НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

НН Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

До захисту допущено:

Завідувач кафедри

_____ Оксана ТИМОЩУК

«___»_____2023 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системний аналіз і управління»

спеціальності 124 «Системний аналіз»

на тему: «Аналіз та прогнозування кредитних ризиків у банківській сфері»

Виконала:
студентка IV курсу, групи КА-95
Тіщенко Вікторія Юріївна

_____

Керівник:
д.т.н., професор кафедри ММСА,
Кузнєцова Н.В. _____

Консультант з економічного розділу:
к.е.н., доцент кафедри ТПЕ
Рощина Н.В. _____

Консультант з нормоконтролю::
к.ф.-м.н., доцент кафедри ММСА,
Статкевич В.М. _____

Рецензент:
доцент, к.т.н., доцент кафедри СП,
Безносик Олександр Юрійович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
НН Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 124 «Системний аналіз»
Освітня програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Оксана ТИМОЩУК
«__» травня 2023 р.

ЗАВДАННЯ
на дипломну роботу студенту
Тіщенко Вікторії Юріївні

1. Тема роботи «Аналіз та прогнозування кредитних ризиків у банківській сфері», керівник роботи Кузнєцова Наталія Володимирівна, д.т.н., професор, затверджені наказом по університету від «__» травня 2023 р. № _____

2. Термін подання студентом роботи __.06.2023.

3. Вихідні дані до роботи

1. Дані про клієнтів банку з відкритого набору на Kaggle
2. Мова програмування Python
3. Середовище розробки – Jupiter Notebook
4. Бібліотеки та модулі, що використовувалися: NumPy, Pandas, Sklearn, ImbLearn, Matplotlib, Seaborn, missingno

4. Зміст роботи

1. Проаналізувати попередні дослідження із теми роботи
2. Провести аналіз та вибір алгоритмів для прогнозування
3. Провести аналіз та обробку даних для тренування моделей
4. Створити систему для прогнозування на основі обраних алгоритмів та оброблених даних, визначити найкращій варіант
5. Провести економічний аналіз розробленого програмного продукту

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

1. Презентація

6. Консультанти розділів роботи[1*]

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ РОЗРОБКИ ЗАПРОПОНОВАНОЇ СИСТЕМИ	Рощина Н.В., доцент, к.е.н.	01.05.2023	29.05.2023
НОРМОКОНТРОЛЬ	Статкевич В.М., к.ф.-м.н.	15.05.2023	06.06.2023

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Визначення з темою роботи	20.04.2023	виконано
2.	Аналіз основних методів та підході, пошук літератури. Планування структури роботи.	30.04.2023	виконано
3.	Пошук даних для аналізу, огляд існуючих методів прогнозування кредитних	05.05.2023	виконано

[1*] Якщо визначені консультанти. Консультантом не може бути зазначено керівника дипломної роботи.

	ризиків. Написання теоретичної частини роботи.		
4.	Оформлення вступу, перших розділів та списку використаної літератури та джерел	08.05.2023	виконано
5.	Початор роботи над практичною частиною, підготовка даних для роботи з ними, експерементальне моделювання.	15.05.2023	виконано
6.	Написання економічного розділу роботи	20.05.2023	виконано
7.	Завершення роботи над основною частиною дипломної роботи. аналіз результатів	27.05.2023	виконано
8.	Завершення написання економічного розділу. Перевірка економічної частини	01.06.2023	виконано
9.	Оформлення дипломної роботи, створення презентації.	04.06.2023	виконано
10.	Перевірка роботи на нормконтроль.	06.06.2023	виконано

Студент Вікторія ТІЩЕНКО

Керівник Наталія КУЗНЄЦОВА

РЕФЕРАТ

Дипломна робота: 136 с., 12 рис., 9 табл., 2 дод., 39 джерел.

ПРОГНОЗУВАННЯ, КРЕДИТНІ РИЗИКИ, АНСАМБЛЕВІ МОДЕЛІ.

Тема: Аналіз та прогнозування кредитних ризиків у банківській сфері. У роботі розглянуті різні типи моделей для прогнозування кредитних ризиків у банківській сфері.

Об’єкт дослідження: використання методів машинного навчання для прогнозування ризику у банківській сфері на основі даних про клієнтів.

Предмет дослідження: методи машинного навчання та їх застосування у сфері визначення ризиків.

Мета роботи: розробка програмного забезпечення для автоматичного визначення ризиків у банківській сфері на основі алгоритмів машинного навчання.

Результат дослідження: У ході роботи були створені моделі для прогнозування ризиків невиплати кредитів клієнтами. Для роботи був використаний відкритий датасет із сайту Kaggle.

ABSTRACT

Diploma thesis: 136 pages, 12 figures, 9 tables, 2 appendices, 39 references.
FORECASTING, BANK SCORING, ENSEMBLE MODELS.

Theme: Credit scoring system for loan borrowers based on intellectual data analysis. The paper considers various classification models for determining the credit scoring of bank customers.

Object of research: application of intellectual analysis methods for determining the credit scoring of clients of financial institutions.

Subject of research: methods of machine learning and intellectual data analysis and means of their application

Purpose: to develop software for credit scoring in the banking sector based on customer data.

The result of the study: The software was created in the form of models for determining credit scoring for clients of financial institutions. Dataset from the Kaggle website was used to develop the models

ЗМІСТ

ВСТУП.....10
1 АНАЛІЗ ПОПЕРЕДНІХ ДОСЛІДЖЕНЬ В ОБЛАСТІ ПРОГНОЗУВАННЯ БАНКІВСЬКИХ РИЗИКІВ..... 12
1.1. Теоретичні аспекти прогнозування кредитних ризиків у банківській сфері.....12
1.1.1 Визначення кредитного ризику та його основні характеристики……..14
1.1.2 Класифікація кредитних ризиків..... 17
1.1.3 Методи оцінки кредитного ризику..... 19
1.2. Використання прогнозування кредитних ризиків у практиці банківської сфери..... 20
1.2.1 Переваги та недоліки використання прогнозування кредитних ризиків у банківській сфері..... 20
1.2.2 Перспективи використання прогнозування кредитних ризиків у майбутньому.....23
1.3 Висновки до розділу.....24
2 АНАЛІЗ АЛГОРИТМІВ ПРОГНОЗУВАННЯ РИЗИКІВ У БАНКІВСЬКІЙ СФЕРІ..... 26
2.1. Огляд алгоритмів прогнозування ризиків..... 26
2.1.1 Логістична регресія.....26
2.1.2 Нейронні мережі.....27
2.1.3 Випадковий ліс..... 29
2.1.4 Градієнтний бустінг.....31
2.1.5 Опорні вектори..... 33
2.2. Порівняння алгоритмів прогнозування ризиків..... 36
2.2.1 Переваги та недоліки логістичної регресії.....36
2.2.2 Переваги та недоліки нейронних мереж..... 37
2.2.3 Переваги та недоліки випадкового лісу.....38
2.2.4 Переваги та недоліки градієнтного бустинга.....40
2.2.5 Переваги та недоліки опорних векторів.....42
2.3. Використання алгоритмів прогнозування ризиків у практиці банківської сфери:..... 43
2.3.1 Приклади використання алгоритмів прогнозування ризиків в банківській сфері.....43
2.3.2 Переваги та недоліки використання алгоритмів прогнозування... 45

2.3.3 Проблеми, що виникають при використанні алгоритмів прогнозування ризиків у банківській сфері..... 47
2.3.4 Оцінка ефективності використання алгоритмів прогнозування ризиків у банківській сфері..... 48
2.4 Висновки до розділу..... 50
3 АНАЛІЗ ТА ОБРОБКА НАБОРУ ДАНИХ ДЛЯ ПРОГНОЗУВАННЯ.. 52
3.1 Вибір та опис набору даних для прогнозування ризиків у банківській сфері..... 52
3.2 Обробка пропущених даних..... 55
3.3 Видалення викидів..... 59
3.4 Дослідження даних на кореляцію..... 60
3.5 Аналіз даних на розповсюдження..... 62
3.6 Приведення даних в числовий формат..... 65
3.7 Збалансування класів..... 66
3.8 Нормалізація даних та їх розділення..... 68
3.9 Висновки до розділу..... 69
4 СТВОРЕННЯ СИСТЕМИ ПРОГНОЗУВАННЯ БАНКІВСЬКИХ РИЗИКІВ..... 71
4.1 Вибір методів та алгоритмів прогнозування ризиків в банківській сфері..... 72
4.2 Опис процесу побудови моделей прогнозування кредитних ризиків, з урахуванням методів навчання та оцінки моделей..... 74
4.3 Випробування та аналіз результатів системи прогнозування банківських ризиків..... 75
4.4 Опис можливостей та обмежень систем прогнозування банківських ризиків..... 79
4.5 Висновки до розділу..... 80
5 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ РОЗРОБКИ ЗАПРОПОНОВАНОЇ СИСТЕМИ..... 82
5.1 Постановка задачі..... 83
5.2 Обґрунтування функцій програмного продукту..... 84
5.3 Вибір параметрів для програмного продукту..... 89
5.4 Експертний аналіз параметрів..... 91
5.5 Аналіз якості реалізації варіантів функцій..... 95
5.6 Економічний аналіз розробки..... 97
5.7 Вибір оптимального варіанту реалізації ПП..... 103
5.8 Висновки до розділу..... 104
ВИСНОВКИ..... 105

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ..... 107
Додаток А Лістинг..... 120
Додаток Б Презентація……..... 124

ВСТУП

У сучасному економічному середовищі, яке характеризується зростаючою складністю фінансових операцій, банківські установи відіграють все більш важливу роль у забезпеченні функціонування економіки. Однак, разом зі збільшенням обсягу кредитних операцій зростає й ризик, пов'язаний з невиплатою боргів. У зв'язку з цим, аналіз та прогнозування кредитних ризиків стають надзвичайно актуальними для банківської сфери.

Ціль даного дослідження полягає в розкритті сутності кредитного ризику та вивчення методів його аналізу та прогнозування. Кредитний ризик вважається одним із найбільш значущих факторів, які впливають на фінансову стабільність банківських установ. Вирогідність невиплати кредиту, збитки, пов'язані з неповерненням позичених коштів та інші ризики вимагають від банківських установ ретельного аналізу та прогнозування для забезпечення ефективного управління кредитним портфелем.

Основна мета цього дослідження полягає в аналізі різних методів, моделей та підходів до виявлення, оцінки та прогнозування кредитного ризику у банківській сфері. Досліджуючи існуючі методики, я спробую визначити їх переваги та недоліки, а також з'ясувати, які з них є найбільш ефективними у конкретних умовах.

У процесі дослідження будуть враховані різні аспекти, пов'язані з аналізом кредитного ризику. Крім того, буде розглянуто можливості використання сучасних технологій, таких як штучний інтелект та машинне навчання, для поліпшення точності прогнозування.

Результати даного дослідження можуть бути корисними для банківських установ, аналітичних підрозділів та регуляторів у процесі

прийняття рішень щодо кредитування та управління ризиками. Ефективний аналіз та прогнозування кредитних ризиків можуть допомогти знизити втрати банківських установ і сприяти стабільному розвитку фінансової системи в цілому.

В першому розділі буде проведений аналіз попередніх досліджень в області прогнозування банківських ризиків, що дозволить краще зрозуміти поточний стан цієї області та здобути ідеї для покращення вже існуючих рішень.

В другому розділі буде проведений аналіз алгоритмів машинного навчання, що часто використовуються в сфері. Це дозволить побудувати стратегію розробки системи визначення ризиків та обрати методи, що підійдуть для такої задачі якнайкраще.

В третьому розділі буде описано збір та обробку вхідних даних, на якій будуть навчатись моделі. Ця частина роботи буде найважливішою, оскільки правильний підбір даних, їх обробка та створення нових атрибутів більше всього впливає на процес розробки та тренування будь-яких систем машинного навчання.

В четвертому розділі буде проведено розробку моделей для прогнозування ризиків та їх тестування. Також буде утворено прогнози на майбутнє.

В останньому, п’ятому розділі, буде проведено функціонально-вартісний аналіз можливостей реалізації програмного продукту на основі розробленої моделі. В цій частині буде визначено найоптимальніші варіанти створення та впровадження системи передбачення ризику дефолту в позичальників банків.

1 АНАЛІЗ ПОПЕРЕДНІХ ДОСЛІДЖЕНЬ В ОБЛАСТІ ПРОГНОЗУВАННЯ БАНКІВСЬКИХ РИЗИКІВ

1.1. Теоретичні аспекти прогнозування кредитних ризиків у банківській сфері

У банківському секторі прогнозування кредитного ризику є важливим завданням, яке вимагає постійної оцінки ризику для забезпечення стабільності окремих комерційних банків та банківської системи в цілому. Для мінімізації кредитного ризику банки використовують різні моделі та методи, такі як нечіткі множини та специфічні правила, штучні нейронні мережі та підтримувані векторні машини [1]. Однак прогнозування кредитного ризику часто здійснюється в умовах ризику та невизначеності, коли ірраціональна поведінка може вплинути на прийняття рішень. Згідно з [2], теорія перспектив може допомогти в розумінні поведінки осіб, які приймають рішення щодо кредитного ризику в умовах ризику та невизначеності.

Теорія перспектив – це поведінкова економічна теорія, яка вивчає, як люди приймають рішення в умовах ризику та невизначеності. Згідно з цією теорією, люди приймають рішення на основі потенційної вартості втрат і прибутків, а не кінцевого результату. Схильність до втрат і сприйняття ризику є двома основними факторами, які опосередковують поведінку банків при прийнятті рішень щодо кредитних ризиків в умовах ризику та невизначеності. Несхильність до втрат означає схильність індивідів надавати перевагу уникненню втрат над отриманням прибутків тієї ж вартості. Сприйняття ризику, з іншого боку, є суб'єктивною оцінкою ймовірності та тяжкості потенційних втрат.

Застосування теорії перспектив до прогнозування кредитного ризику має свої проблеми. Згідно з [2], теоретична модель теорії перспектив не завжди узгоджується з практичними реаліями банківської галузі. Складність банківської галузі в поєднанні з відсутністю детальних даних про поведінку кредитного ризику ускладнює застосування теорії до прогнозування кредитного ризику. Однак теорія може забезпечити краще розуміння поведінки банків у сфері кредитного ризику та допомогти у визначенні факторів, які впливають на прийняття рішень.

На додаток до теорії перспектив, банки використовують різні методи для оцінки кредитного ризику. Одним з найпоширеніших методів є метод оцінки вартості під ризиком (Value at Risk). Цей метод оцінює максимальну втрату портфеля за певний період часу із заданою ймовірністю (довірчим рівнем) на "нормальному" ринку. Метод Value at Risk використовується для розрахунку процентного ризику за торговими позиціями, фондового та валютного ризиків. Методика дозволяє банкам оцінити максимальну суму очікуваних фінансових втрат за певний період часу із заданим рівнем достовірності [3].

Однак метод VaR має свої обмеження. Базельський комітет з банківського нагляду не дозволяє банкам використовувати моделі з короткостроковим (менше 250 днів) періодом спостереження для розрахунку достатності капіталу. Крім того, модель розрахунку VaR є недостатньою для оцінки очікуваних втрат в умовах сталого розвитку ринку та в кризові періоди. Для подолання цих обмежень були розроблені нові підходи до оцінки ризиків, особливо операційного ризику. Ці підходи дають точні ймовірнісні оцінки навіть за відсутності статистичних даних .

