100000
3	4	3	Graduate	No	8200000	30700000	8	467	18200000
4	5	5	Not Graduate	Yes	9800000	24200000	20	382	12400000
...	...	...	...	...	...	...	...	...	...
4264	4265	5	Graduate	Yes	1000000	2300000	12	317	2800000
4265	4266	0	Not Graduate	Yes	3300000	11300000	20	559	4200000
4266	4267	2	Not Graduate	No	6500000	23900000	18	457	1200000
4267	4268	1	Not Graduate	No	4100000	12800000	8	780	8200000
4268	4269	1	Graduate	No	9200000	29700000	10	607	17800000

4269 rows × 13 columns

Рисунок 2.2 – Приклад ознак у наборі даних

Розглянемо повний список ознак, які є у нашому наборі даних, зробимо їхній опис. Отож даний датасет включає у себе такі колонки:

– loan_id: Унікальний ідентифікатор кожного запису (заявки на кредит). Ця ознака використовується для ідентифікації окремих випадків і, як правило, не застосовується для навчання моделей машинного навчання, оскільки не містить інформації, корисної для прогнозування.

– no_of_dependents: Кількість утриманців заявника. Це показник, що відображає фінансове навантаження на заявника, оскільки чим більше утриманців, тим більші щомісячні витрати можуть бути.

– education: Освіта заявника. Категоріальна ознака, що вказує на рівень освіти (наприклад, "Graduate" - вища освіта, "Not Graduate" - без вищої освіти). Освіта може бути індикатором стабільності доходу та загальної фінансової грамотності.

– self_employed: Статус зайнятості заявника. Категоріальна ознака, яка вказує, чи є заявник самозайнятим ("Yes") або працює за наймом ("No"). Самозайнятість може розглядатися як фактор підвищеного ризику через меншу стабільність доходу порівняно з фіксованою зарплатою.

– income_annum: Річний дохід заявника. Числова ознака, що відображає загальний дохід особи за рік. Це один з найважливіших факторів для оцінки платоспроможності позичальника.

– loan_amount: Сума кредиту. Числова ознака, що відображає запитувану суму позики.

– loan_term: Термін кредиту в роках. Числова ознака, що вказує на період, протягом якого позичальник планує погасити кредит. Довші терміни зазвичай асоціюються з меншими щомісячними платежами, але більшою загальною сумою відсотків.

– cibil_score: Кредитний рейтинг. Числова ознака, що є оцінкою кредитоспроможності заявника. Вищий бал CIBIL (Credit Information Bureau (India) Limited) вказує на кращу кредитну історію та нижчий ризик для кредитора. Це один з найкритичніших показників у кредитному скорингу.

– residential_assets_value: Вартість житлових активів. Числова ознака, що представляє грошову вартість нерухомого майна житлового призначення, яке належить заявнику (наприклад, квартира, будинок). Ці активи можуть виступати як застава або свідчити про загальний фінансовий стан.

– commercial_assets_value: Вартість комерційних активів. Числова ознака, що відображає грошову вартість комерційної нерухомості або інших активів, які використовуються для бізнесу та належать заявнику.

– luxury_assets_value: Вартість розкішних активів. Числова ознака, що відображає вартість високоцінних рухомих активів, таких як дорогі автомобілі, ювелірні вироби, твори мистецтва тощо.

– bank_asset_value: Вартість банківських активів. Числова ознака, що представляє загальну суму коштів на банківських рахунках (депозити, поточні рахунки) та інших ліквідних фінансових активів, що належать заявнику.

– loan_status: Статус схвалення кредиту. Цільова категоріальна ознака, яка вказує, чи була заявка на кредит схвалена ("Approved") або відхилена ("Rejected"). Це те, що ми намагаємося прогнозувати.

Визначимо наявність пропущених значень та типи даних у стовбцях (рис. 2.3).

```
print("\nЗагальна інформація про датасет:")
data.info()
```

```
Загальна інформація про датасет:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4269 entries, 0 to 4268
Data columns (total 13 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   loan_id                   4269 non-null   int64
 1    no_of_dependents         4269 non-null   int64
 2    education                4269 non-null   object
 3    self_employed            4269 non-null   object
 4    income_annum             4269 non-null   int64
 5    loan_amount              4269 non-null   int64
 6    loan_term                4269 non-null   int64
 7    cibil_score              4269 non-null   int64
 8    residential_assets_value 4269 non-null   int64
 9    commercial_assets_value  4269 non-null   int64
 10   luxury_assets_value      4269 non-null   int64
 11   bank_asset_value         4269 non-null   int64
 12   loan_status              4269 non-null   object
dtypes: int64(10), object(3)
memory usage: 433.7+ KB
```

Рисунок 2.3 – Результат виконання функції info()

З рисунку 2.3 видно, що у всьому датасеті відсутні пропущені значення. Це спрощує попередню обробку даних, оскільки не потрібно застосовувати методи імпутації. loan_id — це ідентифікатор і не є ознакою для моделювання. Категоріальні ознаки: education, self_employed, loan_status.

Використаємо функцію describe(), щоб визначити описові статистики числових ознак (рис. 2.4).

```
data.describe()
```

Основні статистичні характеристики числових ознак:

	loan_id	no_of_dependents	income_annum	loan_amount	loan_term	cibil_score	residential_assets_value	commercial_assets_value	luxury_assets_value	bank_asset_value
count	4269.000000	4269.000000	4.269000e+03	4.269000e+03	4269.000000	4269.000000	4.269000e+03	4.269000e+03	4.269000e+03	4.269000e+03
mean	2135.000000	2.498712	5.059124e+06	1.513345e+07	10.900445	599.936051	7.472617e+06	4.973155e+06	1.512631e+07	4.976692e+06
std	1232.498479	1.695910	2.806840e+06	9.043363e+06	5.709187	172.430401	6.503637e+06	4.388966e+06	9.103754e+06	3.250185e+06
min	1.000000	0.000000	2.000000e+05	3.000000e+05	2.000000	300.000000	-1.000000e+05	0.000000e+00	3.000000e+05	0.000000e+00
25%	1068.000000	1.000000	2.700000e+06	7.700000e+06	6.000000	453.000000	2.200000e+06	1.300000e+06	7.500000e+06	2.300000e+06
50%	2135.000000	3.000000	5.100000e+06	1.450000e+07	10.000000	600.000000	5.600000e+06	3.700000e+06	1.460000e+07	4.600000e+06
75%	3202.000000	4.000000	7.500000e+06	2.150000e+07	16.000000	748.000000	1.130000e+07	7.600000e+06	2.170000e+07	7.100000e+06
max	4269.000000	5.000000	9.900000e+06	3.950000e+07	20.000000	900.000000	2.910000e+07	1.940000e+07	3.920000e+07	1.470000e+07

Рисунок 2.4 – Результат виконання функції describe()

За результатами можемо зробити аналіз.

Більшість фінансових ознак (income_annum, loan_amount, assets_value) демонструють дуже широкий діапазон значень та значне стандартне відхилення. Це вказує на велику варіативність фінансового стану та потреб заявників.

Розподіл активів: вартість активів (residential_assets_value, commercial_assets_value, luxury_assets_value, bank_asset_value) має сильно скошені (вправо) розподіли, що свідчить про те, що значна частина заявників має низькі або нульові значення цих активів, тоді як менша частина володіє значними статками. Це є важливим фактором для розуміння демографічного та фінансового профілю клієнтів.

Кредитний рейтинг (cibil_score): Ця ознака знаходиться в типовому діапазоні кредитних балів і, як показали результати моделювання, є дуже потужним предиктором.

Результати describe() підкреслюють необхідність таких кроків попередньої обробки, як масштабування числових ознак, оскільки їх діапазони значно

відрізняються, що може впливати на роботу деяких алгоритмів машинного навчання.

Далі перевіримо кількість унікальних значень в кожному стовбці (рис. 2.5).

```
for col in data.columns:
    print(f"- {col}: {data[col].nunique()} унікальних значень")
    # Для колонок з невеликою кількістю унікальних значень, вивести їх
    if data[col].nunique() < 20:
        print(f"  Унікальні значення: {data[col].unique()}")

print("\n--- Початковий огляд завершено. ---")
```

```
Кількість унікальних значень для кожної колонки:
- loan_id: 4269 унікальних значень
-  no_of_dependents: 6 унікальних значень
  Унікальні значення: [2 0 3 5 4 1]
-  education: 2 унікальних значень
  Унікальні значення: [' Graduate' ' Not Graduate']
-  self_employed: 2 унікальних значень
  Унікальні значення: [' No' ' Yes']
-  income_annum: 98 унікальних значень
-  loan_amount: 378 унікальних значень
-  loan_term: 10 унікальних значень
  Унікальні значення: [12  8 20 10  4  2 18 16 14  6]
-  cibil_score: 601 унікальних значень
-  residential_assets_value: 278 унікальних значень
-  commercial_assets_value: 188 унікальних значень
-  luxury_assets_value: 379 унікальних значень
-  bank_asset_value: 146 унікальних значень
-  loan_status: 2 унікальних значень
  Унікальні значення: [' Approved' ' Rejected']
```

Рисунок 2.5 – Перевірка унікальних значень

Аналіз унікальних значень підтвердив структуру даних:

– loan_id є унікальним ідентифікатором і не повинен використовуватися для моделювання.

– Виявлено декілька бінарних категоріальних ознак (education, self_employed, loan_status) та дискретно-числових/порядкових ознак (no_of_dependents, loan_term), які вимагають відповідного кодування або обробки.

– Числові ознаки (income_annum, loan_amount, cibil_score, різні assets_value) мають достатню варіативність для подальшого аналізу та використання в моделях.

– Звертаємо увагу на наявність зайвих пробілів у категоріальних значеннях (наприклад, ' Graduate', ' No'). Це необхідно буде усунути на етапі очищення даних, щоб забезпечити коректне кодування та уникнути помилок під час подальшого аналізу.

Після очищення значення категоріальних колонок (видалення зайвих пробілів у значеннях категоріальних колонок, якщо такі є), візуалізуємо розподіл цільової змінної (рис. 2.6).

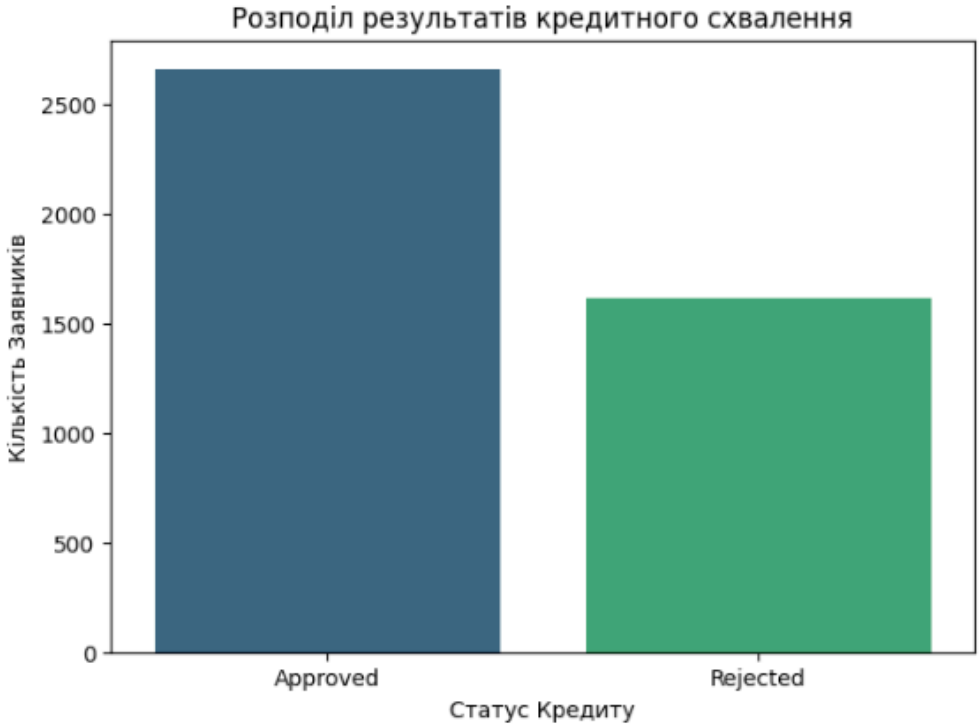

Рисунок 2.6 – Розподіл цільової змінної

З рисунка 2.6 видно помірний дисбаланс класів (у відсотках: 62% та 38%) – це помірний дисбаланс, який не є критичним, але ми пам'ятатимемо про нього на етапі моделювання, щоб обрати відповідні метрики оцінки (наприклад, F1-score, Precision, Recall) та, можливо, розглянути техніки балансування, якщо моделі демонструватимуть низьку точність.

Візуалізуємо розподіл ознак (рис. 2.7).

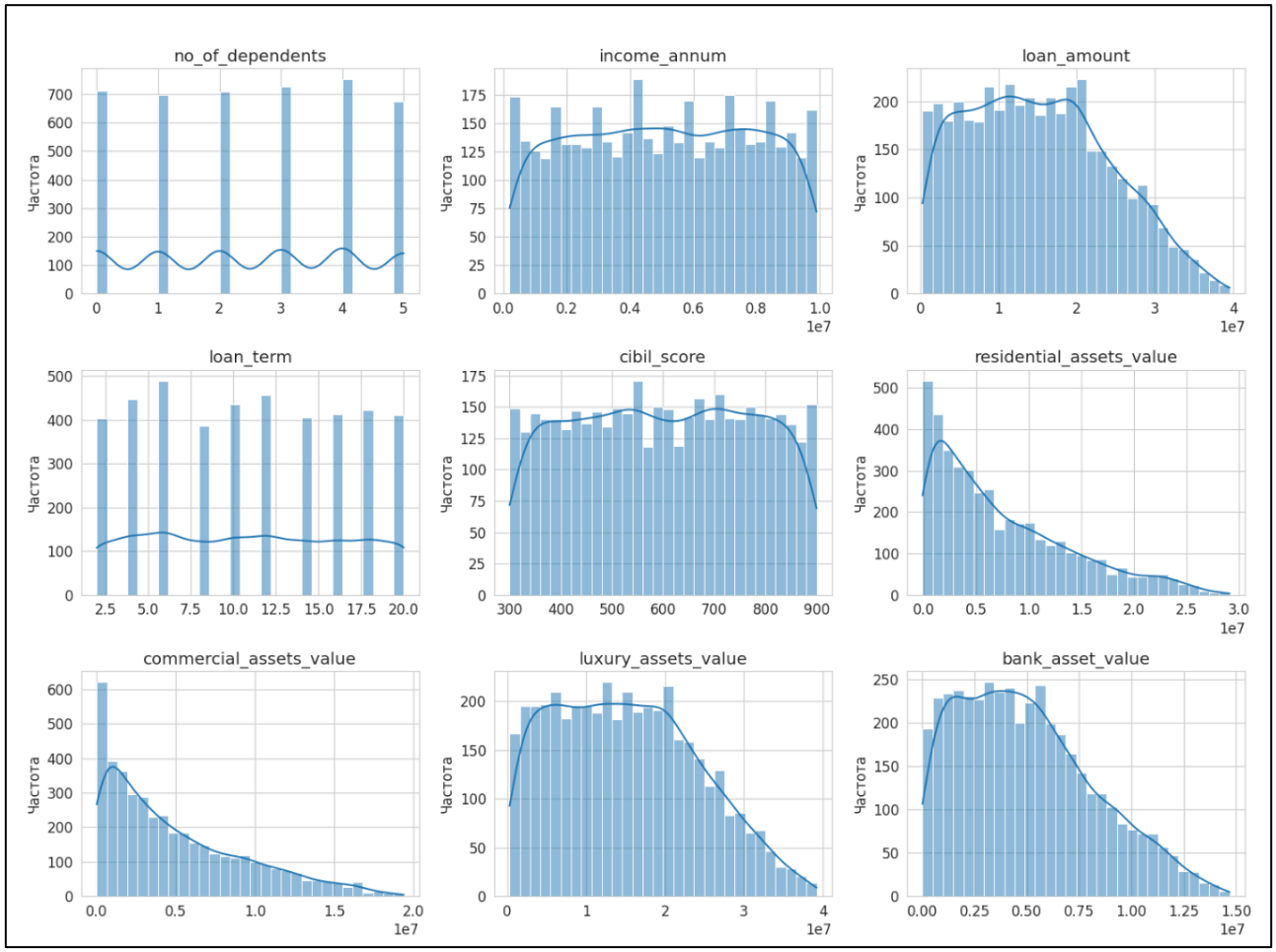

Рисунок 2.7 – Розподіл ознак

З рисунка 2.7 видно, що більшість числових ознак (таких як дохід, сума кредиту) демонструють досить рівномірні або нормальні розподіли, що сприятливо для моделювання.

Ознаки no_of_dependents та loan_term є дискретними з обмеженим набором значень, що підкреслює їхню категоріальну/порядкову природу.

Розподіл cibil_score є відносно нормальним, що підтверджує його як надійну та інформативну ознаку для кредитного скорингу.