Прогнозування кредитного ризику є важливим завданням, яке вимагає постійної оцінки ризику для забезпечення стабільності як окремих комерційних банків, так і банківської системи в цілому. Для оцінки кредитного ризику банки використовують різні моделі та методи, такі як

нечіткі множини та конкретні правила, штучні нейронні мережі та машини з підтримкою векторів. Крім того, теорія перспектив може бути використана для розуміння поведінки осіб, які приймають рішення щодо кредитного ризику, в умовах ризику та невизначеності. Однак застосування теорії перспектив до прогнозування кредитного ризику має свої виклики. Метод VaR є поширеним методом оцінки кредитного ризику, але він має свої обмеження. Нові підходи до оцінки ризиків, особливо операційного ризику, були розроблені для подолання цих обмежень і надання точних ймовірнісних оцінок навіть за відсутності статистичних даних.

1.1.1 Визначення кредитного ризику та його основні характеристики

Кредитний ризик - це ризик неотримання кредитором основної суми боргу та відсотків, що призводить до переривання грошових потоків та збільшення витрат на стягнення заборгованості [4]. Кредитний ризик може характеризувати ризик того, що компанія, яка випустила облігації, не зможе здійснити платіж на вимогу, або що страхова компанія не зможе виплатити страхове відшкодування [5]. Кредитний ризик найпростіше визначити як імовірність того, що позичальник або контрагент банку не зможе виконати свої зобов'язання відповідно до погоджених умов [6]. Кредитний ризик виникає тоді, коли кредитор позичає гроші позичальнику, але може не отримати їх назад. Кредити надаються позичальникам на основі здатності бізнесу або фізичної особи обслуговувати майбутні платіжні зобов'язання (основну суму боргу та відсотки) [7].

Кредитний ризик розраховується на основі загальної здатності позичальника погасити кредит відповідно до його початкових умов. Для оцінки кредитного ризику за споживчим кредитом кредитори часто

розглядають п'ять характеристик кредиту: кредитну історію, платоспроможність, капітал, умови кредиту та пов'язану з ним заставу. Це фактори, які кредитор може проаналізувати про позичальника, щоб допомогти зменшити кредитний ризик. Проведення аналізу на основі цих факторів може допомогти кредитору спрогнозувати ймовірність дефолту позичальника за кредитом [4]. При кредитуванні фізичних осіб кредитори хочуть знати фінансовий стан позичальника – чи є у нього інші активи, інші зобов'язання, який його дохід (відносно всіх його зобов'язань), і як виглядає його кредитна історія? Особисте кредитування, як правило, спирається на особисту гарантію та заставу. З іншого боку, комерційне кредитування набагато складніше; багато бізнес-клієнтів позичають більші суми, ніж фізичні особи. Оцінка ризику комерційного позичальника вимагає застосування різноманітних якісних та кількісних методів [7].

Кредитний ризик вимірюється кредиторами за допомогою власних інструментів оцінки ризику, які відрізняються залежно від фірми або юрисдикції і базуються на тому, чи є боржник фізичною або юридичною особою. Кількісною частиною оцінки кредитного ризику є фінансовий аналіз. Кредитори оцінюють різноманітні показники діяльності та фінансові коефіцієнти, щоб зрозуміти загальний фінансовий стан позичальника. На основі власних методів аналізу, моделей та параметрів оцінки, які використовує кредитор, оцінка кредитоспроможності позичальника виставляється в балах. Цей показник визначає ймовірність того, що позичальник спровокує дефолт. Чим вищий бал/кредитний рейтинг, тим менша ймовірність дефолту позичальника; чим нижчий бал/рейтинг, тим більша ймовірність дефолту позичальника [7].

Кредитний ризик можна частково зменшити за допомогою методів кредитного структурування. Елементи кредитної структури включають, серед іншого, період амортизації, використання (і якість) заставного забезпечення, співвідношення кредиту до вартості (loan-to-value, або LTV)

та кредитні ковенанти. Розуміння будь-якого доступного забезпечення та структурування кредиту відповідно до нього мають першорядне значення. Аналізуючи фактори кредитоспроможності позичальника, такі як поточне боргове навантаження та дохід, кредитори можуть зменшити кредитний ризик [4].

Отже, кредитний ризик – це ймовірність фінансових втрат внаслідок неспроможності позичальника повернути кредит. Кредитний ризик розраховується на основі загальної здатності позичальника погасити кредит відповідно до його початкових умов. Щоб оцінити кредитний ризик за споживчим кредитом, кредитори часто розглядають п'ять факторів кредиту: кредитну історію, платоспроможність, капітал, умови кредиту та пов'язану з ним заставу. Кредитори вимірюють кредитний ризик за допомогою власних інструментів оцінки ризику, які відрізняються залежно від фірми чи юрисдикції і залежать від того, чи є позичальник фізичною особою, чи підприємством. Кредитний ризик можна частково зменшити за допомогою методів структурування кредитів.

1.1.2 Класифікація кредитних ризиків

Кредитний ризик можна класифікувати за кількома категоріями, включаючи ризик кредитного дефолту, ризик концентрації, ризик країн, ризик контрагента та суверенний ризик [8].

- Ризик кредитного дефолту: Ризик кредитного дефолту є найпоширенішим видом кредитного ризику. Він виникає, коли

позичальник не в змозі погасити кредитне зобов'язання в повному обсязі або коли позичальник вже прострочив 90 днів до встановленої дати погашення кредиту. Ризик кредитного дефолту може впливати на всі фінансові операції, чутливі до кредитного ризику, такі як кредити, облігації, цінні папери та деривативи.

- Ризик концентрації: Ризик концентрації – це ризик збитків через вплив на одного позичальника, сектор або географічне розташування. Кредитори можуть зменшити ризик концентрації шляхом диверсифікації свого кредитного портфеля між різними секторами та географічними регіонами.
- Ризик країн: Ризик країни – це ризик, який виникає через політичні, економічні та соціальні фактори в країні проживання позичальника. Ризик країни можна зменшити, обмеживши вплив на країни з високим рівнем політичної та економічної нестабільності. Наприклад в даний момент Україна через політичну ситуацію вважається країною з високим ризиком, тому інвестори з початку 2022 року виводять свої інвестиції за межі країни.
- Ризик контрагента: Ризик контрагента – це ризик, який виникає, коли позичальник не може виконати свої зобов'язання перед кредитором. Ризик контрагента можна зменшити шляхом проведення належної перевірки позичальника та вимоги застави або гарантій.
- Суверенний ризик: Суверенний ризик – це ризик, який виникає, коли позичальник не виконує свої суверенні боргові зобов'язання. Суверенний ризик можна зменшити, обмеживши ризики, пов'язані з країнами з високим рівнем державного боргу та політичною нестабільністю.

На додаток до цих категорій кредитного ризику, кредитори також враховують такі фактори, як кредитна історія позичальника, його платоспроможність, капітал та умови кредиту при оцінці кредитного ризику [9]. Аналізуючи ці фактори, кредитори можуть визначити ймовірність дефолту потенційного позичальника та прийняти обгрунтоване кредитне рішення.

Отже, кредитний ризик можна класифікувати на кілька категорій, включаючи ризик кредитного дефолту, ризик концентрації, ризик країни, ризик контрагента та суверенний ризик. Оцінюючи кредитний ризик, кредитори повинні враховувати такі фактори, як кредитна історія позичальника, його платоспроможність, капітал та умови кредиту. Аналізуючи ці фактори та класифікуючи кредитний ризик, кредитори можуть приймати обґрунтовані кредитні рішення та зменшувати ризик втрат через ризик, пов'язаний з одним позичальником, галуззю або географічним розташуванням.

1.1.3 Методи оцінки кредитного ризику

Існує кілька методів, які кредитори можуть використовувати для оцінки кредитного ризику, включаючи аналіз фінансової звітності, ймовірність дефолту та машинне навчання [8].

- Аналіз фінансової звітності: Аналіз фінансової звітності передбачає перевірку фінансової звітності позичальника, включаючи звіт про прибутки і збитки, баланс і звіт про рух грошових коштів. Кредитор може використовувати фінансові

коефіцієнти та інші показники для оцінки кредитоспроможності позичальника. Наприклад, кредитор може проаналізувати співвідношення боргу до власного капіталу позичальника, коефіцієнт поточної ліквідності та коефіцієнт покриття грошових потоків.

- Моделі ймовірності дефолту: Моделі ймовірності дефолту – це статистичні моделі, які оцінюють ймовірність дефолту позичальника за кредитом. Ці моделі використовують історичні дані для визначення факторів, які найбільше прогнозують дефолт, а потім використовують ці фактори для розрахунку ймовірності дефолту для нового позичальника.
- Машинне навчання: Алгоритми машинного навчання також можна використовувати для оцінки кредитного ризику. Ці алгоритми можуть аналізувати великі обсяги даних і виявляти закономірності, які можуть бути неочевидними для людей-аналітиків. Алгоритми машинного навчання можна використовувати для виявлення позичальників, які мають високий ризик дефолту, і для розробки більш точних моделей кредитного ризику.

Оцінюючи кредитний ризик, кредитори повинні збалансувати витрати та вигоди від прийняття кредитного ризику. Кредитори також повинні враховувати потенційні втрати від невиконання зобов'язань, такі як пропущені платежі та потенційна безнадійна заборгованість. Такі ризики пов'язані з витратами, тому кредитори порівнюють їх з очікуваними вигодами, наприклад, з прибутковістю капіталу, скоригованою на ризик (risk-adjusted return on capital - RAROC) [10].

Отже, кредитори можуть використовувати кілька методів для оцінки кредитного ризику, включаючи аналіз фінансової звітності, моделі ймовірності дефолту та машинне навчання. Одна з відомих методологій

оцінки кредитного ризику включає критерій "прогнозований грошовий потік" для розширення системи показників. Оцінюючи кредитний ризик, кредитори повинні збалансувати витрати та вигоди від прийняття кредитного ризику та врахувати потенційні збитки від невиконання зобов'язань.

1.2. Використання прогнозування кредитних ризиків у практиці банківської сфери

1.2.1 Переваги та недоліки використання прогнозування кредитних ризиків у банківській сфері

Прогнозування кредитного ризику стало важливим аспектом управління ризиками в банківському секторі. Воно дозволяє фінансовим установам оцінювати ризик, пов'язаний з наданням кредитів фізичним чи юридичним особам, та приймати обґрунтовані рішення щодо схвалення чи відхилення кредитних заявок. Однак використання прогнозування кредитних ризиків у банківському секторі має як переваги, так і недоліки.

Переваги:

1. Покращення управління ризиками: Прогнозування кредитного ризику дозволяє банкам визначити потенційні ризики, пов'язані з наданням грошей позичальникам. Ці знання дозволяють банкам приймати обґрунтовані рішення щодо схвалення або відхилення кредитних заявок, тим самим зменшуючи ризик дефолту та фінансових втрат [1].

2. Підвищення ефективності: Прогнозування кредитних ризиків дозволяє банкам автоматизувати процес затвердження кредитів, що може заощадити час і підвищити ефективність. Автоматизовані процеси затвердження кредитів також можуть бути більш точними та послідовними, ніж ручні процеси [1].
3. Краще прийняття рішень: Прогнозування кредитного ризику надає банкам більше інформації про позичальників, що може покращити процес прийняття рішень. Банки можуть використовувати цю інформацію для виявлення тенденцій, закономірностей і потенційних ризиків, пов'язаних з конкретними позичальниками або галузями [10].

Недоліки:

1. Надмірна залежність від моделей: Одним з основних недоліків прогнозування кредитного ризику є те, що банки можуть надмірно покладатися на моделі і не враховувати інші фактори, які можуть вплинути на кредитний ризик. Наприклад, особисті обставини позичальника, такі як втрата роботи або хвороба, можуть вплинути на його здатність погасити кредит, але ці фактори можуть бути не враховані в моделях кредитного ризику [1].
2. Неточність: Моделі прогнозування кредитного ризику не завжди точні, а помилки в моделях можуть призвести до фінансових втрат. Наприклад, банк може схвалити кредит позичальнику, який згодом не зможе його погасити, що призведе до фінансових втрат для банку [1].
3. Брак прозорості: Моделі прогнозування кредитного ризику можуть бути складними, що ускладнює для банків пояснення факторів, які використовуються для оцінки кредитного ризику

для позичальників. Така непрозорість може призвести до браку довіри між банками та позичальниками, а позичальникам може бути складніше зрозуміти, чому їхні кредитні заявки відхиляються [1].

Отже, прогнозування кредитних ризиків має як переваги, так і недоліки в банківському секторі. Хоча воно може покращити управління ризиками, підвищити ефективність та вдосконалити процеси прийняття рішень, воно також може призвести до надмірної залежності від моделей, неточності та недостатньої прозорості. Тому банки повинні збалансувати переваги прогнозування кредитного ризику з потенційними ризиками та забезпечити врахування всіх факторів, які можуть вплинути на кредитний ризик.

1.2.2 Перспективи використання прогнозування кредитних ризиків у майбутньому.

Очікується, що використання прогнозування кредитних ризиків продовжить зростати в майбутньому, оскільки банки прагнуть покращити свої можливості управління ризиками та відповідати регуляторним вимогам. Дослідження [1] свідчить, що до 2025 року ризик-функції в банках повинні бути принципово іншими, ніж сьогодні, з більшим акцентом на аналітиці та неупередженому прийнятті рішень.

Однією з ключових тенденцій, яка формує майбутнє ризик-менеджменту, є зростаюча залежність від передової аналітики та

машинного навчання. Що, звичайно є плюсом для людей як я, що навчаються за напрямком аналітики, адже можна прогнозувати, що попит у кваліфікованих аналітиків буде тільки рости. Очікується, що ризик-функції використовуватимуть моделі для різних цілей, включаючи оцінку кредитів, системи раннього попередження та стягнення заборгованості в роздрібному сегменті та сегменті малих і середніх підприємств.

Ще одна тенденція, яка, ймовірно, вплине на майбутнє прогнозування кредитних ризиків – це зростаюча важливість звітності про ризики. Банкам необхідно буде підвищити якість і своєчасність звітів про ризики, щоб забезпечити можливість прийняття більш ефективних рішень. Це може передбачати заміну паперових звітів інтерактивними планшетними рішеннями, які надають інформацію в режимі реального часу і дозволяють користувачам проводити аналіз першопричин.

Незважаючи на переваги прогнозування кредитних ризиків, все ще існують проблеми, які потребують вирішення. Наприклад, існує ризик надмірної довіри до моделей та недостатньої прозорості процесу оцінки кредитоспроможності. Банкам необхідно забезпечити баланс між перевагами прогнозування кредитного ризику та потенційними ризиками, а також врахувати всі фактори, які можуть вплинути на кредитний ризик. Їм також потрібно буде сформувати сильну культуру управління ризиками і забезпечити, щоб виявлення, оцінка і зниження ризиків стали частиною щоденної роботи всіх співробітників банку [12].

Отже, прогнозування кредитних ризиків має як переваги, так і недоліки в банківському секторі. Хоча воно може покращити управління ризиками, підвищити ефективність та вдосконалити процеси прийняття рішень, воно також може призвести до надмірної залежності від моделей, неточності та недостатньої прозорості. Тому банки повинні збалансувати переваги прогнозування кредитного ризику з потенційними ризиками та

забезпечити врахування всіх факторів, які можуть вплинути на кредитний ризик.