Всі ознаки, що стосуються вартості активів (commercial_assets_value, luxury_assets_value, bank_asset_value, total_assets_value), демонструють сильний позитивний скіс. Це вказує на те, що більшість заявників мають низькі або нульові значення цих активів, тоді як невелика частина володіє дуже значними статками. Така форма розподілу вимагає уваги при виявленні викидів та інтерпретації, але масштабування даних допоможе компенсувати ці відмінності для моделей.

Візуалізуємо розподіли категоріальних ознак та їх зв’язок зі статусом кредиту (рис. 2.8).

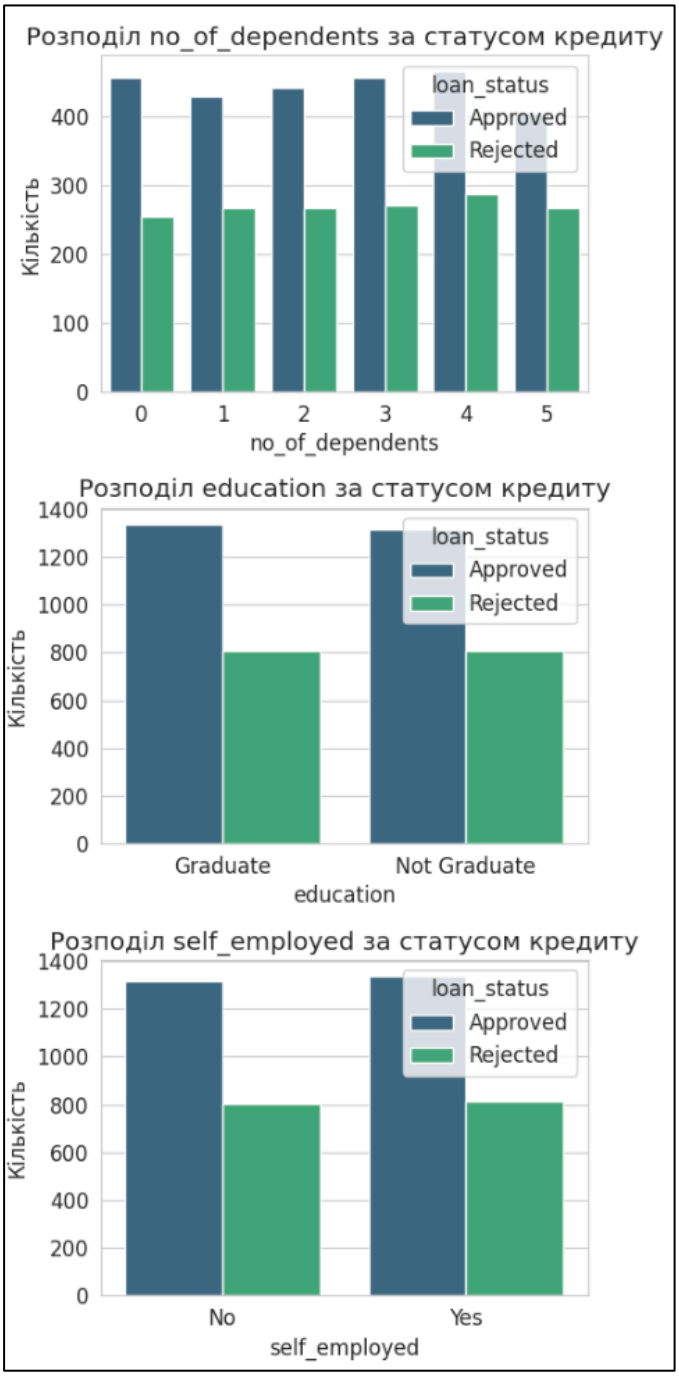

Рисунок 2.8 – Розподіли категоріальних ознак та їх зв’язок зі статусом кредиту

З рисунка 2.8 можна припустити, що ці ознаки можуть мати обмежену індивідуальну прогностичну силу, і їхній внесок у моделі, ймовірно, менший порівняно з числовими ознаками.

Далі проведемо аналіз розподілу числових ознак за статусом кредиту. На рисунку 2.9 зображені боксплоти, які дозволяють візуально оцінити розподіл значень кожної числової ознаки для двох груп: "Approved" (схвалені кредити) та "Rejected" (відхилені кредити), а також виявити потенційні відмінності у медіанах, діапазонах та наявності викидів.

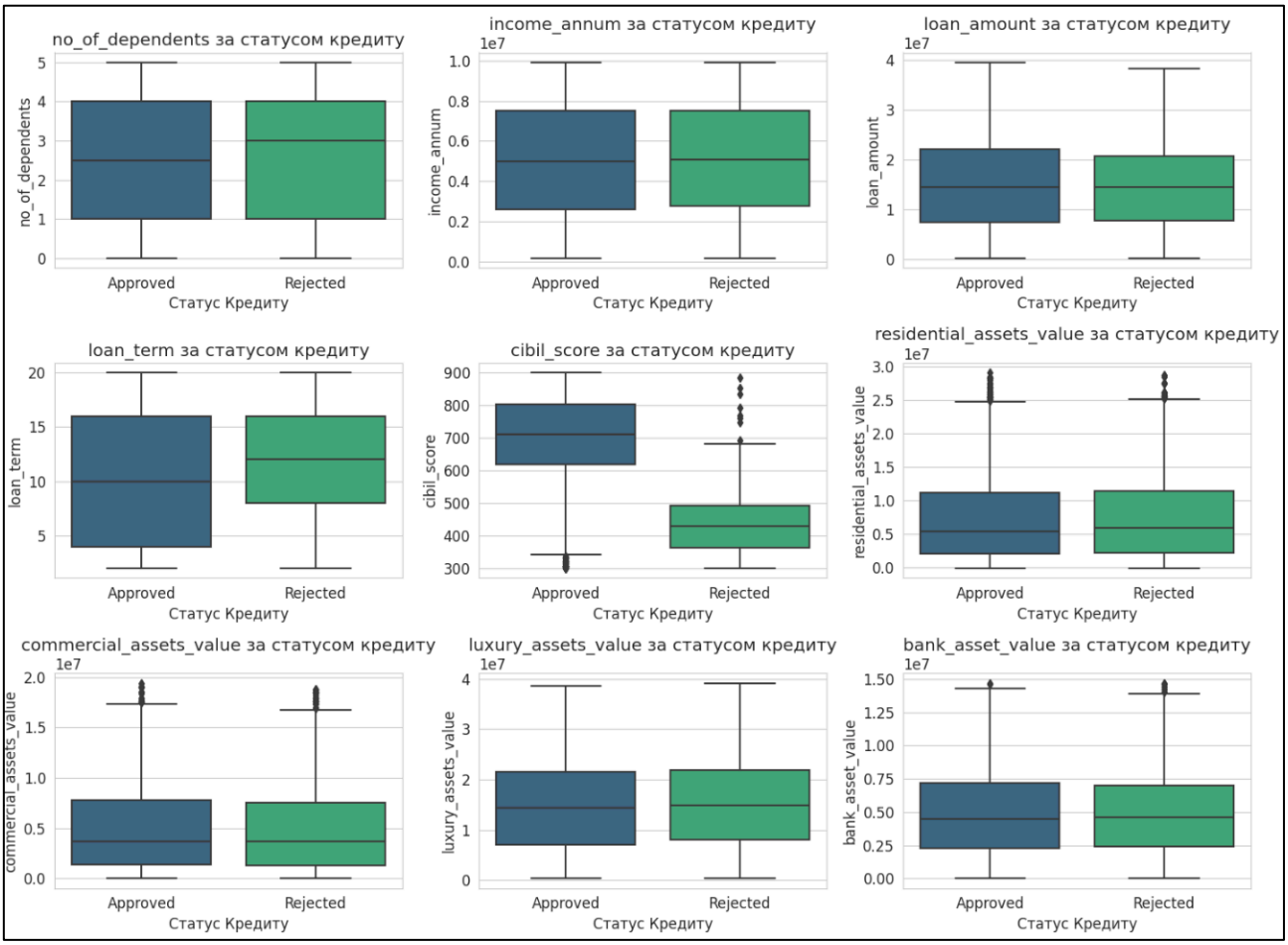

Рисунок 2.9 – Розподіл числових ознак за статусом кредиту

Бачимо, що cibil_score є надзвичайно сильною розрізняючою ознакою для статусу кредиту. Вищий CIBIL Score прямо корелює зі схваленням кредиту, тоді як нижчий – з відхиленням. Це підтверджує попередні висновки про високу кореляцію цієї ознаки.

Більшість інших числових ознак не показують чітких візуальних відмінностей у розподілах між схваленими та відхиленими кредитами. Це означає, що їхній індивідуальний внесок у прогнозування статусу кредиту,

ймовірно, є мінімальним або непрямим, якщо вони і впливають, то у комбінації з іншими ознаками.

Далі розглянемо найбільш важливі числові ознаки і побудуємо попарні розсіювання (scatterplots) та розподіли для них (рис.2.10).

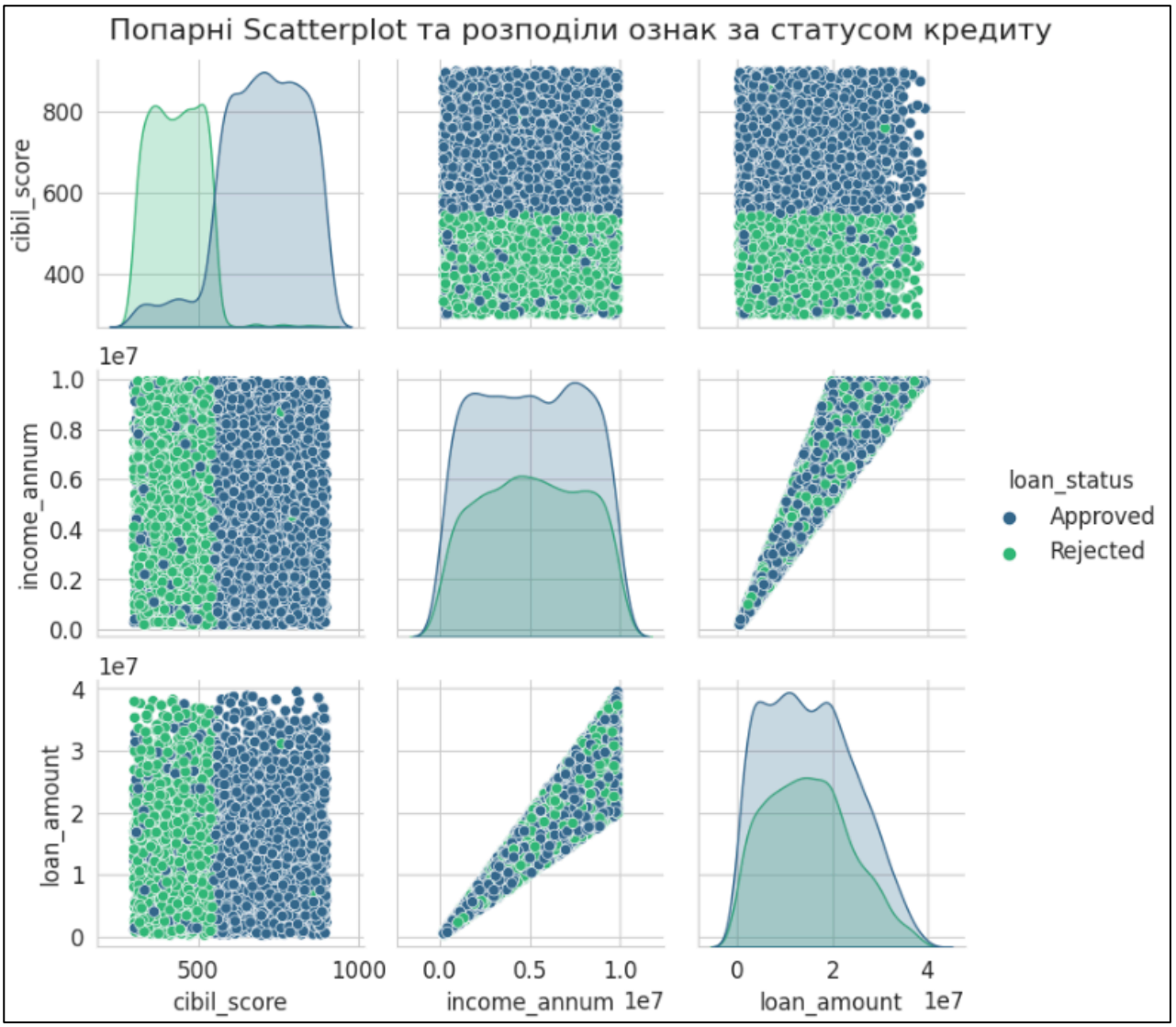

Рисунок 2.10 – Scatterplots числових ознак

Кредитний рейтинг (cibil_score) має найвищу предиктивну силу для прогнозування статусу кредиту. Income_annum та loan_amount демонструють комбінований вплив, але самостійно не є визначальними. Для побудови ефективної моделі доцільно розглядати взаємодії між ознаками та застосовувати методи обробки нелінійностей (Decision Trees, Gradient Boosting).

Розглянемо кореляційну матрицю (рис.2.11).

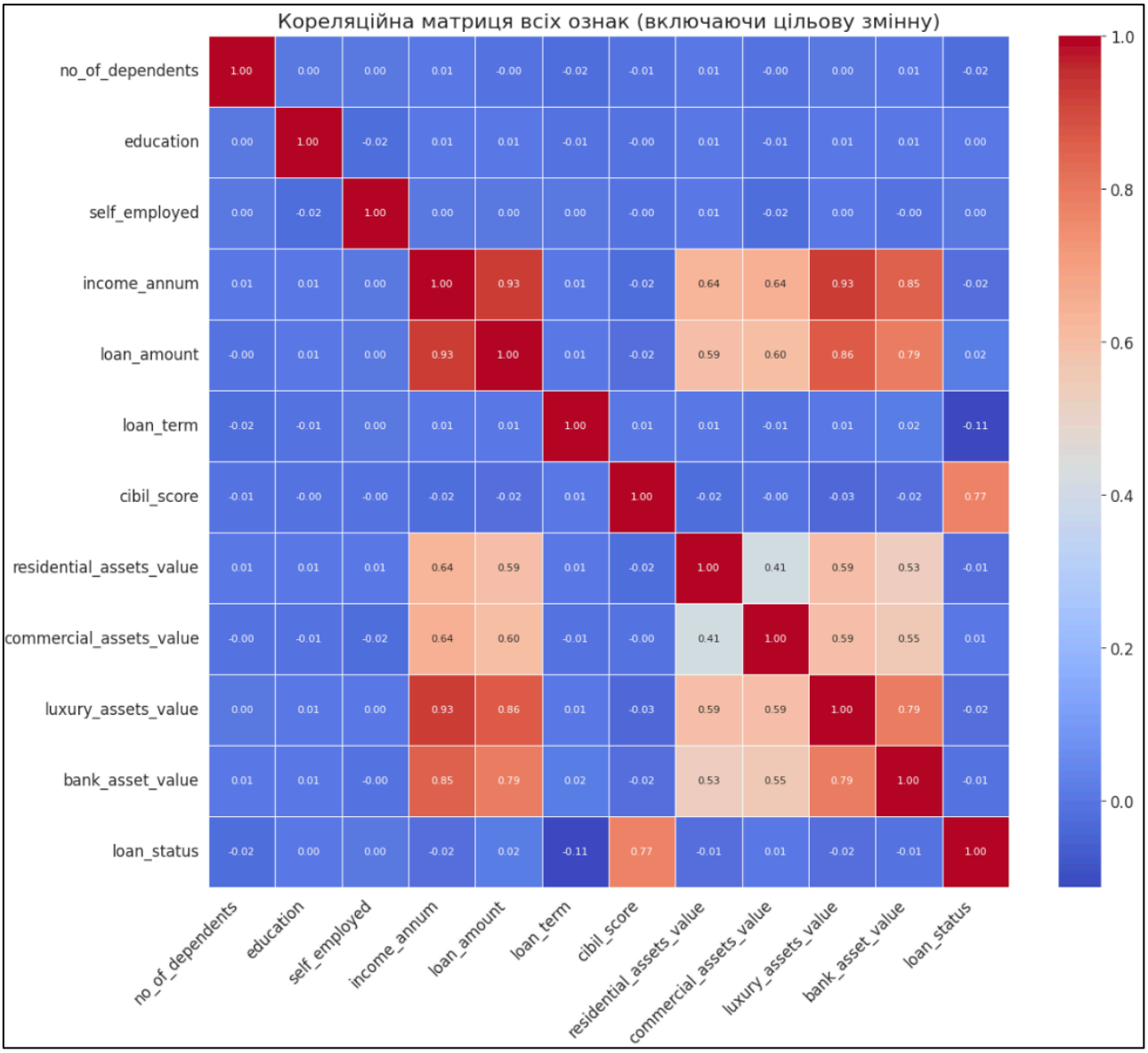

Рисунок 2.11 – Кореляційна матриця

З кореляційної матриці бачимо, що cibil_score, bank_asset_value, luxury_assets_value, commercial_assets_value, residential_assets_value є найважливішими ознаками для прогнозування loan_status з огляду на їхні сильні кореляції.

Наступний етап – підготовка даних до моделювання.

2.4 Висновки

У другому розділі було обґрунтовано вибір інструментальних засобів та архітектури IT-інфраструктури, що є критично важливими для ефективної

реалізації інформаційної технології прогнозування результатів кредитного схвалення. На основі порівняльного аналізу мов програмування та середовищ розробки, як оптимальне рішення було обрано мову Python завдяки її простоті, потужній екосистемі бібліотек для машинного навчання (Scikit-learn, XGBoost) і зручності у прототипуванні. Середовищем розробки обрано Kaggle Notebooks, що забезпечує інтерактивність, готову інфраструктуру та безкоштовний доступ до обчислювальних ресурсів (GPU/TPU).

Було побудовано та описано багаторівневу архітектуру IT-інфраструктури, яка включає рівні збирання, зберігання, обробки, моделювання, оцінки та моніторингу. Це забезпечує масштабованість, живучість, надійність та зручну інтеграцію з фінансовими системами. Визначено необхідні технології: від класичних СУБД до бібліотек візуалізації та інструментів управління IT-проєктом.

У процесі розвідувального аналізу було досліджено відкритий датасет «Loan-Approval-Prediction-Dataset». Проведено аналіз типів даних, структури, унікальності значень, виявлення дисбалансу класів та ключових ознак. Встановлено, що основними предикторами є cibil_score, а також банківські та матеріальні активи, тоді як категоріальні ознаки мають допоміжний характер. Результати аналізу підкреслили необхідність масштабування числових даних, видалення зайвих пробілів у категоріальних значеннях і врахування слабкого дисбалансу класів у моделюванні.

Отримані висновки створюють надійну основу для наступного етапу роботи – побудови та навчання моделей машинного навчання для прогнозування статусу кредиту на основі фінансових та соціальних ознак.

3 РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ДЛЯ ПРОГНОЗУВАННЯ РЕЗУЛЬТАТІВ КРЕДИТНОГО СХВАЛЕННЯ

3.1 Проєктування інформаційної технології

Проєктування інформаційної технології прогнозування результатів кредитного схвалення є ключовим етапом реалізації дослідження. На цьому етапі визначаються архітектурні складові, логіка функціонування, алгоритм обробки даних та взаємодії між підсистемами. Основна мета проєктування — створити ефективну, масштабовану та надійну інформаційну технологію, здатну обробляти фінансові й соціальні показники позичальників та формувати обґрунтовані прогнози щодо схвалення чи відмови у видачі кредиту.

Інформаційна технологія реалізується у вигляді системи, що має модульну структуру. Схема технології наведена на рисунку 3.1.

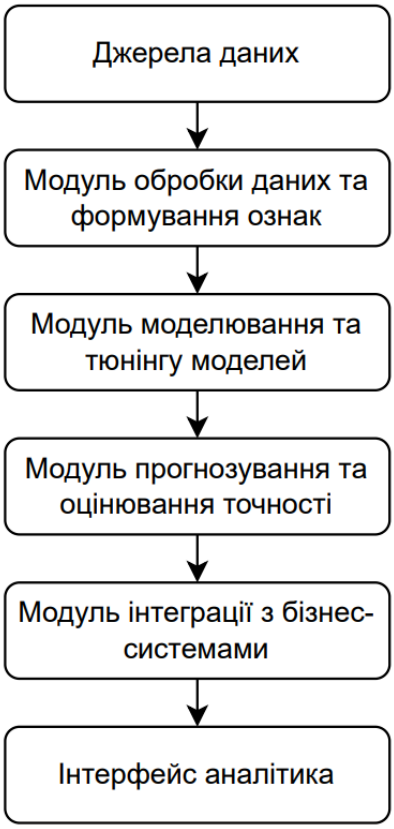

Рисунок 3.1 – Структурна схема технології

Структурна схема технології включає наступні функціональні модулі:

– Джерела даних – включають фінансові системи банку, CRM, кредитні бюро та соціальні реєстри. Дані надходять у вигляді CSV-файлів або API-викликів.

– Модуль обробки даних – очищає та нормалізує дані, виконує масштабування та кодування ознак, усуває пропущені значення та підозрілі записи.

– Модуль формування ознак – здійснює трансформацію вихідних показників у придатну для моделювання форму: категоріальні ознаки переводяться у числові, створюються нові ознаки.

– Модуль моделювання – включає вибір алгоритмів машинного навчання (Logistic Regression, Random Forest, Gradient Boosting), їх навчання та валідацію.

– Модуль тюнінгу моделей – відповідає за автоматизований підбір гіперпараметрів для досягнення найкращих результатів.

– Модуль оцінювання якості – проводить обчислення метрик якості моделі: Accuracy, Precision, Recall, F1-score, ROC AUC.

– Модуль прогнозування – застосовує навчені моделі до нових даних для визначення ймовірності схвалення кредиту.

– Модуль інтеграції з бізнес-системами – дозволяє вбудувати прогноз у фронт-офіс банку, систему видачі кредитів або мобільний застосунок.

– Інтерфейс аналітика – візуалізація результатів у вигляді графіків, таблиць, оцінки ризиків для подальшого аналізу аналітиками банку.

Для забезпечення ефективного функціонування інформаційної технології було спроєктовано алгоритм, представлений на блок-схемі на рисунку 3.2.

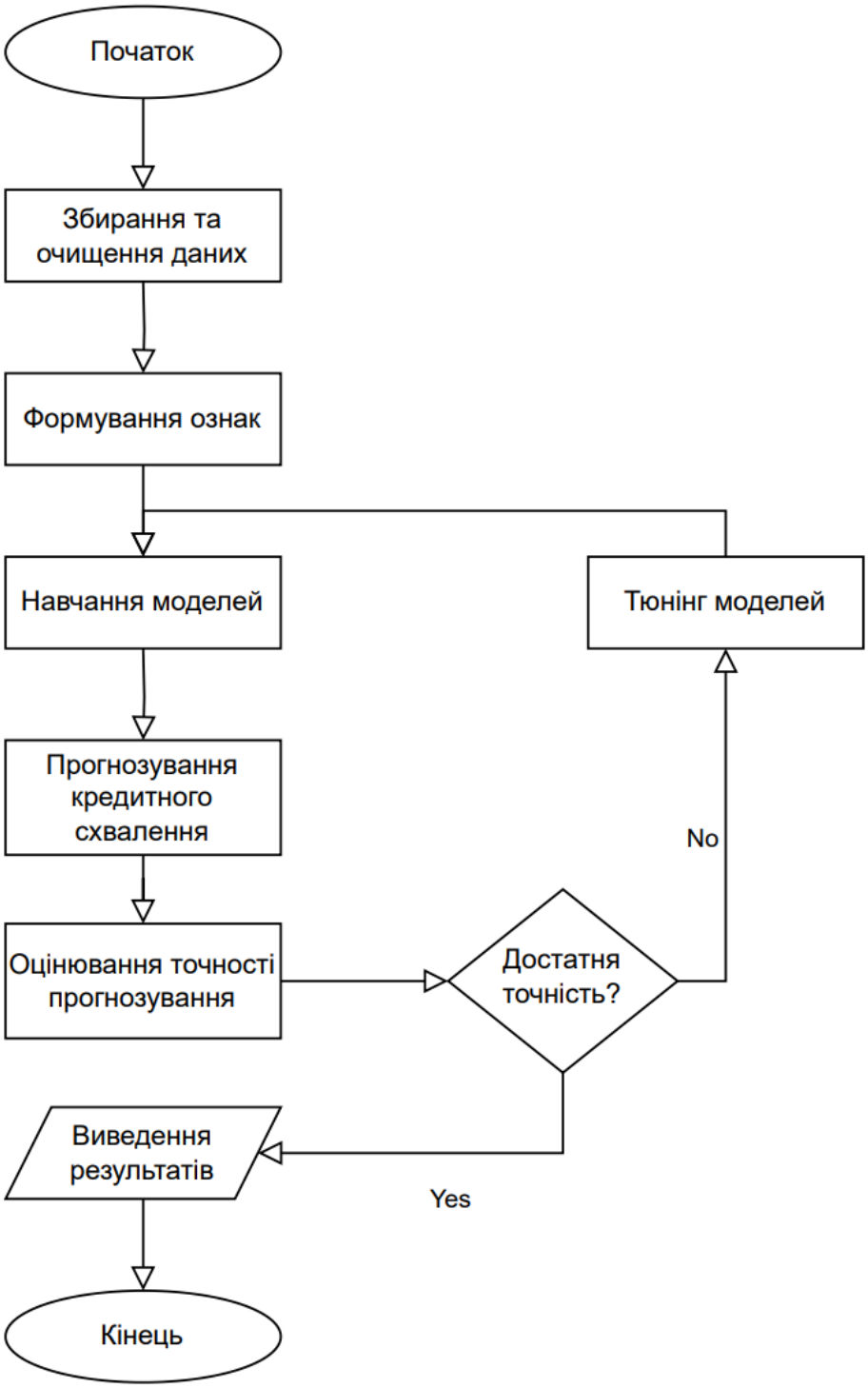

Рисунок 3.2 – Блок-схема алгоритму інформаційної технології

Алгоритм роботи інформаційної технології склдадається з наступних кроків:

1. Початок — ініціалізація системи.
2. Збирання та очищення даних — імпорт інформації з різних джерел, перевірка на пропущені значення, аномалії, форматування.

3. Формування ознак — конструювання релевантних змінних, нормалізація, кодування, масштабування.

4. Навчання моделей — побудова класифікаційних моделей з використанням тренувальних даних.

5. Оцінювання точності — перевірка якості моделі на тестових даних із застосуванням метрик.

6. Перевірка точності — якщо модель не досягла потрібної точності, виконується тюнінг гіперпараметрів і повторне навчання.

7. Прогнозування результату — після досягнення цільових показників модель застосовується до нових кейсів.

8. Виведення результатів — результати виводяться для аналітика або інтегруються у банківське ПЗ.

9. Кінець — завершення виконання алгоритму.

3.2 Підготовка даних для моделювання

Оскільки у нас немає пропущених значень, а також явних викидів, то для початку проведено кодування категоріальних ознак education, self_employed та цільової змінної loan_status. Перетворимо їх на числові (0 та 1), що є коректним для подальшого моделювання. Враховуючи, що вони бінарні, Label Encoding буде простішим і ефективнішим. (рис. 3.3).

```
# Перетворення цільової змінної 'loan_status' на числові значення (0 і 1)
# 'Approved' -> 1, 'Rejected' -> 0
df_processed['loan_status'] = df_processed['loan_status'].map({'Approved': 1, 'Rejected': 0})
print("Цільова змінна 'loan_status' перетворена: 'Approved' -> 1, 'Rejected' -> 0.")

# Кодування бінарних категоріальних ознак за допомогою Label Encoding
# 'education': 'Graduate' -> 1, 'Not Graduate' -> 0
df_processed['education'] = df_processed['education'].map({'Graduate': 1, 'Not Graduate': 0})
print("Ознака 'education' перетворена: 'Graduate' -> 1, 'Not Graduate' -> 0.")

# 'self_employed': 'Yes' -> 1, 'No' -> 0
df_processed['self_employed'] = df_processed['self_employed'].map({'Yes': 1, 'No': 0})
print("Ознака 'self_employed' перетворена: 'Yes' -> 1, 'No' -> 0.")
```

Рисунок 3.3 – Розбиття датасету на трейнові та тестові дані

Також видаляємо 'loan_id', оскільки це ідентифікатор і не є ознакою для моделювання (рис.3.4).

```
# Видалення 'loan_id', оскільки це ідентифікатор і не є ознакою для моделювання
if 'loan_id' in df_processed.columns:
    df_processed = df_processed.drop('loan_id', axis=1)
    print("Колонка 'loan_id' видалена.")
```

Рисунок 3.4 – Видалення «loan_id»

Далі потрібно зробити масштабування ознак. StandardScaler (стандартизація) добре працює з алгоритмами, що базуються на відстанях (SVM), а також з тими, що припускають нормальний розподіл (Логістична Регресія) (рис. 3.5).

```
scaler = StandardScaler()
df_processed[numerical_features_for_scaling] = scaler.fit_transform(df_processed[numerical_features_for_scaling])
print(f"Числові ознаки {numerical_features_for_scaling} масштабовано за допомогою StandardScaler.")
```

Рисунок 3.5 – Створення попередніх процесорів для числових та категоріальних ознак

Перед початком моделювання необхідно розділити оброблений датасет на дві частини: тренувальний набір (training set), який використовується для навчання моделі, та тестовий набір (test set), що призначений для незалежної оцінки її узагальнюючої здатності на нових, невідомих для моделі даних.

Фіксація random_state=42 гарантує відтворюваність результатів розбиття. Кожен запуск коду дасть однакове розбиття, що є критично важливим для порівняльного аналізу моделей.

Параметр stratify=y забезпечує, що відсоткове співвідношення класів (Approved та Rejected) у тренувальному та тестовому наборах буде максимально наближеним до оригінального розподілу в повному датасеті. Це запобігає ситуаціям, коли, наприклад, тестовий набір може містити аномально мало прикладів одного з класів, що призведе до некоректної оцінки моделі (рис. 3.6).

```python
# Визначення ознак (X) та цільової змінної (y)
X = df_processed.drop('loan_status', axis=1)
y = df_processed['loan_status']

# Розбиття на тренувальний та тестовий набори
# Використовуємо stratify=y для збереження пропорцій класів у тренувальному та тестовому наборах,
# що важливо через дисбаланс цільової змінної.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)

print(f"Розмір тренувального набору (X_train, y_train): {X_train.shape}, {y_train.shape}")
print(f"Розмір тестового набору (X_test, y_test): {X_test.shape}, {y_test.shape}")
print(f"Розподіл класів у тренувальному наборі:\n{y_train.value_counts(normalize=True)}")
print(f"Розподіл класів у тестовому наборі:\n{y_test.value_counts(normalize=True)}")
```

```
--- 4.1 Розбиття даних на тренувальний та тестовий набори ---
Розмір тренувального набору (X_train, y_train): (2988, 11), (2988,)
Розмір тестового набору (X_test, y_test): (1281, 11), (1281,)
Розподіл класів у тренувальному наборі:
loan_status
1    0.622155
0    0.377845
Name: proportion, dtype: float64
Розподіл класів у тестовому наборі:
loan_status
1    0.62217
0    0.37783
```

Рисунок 3.6 – Розбиття даних на тренувальний та тестовий набори

Виведення розмірів X_train, y_train, X_test, y_test підтверджує успішне розбиття, stratify=y ефективно зберіг пропорції класів.

Таким чином, датасет повністю підготовлений і готовий до етапу побудови та оцінки моделей.

3.3 Побудова та навчання моделей машинного навчання

Переходимо до етапу вибору та ініціалізації моделей машинного навчання. На цьому етапі ми:

– Використовуємо 6 класифікаційних моделей, включаючи Random Forest, логістичну регресію та Gradient Boosting.

– Оцінюємо ефективність моделей на тренувальних та тестових даних за різними метриками.

– Аналізуємо на предмет недонавчання/перенавчання.

Першим кроком є ініціалізація моделей (рис. 3.7).

```
# Ініціалізація моделей
models = {
    'Logistic Regression': LogisticRegression(random_state=42, solver='liblinear'),
    'Decision Tree': DecisionTreeClassifier(random_state=42),
    'Random Forest': RandomForestClassifier(random_state=42),
    'Gradient Boosting': GradientBoostingClassifier(random_state=42),
    'Support Vector Machine': SVC(random_state=42, probability=True), # probability=True для ROC AUC
    'K-Nearest Neighbors': KNeighborsClassifier()
}
```

Рисунок 3.7 – Ініціалізація моделей

Вибрано широкий спектр класифікаційних моделей:

– Logistic Regression: Проста, лінійна модель, що слугує хорошою базою. solver='liblinear' підходить для невеликих датасетів і бінарної класифікації.

– Decision Tree: Проста, інтерпретована модель, схильна до перенавчання.

– Random Forest: Ансамбль дерев рішень, що зменшує перенавчання та покращує узагальнення.

– Gradient Boosting: Ще один потужний ансамблевий метод, що послідовно виправляє помилки попередніх моделей.

– Support Vector Machine (SVC): Потужний алгоритм, що шукає оптимальну гіперплощину для розділення класів. probability=True дозволяє отримувати ймовірності прогнозу, що необхідно для метрики ROC AUC.

– K-Nearest Neighbors (K-NN): Непараметричний алгоритм, який класифікує нові точки даних, виходячи з більшості голосів її "k" найближчих сусідів у тренувальному наборі.