1.3 Висновки до розділу

За результатами дослідження в першому розділі можна зробити кілька висновків про прогнозування кредитного ризику. Прогнозування кредитного ризику це важливе завдання, яке вимагає постійної оцінки ризику для забезпечення стабільності банківської системи в цілому. Для оцінки кредитного ризику використовуються різні моделі та методи, такі як нечіткі множини, конкретні правила, штучні нейронні мережі та машини з підтримкою векторів. Також кредитний ризик можна класифікувати на кілька категорій, включаючи ризик кредитного дефолту, ризик концентрації, ризик країни, ризик контрагента та суверенний ризик. Кредитори повинні враховувати такі фактори, як кредитна історія позичальника, його платоспроможність, капітал та умови кредиту при оцінці кредитного ризику. Також кредитори можуть використовувати кілька методів для оцінки кредитного ризику, включаючи аналіз фінансової звітності, моделі ймовірності дефолту та машинне навчання. Крім того, існують нові підходи до оцінки ризиків, особливо операційного ризику, які надають точні ймовірнісні оцінки навіть за відсутності статистичних даних. Кінцевим результатом оцінки кредитного ризику повинно бути збалансоване рішення кредитора, що враховує витрати та вигоди від прийняття кредитного ризику та потенційні збитки від невиконання зобов'язань.

2 АНАЛІЗ АЛГОРИТМІВ ПРОГНОЗУВАННЯ РИЗИКІВ У БАНКІВСЬКІЙ СФЕРІ

2.1. Огляд алгоритмів прогнозування ризиків

2.1.1 Логістична регресія

Логістична регресія – це популярний алгоритм машинного навчання, який використовується в задачах бінарної класифікації, де цільова змінна має два значення класу. Логістична функція, також відома як сигмоїдна функція, використовується в основі методу для перетворення будь-якого дійсного числа в значення між 0 і 1, але ніколи на цих межах. Модель логістичної регресії оцінюється за допомогою процесу, який називається оцінкою максимальної ймовірності. Робити прогнози за допомогою логістичної регресії легко, а підготовка даних для логістичної регресії дуже схожа на лінійну регресію [13].

Логістична регресія використовується, коли залежна змінна (вихідні дані) має двійковий формат, наприклад, 0 (неправда, false) або 1 (правда, true) [14]. Логістична регресія є основним методом для задач бінарної класифікації, і це найбільш поширений алгоритм машинного навчання [15].

Припущення, що незалежні змінні повинні бути нормально розподілені, лінійно пов'язані або мати однакову дисперсію в кожній групі, не є правильним для логістичної регресії. Алгоритми логістичної або лінійної регресії припускають, що між незалежними та залежними змінними існує лінійний зв'язок, але вони не мають припущення про те,

що незалежні змінні мають будь-який конкретний розподіл. Однак перетворення неперервних незалежних змінних до форми нормального розподілу краще виявляє цей лінійний зв'язок [13].

Без більшої репрезентативної вибірки логістична регресія може не мати достатньої статистичної потужності для виявлення значущого ефекту. Лінійні регресійні моделі використовуються для визначення зв'язку між неперервною залежною змінною та однією або кількома незалежними змінними, тоді як логістична регресія використовується для прогнозування категоріальної змінної порівняно з неперервною. Обидві моделі використовуються в регресійному аналізі для прогнозування майбутніх результатів. Лінійну регресію зазвичай легше зрозуміти, ніж логістичну [16].

У випадку, коли очікується, що ймовірність успіху не досягне 100%, фактично, реалістичні ймовірності знаходяться в діапазоні від 0 до a%, тоді як a невідоме, логістична регресія може бути не найкращим варіантом для вирішення цієї проблеми. У цьому випадку краще проконсультуватися зі справжнім математиком, який може допомогти правильно обмежити модель відповідно до того, що потрібно.

2.1.2 Нейронні мережі

Нейронні мережі широко використовуються в машинному навчанні (Machine Learning або ML), оскільки вони пропонують гнучку альтернативу, яка добре працює в багатьох випадках. Нейронні мережі використовуються для багатьох застосувань, включаючи комп'ютерний зір, обробку природної мови, а також ігри, прийняття рішень і моделювання. [17]

Нейронні мережі – це мережа функцій, які використовують набір ваг, що представляють зв'язки між кожним шаром нейронної мережі та шаром під ним. Нижній шар може бути іншим шаром нейронної мережі або будь-яким іншим шаром. Нейронні мережі використовуються для того, щоб зрозуміти і перетворити вхідну інформацію одного типу в заданий вихід, іноді в інший тип [18].

Однією з основних відмінних характеристик глибинного навчання є використання алгоритмів штучних нейронних мереж. На високому рівні нейронні мережі можна уявити як конфігурацію "обробних блоків", де вихід кожного блоку є входом іншого. Кожен з цих блоків може приймати один або багато входів і, по суті, виконує зважену суму своїх входів, застосовує зміщення (або "зсув"), а потім нелінійну функцію перетворення (яка називається "активація"). Різні комбінації цих компонентів були використані для опису меж рішень у класифікації, регресійних функціях та інших структурах, що є центральними для задач машинного навчання [17].

Нейронні мережі використовуються для багатьох застосувань, включаючи комп'ютерний зір, обробку природної мови, а також ігри, прийняття рішень і моделювання. Вони використовуються для обробки зображень, а також для різних типів вхідних даних, таких як аудіо. Згорткові нейронні мережі (Convolutional Neural Networks або CNN) – це категорія глибоких нейронних мереж, які найчастіше застосовуються для аналізу візуальної обробки. Їх також називають штучними нейронними мережами, інваріантними до зсуву або інваріантними до області (Shift-Invariant Artificial Neural Network або SIANN), виходячи з їхньої конструкції зі спільними вагами та незмінними характеристиками при перекладі. LSTM (Long-Short Term Memory) використовуються для класифікації, обробки та прогнозування на основі статистичних даних [18].

Нейронні мережі не завжди є кращими, ніж просто перетинання ознак, але вони є гнучкою альтернативою, яка добре працює в багатьох випадках. Класичні методи машинного навчання, такі як дерева з градієнтним підсиленням (XGBoost, LightGBM і CatBoost), схоже, мають перевагу для табличних даних. Для менш структурованих даних, таких як текст і зображення, нейронні мережі, як правило, працюють краще. Найкращий підхід – це завжди експериментувати з вашим конкретним джерелом даних і варіантом використання, щоб визначити, які методи найкраще підходять для вашої проблеми.

Отже, нейронні мережі широко використовуються в машинному навчанні і застосовуються в багатьох сферах, включаючи комп'ютерний зір, обробку природної мови, а також в іграх, прийнятті рішень і моделюванні. Нейронні мережі – це мережа функцій, які використовують набір ваг, що представляють зв'язки між кожним шаром нейронної мережі і шаром під ним. Вони використовуються для обробки зображень, а також для різних типів вхідних даних, таких як аудіо. Однак вони не завжди є кращими за класичні методи машинного навчання, і найкращим підходом завжди є експеримент з вашим конкретним джерелом даних і варіантом використання, щоб визначити, які методи найкраще підходять для вашої проблеми.

2.1.3 Випадковий ліс

Випадкові ліси – це популярний алгоритм машинного навчання, який об'єднує вихідні дані декількох дерев рішень для досягнення єдиного результату. Він може використовуватися як для класифікації, так і для регресійних задач в ML, і базується на концепції ансамблевого навчання,

яке є процесом об'єднання декількох класифікаторів для вирішення складної задачі та покращення продуктивності моделі [19].

Алгоритм випадкового лісу є розширенням методу bagging, оскільки він використовує як “bags”, так і випадковість ознак для створення некорельованого лісу дерев рішень. Випадковість ознак генерує випадкову підмножину ознак, що забезпечує низьку кореляцію між деревами рішень. Це ключова відмінність між деревами рішень та випадковими лісами. У той час як дерева рішень розглядають всі можливі розбиття ознак, випадкові ліси вибирають лише підмножину цих ознак [20].

Випадкові ліси мають три основні гіперпараметри, які необхідно встановити перед навчанням. До них відносяться розмір вузла, кількість дерев та кількість вибіркових ознак. Після цього класифікатор випадкових лісів можна використовувати для розв'язання задач регресії або класифікації.

Однією з ключових переваг випадкових лісів є те, що вони зменшують ризик надмірного навчання, оскільки дерева рішень мають тенденцію сильно підганятись до всіх зразків у навчальних даних. Однак, коли у випадковому лісі є достатня кількість дерев рішень, класифікатор не буде перенавчати модель, оскільки усереднення некорельованих дерев знижує загальну дисперсію та похибку прогнозування. Випадкові ліси також забезпечують гнучкість, оскільки вони можуть обробляти завдання регресії та класифікації з високим ступенем точності. Пакетування (bagging) ознак також робить класифікатор випадкових лісів ефективним інструментом для оцінки відсутніх значень, оскільки він зберігає точність, коли частина даних відсутня.

Випадкові ліси використовуються в багатьох різних сферах, як банківська справа, фондовий ринок, медицина та електронна комерція. У банківській справі алгоритм використовується для ідентифікації кредитного ризику. У медицині його можна використовувати для

виявлення тенденцій розвитку хвороб та ризиків захворювання. У землекористуванні – для виявлення ділянок зі схожим землекористуванням. У маркетингу – для виявлення маркетингових тенденцій.

Випадкові ліси можна реалізувати за допомогою Python. Першим кроком є імпорт класу RandomForestClassifier з бібліотеки sklearn.ensemble. Об'єкт класифікатора приймає такі параметри, як необхідна кількість дерев у випадковому лісі та функція для аналізу точності розбиття.

Отже, Випадкові ліси – це популярний алгоритм машинного навчання, який об'єднує вихід декількох дерев рішень для досягнення єдиного результату. Вони зменшують перенавчання у дерев рішень та використовують ансамблевий метод навчання для покращення результатів. Випадкові ліси широко використовуються у сфері прогнозування, в тому числі в банківській сфері.

2.1.4 Градієнтний бустінг

Градієнтний бустінг – це популярний алгоритм, який використовується в машинному навчанні для задач класифікації та регресії. Це ансамблевий метод, який об'єднує кілька слабких моделей в одну сильну. Алгоритм навчає модель послідовно, де кожна нова модель намагається виправити попередню модель. Gradient Boosting працює за принципом, що багато слабких моделей (наприклад, неглибокі дерева) можуть разом зробити більш точний предиктор [20].

У Gradient Boosting кожна нова модель навчається мінімізувати функцію втрат попередньої моделі за допомогою градієнтного спуску. На кожній ітерації алгоритм обчислює градієнт функції втрат по відношенню

до прогнозів поточного ансамблю, а потім навчає нову слабку модель, щоб мінімізувати цей градієнт. Передбачення нової моделі додаються до ансамблю, і процес повторюється доти, доки не буде досягнуто критерію зупинки [20].

Градієнтний бустінг будує модель поетапно і узагальнює модель, дозволяючи оптимізувати довільну диференційовану функцію втрат. По суті, він об'єднує слабких учнів в одного сильного учня в ітеративному режимі. З додаванням кожного слабкого елемента підбирається нова модель, яка дає більш точну оцінку відповідної змінної. Нові слабкі нейрони максимально корелюють з від'ємним градієнтом функції втрат, пов'язаним з усім ансамблем [21].

Gradient Boosting є дуже потужною технікою для побудови прогнозуючих моделей. Він може бути застосований до багатьох різних задач визначення ризику і оптимізує точність прогнозування моделі, що є перевагою перед традиційними методами навчання. Це дає свободу в розробці моделей. Він також вирішує проблеми мультиколінеарності, тобто проблеми, коли існують високі кореляції між двома або більше предикторними змінними. Моделі Gradient Boosting продемонстрували успіх у практичному застосуванні, а також у різних задачах машинного навчання та інтелектуального аналізу даних [21].

Ансамбль складається з M дерев. Дерево1 навчається за допомогою матриці ознак X та міток y. Передбачення, позначені мітками $\hat{y}1$, використовуються для визначення залишкових помилок навчальної множини r1. Потім навчається дерево2, використовуючи матрицю ознак X та залишкові помилки r1 дерева 1 як мітки. Прогнозовані результати $\hat{r}1$ потім використовуються для визначення залишкових r2. Процес повторюється до тих пір, поки не будуть навчені всі M дерев, що утворюють ансамбль. Існує важливий параметр, який використовується в

цій методиці, відомий як усадка(shrinkage). Усадка означає, що прогноз кожного дерева в ансамблі зменшується після того, як його помножено на швидкість навчання (eta), яка коливається від 0 до 1. Існує компроміс між eta та кількістю оцінок: зменшення швидкості навчання потрібно компенсувати збільшенням кількості оцінок, щоб досягти певної продуктивності моделі. Оскільки всі дерева вже навчені, можна робити прогнози. Кожне дерево прогнозує мітку, а остаточний прогноз задається формулою

$$y(pred) = y1 + (eta * r1) + (eta * r2) + \ldots\ldots + (eta * rN)$$

Отже, градієнтний бустинг це потужний алгоритм, який покращує результати роботи малих менш точних моделей, що зазвичай заснован на деревах рішень. Цей алгоритм використовує метод градієнтного спуску, щоб послідовно збільшувати точність передбачень моделей одна за одною на основі помилок попередніх моделей. GradientBoosting є доволі популярним методом передбачень у сфері машинного навчання та може використовуватись для задач регресії та класифікації, в тому числі для прогнозування кредитних ризиків у фінансовій сфері.

2.1.5 Опорні вектори

Машини опорних векторів (Support Vector Machines або SVM) – популярний алгоритм керованого навчання, який використовується для задач класифікації, регресії та виявлення викидів. Вони ефективні у просторах високої розмірності і все ще добре працюють, коли кількість вимірів перевищує кількість зразків. SVM працюють шляхом знаходження гіперплощини, яка розділяє дані на два класи з максимальним відривом, де відрив – це відстань між гіперплощиною і найближчими точками даних з

кожного класу. Точки, найближчі до гіперплощини, називаються опорними векторами, і вони використовуються у функції прийняття рішення. SVM є універсальним інструментом, тому для функції прийняття рішення можна вказати різні функції ядра. Надаються стандартні ядра, але також можна вказати власні ядра [22].

SVM – це потужний керований алгоритм, який найкраще працює на невеликих, але складних наборах даних. Їх можна використовувати як для задач регресії, так і для задач класифікації, але, як правило, найкраще вони працюють в задачах класифікації. В основі SVM лежить концепція пошуку гіперплощини, яка найкраще розділяє дані на класи. SVM призначені для пошуку гіперплощини, яка максимізує відстань між двома класами. Межа – це відстань між гіперплощиною і найближчими точками даних з кожного класу. SVM ефективні у просторах високої розмірності і все ще добре працюють, коли кількість вимірів перевищує кількість вимірів. SVM використовують підмножину навчальних точок у функції рішення (так звані опорні вектори), тому вони ефективно використовують пам'ять [23].

За час свого існування SVM стали добре відомим інструментом у машинному навчанні. Вони добре працюють на практиці і зараз використовуються в широкому спектрі застосувань від розпізнавання рукописних цифр до ідентифікації осіб, категоризації тексту, біоінформатики та маркетингу баз даних. Вони були надзвичайно популярні в той час, коли були розроблені в 1990-х роках, і продовжують залишатися основним методом для отримання високопродуктивного алгоритму з невеликою кількістю налаштувань.

SVM добре працюють на практиці і зараз використовуються в широкому спектрі задач. Існує два основних типи SVM: лінійні та нелінійні. Лінійні SVM використовуються для даних, що лінійно розділяються в просторі, тоді як нелінійні SVM використовуються для даних, що не розділяються лінійно. Нелінійні SVM використовують

функції ядра для перетворення даних у простір вищої розмірності, де вони вже розділяються лінійно. SVM безпосередньо не надають оцінок ймовірності, але їх можна обчислити за допомогою дорогої п'ятикратної перехресної перевірки.

SVM реалізовано у scikit-learn з використанням libsvm та liblinear для виконання всіх обчислень. Ці бібліотеки сворені за допомогою C та Python.

Для впровадження SVM в модель машинного навчання необхідно попередньо обробити дані, визначити SVM-класифікатор та його гіперпараметри. Модель повинна бути пристосована до навчальних даних, а прогнози для нових даних можуть бути зроблені за допомогою методу прогнозування [24].

Отже, машини опорних векторів (SVM) – це потужний алгоритм керованого навчання, що використовується для задач класифікації, регресії та виявлення викидів. В основі SVM лежить пошук гіперплощини, яка розділяє дані на класи з максимальним відривом. Точки, найближчі до гіперплощини, називаються опорними векторами і вони використовуються у функції прийняття рішення. SVM є ефективними в просторах високої розмірності та використовують підмножину навчальних точок, що робить їх ефективними у використанні пам'яті. Вони є універсальним інструментом, тому для функції прийняття рішення можна вказати різні функції ядра. SVM добре працюють на практиці і використовуються в широкому спектрі застосувань, від розпізнавання рукописних цифр до ідентифікації осіб, категоризації тексту, біоінформатики та маркетингу баз даних. SVM були розроблені в 1990-х роках та були надзвичайно популярні в той час і залишаються основним методом для отримання високопродуктивного алгоритму з невеликою кількістю налаштувань.

2.2. Порівняння алгоритмів прогнозування ризиків

В сучасному світі ризики є складною та невід'ємною частиною бізнесу, фінансів та інших сфер діяльності. Щоб ефективно управляти ризиками, необхідно мати знання та розуміння їх впливу на бізнес-процеси. Одним із методів управління ризиками є їх прогнозування. У цьому розділі будуть порівняні різні алгоритми прогнозування ризиків, їх переваги та недоліки, та визначені оптимальні умови застосування кожного з них. Це допоможе зрозуміти, які алгоритми є більш ефективними для конкретних завдань та як вони можуть бути використані для підвищення ефективності управління ризиками в галузі кредитування.

2.2.1 Переваги та недоліки логістичної регресії

Однією з найбільших переваг логістичної регресії є те, що коли у вас є складна лінійна задача і не так багато даних, то логістична регресія все одно здатна створювати досить корисні прогнози. Це перевага, яка притаманна математичним основам логістичної регресії і неможлива для більшості інших моделей машинного навчання [25].

Однак у логістичної регресії є кілька недоліків. Якщо є ознака, яка ідеально розділяє два класи, то модель більше не може бути навчена. Це називається повним розділенням. Однак у більшості випадків повне розділення можна вирішити, визначивши попередній розподіл ймовірностей ваг або запровадивши штрафування ваг. Логістична регресія може призвести до надмірної підгонки, якщо кількість ознак перевищує кількість спостережень. Крім того, складно отримати складні

взаємозв'язки, і алгоритми, такі як нейронні мережі, є більш придатними та потужними для таких завдань [26].

Підсумовуючи, до переваг логістичної регресії можна віднести її простоту, легкість реалізації, ефективність у навчанні та здатність надавати ймовірності. Недоліком логістичної регресії є те, що вона може призвести до надмірної підгонки, якщо кількість ознак перевищує кількість спостережень, а також складність отримання складних взаємозв'язків. Однак логістична регресія чудово себе показує з невеликим навчанням, що є великою перевагою в складних лінійних задачах з невеликою кількістю даних.

2.2.2 Переваги та недоліки нейронних мереж

Нейронні мережі мають ряд переваг, таких як здатність навчатися і моделювати нелінійні та складні взаємозв'язки, що є важливим, оскільки багато взаємозв'язків між входами і виходами є нелінійними в реальних життєвих ситуаціях. Вони також добре адаптуються до різних типів даних і можуть використовуватися як для керованого, так і для некерованого навчання. Крім того, нейронні мережі здатні обробляти дані високої розмірності і можуть бути навчені розпізнавати закономірності у великих наборах даних [27].

Однак, використання нейронних мереж має і ряд недоліків. Одним з найбільш суттєвих недоліків є їхня природа "чорної скриньки", що означає, що важко зрозуміти, як і чому нейронна мережа прийшла до певного результату. Це особливо актуально, коли мережа приймає рішення на основі складних або абстрактних даних, таких як зображення або природна мова. Крім того, нейронні мережі можуть бути дорогими в

обчислювальному плані, вимагаючи великої кількості часу і ресурсів для навчання та оптимізації. Для ефективної роботи їм також потрібні великі обсяги даних, а їхня продуктивність може сильно залежати від якості та кількості навчальних даних. Нарешті, нейронні мережі можуть бути чутливими до перенавчання, яке відбувається, коли мережа стає занадто складною і починає занадто точно відповідати навчальним даним, що призводить до поганого узагальнення нових даних.

Таким чином, нейронні мережі мають ряд переваг, включаючи здатність моделювати складні взаємозв'язки та адаптивність до різних типів даних. Однак вони також мають ряд недоліків, таких як зазначений вище "чорний ящик", обчислювальні витрати, чутливість до перенавчання та залежність від великих обсягів даних. Важливо враховувати ці переваги та недоліки при прийнятті рішення про використання нейронних мереж для вирішення конкретного завдання.

2.2.3 Переваги та недоліки випадкового лісу

Випадковий ліс або Random Forest – це популярний алгоритм машинного навчання, який використовується як для класифікації, так і для регресії. Він відомий своєю здатністю обробляти великі обсяги даних і високою точністю. Однією з головних переваг випадкового лісу є його висока точність порівняно з іншими алгоритмами машинного навчання. Це пов'язано з тим, що він об'єднує прогнози декількох дерев рішень, що допомагає зменшити дисперсію в моделі та покращити загальну продуктивність. Ще однією перевагою випадкового лісу є те, що він може обробляти великі обсяги даних, що робить його гарним вибором для додатків, що працюють з великими обсягами даних. Він також здатний

обробляти дані високої розмірності, наприклад, дані з багатьма характеристиками. Випадковий ліс може обробляти відсутні значення в даних, використовуючи середнє або медіанне значення ознаки для заміни відсутніх значень. Він також може обробляти категоріальні дані, створюючи фіктивні змінні для кожної категорії і розглядаючи їх як окремі ознаки. Випадковий ліс можна використовувати як для класифікації, так і для регресійних задач, що робить його універсальним вибором для широкого спектру застосувань [28].

Однак є й певні недоліки використання випадкового лісу. Одним з головних недоліків є те, що навчання випадкового лісу може бути обчислювально дорогим, особливо для великих наборів даних. Це може зробити процес навчання тривалим і вимагати використання високопродуктивного комп'ютера. Ще одним недоліком випадкового лісу є те, що його може бути важко інтерпретувати, оскільки він поєднує в собі прогнози декількох дерев рішень. Це ускладнює розуміння того, як модель робить свої прогнози, і виявлення важливих особливостей у даних. Хоча алгоритм випадкового лісу менш схильний до перенавчання, ніж одне дерево рішень, він все одно може перенавчатися, якщо кількість дерев у лісі занадто велика. Важливо ретельно налаштовувати гіперпараметри моделі, щоб запобігти надмірному пристосуванню. Random Forest може бути упередженим при роботі з категоріальними змінними, і він може надавати перевагу змінним з більшою кількістю рівнів, що може становити ризик для прогнозування [29].

Підсумовуючи, можна сказати, Random Forest - це потужний алгоритм, який забезпечує точні прогнози як для задач класифікації, так і для задач регресії. Він стійкий до викидів, може обробляти зашумлені дані і добре узагальнює нові дані. Однак він має певні обмеження з точки зору обчислювальних витрат, інтерпретованості та надмірного пристосування. Важливо ретельно налаштовувати гіперпараметри, щоб запобігти

надмірному пристосуванню та врахувати потенційне зміщення в бік змінних з більшою кількістю рівнів. Тим не менш, Random Forest може бути чудовим вибором для великих наборів даних і додатків для роботи з великими даними, де його здатність обробляти багатовимірні дані і висока точність роблять його цінним інструментом.

2.2.4 Переваги та недоліки градієнтного бустинга

Градієнтний бустінг або Gradient Boosting – це потужний алгоритм машинного навчання, який широко використовується для задач класифікації та регресії. Однією з головних переваг градієнтного бустінгу є його здатність оптимізувати на різних функціях втрат і надавати кілька варіантів налаштування гіперпараметрів, що робить його дуже гнучким алгоритмом. Градієнтний бустінг може добре працювати як з категоріальними, так і з числовими значеннями, не вимагаючи попередньої обробки даних. Крім того, градієнтний бустінг може без проблем обробляти відсутні значення, викиди та категоріальні значення з високою кардинальністю на ознаках [30].

Однак, у використанні Gradient Boosting є й певні недоліки. Одним з основних недоліків є те, що він може бути схильний до надмірної адаптації, особливо якщо дані є зашумленими. Цю проблему можна вирішити, застосовуючи регуляризацію L1 і L2 або використовуючи низьку швидкість навчання. Моделі з градієнтним прискоренням можуть бути дорогими в обчислювальному плані і потребують багато часу для навчання, особливо на центральних процесорах. Висока гнучкість Gradient Boosting призводить до того, що багато параметрів взаємодіють і сильно впливають на поведінку підходу, наприклад, кількість ітерацій, глибина

дерева і параметри регуляризації. Це вимагає пошуку на великій сітці під час налаштування, що може зайняти багато часу. Gradient Boosting також є менш інтерпретативним за своєю природою, що може ускладнити розуміння того, як модель робить свої прогнози, та виявлення важливих особливостей у даних.

Отже, градієнтний бустінг - це потужний алгоритм, який забезпечує гнучкість і високу точність як для задач класифікації, так і для задач регресії. Він може без проблем обробляти пропущені значення, викиди та категоріальні значення високої кардинальності. Однак він може бути схильний до надмірного пристосування, обчислювально дорогий і менш інтерпретативний за своєю природою. Важливо ретельно налаштовувати гіперпараметри, щоб запобігти надмірному пристосуванню та врахувати потенційне зміщення в бік певних параметрів. Тим не менш, Gradient Boosting може бути чудовим вибором для широкого спектру застосувань, де його гнучкість і висока точність роблять його цінним інструментом.

2.2.5 Переваги та недоліки опорних векторів

Машина опорних векторів (Support Vector Machine або SVM) – це потужний алгоритм машинного навчання, який можна використовувати як для класифікації, так і для регресії. SVM мають ряд переваг, таких як здатність працювати з даними високої розмірності, невеликими наборами даних і моделювати нелінійні межі рішень. SVM також стійкі до шуму в даних і мають хороші показники узагальнення. SVM мають розріджений

розв'язок, що означає, що вони використовують лише підмножину навчальних даних для прогнозування, що робить алгоритм більш ефективним і менш схильним до перенавчання. SVM можуть бути застосовані до широкого спектру використання, такого як обробка природної мови, комп'ютерний зір та біоінформатика [32]

Однак, використання SVM має і певні недоліки. Одним з основних недоліків є те, що SVM можуть бути обчислювально дорогими для великих наборів даних, оскільки алгоритм вимагає розв'язання задачі квадратичної оптимізації. SVM також можуть бути чутливими до вибору ядра, що може сильно вплинути на продуктивність алгоритму, і може бути важко визначити найкраще ядро для даного набору даних. SVM можуть бути чутливими до вибору параметрів, таких як параметр регуляризації, і може бути важко визначити оптимальні значення параметрів для даного набору даних. SVM також обмежені двокласовими задачами, хоча багатокласові задачі можуть бути вирішені за допомогою стратегій "один проти одного" або "один проти всіх". SVM не надають ймовірнісної інтерпретації границі рішення, що може бути недоліком у деяких додатках. Нарешті, SVM не підходять для великих наборів даних з великою кількістю ознак і наборів даних з пропущеними значеннями, оскільки вони вимагають повних наборів даних без пропущених значень.

Отже, SVM - це потужний алгоритм, який забезпечує високу точність як для задач класифікації, так і для задач регресії. Вони можуть обробляти дані високої розмірності, невеликі набори даних і можуть моделювати нелінійні межі рішень. Однак, вони можуть бути обчислювально дорогими для великих наборів даних і можуть бути чутливими до вибору ядра і параметрів. SVM обмежені задачами двох класів, не надають ймовірнісної інтерпретації межі рішення і не підходять для великих наборів даних з великою кількістю ознак та наборів даних з відсутніми значеннями. Тим не менш, SVM можуть бути чудовим вибором

для широкого спектру застосувань, де їхня здатність обробляти багатовимірні дані та висока точність роблять їх цінним інструментом.

2.3. Використання алгоритмів прогнозування ризиків у практиці банківської сфери

2.3.1 Приклади використання алгоритмів прогнозування ризиків в банківській сфері

Банківський сектор є однією з галузей, які отримали значну користь від використання алгоритмів прогнозування ризиків. Алгоритми прогнозування ризиків використовуються для передбачення можливості виникнення ризиків у банківській системі, що дозволяє банкам вживати превентивних заходів для зниження рівня ризику. В останні роки алгоритми машинного навчання все частіше використовуються для прогнозування ризиків на фінансових ринках. Наприклад, в одному з досліджень було запропоновано систему прогнозування ризиків фінансових ринків на основі алгоритмів машинного навчання, яка змогла досягти 95,64% успішності прогнозування на тестовому наборі [32]. В іншому дослідженні було запропоновано метод прогнозування банківських фінансових ризиків на основі великих даних, який використовував алгоритми Лассо та лінійної регресії за допомогою функцій великих даних та фреймворкових технологій для підвищення точності прогнозування та скорочення часу прогнозування [33].

Методи прогнозування банківських фінансових ризиків, запропоновані в цих дослідженнях, передбачали використання

різноманітних ознак, включаючи фінансову та нефінансову інформацію. Фінансові ознаки були розраховані та вилучені з використанням бухгалтерської інформації у фінансовій звітності, що регулярно публікується банком, тоді як нефінансові ознаки були вилучені з використанням даних розкриття інформації у вигляді фінансових звітів, новин та інших текстів, пов'язаних з банком. Нефінансові ознаки були перетворені на структуровані дані після простої обробки, які можна було безпосередньо використовувати як вхідні дані для алгоритму навчання. Текстові ознаки були вилучені за допомогою моделі bag of words та методу зважування частоти слів, зворотної до частоти документів (TF-IDF), а для фільтрації вилучених початкових текстових ознак та збереження важливих ознак для забезпечення якості ознак був використаний метод посилення інформативності [33].

Однією з переваг використання алгоритмів прогнозування ризиків у банківському секторі є можливість систематично прогнозувати непередбачувані ризики та вживати превентивних заходів для зниження рівня ризику. Використання алгоритмів машинного навчання, функцій великих даних та фреймворк-технологій дозволило підвищити точність прогнозування та скоротити час прогнозування. Однак, все ще існують певні обмеження щодо використання алгоритмів прогнозування ризиків у банківському секторі. Наприклад, обмеженість каналів збору даних може вплинути на прогностичний ефект моделі. Крім того, високорозмірні текстові ознаки, витягнуті з нефінансової інформації, можуть містити деякі надлишкові та нерелевантні ознаки, що може вплинути на якість ознак. Тим не менш, використання алгоритмів прогнозування ризиків у банківському секторі довело свою ефективність і цінність, і подальші дослідження проводяться з метою розширення багатоджерельної інформації та збору даних про фінансові ризики банків у реальному часі для підвищення точності моделі.