Кожна модель навчається на тренувальному наборі. Прогнози робляться як на тренувальному, так і на тестовому наборах. Це дозволяє оцінити як якість навчання (на тренувальному), так і узагальнюючу здатність (на тестовому). Для метрики ROC AUC потрібні ймовірності приналежності до класу. Це дозволяє оцінити здатність моделі ранжувати приклади за ймовірністю приналежності до позитивного класу (рис. 3.8).

```python
for name, model in models.items():
    print(f"\nНавчання моделі: {name}...")
    model.fit(X_train, y_train)

    # Прогнози на тренувальному та тестовому наборах
    y_train_pred = model.predict(X_train)
    y_test_pred = model.predict(X_test)

    # Прогнози ймовірностей для ROC AUC (якщо модель підтримує)
    y_train_proba = model.predict_proba(X_train)[:, 1] if hasattr(model, 'predict_proba') else None
    y_test_proba = model.predict_proba(X_test)[:, 1] if hasattr(model, 'predict_proba') else None

    # Оцінка метрик на тренувальному наборі
    train_accuracy = accuracy_score(y_train, y_train_pred)
    train_precision = precision_score(y_train, y_train_pred)
    train_recall = recall_score(y_train, y_train_pred)
    train_f1 = f1_score(y_train, y_train_pred)
    train_roc_auc = roc_auc_score(y_train, y_train_proba) if y_train_proba is not None else 'N/A'

    # Оцінка метрик на тестовому наборі
    test_accuracy = accuracy_score(y_test, y_test_pred)
    test_precision = precision_score(y_test, y_test_pred)
    test_recall = recall_score(y_test, y_test_pred)
    test_f1 = f1_score(y_test, y_test_pred)
    test_roc_auc = roc_auc_score(y_test, y_test_proba) if y_test_proba is not None else 'N/A'
```

Рисунок 3.8 – Навчання моделей

Як бачимо з рисунка 3.8, обчислюється низка стандартних метрик класифікації для обох наборів даних:

– Accuracy (Точність): Загальна частка правильних прогнозів. Добра для збалансованих класів, але менш інформативна при сильному дисбалансі.

– Precision (Точність): Частка істинно позитивних прогнозів серед усіх, які модель передбачила як позитивні. Важлива, коли вартість хибнопозитивних спрацювань висока (наприклад, схвалення неплатоспроможного кредиту).

– Recall (Повнота/Чутливість): Частка істинно позитивних прогнозів серед усіх фактично позитивних випадків. Важлива, коли вартість хибнонегативних спрацювань висока (наприклад, відмова платоспроможному клієнту).

– F1-Score: Гармонійне середнє між Precision та Recall. Корисна, коли потрібен баланс між точністю та повнотою, особливо при дисбалансі класів.

– ROC AUC (Площа під кривою робочих характеристик): Дуже надійна метрика для оцінки якості бінарної класифікації, особливо при дисбалансі класів. Вона показує здатність моделі розрізняти класи на різних

порогах класифікації. Значення 0.5 означає випадкове вгадування, 1.0 – ідеальний класифікатор.

Генерується детальний звіт про класифікацію, який включає такі метрики, як Accuracy, Precision, Recall, F1-score, ROC-AUC.

Precision (точність) – частка правильно передбачених позитивних випадків серед усіх передбачених позитивних.

Recall (повнота/чутливість) – частка правильно передбачених позитивних випадків серед усіх фактично позитивних випадків.

F1-score - гармонійне середнє Precision та Recall, що є хорошою метрикою для незбалансованих наборів даних.

AUC-ROC є надійною метрикою точності класифікатора, яка оцінює його здатність розрізняти класи, незалежно від порогу класифікації. Вона особливо корисна для незбалансованих наборів даних. Високе значення AUC-ROC (ближче до 1) вказує на кращу розділювальну здатність моделі.

Обчислюється матриця похибок (confusion matrix), яка є табличним представленням кількості True Positives (TP), True Negatives (TN), False Positives (FP) та False Negatives (FN).

Матриця похибок є чудовим інструментом для візуалізації точності класифікаційної моделі. Вона дозволяє швидко побачити, які типи помилок (наприклад, помилково прогнозовані позитивні рішення – FP, або пропущені фактичні відмови – FN) модель робить, що є критично важливим для розуміння її практичних наслідків.

Також виконується оцінка моделей на тренувальних та тестових даних для виявлення перенавчання (overfitting) або недонавчання (underfitting).

На рисунках 3.9-3.11 наведені детальні результати деяких моделей.

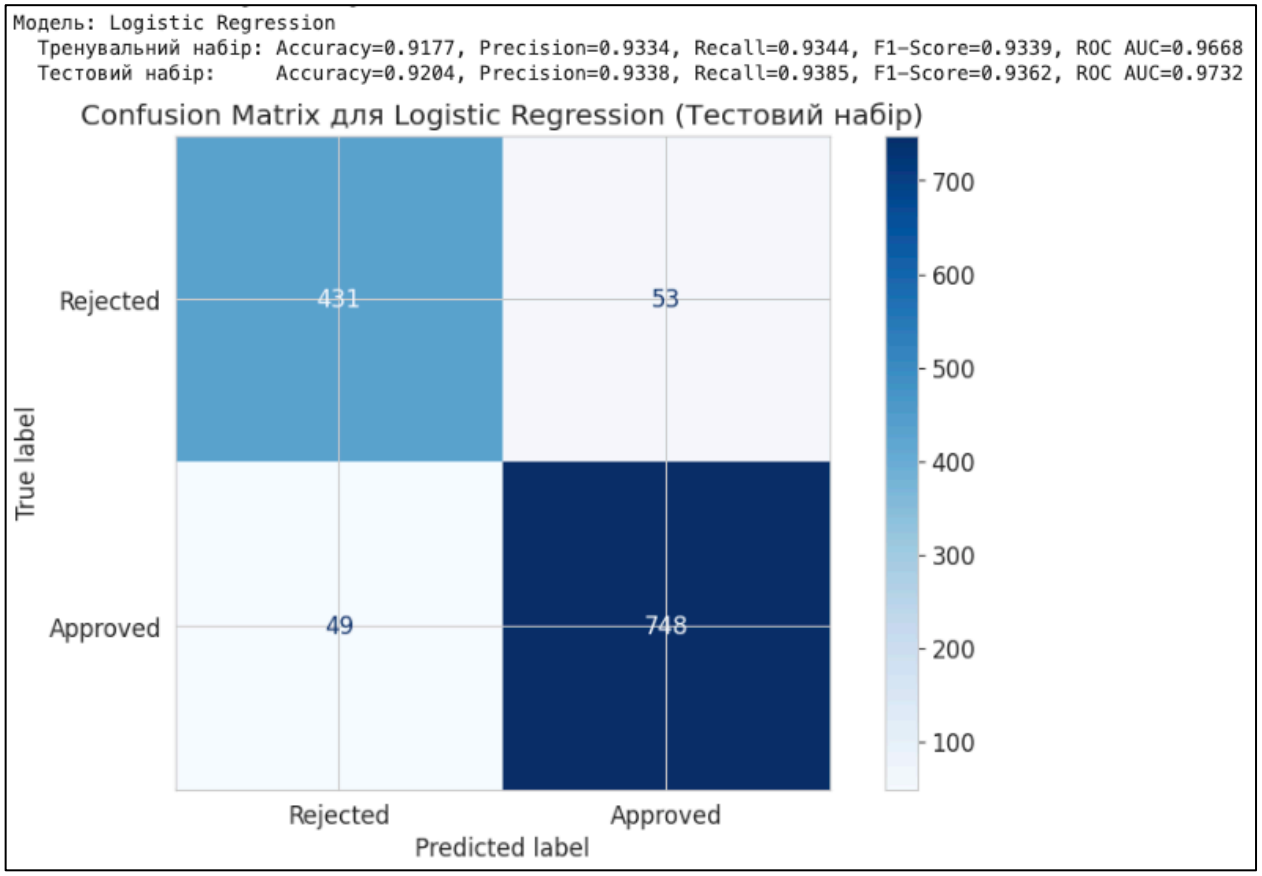

Рисунок 3.9 – Оцінка моделі Logistic Regression

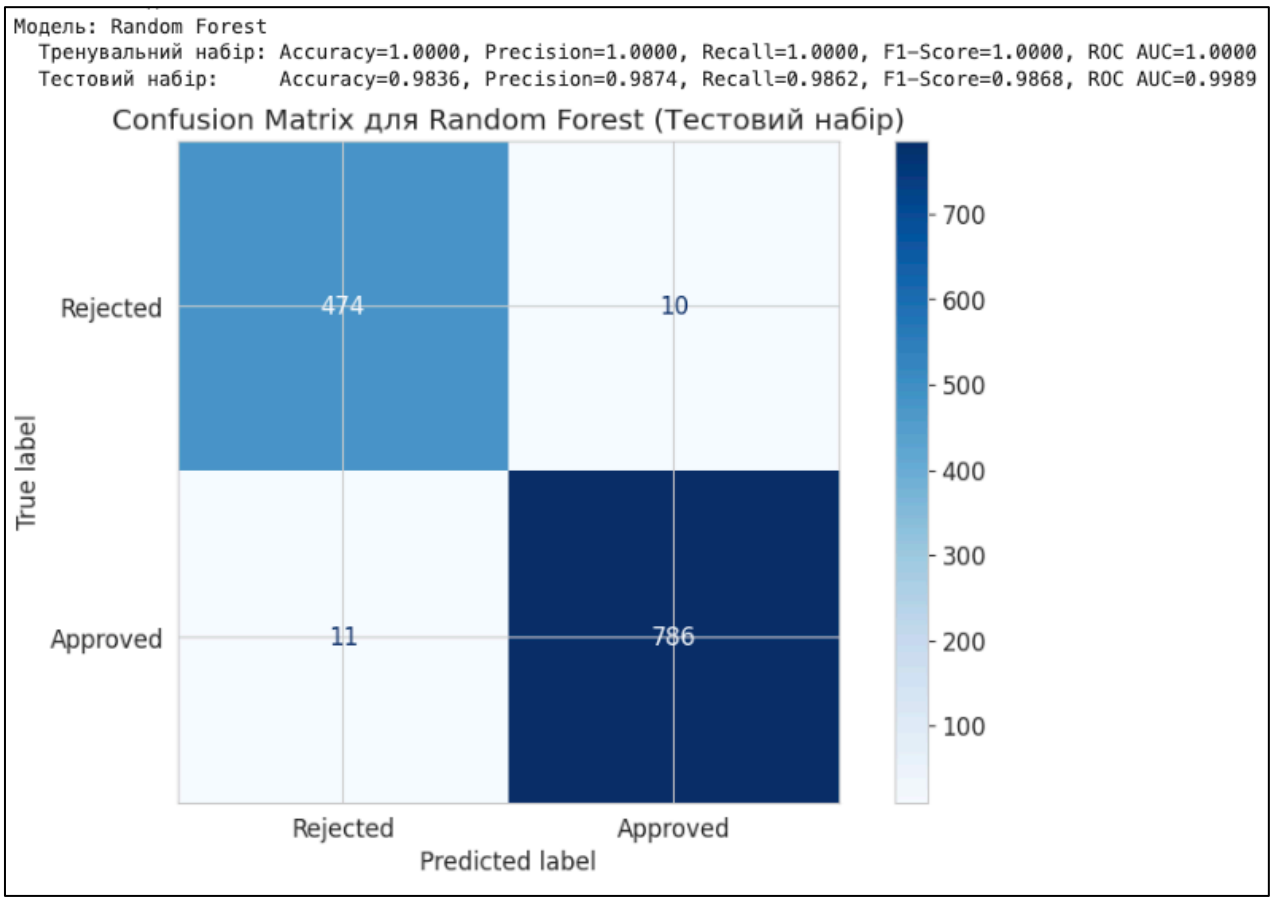

Рисунок 3.10 – Оцінка моделі Random Forest

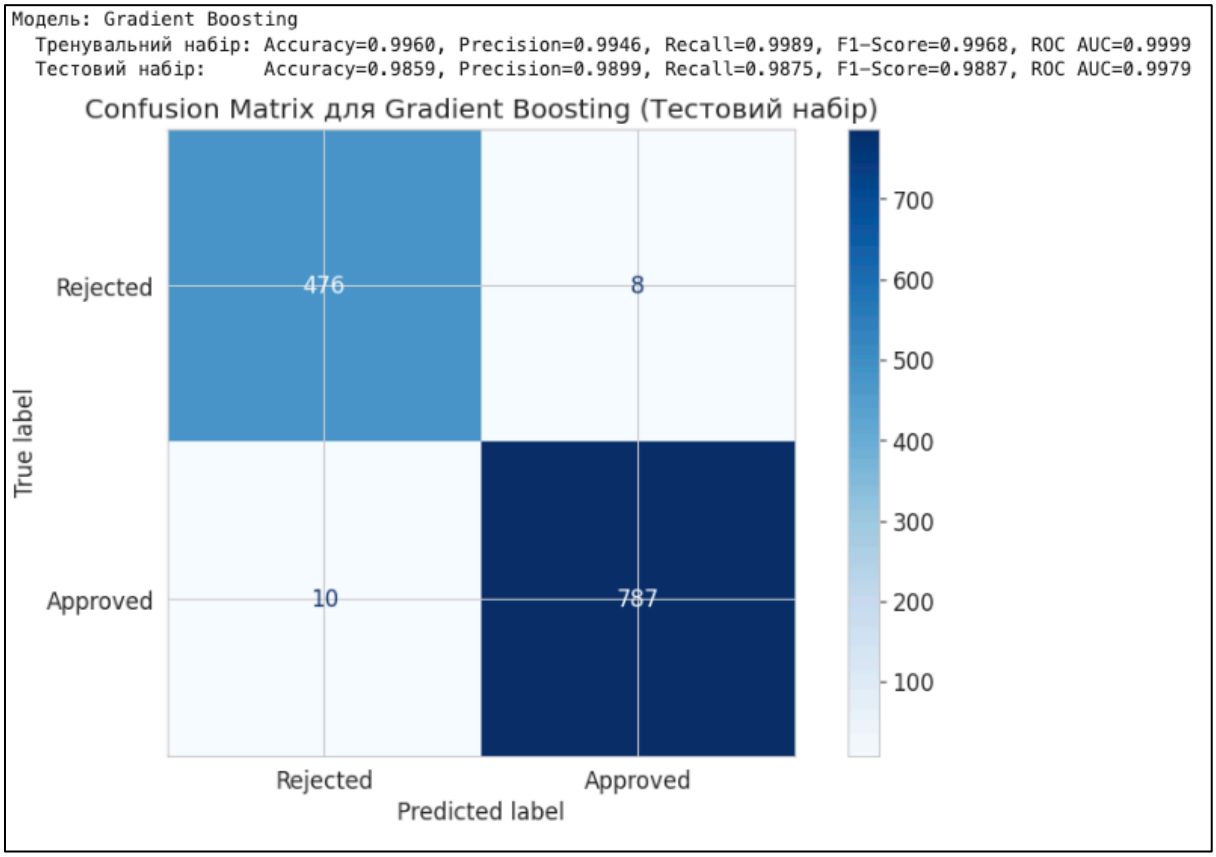

Рисунок 3.11 – Оцінка моделі Gradient Boosting

Переходимо до етапу фінального порівняння точності всіх розроблених моделей на тренувальних та тестових даних. Цей зведений аналіз є вирішальним для вибору оптимальної моделі для завдання прогнозування кредитного схвалення, а також для діагностики потенційного перенавчання чи недонавчання (рис. 3.13).

Модель	Accuracy (Train)	Accuracy (Test)
Logistic Regression	0.92	0.92
Decision Tree	1	0.98
Random Forest	1	0.98
Gradient Boosting	0.99	0.99
Support Vector Machine	0.95	0.94
K-Nearest Neighbors	0.94	0.88

Рисунок 3.13 – Порівняння точності класифікаційних моделей

З огляду на метрики на тестовому наборі (особливо ROC AUC та F1-Score), Gradient Boosting є основним кандидатом на найкращу модель для цієї задачі. Він демонструє найкращу узагальнюючу здатність та найменший ступінь перенавчання серед усіх протестованих алгоритмів.

Для повноцінного аналізу також проведемо аналіз важливості ознак (Feature Importance). Розуміння того, які ознаки найбільше впливають на прогнози моделі має величезне практичне значення для розробки стратегій схвалення позик.

Побудуємо діаграму важливості ознак (рис.3.14).

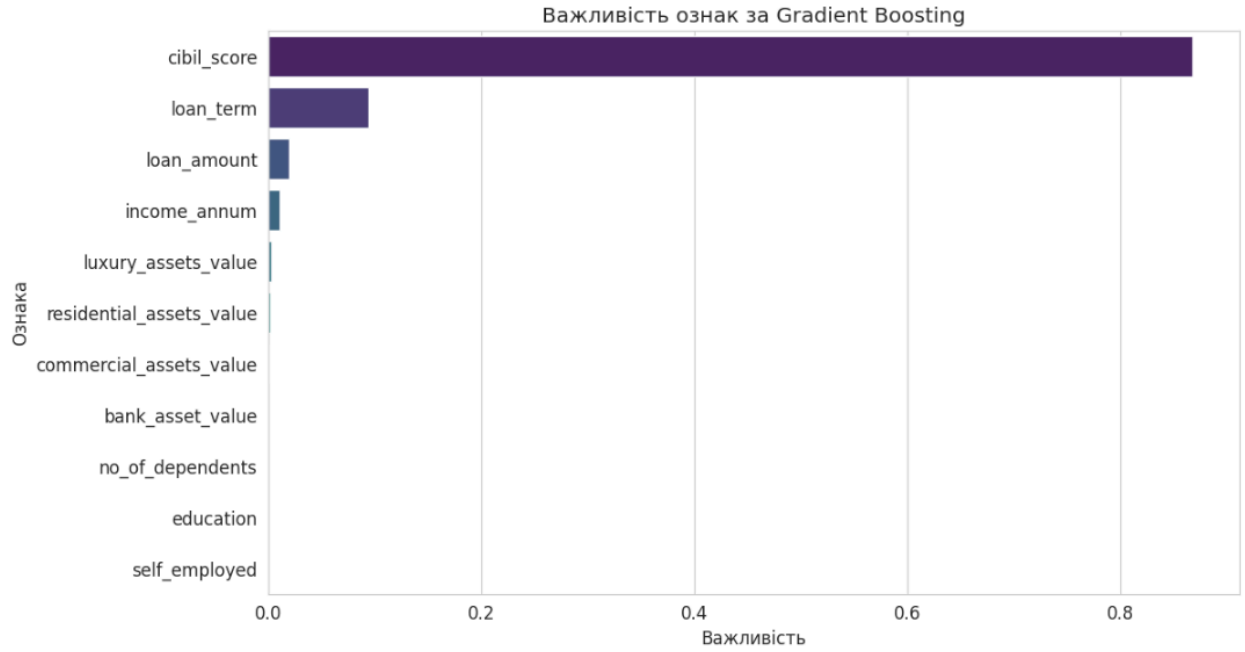

Рисунок 3.14 – Важливість ознак моделі Gradient Boosting

Наведемо окремо топ 5 найважливіших ознак для моделі Gradient Boosting (рис.3.15).