2.3.2 Переваги та недоліки використання алгоритмів прогнозування

Алгоритми прогнозування можуть запропонувати кілька переваг для бізнесу, таких як здатність охоплювати складні та нелінійні моделі, використовувати численні та різнорідні джерела даних, автоматизувати процес прогнозування та надавати точну та надійну інформацію для прийняття рішень. Алгоритми прогнозування можуть скоротити час, зусилля та витрати на прогнозування результатів бізнесу, а такі змінні, як фактори навколишнього середовища, конкурентна розвідка, зміни в регулюванні та ринкові умови, можуть бути враховані в математичних розрахунках, щоб надати більш повне уявлення за відносно низьких витрат [34].

Однак використання алгоритмів прогнозування має й певні недоліки. Одним з головних недоліків є те, що алгоритми прогнозування можуть бути дорогими у впровадженні та підтримці, особливо якщо потрібне спеціалізоване обладнання, програмне забезпечення та експертиза. Крім того, алгоритми прогнозування можуть бути складними для розуміння та пояснення, особливо якщо використовуються глибокі та складні моделі, такі як нейронні мережі. Ще одним недоліком є те, що продуктивність і надійність прогнозів можуть постраждати, якщо дані є неповними, неточними, застарілими або упередженими. Крім того, на алгоритми прогнозування можуть впливати зовнішні фактори та події, які є непередбачуваними або невідомими, що призводить до помилок або відхилень у прогнозах [35]. Нарешті, використання алгоритмів прогнозування може не підходити для всіх ситуацій або контекстів, і при

розгляді питання про те, який процес прогнозування підходить для бізнесу, слід зважити всі переваги та недоліки [36].

Таким чином, алгоритми прогнозування можуть забезпечити кілька переваг для бізнесу, таких як підвищення точності, гнучкості, масштабованості та інноваційності, а також скоротити час, зусилля та витрати на прогнозування результатів бізнесу. Однак алгоритми прогнозування також можуть бути дорогими у впровадженні та підтримці, складними для розуміння та пояснення, а на продуктивність та надійність прогнозів можуть впливати неповні, неточні, застарілі або упереджені дані. Крім того, на алгоритми прогнозування можуть впливати зовнішні фактори та події, які є непередбачуваними або невідомими, що призводить до помилок або відхилень у прогнозах. Тим не менш, алгоритми прогнозування можуть бути цінним інструментом для бізнесу, а переваги та недоліки повинні бути зважені при розгляді питання про те, який процес прогнозування є правильним для бізнесу.

2.3.3 Проблеми, що виникають при використанні алгоритмів прогнозування ризиків у банківській сфері

Використання алгоритмів прогнозування ризиків у банківському секторі також може спричинити певні проблеми. Одна з основних проблем пов'язана з модельним ризиком, оскільки моделі машинного навчання можуть посилювати деякі елементи модельного ризику [37]. Традиційні системи та практики перевірки моделей можуть бути недостатніми для

подолання ризиків, пов'язаних з моделями машинного навчання, і банкам необхідно вдосконалити управління ризиками моделей для вирішення проблем, пов'язаних з моделями машинного навчання. Крім того, використання алгоритмів прогнозування ризиків може бути дорогим у впровадженні та підтримці, особливо якщо потрібне спеціалізоване обладнання, програмне забезпечення та експертиза. Складність моделей також може ускладнити розуміння та пояснення того, як вони працюють.

Ще однією проблемою, що виникає при використанні алгоритмів прогнозування ризиків, є потенційна можливість виникнення етичних та регуляторних питань. Наприклад, банки можуть опинитися в ситуації порушення антидискримінаційного законодавства і понести значні штрафи, якщо моделі будуть упередженими або дискримінаційними. Використання алгоритмів прогнозування ризиків також може викликати занепокоєння щодо конфіденційності та безпеки даних, оскільки алгоритми можуть вимагати доступу до конфіденційних даних клієнтів [38]. Крім того, продуктивність і надійність прогнозів можуть постраждати, якщо дані є неповними, неточними, застарілими або упередженими. Використання алгоритмів прогнозування ризиків може не підходити для всіх ситуацій або контекстів, і при розгляді питання про те, який процес прогнозування підходить для бізнесу, слід зважити всі переваги та недоліки [39].

Таким чином, використання алгоритмів прогнозування ризиків у банківському секторі може спричинити низку проблем, таких як ризик моделі, високі витрати на впровадження та обслуговування, складність, етичні та регуляторні питання, проблеми з конфіденційністю та безпекою даних, а також проблеми з продуктивністю та надійністю. Банкам необхідно вдосконалити управління ризиками, пов'язаними з моделями, щоб впоратися з ризиками моделей машинного навчання, а переваги та недоліки використання алгоритмів прогнозування ризиків повинні бути

зважені при розгляді питання про те, який процес прогнозування є правильним для бізнесу.

2.3.4 Оцінка ефективності використання алгоритмів прогнозування ризиків у банківській сфері

Оцінка ефективності алгоритмів прогнозування ризиків у банківському секторі може включати оцінку точності прогнозування та часових витрат моделей. Запропонований гібридний метод, що використовує алгоритми Лассо та лінійної регресії з використанням функцій великих даних та фреймворкових технологій у [33], показав, що він ефективно підвищує точність прогнозування та витрачає порівняно менше часу на прогнозування ризиків. Однак у дослідженні не розглядався ефект прогнозування інших можливих і корисних джерел даних, і подальші дослідження необхідні для розширення багатоджерельної інформації та збору даних про банківські фінансові ризики в режимі реального часу для перевірки ефекту моделі прогнозування банківських фінансових ризиків. У роботі [39] Лессманн та його колеги провели порівняльний аналіз найсучасніших алгоритмів класифікації для кредитного скорингу та актуалізували дослідження на цю тему. У дослідженні оцінювалася продуктивність різних алгоритмів класифікації, таких як дерева рішень, випадкові ліси, машини опорних векторів і нейронні мережі, з використанням різних метрик продуктивності. Результати показали, що ансамблеві методи, такі як packing та бустінг, покращують точність класифікації дерев рішень та випадкових лісів. Однак дослідження також показало, що продуктивність алгоритмів варіюється залежно від набору даних і використовуваної метрики

продуктивності. Тому важливо ретельно оцінювати ефективність алгоритмів прогнозування ризиків у банківському секторі перед їх впровадженням.

Окрім оцінки точності прогнозування та часових витрат моделей, оцінка ефективності алгоритмів прогнозування ризиків у банківському секторі може також включати оцінку здатності моделей ідентифікувати ризики та управляти ними. Згідно з [33], прогнозування фінансових ризиків є важливою технікою для систематичного передбачення непередбачуваних ризиків у банківських системах. Запропонована модель прогнозування фінансових ризиків банків на основі великих даних використовує алгоритми Лассо та лінійної регресії з використанням особливостей великих даних та фреймворкових технологій для фільтрації початкових текстових ознак, попередньої обробки текстових даних річного звіту та вилучення текстових ознак прогнозування фінансових ризиків. Модель прогнозування фінансових ризиків банку побудована на основі алгоритму адаптивного випадкового підпростору зваженого злиття (weighted fusion adaptive random subspace algorithm), який може ефективно підвищити точність прогнозування ризиків за коротший проміжок часу прогнозування ризиків. Однак у дослідженні також зазначається, що через обмеженість каналів збору даних інші можливі та корисні джерела даних не були розглянуті. Тому необхідні подальші дослідження для розширення інформації з різних джерел та збору даних про фінансові ризики банків у режимі реального часу для перевірки ефекту моделі прогнозування банківських фінансових ризиків.

Отже, оцінка ефективності алгоритмів прогнозування ризиків у банківському секторі може включати оцінку точності прогнозування та часових витрат моделей, їхньої здатності ідентифікувати та управляти ризиками, а також ретельну оцінку продуктивності алгоритмів перед їхнім впровадженням.

2.4 Висновки до розділу

Отже, кредитори можуть використовувати кілька методів для оцінки кредитного ризику, включаючи аналіз фінансової звітності, моделі ймовірності дефолту та машинне навчання. Оцінюючи кредитний ризик, кредитори повинні збалансувати витрати та вигоди від прийняття кредитного ризику та врахувати потенційні збитки від невиконання зобов'язань.

Прогнозування кредитних ризиків може бути корисним для управління ризиками в банківському секторі, оскільки воно може підвищити ефективність та вдосконалити процеси прийняття рішень. Однак, на відміну від точних методів, таких як аналіз фінансової звітності, прогнозування кредитного ризику може мати неточності та недостатню прозорість. Банки повинні збалансувати переваги прогнозування кредитного ризику з потенційними ризиками та забезпечити врахування всіх факторів, які можуть вплинути на кредитний ризик. Це означає, що кредитори повинні здійснювати оцінку ризику, враховуючи різні методи оцінки кредитного ризику та критеріїв, таких як прогнозований грошовий потік, щоб забезпечити найбільш точну та прозору оцінку.

3 АНАЛІЗ ТА ОБРОБКА НАБОРУ ДАНИХ ДЛЯ ПРОГНОЗУВАННЯ

3.1 Вибір та опис набору даних для прогнозування ризиків у банківській сфері

Обраний датасет має дані про клієнтів, котрим надавались позики в банку. Надання позик є основним джерелом доходів для банків, але також пов'язане з ризиком неповернення позикових коштів боржниками. З метою зменшення цього ризику банки зібрали історичні дані про позичальників та пропонують розробити потужну модель машинного навчання для класифікації нового позичальника на ймовірність неповернення позики.

Наданий набір даних величезний і містить кілька детермінованих факторів, таких як дохід позичальника, стать, ціль позики та інше. Атрибути, присутні в датасеті, описані нижче в таблиці 3.1:

Таблиця 3.1 - Атрибути та їх описи

Атрибут	Опис
id	Унікальний ідентифікатор
loan_limit	Ліміт позики
gender	Стать позичальника
approv_in_adv	Одобрення вперед (заздалегідь)
loan_type	Тип позики
loan_purpose	Мета позики
credit_worthiness	Кредитна здатність
open_credit	Відкритий кредит
business_or_commercial	Бізнес або комерційна ціль
loan_amount	Сума позики
Продовження таблиці 3.1	

rate_of_interest	Процентна ставка
interest_rate_spread	Різниця процентних ставок
upfront_charges	Передоплата
term	Термін позики
neg_ammortization	Негативна амортизація
interest_only	Тільки відсотки
lump_sum_payment	Одноразовий платіж
property_value	Вартість нерухомості
construction_type	Тип будівництва
occupancy_type	Тип зайнятості
secured_by	Забезпечено чим (Тип забезпеченості)
total_units	Загальна кількість одиниць
income	Дохід позичальника
credit_type	Тип кредиту
credit_score	Кредитний бал
co-applicant_credit_type	Тип кредиту співзаявника
age	Вік позичальника
submission_of_application	Подання заявки
ltv	Відношення вартості кредиту до вартості нерухомості
region	Регіон
security_type	Тип забезпечення
status	Статус позики
dtir1	Відношення боргу до доходу перед одержанням позики

Як видно із таблиці вище, датасет має величезну кількість атрибутів, що можуть значним чином вплинути на роботу моделі, даючи їй широкий обсяг інформації про клієнта, щоб знаходити зв’язки із виплатою або невиплатою кредита.

Датасет має аж 148 670 рядків із даними про клієнта, що є достатньо надійним набором, навіть в контексті великих комерційних досліджень, що виконуються банками під замовлення, а отже, і моделі мають більш точно працювати на таких обсягах даних. Детальні дані про кожен атрибут описані в таблиці 3.2.

Таблиця 3.2 - Детальний опис кожного атрибуту даних

Номер	Атрибут	Кількість пропущених значень	Відсоток пропущених значень	Тип даних
0	id	0	0,00%	object
1	loan_limit	3344	2,25%	object
2	gender	0	0,00%	object
3	approv_in_adv	908	0,61%	object
4	loan_type	0	0,00%	object
5	loan_purpose	134	0,09%	object
6	credit_worthiness	0	0,00%	object
7	open_credit	0	0,00%	object
8	business_or_commercial	0	0,00%	object
9	loan_amount	0	0,00%	int64
10	rate_of_interest	36439	24,51%	float64
11	interest_rate_spread	36639	24,64%	float64
12	upfront_charges	39642	26,66%	float64
13	term	41	0,03%	float64
14	neg_ammortization	121	0,08%	object
15	interest_only	0	0,00%	object
16	lump_sum_payment	0	0,00%	object
17	property_value	15098	10,16%	float64
18	construction_type	0	0,00%	object
19	occupancy_type	0	0,00%	object
20	secured_by	0	0,00%	object
21	total_units	0	0,00%	object
22	income	9150	6,15%	float64
23	credit_type	0	0,00%	object
24	credit_score	0	0,00%	int64

Продовження таблиці 3.2

25	co-applicant_credit_type	0	0,00%	object
26	age	200	0,13%	object
27	submission_of_application	200	0,13%	object
28	ltv	15098	10,16%	float64
29	region	0	0,00%	object
30	security_type	0	0,00%	object
31	status	0	0,00%	int64
32	dtir1	24121	16,22%	float64

Як видно з наведеної таблиці, більшість атрибутів не мають значну кількість пропущених значень, проте атрибути dtir1, ltv, rate_of_interest, interest_rate_spread, upfront_charges, term в цей перелік не входять. Із ними доведеться провести заповнення пропущених значень в майбутньому.

3.2 Обробка пропущених даних

Далі, для кращої візуалізації та розуміння було створено heatmap для всіх пропущених значень по рядках датасета (Рис. 3.1).

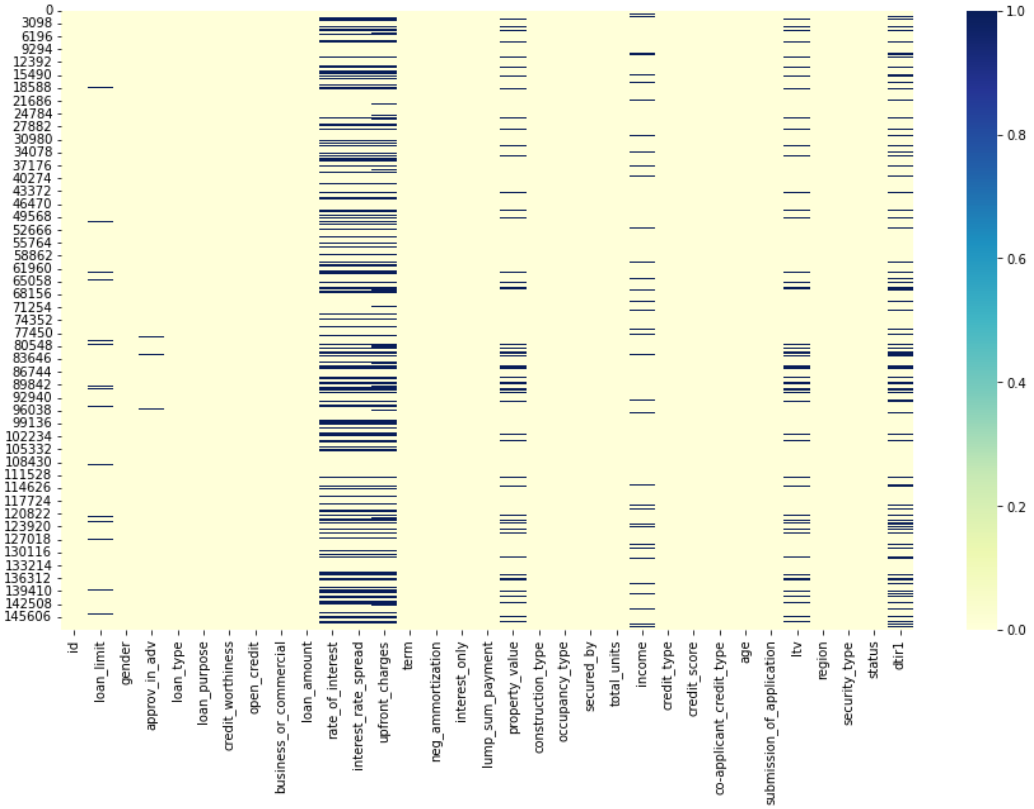


Рисунок 3.1 – Heatmap для пропущених значень

Як видно з рисунку наведеного вище, рядки, що мають пропущені дані здебільшого співпадають. Тобто, рядки, що мають пропущені дані, скоріш за все мають їх в декількох атрибутах одразу.