Топ 5 ознак за Gradient Boosting:

	Feature	Importance
6	cibil_score	0.867404
5	loan_term	0.094317
4	loan_amount	0.020008
3	income_annum	0.011015
9	luxury_assets_value	0.003034

Рисунок 3.15 – Топ 5 найважливіших ознак для моделі Gradient Boosting

Очевидно, що cibil_score є домінуюче важливою ознакою для прогнозування кредитного схвалення. Її важливість становить майже 87% від загальної важливості всіх ознак, що є винятково високим показником.

Це повністю корелює з попереднім аналізом кореляційної матриці, де cibil_score мала найсильнішу позитивну кореляцію з loan_status. Модель Gradient Boosting підтверджує, що кредитний бал є практично єдиним визначальним фактором для прийняття рішення про кредит. Це вказує на те, що якість кредитної історії та платіжної дисципліни є першочерговою для банку.

Будь-які подальші зусилля з покращення моделі повинні починатися з ретельного вивчення цієї ознаки. Якщо дані cibil_score є неточними або містять аномалії, це матиме величезний вплив на точність моделі.

Термін кредиту є другою за важливістю ознакою, хоча її вплив значно менший, ніж у cibil_score (близько 9.4% від загальної важливості).

У кореляційній матриці loan_term мала дуже слабку негативну кореляцію (-0.11) з loan_status. Це приклад того, як кореляційна матриця (яка показує лінійні зв'язки) може не повністю відображати важливість ознаки в складних нелінійних моделях, таких як Gradient Boosting. Можливо, модель виявила нелінійну залежність або взаємодію loan_term з іншими ознаками, що робить її більш значущою, ніж здається з лінійного аналізу. Довший термін кредиту може асоціюватися з більшим ризиком або іншими умовами, що впливають на схвалення.

Інші ознаки, які не увійшли в топ-5, можливо, можна було б виключити з моделі без суттєвої втрати точності, або ж вони важливі лише у дуже специфічних, тонких взаємодіях, які не були виявлені цим методом.

3.4 Висновки

У третьому розділі було виконано проєктування інформаційної технології прогнозування результатів кредитного схвалення, яке охоплює побудову архітектури, визначення логіки функціонування, розробку та оцінку моделей, забезпечення їхньої інтеграції у реальні системи. Ретельне структурування

модулів дозволяє створити гнучку, ефективну та масштабовану систему, яка відповідає сучасним вимогам фінансового сектору щодо точності, надійності та прозорості прийняття рішень.

Було повністю підготовлено вхідні дані для моделювання: видалено технічні ознаки, закодовано бінарні категоріальні змінні, масштабовано числові показники та реалізовано коректне розбиття на тренувальний і тестовий набори з урахуванням розподілу класів. Завдяки використанню LabelEncoder та StandardScaler було забезпечено відповідність даних вимогам більшості алгоритмів машинного навчання.

Було проведено навчання шести різних класифікаційних моделей, що дозволило здійснити повноцінне порівняння якості прогнозування за допомогою набору стандартних метрик: Accuracy, Precision, Recall, F1-Score та ROC AUC. Результати аналізу підтвердили перевагу ансамблевих методів, зокрема Gradient Boosting (Accuracy=0,99), який продемонстрував найкращу узагальнюючу здатність без ознак перенавчання.

Додатково було оцінено важливість ознак, що впливають на прийняття рішень моделлю. Кредитний рейтинг (cibil_score) виявився домінуючим фактором, що підтверджує його центральне значення у системах кредитного скорингу. Важливість цієї ознаки в моделі Gradient Boosting склала понад 85%, що узгоджується з попереднім аналітичним дослідженням.

Таким чином, на основі ретельно підготовлених даних і використання сучасних методів машинного навчання було створено ефективну прогностичну модель для задачі кредитного схвалення, що відповідає сучасним вимогам до якості, інтерпретованості та практичної придатності в реальному банківському середовищі.

ВИСНОВКИ

У ході виконання бакалаврської кваліфікаційної роботи досягнуто поставлену мету — підвищення точності прогнозування результатів кредитного схвалення шляхом розроблення ефективної інформаційної технології, що базується на фінансових та соціальних показниках із використанням методів машинного навчання.

У процесі дослідження було розв’язано всі поставлені задачі. Проведено системний аналіз предметної області, охарактеризовано об’єкт дослідження як складну інформаційну систему, що функціонує в умовах невизначеності. Розглянуто методологічні та технічні аспекти кредитного скорингу, визначено найбільш релевантні алгоритми машинного навчання та сформовано обґрунтовану вибірку моделей для реалізації.

Здійснено вибір і обґрунтування IT-інфраструктури, оптимальної для реалізації поставленого завдання. Як основні інструменти обрано мову програмування Python, середовище розробки Kaggle Notebooks та бібліотеки Scikit-learn і XGBoost, що забезпечують гнучкість, продуктивність і високу точність моделювання. Побудовано архітектуру інформаційної технології з урахуванням вимог до надійності, масштабованості, живучості та зручності інтеграції з банківськими системами.

Проведено розвідувальний аналіз відкритого датасету, підготовлено вхідні дані до моделювання шляхом очищення, кодування, масштабування та стратифікованого розбиття. Це забезпечило стабільність та узгодженість моделей на всіх етапах навчання.

На завершальному етапі було розроблено та протестовано декілька моделей машинного навчання для бінарної класифікації. Найкращі результати продемонстрував алгоритм Gradient Boosting, досягнувши високої точності (Accuracy=0.99) та найкращих показників за метриками F1-score і ROC AUC. Аналіз важливості ознак підтвердив, що найвпливовішим фактором для прийняття рішень є кредитний рейтинг (cibil_score), що є логічним і узгодженим із фінансовою практикою.

Таким чином, реалізована інформаційна технологія відповідає вимогам сучасних фінансових установ до точності, надійності та адаптивності, що свідчить про успішне виконання мети та задач дослідження. Отримані результати мають практичну цінність і можуть бути інтегровані в системи прийняття рішень у сфері кредитування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A. Gupta, V. Pant, S. Kumar and P. K. Bansal, "Bank Loan Prediction System using Machine Learning," *2020 9th International Conference System Modeling and Advancement in Research Trends (SMART)*, Moradabad, India, 2020, pp. 423-426, doi: 10.1109/SMART50582.2020.9336801.

2. M. Z. Hussain *et al.*, "Bank Loan Prediction System Using Machine Learning Models," *2024 IEEE 9th International Conference for Convergence in Technology (I2CT)*, Pune, India, 2024, pp. 1-5, doi: 10.1109/I2CT61223.2024.10543786.

3. Haque, F M Ahosanul & Hassan, Md. Mahedi. (2024). Bank Loan Prediction Using Machine Learning Techniques. doi:10.48550/arXiv.2410.08886.

4. Haque, F. M. A., & Hassan, Md. M. (2024). Bank Loan Prediction Using Machine Learning Techniques. *American Journal of Industrial and Business Management*, 14, 1690-1711. doi: 10.4236/ajibm.2024.1412085.

5. Logistic Regression Explained in 7 Minutes. URL: https://medium.com/data-science/logistic-regression-explained-in-7-minutes-f648bf44d53e (дата звернення: 22.05.2025)

6. Master the Art of Feature Selection: Turbocharge Your Data Analysis with LDA! URL: https://medium.com/@tushar.babbar08/master-the-art-of-feature-selection-turbocharge-your-data-analysis-with-lda-c021065ab136 (дата звернення: 22.05.2025)

7. Credit Scoring Prediction. URL: https://medium.com/@m.ariefrachmaann/credit-scoring-prediction-acd1f576fd86 (дата звернення: 22.05.2025).

8. Yousheng Zhou. A privacy-preserving logistic regression-based diagnosis scheme for digital healthcare. *Elsevier*, 2023, 63-73p. URL: https://research.google/pubs/logistic-regression-the-importance-of-being-improper/.

9. Support Vector Machines (SVM) In Machine Learning Made Simple & How To Tutorial. URL: https://spotintelligence.com/2024/05/06/support-vector-machines-svm/ (дата звернення: 22.05.2025).

10. What Is a Decision Tree? URL: https://www.mastersindatascience.org/learning/machine-learning-algorithms/decision-tree/ (дата звернення: 22.05.2025).

11. Intrusion detection using decision tree classifier with feature reduction technique/ Syed Atir Raza, Sania Shamim, Abdul Hannan Khan, Aqsa Anwar. *Mehran University Research Journal Of Engineering & Technology*, 2023, 30-37p. URL: https://search.informit.org/doi/abs/10.3316/informit.002355033595975 .

12. E. Hussein Sayed, A. Alabrah, K. Hussein Rahouma, M. Zohaib and R. M. Badry, "Machine Learning and Deep Learning for Loan Prediction in Banking: Exploring Ensemble Methods and Data Balancing," in *IEEE Access*, vol. 12, pp. 193997-194019, 2024, doi: 10.1109/ACCESS.2024.3509774.

13. RandomForestClassifier — scikit-learn 1.6.1 documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (дата звернення: 22.05.2025).

14. Zhu, Lin & Qiu, Dafeng & Ergu, Daji & Ying, Cai & Liu, Kuiyi. (2019). A study on predicting loan default based on the random forest algorithm. Procedia Computer Science. 162. 503-513. 10.1016/j.procs.2019.12.017.

15. Manzali, Y., Elfar, M. Random Forest Pruning Techniques: A Recent Review. *Oper. Res. Forum* 4, 43 (2023). URL: https://doi.org/10.1007/s43069-023-00223-6 .

16. Random Forest Simple Explanation. URL: https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d (дата звернення: 22.05.2025).

17. Amala Sonny, Abhinav Kumar, Linga Reddy Cenkeramaddi. Carry Object Detection Utilizing mmWave Radar Sensors and Ensemble-Based Extra Tree Classifiers on the Edge Computing Systems. *IEEE Sensors Journal*, 2023, 20137 - 20149p. URL: https://ieeexplore.ieee.org/abstract/document/10188595.

18. Intrusion Detection using hybridized Meta-heuristic techniques with Weighted XGBoost Classifier/ Ghulam Mohiuddin, Zhijun Lin та ін. за ред. Ghulam Mohiuddin. *Expert Systems with Applications*, 2023. URL: https://www.sciencedirect.com/science/article/pii/S0957417423010989 .

19. Odegua, Rising. (2020). Predicting Bank Loan Default with Extreme Gradient Boosting. 10.48550/arXiv.2002.02011.

20. XGBoost Documentation, 2022. [Електронний ресурс]. URL: https://xgboost.readthedocs.io/en/stable/

21. Intrusion Detection using hybridized Meta-heuristic techniques with Weighted XGBoost Classifier/ Ghulam Mohiuddin, Zhijun Lin та ін. за ред. Ghulam Mohiuddin. *Expert Systems with Applications*, 2023. URL: https://www.sciencedirect.com/science/article/pii/S0957417423010989 (дата звернення: 28.05.2025).

22. K, S., Thilagam, P.S. Multi-layer perceptron based fake news classification using knowledge base triples. *Appl Intell* 53, 6276–6287 (2023). [Електронний ресурс]. – URL: https://link.springer.com/article/10.1007/s10489-022-03627-9 .

23. Мокін В. Б., Дратований М. В. Наука про дані: машинне навчання та інтелектуальний аналіз даних : електронний навчальний посібник комбінованого (локального та мережевого) використання [Електронний ресурс]. Вінниця : ВНТУ, 2024, 258 с. URL: https://docs.vntu.edu.ua/card.php?id=8163 (дата звернення: 28.05.2025).

24. Путівник мовою програмування Python. URL: https://pythonguide.rozh2sch.org.ua/#_короткий_опис (дата звернення: 28.05.2025).

25. Популярність Python. URL: http://www.itschool.vn.ua/the-popularity-of-python/ (дата звернення: 28.05.2025).

26. Predicting Loan Default in R. URL: https://www.geeksforgeeks.org/predicting-loan-default-in-r/ (дата звернення: 28.05.2025).

27. What is R? URL: https://www.r-project.org/about.html (дата звернення: 28.05.2025).

28. R (programming language). URL: https://en.wikipedia.org/wiki/R_(programming_language) (дата звернення: 28.05.2025).

29. Machine learning with Java. URL: https://www.geeksforgeeks.org/machine-learning-with-java/ (дата звернення: 28.05.2025).

30. Java (programming language). URL: https://en.wikipedia.org/wiki/Java_(programming_language) (дата звернення: 28.05.2025).

31. Jupyter. URL: https://jupyter.org/about (дата звернення: 28.05.2025).

32. Introduction to machine learning with Jupyter notebooks. URL: https://developers.redhat.com/articles/2021/05/21/introduction-machine-learning-jupyter-notebooks# (дата звернення: 28.05.2025).

33. VS Code vs. Pycharm for machine learning development. URL: https://ozanciga.wordpress.com/2023/12/07/vscode-vs-pycharm-for-machine-learning-development/ (дата звернення: 28.05.2025).

34. Google Colab: the power of the cloud for machine learning. URL: https://datascientest.com/en/google-colab-the-power-of-the-cloud-for-machine-learning (дата звернення: 28.05.2025).

35. How to use Google Colab. URL: https://www.geeksforgeeks.org/how-to-use-google-colab/ (дата звернення: 28.05.2025).

36. How to Use Kaggle. URL: https://www.kaggle.com/docs/notebooks (дата звернення: 28.05.2025).

37. Kaggle Wiki. URL: https://uk.wikipedia.org/wiki/Kaggle (дата звернення: 28.05.2025).

38. Loan-Approval-Prediction-Dataset. Kaggle. URL: https://www.kaggle.com/datasets/architsharma01/loan-approval-prediction-dataset/data (дата звернення: 28.05.2025).

Додаток А

(обов’язковий)

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
“____”_____ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на бакалаврську кваліфікаційну роботу

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПРОГНОЗУВАННЯ РЕЗУЛЬТАТІВ КРЕДИТНОГО СХВАЛЕННЯ НА ОСНОВІ ФІНАНСОВИХ ТА СОЦІАЛЬНИХ ПОКАЗНИКІВ

08-34.БКР.019.00.000 ТЗ

Керівник: к.т.н., доц.
_____ Олексій КОЗАЧКО
«__» _____ 2025 р.
Розробив студент гр. 2ІСТ-21б
_____ Максим СІЧКАР
«__» _____ 2025 р.

Вінниця 2025

1. Підстава для проведення робіт.

Підставою для виконання роботи є наказ №__ по ВНТУ від «__» _____2025р., та індивідуальне завдання на БКР, затверджене протоколом №__ засідання кафедри САІТ від «__» _____ 2025р.

2. Джерела розробки:

1) Haque, F M Ahosanul & Hassan, Md. Mahedi. (2024). Bank Loan Prediction Using Machine Learning Techniques. doi:10.48550/arXiv.2410.08886.

2) Haque, F. M. A., & Hassan, Md. M. (2024). Bank Loan Prediction Using Machine Learning Techniques. *American Journal of Industrial and Business Management*, 14, 1690-1711. doi: 10.4236/ajibm.2024.1412085.

3. Мета і призначення роботи.

Метою дослідження є підвищення точності прогнозування результатів кредитного схвалення на основі фінансових та соціальних показників шляхом розроблення інформаційної технології з використанням методів машинного навчання.

4. Вихідні дані для проведення робіт:

Kaggle Dataset «Loan Approval Prediction Dataset»
https://www.kaggle.com/datasets/architsharma01/loan-approval-prediction-dataset/data

5. Методи дослідження: методи обробки та аналізу даних, методи машинного навчання, методи візуалізації даних, методи оцінювання ефективності моделей, програмні методи реалізації.

6. Етапи роботи і терміни їх виконання:

a) Аналіз предметної області _____ – _____
b) Вибір інструментів _____ – _____
c) Розвідувальний аналіз даних _____ – _____
d) Розроблення інформаційної технології _____ – _____
e) Оформлення матеріалів до захисту БКР _____ – _____

7. Очікувані результати та порядок реалізації

Отримання інформаційної технології прогнозування результатів кредитного схвалення.

8. Вимоги до розробленої документації

Текстова та ілюстративна частини роботи оформлені у відповідності до вимог «Методичних вказівок до виконання бакалаврських кваліфікаційних робіт для студентів спеціальностей: 124 «Системний аналіз», 126 «Інформаційні системи та технології» (освітня програма «Прикладні інформаційні технології»)».

9. Порядок приймання роботи

Публічний захист «___» _____ 2025 р.
Початок розробки «___» _____ 2025 р.
Граничні терміни виконання БКР «___» _____ 2025 р.

Розробив студент групи 2ІСТ-21б _____ Максим СІЧКАР

Додаток Б

(обов’язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Інформаційна технологія прогнозування результатів кредитного схвалення на основі фінансових та соціальних показників»

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ: кафедра САІТ, ФІІТА, гр. 2ІСТ-21б

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 2,13 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне):

- ■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту
- □ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- □ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Віталій МОКІН, зав. каф. САІТ _____ (підпис)

Євгеній КРИЖАНОВСЬКИЙ, доц. каф. САІТ _____ (підпис)

Особа, відповідальна за перевірку _____ (підпис) Сергій ЖУКОВ

З висновком експертної комісії ознайомлений(-на)

Керівник _____ (підпис) Олексій КОЗАЧКО, к.т.н., доц. каф. САІТ

Здобувач _____ (підпис) Максим СІЧКАР

Додаток В

(довідниковий)

Фрагмент лістингу програми

```
# Перевірка унікальних значень для кожної колонки (особливо для категоріальних)
print("\nКількість унікальних значень для кожної колонки:")
for col in data.columns:
    print(f"- {col}: {data[col].nunique()} унікальних значень")
    # Для колонок з невеликою кількістю унікальних значень, вивести їх
    if data[col].nunique() < 20:
        print(f"  Унікальні значення: {data[col].unique()}")

print("\n--- Початковий огляд завершено. ---")

# --- Попередня обробка назв колонок та значень для зручності ---
# Видалення зайвих пробілів у назвах колонок
data.columns = data.columns.str.strip()
print("\nОчищені назви колонок:")
print(data.columns)

# Видалення зайвих пробілів у значеннях категоріальних колонок, якщо такі є
for col in ['education', 'self_employed', 'loan_status']:
    if data[col].dtype == 'object':
        data[col] = data[col].str.strip()
print("\nОчищені значення категоріальних колонок (приклад education, self_employed, loan_status):")
print(data['education'].unique())
print(data['self_employed'].unique())
print(data['loan_status'].unique())

# --- 2.1 Аналіз розподілу цільової змінної ('loan_status') ---
print("\n--- 2.1 Аналіз розподілу цільової змінної ('loan_status') ---")
plt.figure(figsize=(7, 5))
sns.countplot(x='loan_status', data=data, palette='viridis')
plt.title('Розподіл результатів кредитного схвалення')
plt.xlabel('Статус Кредиту')
```

```
plt.ylabel('Кількість Заявників')
plt.show()

# Вивід відсоткового співвідношення
status_counts = data['loan_status'].value_counts()
print("\nКількість випадків для кожного статусу схвалення:")
print(status_counts)
print("\nВідсоткове співвідношення статусів схвалення:")
print(status_counts / len(data) * 100)

# --- 2.2 Розподіли числових ознак (гістограми, KDE plots) ---
print("\n--- 2.2 Розподіли числових ознак ---")

# Визначимо числові колонки, виключаючи 'loan_id'
numerical_cols = data.select_dtypes(include=np.number).columns.tolist()
if 'loan_id' in numerical_cols:
    numerical_cols.remove('loan_id')

# Побудова гістограм та KDE plot для кожної числової ознаки
for col in numerical_cols:
    plt.figure(figsize=(10, 6))
    sns.histplot(data[col], kde=True, bins=30, palette='viridis')
    plt.title(f'Розподіл ознаки: {col}')
    plt.xlabel(col)
    plt.ylabel('Частота')
    plt.show()

# --- 2.2 Розподіли числових ознак (гістограми, KDE plots) - Компактний вивід 3x3 ---
print("\n--- 2.2 Розподіли числових ознак (компактний вивід 3x3) ---")

n_rows = 3
n_cols = 3

plt.figure(figsize=(n_cols * 5, n_rows * 4)) # Загальний розмір фігури (3*5 = 15 ширини, 3*4 = 12 висоти)

for i, col in enumerate(numerical_cols):
```

```
    if i >= n_rows * n_cols: # Забезпечуємо, що не вийдемо за межі 3x3, якщо ознак більше 9
        break
    plt.subplot(n_rows, n_cols, i + 1) # Створення subplot
    sns.histplot(data[col], kde=True, bins=30, palette='viridis')
    plt.title(f'{col}') # Заголовок для кожного subplot'a
    plt.xlabel('') # Приховуємо xlabel для компактності
    plt.ylabel('Частота')

plt.suptitle('Розподіли числових ознак', y=1.02, fontsize=16) # Загальний заголовок для всіх графіків
plt.tight_layout(rect=[0, 0.03, 1, 0.98]) # Регулювання відступів для компактності
plt.show()

# --- 2.3 Аналіз категоріальних ознак (bar plots, count plots) ---
print("\n--- 2.3 Аналіз категоріальних ознак ---")

categorical_cols = ['no_of_dependents', 'education', 'self_employed'] # loan_status - цільова змінна, вже проаналізована

# Побудова countplot для кожної категоріальної ознаки
n_categorical_cols = len(categorical_cols)
n_cols_cat = 1 # Кількість колонок графіків у сітці
n_rows_cat = (n_categorical_cols + n_cols_cat - 1) // n_cols_cat

plt.figure(figsize=(n_cols_cat * 5, n_rows_cat * 4))

for i, col in enumerate(categorical_cols):
    plt.subplot(n_rows_cat, n_cols_cat, i + 1)
    sns.countplot(x=col, data=data, hue='loan_status', palette='viridis') # Відображення зв'язку з цільовою змінною
    plt.title(f'Розподіл {col} за статусом кредиту')
    plt.xlabel(col)
    plt.ylabel('Кількість')
    plt.xticks(rotation=0) # Без повороту підписів для кращої читабельності

plt.suptitle('Розподіли категоріальних ознак та їх зв\'язок зі статусом кредиту', y=1.02, fontsize=16)
plt.tight_layout(rect=[0, 0.03, 1, 0.98])
plt.show()
```

```
# --- 2.4 Кореляційні матриці та теплові карти для числових ознак ---
print("\n--- 2.4 Кореляційний аналіз числових ознак ---")

# Розрахунок кореляційної матриці
correlation_matrix = data[numerical_cols].corr()

# Побудова теплової карти
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
plt.title('Кореляційна матриця числових ознак')
plt.show()

# Вивід сильно корельованих пар (за бажанням, можна налаштувати поріг)
print("\nПари числових ознак з високою кореляцією (понад 0.7 або менше -0.7):")
highly_correlated_pairs = {}
for i in range(len(correlation_matrix.columns)):
    for j in range(i):
        if abs(correlation_matrix.iloc[i, j]) > 0.7: # Поріг для високої кореляції
            col1 = correlation_matrix.columns[i]
            col2 = correlation_matrix.columns[j]
            highly_correlated_pairs[f"{col1} - {col2}"] = correlation_matrix.iloc[i, j]

if highly_correlated_pairs:
    for pair, corr_val in highly_correlated_pairs.items():
        print(f"- {pair}: {corr_val:.2f}")
else:
    print("Високо корельованих пар числових ознак (понад 0.7) не виявлено.")

# --- 2.5 Боксплоти для виявлення викидів та зв'язку числових ознак з цільовою змінною ---
print("\n--- 2.5 Аналіз викидів та зв'язку числових ознак з 'loan_status' ---")

# Побудова боксплотів для кожної числової ознаки відносно loan_status
n_numerical_cols = len(numerical_cols)
n_cols_box = 3
n_rows_box = (n_numerical_cols + n_cols_box - 1) // n_cols_box
```

```
plt.figure(figsize=(n_cols_box * 5, n_rows_box * 4))

for i, col in enumerate(numerical_cols):
    plt.subplot(n_rows_box, n_cols_box, i + 1)
    sns.boxplot(x='loan_status', y=col, data=data, palette='viridis')
    plt.title(f'{col} за статусом кредиту')
    plt.xlabel('Статус Кредиту')
    plt.ylabel(col)

plt.suptitle('Боксплоти числових ознак за статусом кредиту', y=1.02, fontsize=16)
plt.tight_layout(rect=[0, 0.03, 1, 0.98])
plt.show()

print("\n--- Детальний EDA завершено. ---")

# --- 2.6 Scatterplot матриця всіх ознак з маркуванням цільової змінної ---
print("\n--- 2.6 Візуалізація попарних взаємозв'язків ознак (Pairplot) ---")

# Для pairplot краще використовувати підмножину ознак, якщо їх багато,
# або бути готовим до того, що графік буде дуже щільним.
# Розглянемо найбільш важливі числові ознаки та cibil_score
# Виключаємо loan_id, оскільки він не є ознакою
# Виключаємо education та self_employed, оскільки вони бінарні і pairplot для них не дуже інформативний.

features_for_pairplot = [
    'cibil_score',
    'income_annum',
    'loan_amount',
    #'loan_term',

    'loan_status' # Цільова змінна для маркування
]

# Перевіряємо, чи всі вибрані ознаки існують у df_processed
existing_features_for_pairplot = [col for col in features_for_pairplot if col in data.columns]
```

```
if len(existing_features_for_pairplot) < len(features_for_pairplot):
    print(f"Попередження: Деякі ознаки для pairplot не знайдено у датафреймі. Використовуються: {existing_features_for_pairplot}")



# Побудова pairplot
# Зверніть увагу: це може зайняти багато часу та ресурсів для великих датасетів/великої кількості ознак.
# Залежно від кількості ознак, варто розглянути PairGrid або точковий plot для конкретних пар.
# Для 15 ознак (навіть 9+4 нові) pairplot буде дуже великим. Оберемо найбільш релевантні.
print("Побудова Pairplot. Це може зайняти кілька хвилин...")
sns.pairplot(data[existing_features_for_pairplot], hue='loan_status', palette='viridis', diag_kind='kde')
plt.suptitle('Попарні Scatterplot та розподіли ознак за статусом кредиту', y=1.02, fontsize=16)
plt.show()
print("Pairplot завершено.")

print("\n--- EDA завершено. ---")

# Створення копії датафрейму, щоб не змінювати оригінальні дані EDA
df_processed = data.copy()

# Видалення 'loan_id', оскільки це ідентифікатор і не є ознакою для моделювання
if 'loan_id' in df_processed.columns:
    df_processed = df_processed.drop('loan_id', axis=1)
    print("Колонка 'loan_id' видалена.")

# Перетворення цільової змінної 'loan_status' на числові значення (0 і 1)
# 'Approved' -> 1, 'Rejected' -> 0
df_processed['loan_status'] = df_processed['loan_status'].map({'Approved': 1, 'Rejected': 0})
print("Цільова змінна 'loan_status' перетворена: 'Approved' -> 1, 'Rejected' -> 0.")

# Кодування бінарних категоріальних ознак за допомогою Label Encoding
# 'education': 'Graduate' -> 1, 'Not Graduate' -> 0
df_processed['education'] = df_processed['education'].map({'Graduate': 1, 'Not Graduate': 0})
print("Ознака 'education' перетворена: 'Graduate' -> 1, 'Not Graduate' -> 0.")

# 'self_employed': 'Yes' -> 1, 'No' -> 0
```

```
df_processed['self_employed'] = df_processed['self_employed'].map({'Yes': 1, 'No': 0})
print("Ознака 'self_employed' перетворена: 'Yes' -> 1, 'No' -> 0.")

print("\n--- 3.5 Масштабування числових ознак ---")

# Визначимо числові колонки для масштабування (всі числові, крім 'loan_status')
numerical_features_for_scaling = df_processed.select_dtypes(include=np.number).columns.tolist()
if 'loan_status' in numerical_features_for_scaling:
    numerical_features_for_scaling.remove('loan_status')

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df_processed[numerical_features_for_scaling] =
scaler.fit_transform(df_processed[numerical_features_for_scaling])
print(f"Числові ознаки {numerical_features_for_scaling} масштабовано за допомогою StandardScaler.")

print("\nВигляд обробленого датафрейму (перші 5 рядків):")
print(df_processed.head())
print("\n--- Підготовка даних завершена. ---")

# --- 4.1 Розбиття даних на тренувальний та тестовий набори ---
print("\n--- 4.1 Розбиття даних на тренувальний та тестовий набори ---")

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score,
confusion_matrix, ConfusionMatrixDisplay

# Визначення ознак (X) та цільової змінної (y)
X = df_processed.drop('loan_status', axis=1)
y = df_processed['loan_status']

# Розбиття на тренувальний та тестовий набори
# Використовуємо stratify=y для збереження пропорцій класів у тренувальному та тестовому наборах,
# що важливо через дисбаланс цільової змінної.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)
```

```
print(f"Розмір тренувального набору (X_train, y_train): {X_train.shape}, {y_train.shape}")
print(f"Розмір тестового набору (X_test, y_test): {X_test.shape}, {y_test.shape}")
print(f"Розподіл класів у тренувальному наборі:\n{y_train.value_counts(normalize=True)}")
print(f"Розподіл класів у тестовому наборі:\n{y_test.value_counts(normalize=True)}")

# --- 4.2 Використання класифікаційних моделей та оцінка ефективності ---
print("\n--- 4.2 Використання класифікаційних моделей та оцінка ефективності ---")

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC # Support Vector Machine
from sklearn.neighbors import KNeighborsClassifier # k-Nearest Neighbors

# Ініціалізація моделей
models = {
    'Logistic Regression': LogisticRegression(random_state=42, solver='liblinear'),
    'Decision Tree': DecisionTreeClassifier(random_state=42),
    'Random Forest': RandomForestClassifier(random_state=42),
    'Gradient Boosting': GradientBoostingClassifier(random_state=42),
    'Support Vector Machine': SVC(random_state=42, probability=True), # probability=True для ROC AUC
    'K-Nearest Neighbors': KNeighborsClassifier()
}

# Словник для зберігання результатів
results = {}

for name, model in models.items():
    print(f"\nНавчання моделі: {name}...")
    model.fit(X_train, y_train)

    # Прогнози на тренувальному та тестовому наборах
    y_train_pred = model.predict(X_train)
    y_test_pred = model.predict(X_test)

    # Прогнози ймовірностей для ROC AUC (якщо модель підтримує)
```

```
    y_train_proba = model.predict_proba(X_train)[:, 1] if hasattr(model, 'predict_proba') else None
    y_test_proba = model.predict_proba(X_test)[:, 1] if hasattr(model, 'predict_proba') else None

    # Оцінка метрик на тренувальному наборі
    train_accuracy = accuracy_score(y_train, y_train_pred)
    train_precision = precision_score(y_train, y_train_pred)
    train_recall = recall_score(y_train, y_train_pred)
    train_f1 = f1_score(y_train, y_train_pred)
    train_roc_auc = roc_auc_score(y_train, y_train_proba) if y_train_proba is not None else 'N/A'

    # Оцінка метрик на тестовому наборі
    test_accuracy = accuracy_score(y_test, y_test_pred)
    test_precision = precision_score(y_test, y_test_pred)
    test_recall = recall_score(y_test, y_test_pred)
    test_f1 = f1_score(y_test, y_test_pred)
    test_roc_auc = roc_auc_score(y_test, y_test_proba) if y_test_proba is not None else 'N/A'

    results[name] = {
        'Train Accuracy': train_accuracy,
        'Train Precision': train_precision,
        'Train Recall': train_recall,
        'Train F1-Score': train_f1,
        'Train ROC AUC': train_roc_auc,
        'Test Accuracy': test_accuracy,
        'Test Precision': test_precision,
        'Test Recall': test_recall,
        'Test F1-Score': test_f1,
        'Test ROC AUC': test_roc_auc
    }

    # Вивід результатів
    print(f"Модель: {name}")
    print(f"  Тренувальний набір: Accuracy={train_accuracy:.4f}, Precision={train_precision:.4f},
Recall={train_recall:.4f}, F1-Score={train_f1:.4f}, ROC AUC={train_roc_auc:.4f}")
    print(f"  Тестовий набір:     Accuracy={test_accuracy:.4f}, Precision={test_precision:.4f},
Recall={test_recall:.4f}, F1-Score={test_f1:.4f}, ROC AUC={test_roc_auc:.4f}")
```

```
    # Візуалізація Confusion Matrix для тестового набору
    cm = confusion_matrix(y_test, y_test_pred)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Rejected', 'Approved'])
    disp.plot(cmap=plt.cm.Blues)
    plt.title(f'Confusion Matrix для {name} (Тестовий набір)')
    plt.show()

# Вивід всіх результатів у вигляді DataFrame для порівняння
print("\n--- Зведена таблиця результатів моделей ---")
results_df = pd.DataFrame.from_dict(results, orient='index')
print(results_df.round(4))

print("\n--- Побудова моделей завершена. ---")

from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from scipy.stats import uniform, randint

# Словник для збереження найкращих моделей після тюнінгу
best_models = {}

# --- 5.1 Тюнінг Logistic Regression ---
print("\n--- 5.1 Тюнінг Logistic Regression ---")
# Параметри для тюнінгу Logistic Regression: C (сила регуляризації), solver
param_dist_lr = {
    'C': uniform(loc=0, scale=4), # Регуляризація, значення від 0 до 4
    'solver': ['liblinear', 'lbfgs', 'saga'] # 'liblinear' для L1/L2 регуляризації, 'lbfgs' для L2, 'saga' для обох
}
# Використаємо RandomizedSearchCV для прискорення
random_search_lr = RandomizedSearchCV(LogisticRegression(random_state=42),
param_distributions=param_dist_lr,
                                      n_iter=50, cv=5, scoring='f1', random_state=42, n_jobs=-1, verbose=1)
random_search_lr.fit(X_train, y_train)

best_lr_model = random_search_lr.best_estimator_
best_models['Logistic Regression'] = best_lr_model

print(f"Найкращі параметри для Logistic Regression: {random_search_lr.best_params_}")
```

```
print(f"F1-Score на крос-валідації для Logistic Regression: {random_search_lr.best_score_:.4f}")

# Оцінка найкращої моделі на тестовому наборі
y_test_pred_lr = best_lr_model.predict(X_test)
y_test_proba_lr = best_lr_model.predict_proba(X_test)[:, 1]

test_f1_lr = f1_score(y_test, y_test_pred_lr)
test_accuracy_lr = accuracy_score(y_test, y_test_pred_lr)
test_roc_auc_lr = roc_auc_score(y_test, y_test_proba_lr)

print(f"Оцінка найкращої Logistic Regression на тестовому наборі:")
print(f"  Accuracy={test_accuracy_lr:.4f}, F1-Score={test_f1_lr:.4f}, ROC AUC={test_roc_auc_lr:.4f}")

# --- 5.2 Тюнінг Random Forest ---
print("\n--- 5.2 Тюнінг Random Forest ---")
# Параметри для тюнінгу Random Forest: n_estimators, max_depth, min_samples_split, min_samples_leaf
param_dist_rf = {
   'n_estimators': randint(50, 200), # Кількість дерев
   'max_depth': randint(5, 20),      # Максимальна глибина дерева
   'min_samples_split': randint(2, 10), # Мінімальна кількість зразків для розділення вузла
   'min_samples_leaf': randint(1, 5)    # Мінімальна кількість зразків у листовому вузлі
}
# Використаємо RandomizedSearchCV
random_search_rf = RandomizedSearchCV(RandomForestClassifier(random_state=42),
param_distributions=param_dist_rf,
                                      n_iter=50, cv=5, scoring='f1', random_state=42, n_jobs=-1, verbose=1)
random_search_rf.fit(X_train, y_train)

best_rf_model = random_search_rf.best_estimator_
best_models['Random Forest'] = best_rf_model

print(f"Найкращі параметри для Random Forest: {random_search_rf.best_params_}")
print(f"F1-Score на крос-валідації для Random Forest: {random_search_rf.best_score_:.4f}")

# Оцінка найкращої моделі на тестовому наборі
y_test_pred_rf = best_rf_model.predict(X_test)
y_test_proba_rf = best_rf_model.predict_proba(X_test)[:, 1]
```

```
test_f1_rf = f1_score(y_test, y_test_pred_rf)
test_accuracy_rf = accuracy_score(y_test, y_test_pred_rf)
test_roc_auc_rf = roc_auc_score(y_test, y_test_proba_rf)

print(f"Оцінка найкращого Random Forest на тестовому наборі:")
print(f"  Accuracy={test_accuracy_rf:.4f}, F1-Score={test_f1_rf:.4f}, ROC AUC={test_roc_auc_rf:.4f}")

# --- 5.3 Тюнінг Gradient Boosting ---
print("\n--- 5.3 Тюнінг Gradient Boosting ---")
# Параметри для тюнінгу Gradient Boosting: n_estimators, learning_rate, max_depth
param_dist_gb = {
   'n_estimators': randint(50, 200),
   'learning_rate': uniform(loc=0.01, scale=0.2), # Швидкість навчання
   'max_depth': randint(3, 10),
   'subsample': uniform(loc=0.6, scale=0.4) # Частка зразків для кожного дерева
}
random_search_gb = RandomizedSearchCV(GradientBoostingClassifier(random_state=42),
param_distributions=param_dist_gb,
                                      n_iter=50, cv=5, scoring='f1', random_state=42, n_jobs=-1, verbose=1)
random_search_gb.fit(X_train, y_train)

best_gb_model = random_search_gb.best_estimator_
best_models['Gradient Boosting'] = best_gb_model

print(f"Найкращі параметри для Gradient Boosting: {random_search_gb.best_params_}")
print(f"F1-Score на крос-валідації для Gradient Boosting: {random_search_gb.best_score_:.4f}")

# Оцінка найкращої моделі на тестовому наборі
y_test_pred_gb = best_gb_model.predict(X_test)
y_test_proba_gb = best_gb_model.predict_proba(X_test)[:, 1]

test_f1_gb = f1_score(y_test, y_test_pred_gb)
test_accuracy_gb = accuracy_score(y_test, y_test_pred_gb)
test_roc_auc_gb = roc_auc_score(y_test, y_test_proba_gb)

print(f"Оцінка найкращого Gradient Boosting на тестовому наборі:")
```

```
print(f"  Accuracy={test_accuracy_gb:.4f}, F1-Score={test_f1_gb:.4f}, ROC AUC={test_roc_auc_gb:.4f}")


print("\n--- Тюнінг моделей завершено. ---")

# --- 6.2 Важливість ознак для Gradient Boosting ---
print("\n--- 6.2 Важливість ознак для Gradient Boosting ---")
feature_importances_gb = pd.DataFrame({
   'Feature': X.columns,
   'Importance': gb_model.feature_importances_
}).sort_values(by='Importance', ascending=False)

print("Важливість ознак за Gradient Boosting:")
print(feature_importances_gb)

# Візуалізація важливості ознак Gradient Boosting
plt.figure(figsize=(12, 7))
sns.barplot(x='Importance', y='Feature', data=feature_importances_gb, palette='viridis')
plt.title('Важливість ознак за Gradient Boosting')
plt.xlabel('Важливість')
plt.ylabel('Ознака')
plt.show()
```