В такому разі гарним варіантом було би видалити рядки із найбільшою кількістю пропущених рядків, а всі інші рядки видалити, враховуючи значну кількість даних, що в цій роботі доступна. Проте, для виконання роботи по всіх стандартах пропущені дані будуть заповнені із використанням різних методів.

Для роботи із пропущеними значеннями було використано декілька методів:

1. KNNInputer
2. Mode inputing

KNN Imputer є одним з методів для заповнення пропущених значень в наборі даних. Цей метод використовує алгоритм K-найближчих сусідів (K-Nearest Neighbors) для визначення значень пропущених даних на основі найближчих сусідів з уже наявних даних.

Процес заповнення пропущених значень за допомогою KNN Imputer включає наступні кроки:

1. Обчислення відстаней – розрахунок відстаней між зразками з наявними значеннями і зразками з пропущеними значеннями.
2. Вибір найближчих сусідів – вибір K найближчих сусідів для кожного зразка з пропущеними значеннями на основі обчислених відстаней.
3. Заміна пропущених значень – заміна пропущених значень в зразках з прогнозованими значеннями, які отримані від K найближчих сусідів.

Основна ідея KNN Imputer полягає в тому, що зразки з подібними характеристиками мають схожі значення. Таким чином, використання інформації від найближчих сусідів допомагає зробити більш точні прогнози для пропущених значень.

Одним з важливих параметрів KNN Imputer є параметр K, який визначає кількість найближчих сусідів, які використовуються для прогнозу пропущених значень. Вибір оптимального значення K може впливати на якість заповнення пропущених даних. В нашому випадку кількість сусідів була визначена на рівні трьох.

KNN працює тільки із числовими даними, тому цей метод був застосований тільки до числових даних, що мають пропущені значення.

Далі до всіх категорійних даних був застосований метод Mode. Даний метод є одним з простих методів заповнення пропущених значень в наборі даних. Він використовує найпоширеніше значення (моду) атрибута для заповнення відсутніх даних.

Процес заповнення пропущених значень за допомогою методу Mode включає наступні кроки:

1. Обчислення моди – обчислення найпоширенішого значення для кожного атрибута з наявними даними.
2. Заміна пропущених значень – заміна пропущених значень в атрибутах модою, тобто найпоширенішим значенням.

Основна ідея методу Mode полягає в тому, що найпоширеніше значення в атрибуті може бути репрезентативним для заповнення пропущених даних. Використання моди дозволяє замінити пропущені значення таким чином, щоб їх розподіл залишався близьким до розподілу наявних значень.

При використанні цього методу варто враховувати, що Mode не враховує іншу інформацію, яка може бути корисною для заповнення пропущених даних. Крім того, якщо атрибут має декілька мод, може бути важко визначити найбільш репрезентативний варіант для заповнення.

Після заповнення таблиці із даними Heatmap отримав такий вигляд, як вказано на Рис. 3.2:

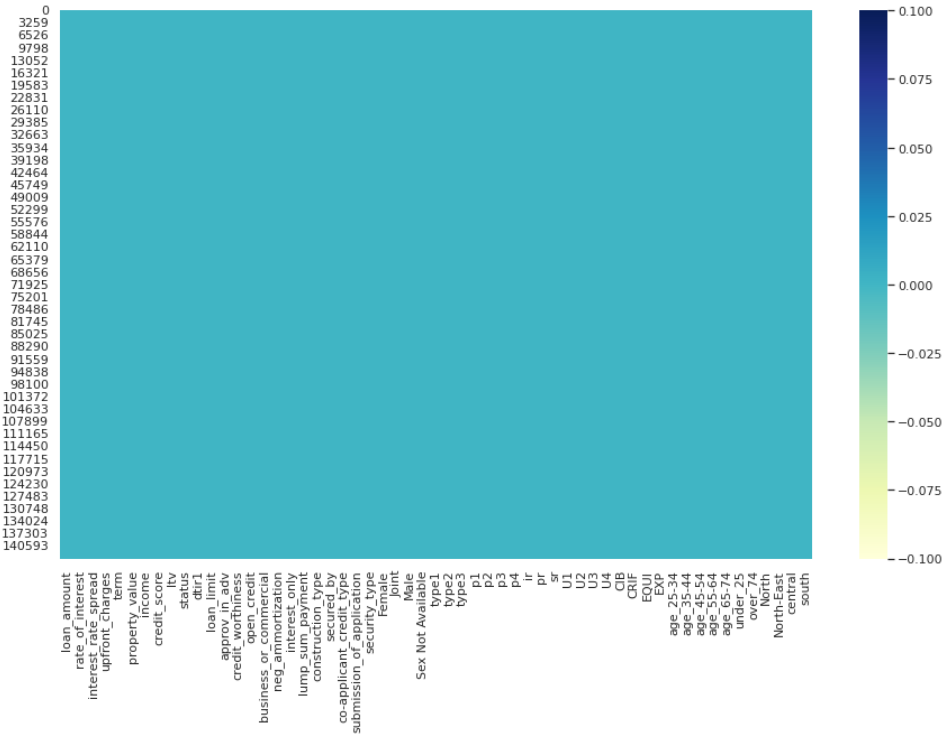


Рисунок 3.2 – Heatmap таблиці з даними після заповнення пропущених значень

3.3 Видалення викидів

Видалення викидів є одним з методів обробки викидів (outliers) в наборі даних. Викиди – це значення або спостереження, які суттєво відрізняються від інших даних в наборі, тому якщо їх залишати та використовувати в аналізі – можуть значно вплинути на достовірність результатів. Вони можуть виникати через помилки вимірювання, випадкові аномалії або систематичні збої.

Одним з поширених підходів до видалення викидів є використання статистичних методів, таких як правило трьох сигм (3-сигма правило) або міжквартильний розмах (IQR). За цими методами викиди визначаються як

значення, які знаходяться за межами певного діапазону, що базується на статистичних характеристиках даних. IQR обраховується за формулою:

$$IQR = Q3 - Q1,$$

де Q1 представляє перший квартиль (25-й процентиль), а

Q3 – третій квартиль (75-й процентиль)

Видалення викидів може бути корисним для покращення якості аналізу та моделювання даних. Викиди можуть впливати на розподіл даних, статистичні міри центру та розмаху, а також на побудовані моделі. Видалення викидів може покращити стабільність та достовірність аналізу, а також забезпечити кращі результати передбачення або класифікації. В нашому випадку було використано саме метод IQR.

Після обрахування квартилів та проведення визначення викидів було видалено 4812 рядків із датасету, що становить 3% від оригінального датасету. В результаті обробки в датасеті залишилось пропорційно 0.967% даних, що досі залишається достатньою кількістю, адже вибраний датасет досить великий, як я зазначала вище.

3.4 Дослідження даних на кореляцію

Дослідження даних на кореляцію є важливим етапом в аналізі даних, який дозволяє виявити зв'язки та взаємозв'язки між різними атрибутами або змінними в наборі даних. Кореляція вимірює ступінь лінійної залежності між двома змінними, допомагаючи з'ясувати, чи існує певний шаблон або тенденція між ними.

Дослідження кореляції може допомогти відповісти питання чи існує позитивний або негативний зв'язок між двома змінними, яка сила кореляції

між двома змінними, які атрибути найбільше впливають на цільову змінну та чи можна прогнозувати значення однієї змінної на підставі іншої змінної.

Дослідження кореляції часто використовує теплові карти, щоб візуалізувати матрицю кореляції між змінними.

Теплова карта, що демонструє кореляцію для обраного датасета виглядає таким чином як на Рис. 3.3:

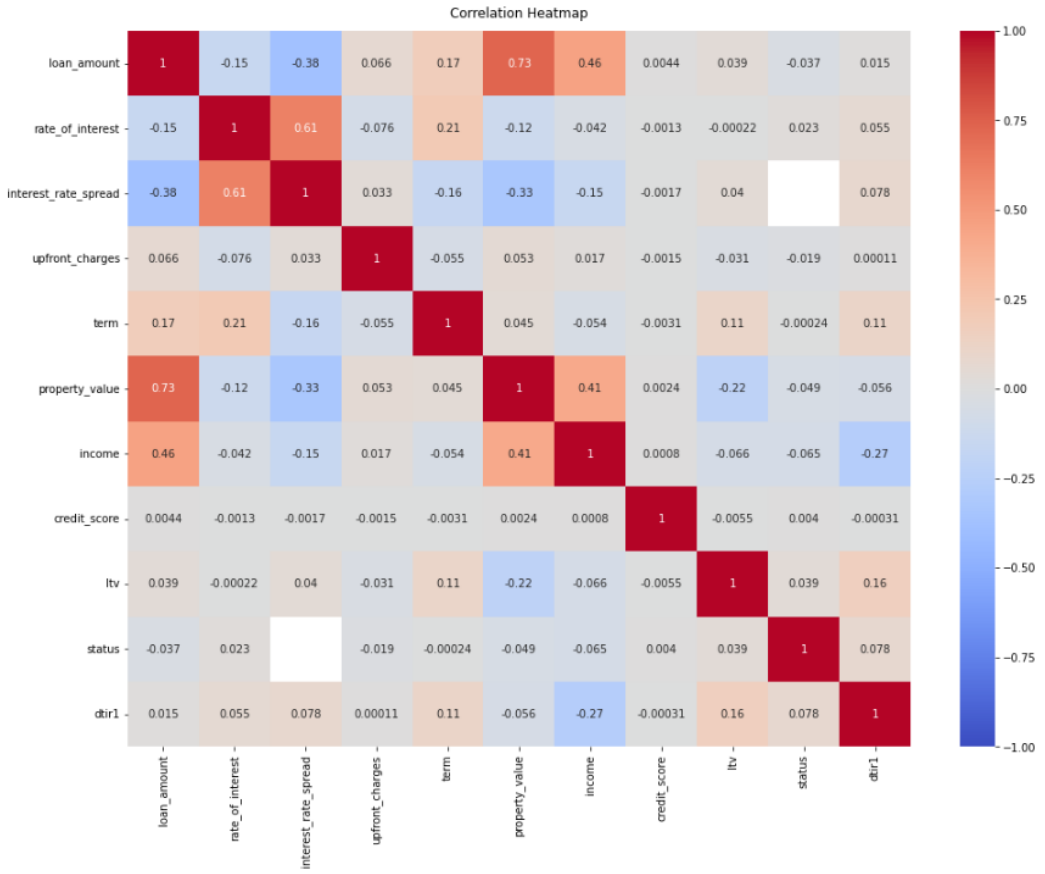


Рисунок 3.3 – Кореляційна матриця для даних з датасету

Отже, на основі рисунку вище, можна побачити, що ми не маємо якихось неочікуваних результатів в тепловій карті. Цільова змінна Status, що демонструє наявність дефолту у клієнта не демонструє помітних кореляцій із іншими атрибутами, а отже, додаткових атрибутів позбуватись необхідності немає.

3.5 Аналіз даних на розповсюдження

В першу чергу слід дослідити розподіл цільової змінної Status, щоб дослідити дисбаланс та помітити проблеми, що можуть виникнути на основі цього (Рис. 3.4).

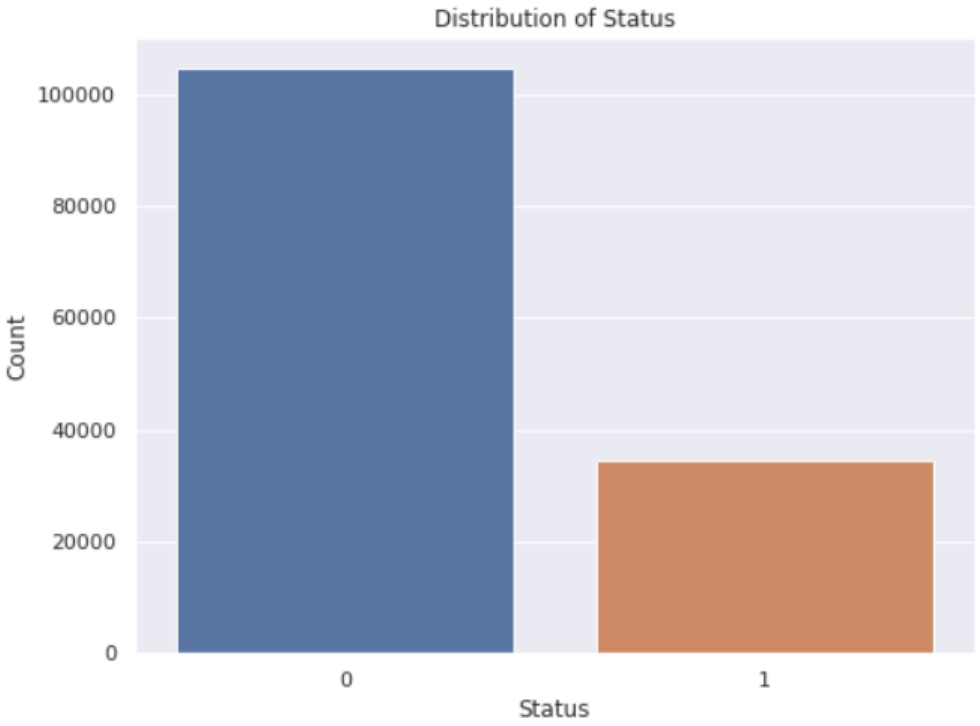


Рисунок 3.4 – Розподіл цільових даних в датасеті

Як видно із графіку вище, в датасеті присутній значний дисбаланс на цільовому атрибуті. Це призводить до упередженості моделі до більшого класу, що стає причиною меншої точності моделі в результаті. Тому в майбутньому при обробці даних буде необхідно вирішити цю проблему.