Додаток Г

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПРОГНОЗУВАННЯ РЕЗУЛЬТАТІВ КРЕДИТНОГО СХВАЛЕННЯ НА ОСНОВІ ФІНАНСОВИХ ТА СОЦІАЛЬНИХ ПОКАЗНИКІВ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2025 р.

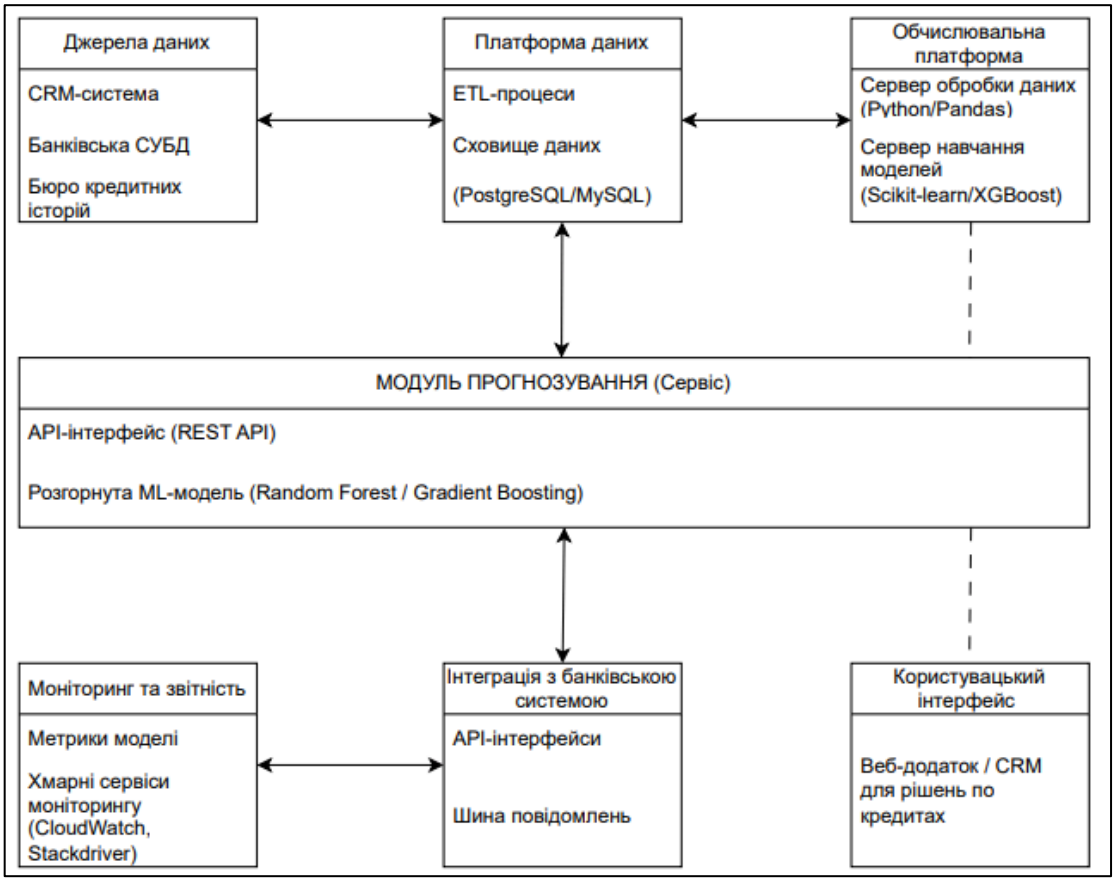

Рисунок Г.1 – Діаграма архітектури системи

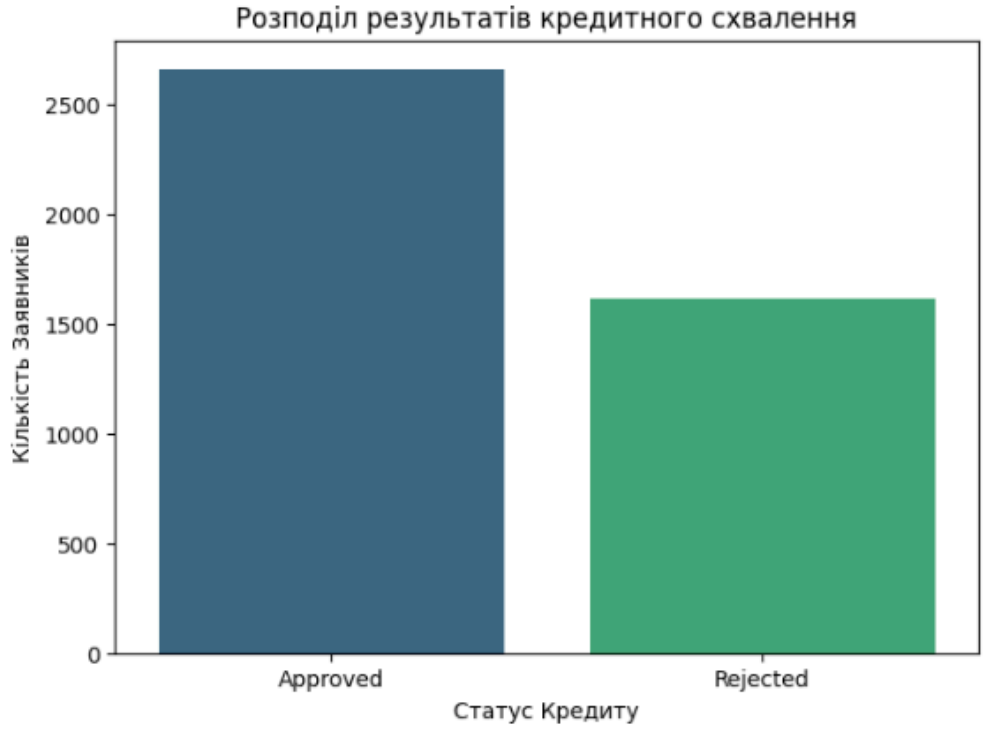

Рисунок Г.2 – Розподіл цільової змінної

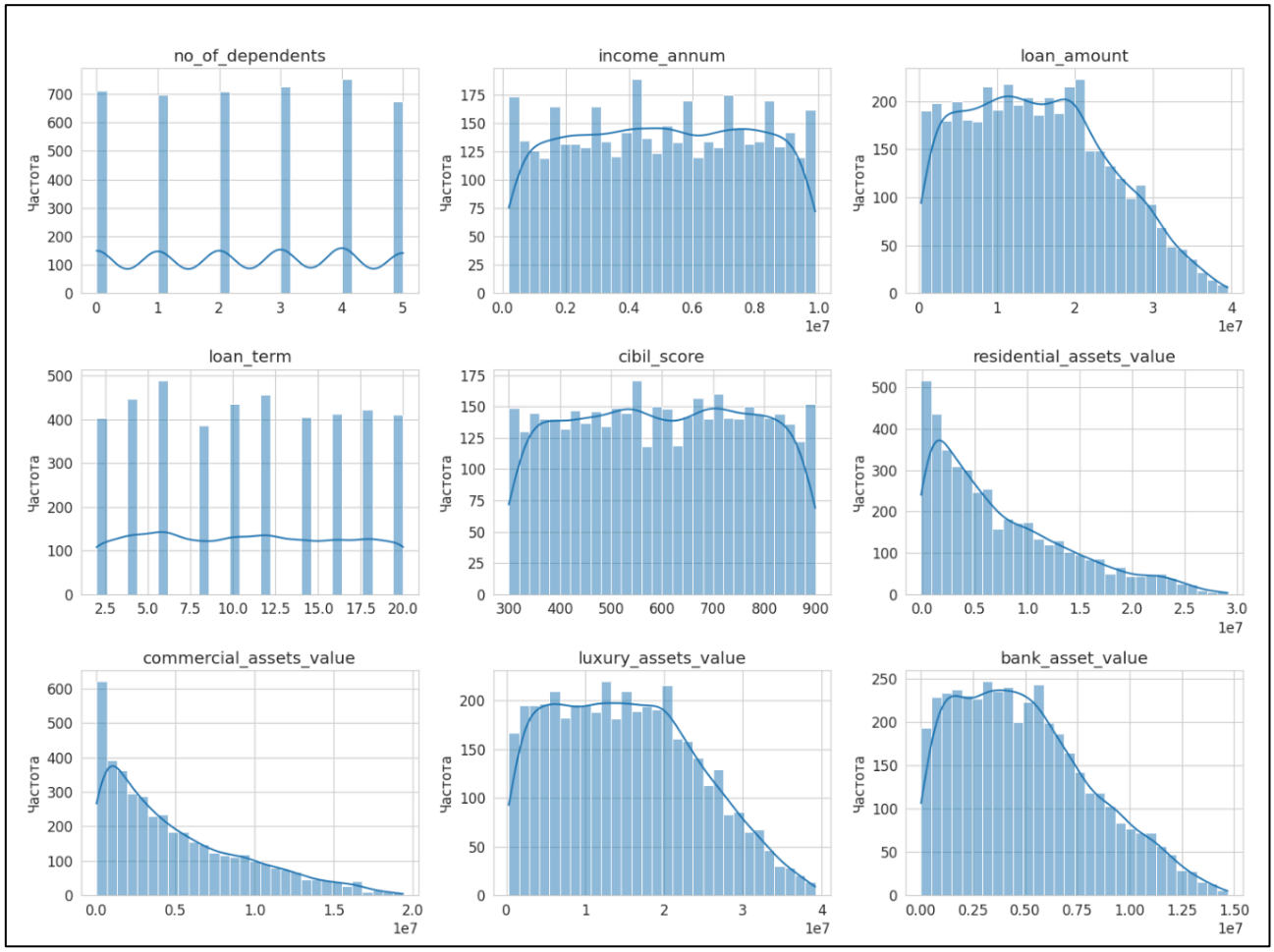

Рисунок Г.3 – Розподіл ознак

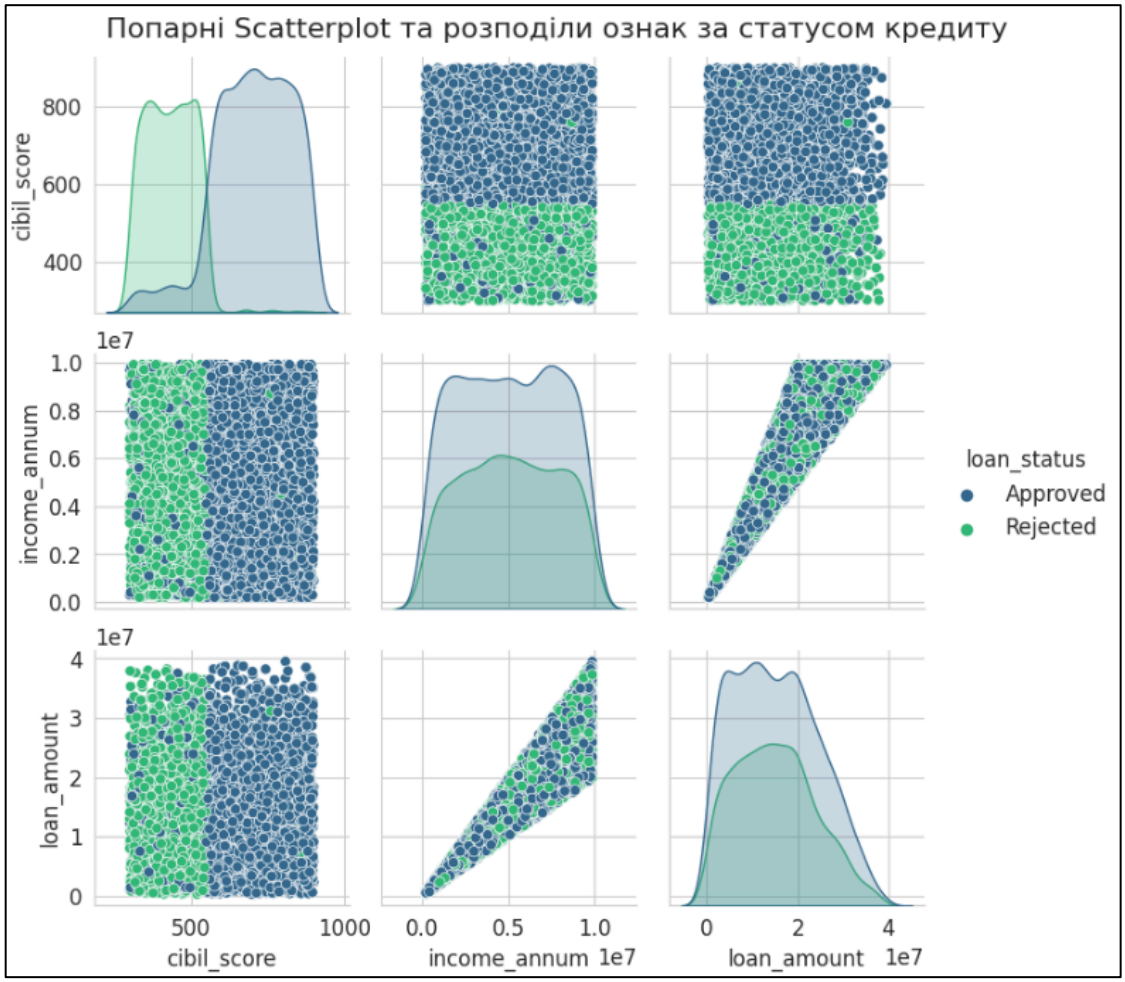

Рисунок Г.4 – Scatterplots числових ознак

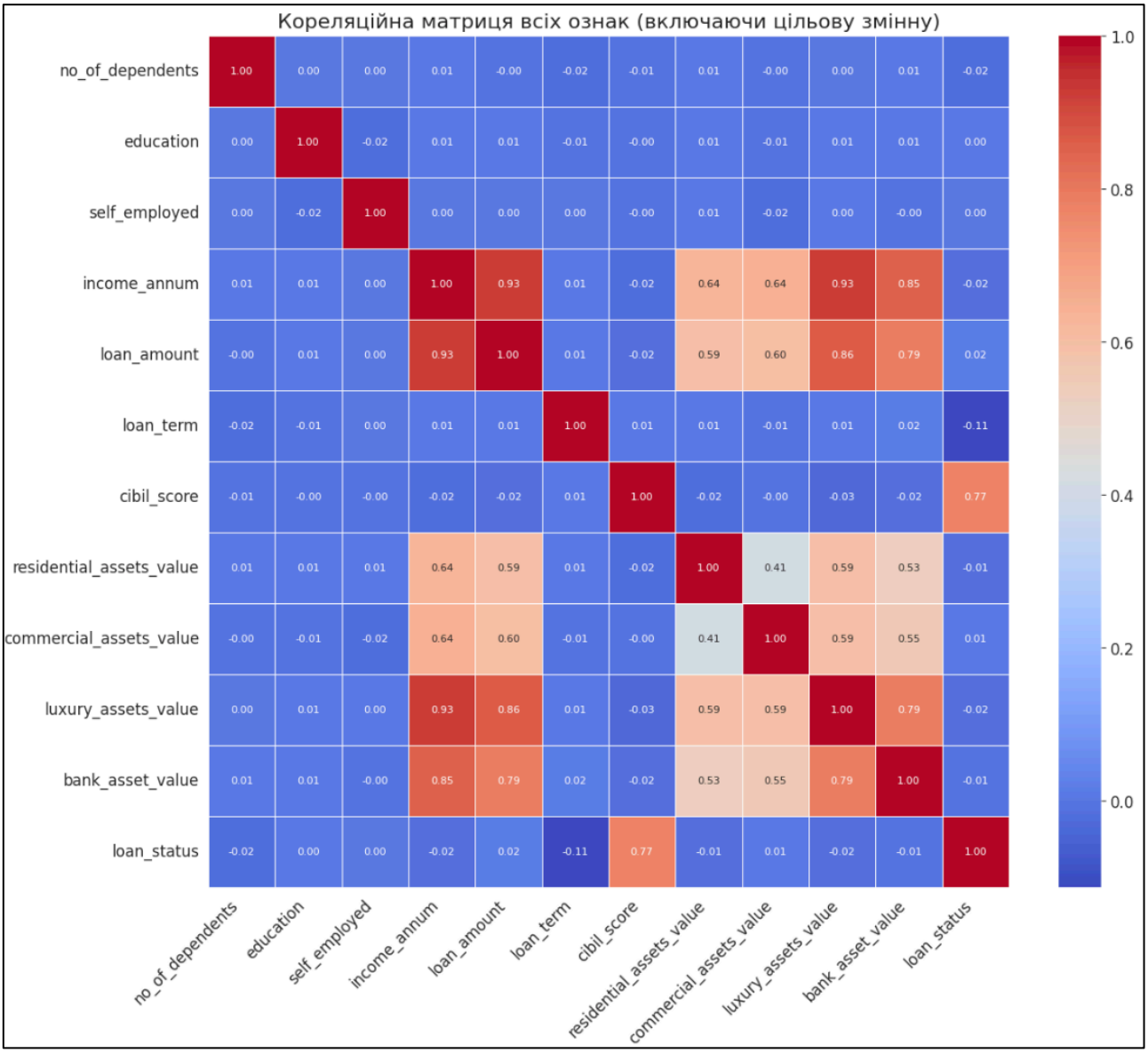

Рисунок Г.5 – Кореляційна матриця

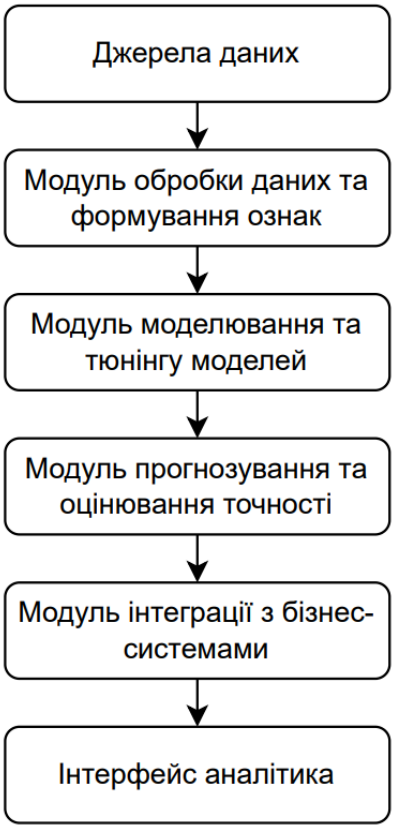

Рисунок Г.6 – Структурна схема технології

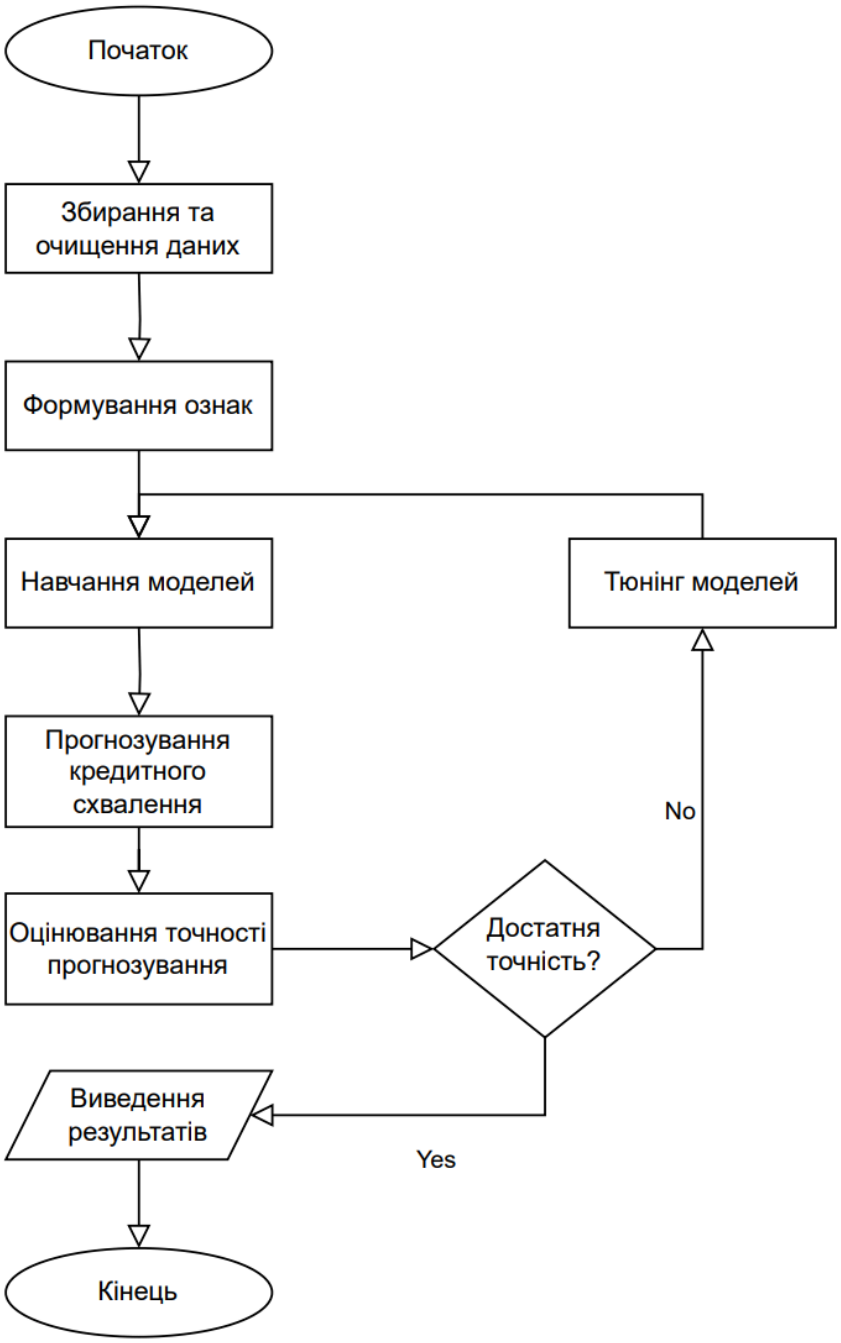

Рисунок Г.7 – Блок-схема алгоритму інформаційної технології

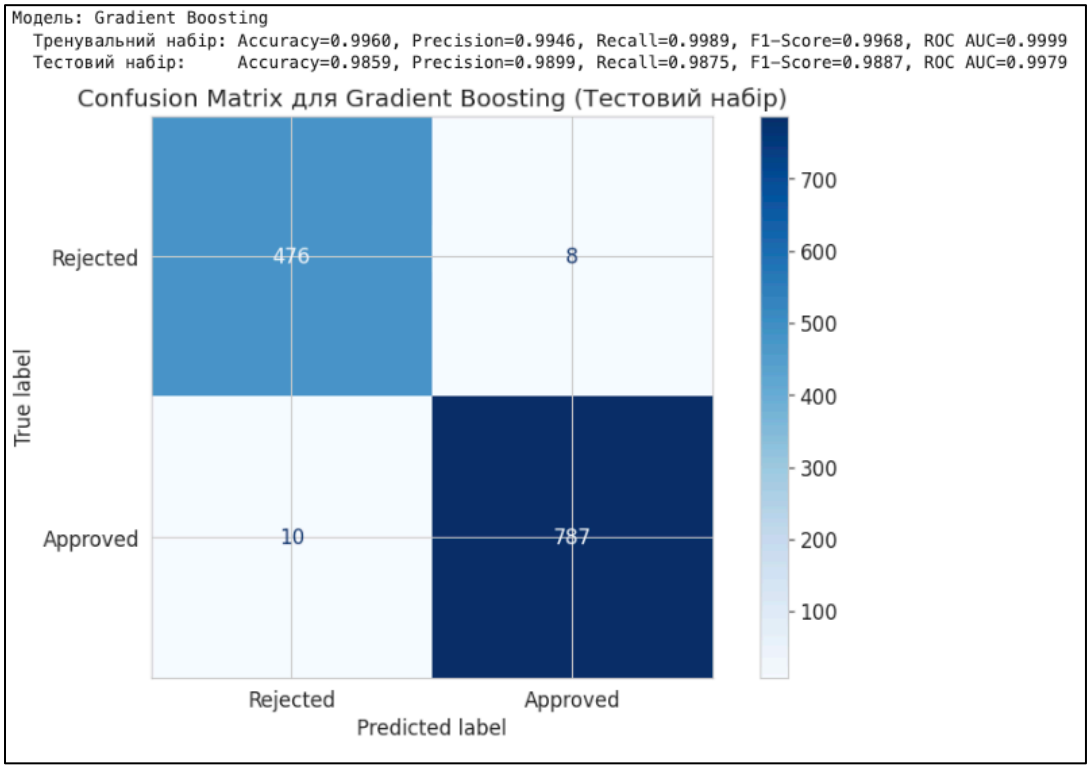

Рисунок Г.8 – Оцінка моделі Gradient Boosting

Модель	Accuracy (Train)	Accuracy (Test)
Logistic Regression	0.92	0.92
Decision Tree	1	0.98
Random Forest	1	0.98
Gradient Boosting	0.99	0.99
Support Vector Machine	0.95	0.94
K-Nearest Neighbors	0.94	0.88

Рисунок Г.9 – Порівняння точності класифікаційних моделей

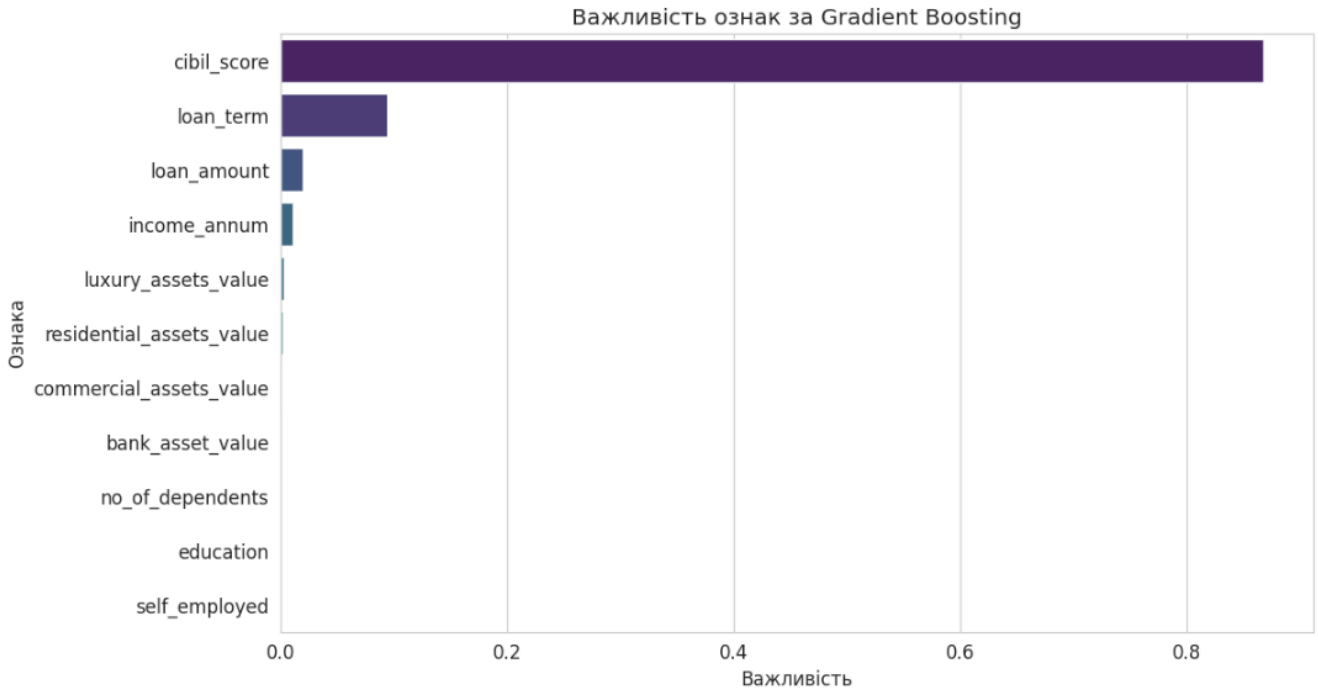

Рисунок Г.10 – Важливість ознак моделі Gradient Boosting