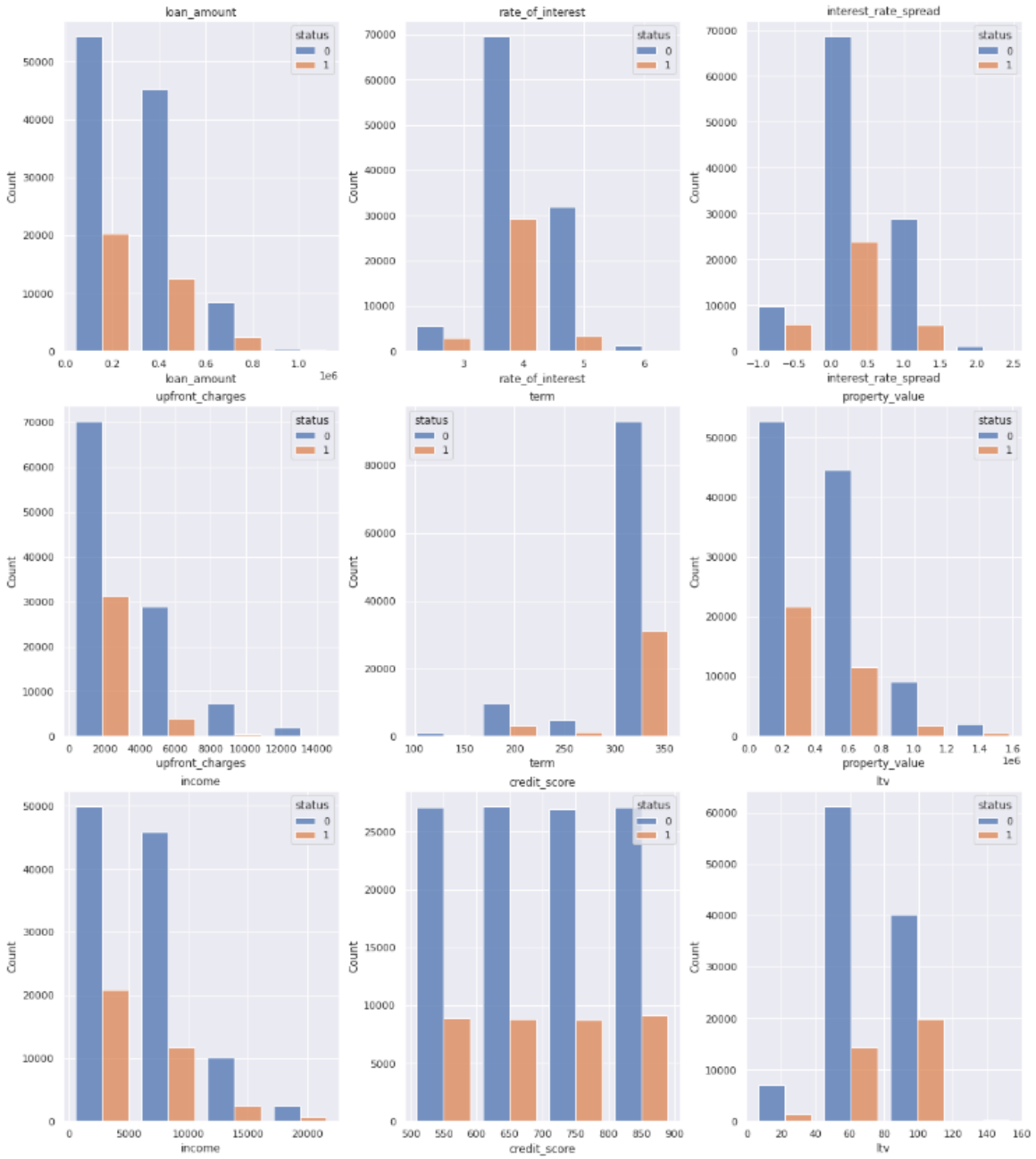


Рисунок 3.5 – Розподіл категорійних даних в залежності від цільової змінної

Як бачимо на графіках (Рис. 3.5), розподіл по всіх категоріях приблизно рівний, отже, значних зміщень в сторону якогось атрибуту і кореляції між цільовим та категорійним атрибутом не спостерігається.

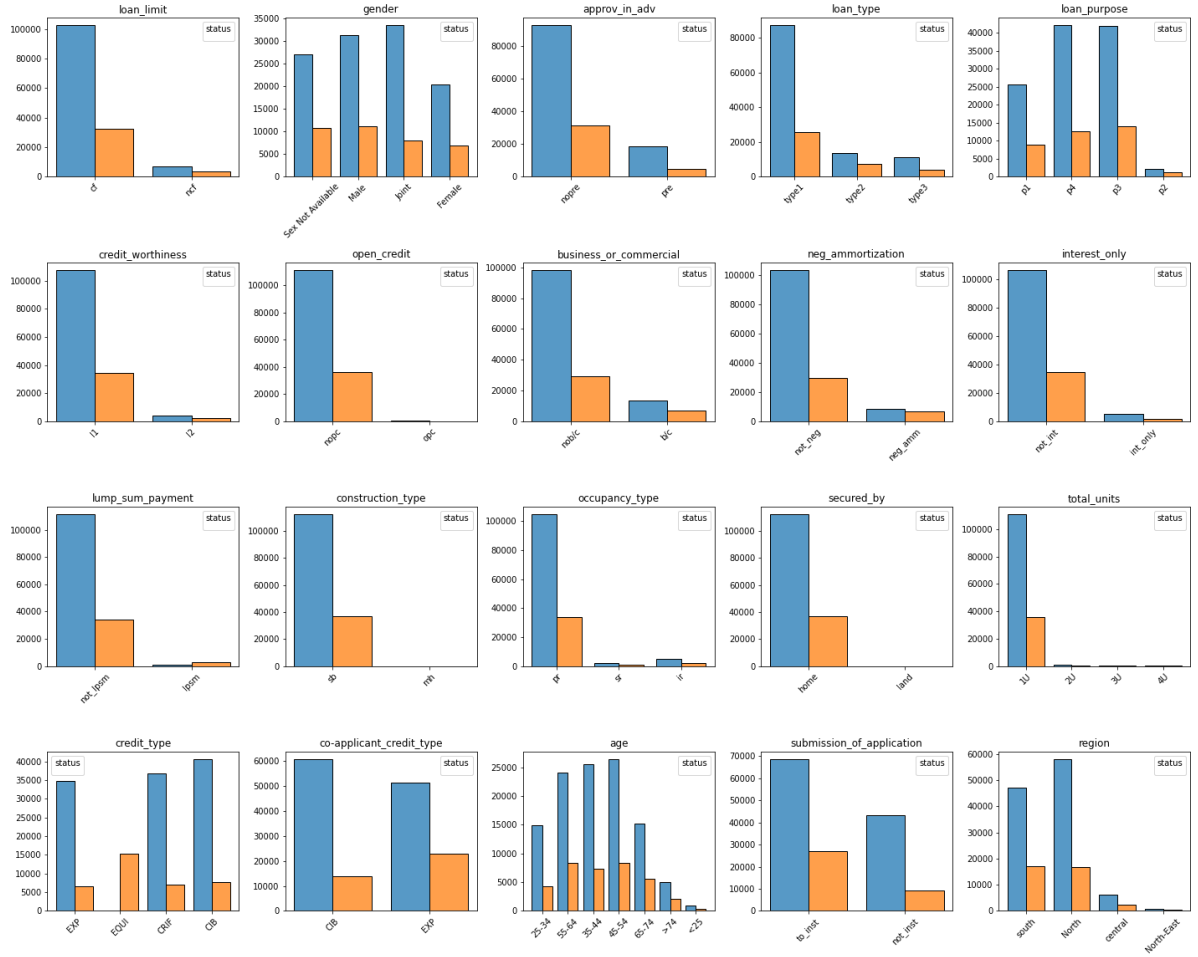


Рисунок 3.6 - Розподіл категоріальних змінних за цільовим атрибутом

Як видно із графіків вище (Рис. 3.6), в даному випадку значних перекосів в сторону якогось класу, не враховуючи розподіл цільових змінних, не відбувається.

3.6 Приведення даних в числовий формат

В першу чергу при обробці даних було проведено їх розділення на дані для бінаризації, послідовного енкодінгу та OneHot методу.

Для бінаризації були обрані колонки:

1. security_type
2. submission_of_application
3. co-applicant_credit_type
4. secured_by
5. lump_sum_payment
6. interest_only
7. neg_ammortization
8. construction_type
9. business_or_commercial
10. open_credit
11. credit_worthiness
12. approv_in_adv
13. loan_limit
14. status

Категорійні колонки були визначені автоматично, до них увійшли:

1. id
2. loan_limit
3. gender
4. approv_in_adv
5. loan_type
6. loan_purpose
7. credit_worthiness
8. open_credit
9. business_or_commercial
10. neg_ammortization
11. interest_only
12. lump_sum_payment

13.construction_type
14.occupancy_type
15.secured_by
16.total_units
17.credit_type
18.co-applicant_credit_type
19.age
20.submission_of_application
21.region
22.security_type

Колонки для бінаризації були закодовані із використанням методу LabelEncoding, що послідовно призначає категоріям свої числа та записує їх в один атрибут. Категорійні колонки були закодовані із використанням методу OneHot, що створює новий атрибут для кожного унікального значення в колонці та ставить 1 там, де ця колонка відповідає атрибуту.

3.7 Збалансування класів

Для розв'язання проблеми незбалансованості класів у задачах класифікації існує кілька підходів. Один з таких підходів – використання методу Random Under Sampling за допомогою бібліотеки Imbalanced-learn, що спеціалізується на роботі зі незбалансованими наборами даних.

Метод Random Under Sampling полягає у випадковому видаленні зразків з більшої кількості класу, щоб досягти балансу між класами. Це означає, що зразки з більш чисельного класу будуть випадково видалені до рівня зразків менш чисельного класу. Таким чином, отримується нова

підвибірка даних, в якій кількість зразків у кожному класі стає рівною. Із огляду на наш випадок, де кількість даних є доволі значною, використання методів андерсемплінгу є більш оптимальним, ніж апсемплінгу, оскільки вони не спотворюють дані.

Після використання методу RandomUnderSampling баланс класів у наборі почав виглядати таким чином (Рис 3.7):

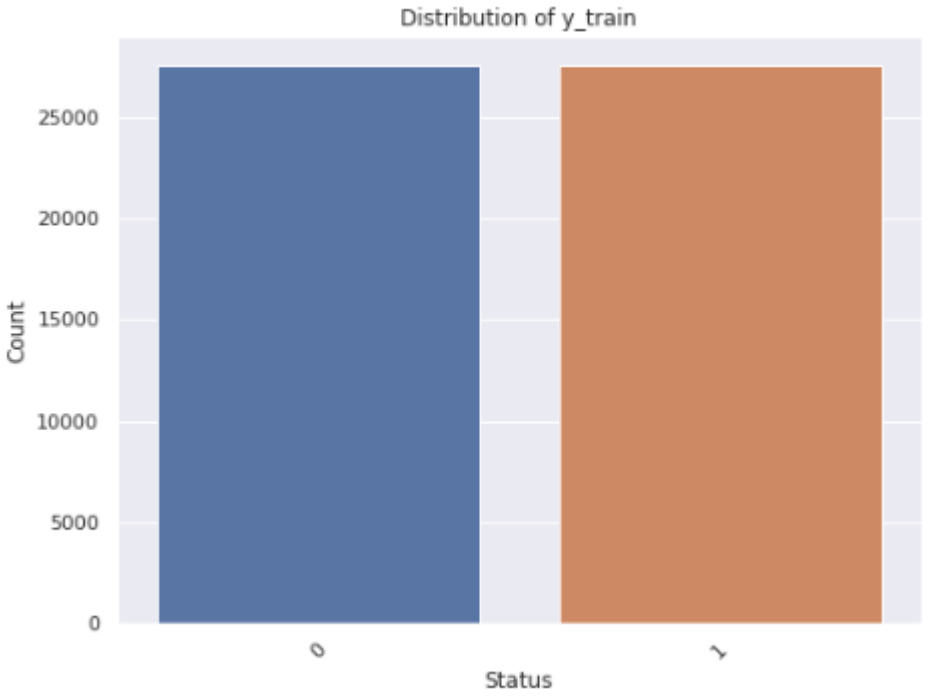


Рисунок 3.7 – Вигляд даних після збалансування

3.8 Нормалізація даних та їх розділення

Нормалізація даних – це процес приведення значень різних атрибутів до спільного діапазону, зазвичай від 0 до 1. Це робиться для забезпечення однорідності і порівнянності атрибутів у моделях машинного

навчання. Один з популярних методів нормалізації даних – це метод Min-Max Scaling.

Метод Min-Max Scaling, також відомий як метод мінімаксного масштабування, приводить значення атрибутів до діапазону від 0 до 1. Цей метод використовує формулу:

$$X\_norm = \frac{(X - X\_min)}{(X\_max - X\_min)},$$

де X_norm – нормалізоване значення,

X – вихідне значення атрибута,

X_min – мінімальне значення атрибута,

X_max – максимальне значення атрибута.

Метод Min-Max Scaling добре підходить для атрибутів, які мають відомий межовий діапазон, і забезпечує збереження відносних пропорцій між значеннями атрибутів. Він корисний у випадках, коли значення атрибутів мають різну шкалу, і необхідно привести їх до спільного діапазону.

Після нормалізації в даних було визначено цільовий атрибут y та тренувальні дані X. Потім дані було розбито на дві частини для тренування та для тестування роботи моделі. Дані для тестування були незбалансованими, щоб максимально повторювати реальні умови, в той час, як тренувальні дані були збалансованими задля запобігання упередженості моделі між класами.

3.9 Висновки до розділу

У цьому розділі було проведено аналіз та обробку набору даних для прогнозування ризиків у банківській сфері. Почала я з вибору та опису набору даних, що містить різні атрибути про кредитних позичальників.

Далі я розглянула процес обробки пропущених даних. Застосувала різні стратегії, такі як заповнення недостаючих значень методом моди, середнім значенням та видалення рядків з пропущеними даними. Це дозволило мені забезпечити повноту даних та уникнути втрати важливої інформації.

Потім я перейшла до видалення викидів. Використовуючи метод міжквартильного розмаху, я ідентифікувала аномальні значення у числових атрибутах та видалила їх з набору даних. Це допомогло покращити якість даних та уникнути негативного впливу викидів на аналіз та моделювання.

Далі я провела дослідження даних на кореляцію, виявивши зв'язки між різними атрибутами. Це дозволило мені отримати уявлення про взаємозв'язок між змінними та їх вплив на цільову зміну.

Також я зайнялася приведенням даних у числовий формат, що було необхідним для подальшого моделювання та аналізу.

Для забезпечення збалансованості класів я використала метод RandomUnderSampling, який дозволив зменшити перевагу більшого класу та забезпечити рівновагу між класами.

На завершення розділу я провела нормалізацію даних за допомогою методу MinMaxScaling, що привело значення атрибутів до спільного діапазону та забезпечило стабільність моделей при роботі зі змішаними шкалами даних.

В результаті проведених процедур обробки та аналізу даних я підготували набір даних для подальшого моделювання та прогнозування ризиків у банківській сфері.

4 СТВОРЕННЯ СИСТЕМИ ПРОГНОЗУВАННЯ БАНКІВСЬКИХ РИЗИКІВ

У цьому розділі я розгляну процес створення системи прогнозування банківських ризиків. Почну я з вибору методів та алгоритмів прогнозування, які найбільш підходять для вирішення даної задачі. Розгляну різні підходи та алгоритми, такі як логістична регресія, дерева рішень, випадковий ліс, градієнтний бустінг та нейронні мережі, та проаналізую їхні переваги та недоліки.

Далі перейду до побудови моделі прогнозування кредитних ризиків. Розгляну процес вибору методів навчання, таких як навчання з вчителем та навчання без вчителя, та оберу найбільш підходящі методи для нашої задачі. Оціню ефективність моделі за допомогою метрик оцінки та проведу перехресну перевірку, щоб забезпечити її стійкість та здатність до узагальнення.

Після цього я проведу випробування та аналіз результатів системи прогнозування банківських ризиків. Оціню точність та надійність моделей, проаналізую їх здатність до передбачення ризикованих ситуацій та зроблю висновки щодо їх ефективності.

Потім я створю ансамблеву модель на основі всіх методів та порівняю результати із кожною окремою моделлю в індивідуальному порядку.

Описавши можливості та обмеження системи прогнозування банківських ризиків, я підсумую результати моєї роботи в цьому розділі. Висвітливши переваги та потенційні обмеження системи, зроблю висновки щодо її використання та можливостей подальшого вдосконалення.

В результаті розділу 4 я матиму глибоке розуміння системи прогнозування банківських ризиків, її потенціалу та можливостей, а також мату практичні результати та рекомендації щодо подальшого розвитку системи.

4.1 Вибір методів та алгоритмів прогнозування ризиків в банківській сфері.

У цьому пункті я розгляну різні методи та алгоритми, які можна використовувати для прогнозування ризиків в банківській сфері. З урахуванням особливостей мого датасету та мети прогнозування, я проведу аналіз різних підходів і оберу найбільш ефективні для моєї задачі.

Під час проведення дослідження в розділі 2 вибір пав на використання моделей випадкових лісів, логістичної регресії, нейромереж та градієнтного бустингу.

В додаток до цих моделей я скористаюсь методом Stacking для порівняння всіх моделей та використаю логістичну регресію як класифікатор на виході, що буде підбирати коефіцієнти для кожної моделі та оцінювати їх роботу за точністю передбачень.

Метод стекінгу є одним з популярних ансамблевих методів машинного навчання. Він поєднує прогнози кількох базових моделей, використовуючи їх як вхідні дані для фінальної моделі, яка здійснює остаточне прогнозування. Ідея стекінгу полягає в тому, що фінальна модель навчається використовуючи прогнози базових моделей як нові ознаки.

Процес стекінгу можна поділити на два основних кроки. Першим кроком є навчання базових моделей на вихідних даних. Кожна базова

модель може використовувати різні алгоритми та параметри. Наступним кроком є створення набору прогнозів базових моделей на невиданих даних. Ці прогнози стають новими ознаками, які використовуються для навчання фінальної моделі.

Фінальна модель може бути будь-яким алгоритмом машинного навчання, таким як логістична регресія, випадковий ліс або градієнтний бустінг. Вона навчається використовуючи набір прогнозів базових моделей, які стали новими ознаками. Це дозволяє фінальній моделі використовувати складніші залежності в даних та здійснювати більш точне прогнозування. В моєму випадку буде використана логістична регресія, оскільки вона дозволяє оцінювати важливість кожної моделі лінійним способом і може бути з легкістю інтерпретована.

Однією з переваг методу стекінгу є його здатність до комбінування прогнозів різних моделей, що дозволяє отримати кращу прогностичну точність. Крім того, стекінг може допомогти зменшити проблему перенавчання і покращити стійкість моделі до шуму в даних.

Проте, метод стекінгу також має свої обмеження. Він може бути витратним з обчислювальної точки зору, оскільки потребує навчання кількох моделей та створення багатьох прогнозів. Крім того, ефективність стекінгу залежить від якості базових моделей та їхньої різноманітності. Недостатня різноманітність може призвести до незначного поліпшення прогнозів.

4.2 Опис процесу побудови моделей прогнозування кредитних ризиків, з урахуванням методів навчання та оцінки моделей

Для метрики точності було використано звичайну точність “accuracy” та побудовано матриці суперечностей, що дозволяють оцінити відразу Precision, Recall та побачити наочно упередженість моделі в сторону певного класу.

Accuracy (точність) – це вимірює загальну точність моделі у класифікації. Вона вказує на відношення кількості правильно класифікованих прикладів до загальної кількості прикладів у наборі даних. Формула для обчислення accuracy:

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

де TP (True Positive) – це кількість правильно виявлених позитивних класів,

TN (True Negative) – це кількість правильно виявлених негативних класів.,

FP (False Positive) – це кількість неправильно виявлених позитивних класів,

FN (False Negative) – це кількість неправильно виявлених негативних класів.

Precision (точність) – вимірює точність моделі у виявленні позитивних класів. Вона вказує на відношення кількості правильно виявлених позитивних класів (True Positives) до загальної кількості класифікованих як позитивні (True Positives + False Positives). Precision відповідає на питання: "Яка частка об'єктів, класифікованих як позитивні, є дійсно позитивними?". Формула для обчислення precision:

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

Recall (повнота) – вимірює здатність моделі знайти всі позитивні класи. Вона вказує на відношення кількості правильно виявлених позитивних класів (True Positives) до загальної кількості дійсних позитивних класів (True Positives + False Negatives). Recall відповідає на питання: "Яка частка дійсних позитивних класів була правильно виявлена?". Формула для обчислення recall:

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

Ці метрики допомагають зрозуміти, наскільки ефективно модель класифікує дані та є популярним вибором у сфері оцінки роботи моделей класифікації. Для різних ситуацій буває важливо збалансувати точність та повноту залежно від контексту задачі.

4.3 Випробування та аналіз результатів системи прогнозування банківських ризиків

В цьому пункті проводиться випробування системи прогнозування банківських ризиків та аналіз отриманих результатів. Після побудови моделі прогнозування ризиків на попередніх етапах дослідження, я перейшла до етапу перевірки її ефективності та точності.

Як було згадано в пункті 4.2, перевірка роботи моделей буде відбуватись із використанням метрики Precicion, Recall та Accuracy. Кожна з моделей буде оцінена на даних для тестування та їх результати будуть порівнянні між собою для визначення найбільш оптимального варіанту. В тому числі буде проведено тестування на моделі, що зіставляє роботу всіх

випробуваних моделей для перевірки результатів роботи ансамблевих моделей.

Рисунок 4.1 - Результати оцінки роботи моделей

Отже, як видно із рисунку 4.1, найкраще себе показали моделі випадкових лісів та градієнтного бустингу. Використання методу XGBoost є безперечним лідером в даному випробуванні, оскільки ця модель обганяє всі інші за точністю на значний проміжок. Від другого місця (RandomForest) до першого місця є проміжок в цілих 8%.

Інші дві моделі відзначились упередженістю стосовно першого класу. Це навряд є наслідком невірної обробки даних, оскільки дані були

збалансовані, а інші моделі показали себе значно краще. Вірогідно, такі результати є наслідками недоліків моделей та їхньої нездатності успішно апроксимувати, уловлювати нелінійні зв’язки та працювати із великими обсягами атрибутів на вході.

Далі була проведена оцінка роботи ансамблевої моделі на основі методу стакінгу, що поєднував всі моделі та використовував логістичну регресію як основу для проведення фінальної класифікації на основі передбачень всіх інших моделей (Рис. 4.2).

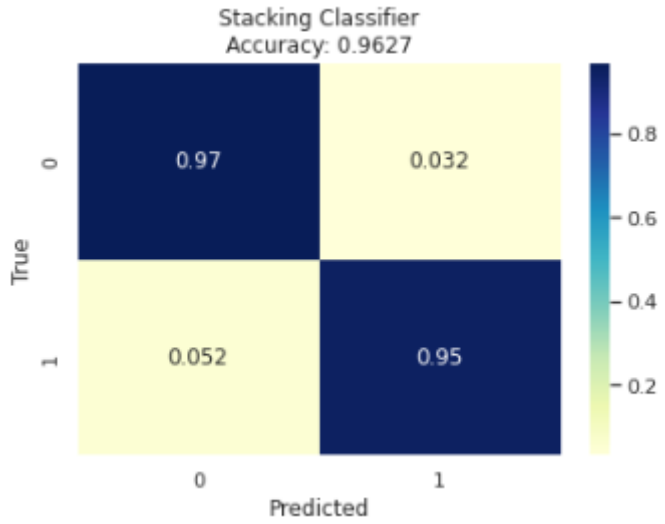


Рисунок 4.2 – Результати роботи StackingClassifier

Як видно із результатів, наведених вище на рисунку 4.2, модель StackingClassifier видає результати доволі близькі до результатів моделі XGBoost. Різниця між точністю двох моделей знаходиться в проміжку десятих відсотків, тому можна сказати, що вони показують рівні результати.

Далі було проведено виявлення важливостей для кожної моделі, що становили коефіцієнти, на які множились їх передбачення моделлю логістичної регресії. (Рис. 4.3)

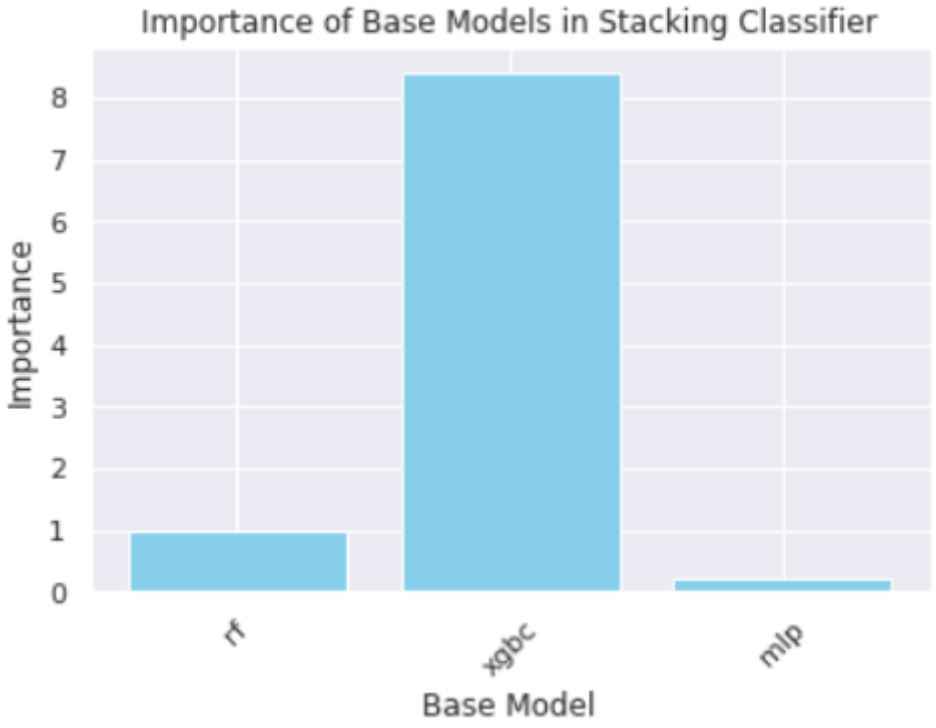


Рисунок 4.3 – Важливості моделей StackingClassifier

Отже, як і очікувалось, модель StackingClassifier віддала близько 90% ваги в рішеннях до алгоритму XGBoost, що видно на рисунку 4.3. Подібні ансамблеві збірки зазвичай показують кращі результати в купі з іншими моделями, близькими за результатами, що дозволяє позбуватись аномалій в їх роботі та перекривати недоліки одне одного, проте, в даному випадку результати роботи моделей зовсім не близькі, що і є наслідком таких результатів.

Далі була створена таблиця з результатами роботи всіх моделей для наглядності. (Табл. 4.1)

Таблиця 4.1 – Результати роботи моделей

Number	Model	Accuracy
0	Random Forest	0.885564
1	XGBoost	0.966596
2	Logistic Regression	0.596099
3	Neural Network	0.535146
4	Stacking Classifier	0.962681

4.4 Опис можливостей та обмежень систем прогнозування банківських ризиків

Розроблені моделі прогнозування банківських ризиків значною мірою залежать від якості даних, які надаються для тренування. Якість та достовірність даних впливають на точність та надійність моделей. Чим більш чисті та повні дані, тим краще можна навчити модель розпізнавати ризикові ситуації та приймати адекватні рішення.

Однак, коли моделі вже розроблені та навчені, вони можуть працювати швидко та в режимі реального часу. Це означає, що вони

можуть швидко аналізувати вхідні дані та надавати прогнози щодо кредитних ризиків на основі навчених правил та алгоритмів.

Крім того, розроблені моделі можуть бути покращені з часом. Шляхом дотренування моделей на нових даних, вони можуть покращувати свою точність та надійність. Це дозволяє моделям адаптуватись до змінних умов та змінюючихся трендів у банківській сфері.

Найкраща розроблена модель в даному дослідженні має вражаючу точність на рівні 96,6%. Це досить високий результат, що свідчить про високу якість моделі та її здатність до ефективного прогнозування банківських ризиків. Проте, варто зазначити, що точність моделі може варіюватись залежно від специфіки даних та умов їх використання. Тому важливо продовжувати спостерігати та оцінювати роботу моделей, а також вдосконалювати їх з часом.

4.5 Висновки до розділу

У цьому розділі було розглянуто процес створення системи прогнозування банківських ризиків. Були вибрані методи та алгоритми прогнозування, які найкраще відповідають вимогам банківської сфери. Описано процес побудови моделей прогнозування кредитних ризиків, включаючи вибір методів навчання та оцінки моделей.

Після побудови моделей було проведено випробування та аналіз результатів системи прогнозування банківських ризиків. Результати виявилися досить обнадійливими, з високою точністю та надійністю прогнозів. Моделі продемонстрували здатність ефективно розпізнавати ризикові ситуації та приймати адекватні рішення.

Важливим кроком було також описання можливостей та обмежень систем прогнозування банківських ризиків. Розроблені моделі мають деякі обмеження, пов'язані з якістю наданих даних та специфікою їх використання. Однак, вони також мають значний потенціал та можуть бути дотреновані з часом на нових даних для подальшого покращення результатів.

Найкраща розроблена модель демонструє дуже високу точність, досягаючи 96,6%. Це підтверджує успішність побудованих моделей та їх здатність до ефективного прогнозування банківських ризиків. Проте, важливо продовжувати вдосконалювати систему та враховувати її можливості та обмеження для забезпечення надійності та актуальності прогнозів.

5 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ РОЗРОБКИ ЗАПРОПОНОВАНОЇ СИСТЕМИ

У даному розділі проводиться оцінка основних характеристик системи аналізу та прогнозування кредитних ризиків у банківській сфері. Метою цього дослідження є створення програмного продукту, який забезпечує якісне дослідження та аналіз кредитних ризиків не лише в Україні, але й у всьому світі.

У рамках дослідження розглядаються різні варіанти реалізації системи, зокрема використання функціонально-вартісного аналізу для вибору оптимальної стратегії та забезпечення економічної ефективності системи. Функціонально-вартісний аналіз дозволяє оцінити реальну вартість продукту або послуги, незалежно від організаційної структури компанії. Це допомагає виявити резерви зниження витрат та забезпечити оптимальне співвідношення між споживчою вартістю та витратами на розробку та функціонування системи.

Дослідження показало, що розроблені моделі аналізу та прогнозування кредитних ризиків залежать від якості наданих даних для тренування. Вони демонструють швидку та в прямому часі роботу, а також здатність до дотренування на нових даних для покращення результатів. Найкраща модель, яка має точність на рівні 96,6%, є доказом успішності системи прогнозування кредитних ризиків.

5.1 Постановка задачі

Дослідження використовує метод функціонально-вартісного аналізу (ФВА) для проведення техніко-економічного аналізу системи прогнозування стійкості фінансових показників. Оскільки рішення, пов'язані з проектуванням та реалізацією компонентів системи, мають вплив на всю систему в цілому, кожна окрема підсистема повинна задовольняти вимоги системи в цілому. Тому фактичний аналіз полягає в аналізі функцій програмного продукту, який призначений для збору, обробки та аналізу даних компанії.

Технічні вимоги до програмного продукту включають наступні аспекти:

1. Функціонування на персональних комп'ютерах із стандартним набором компонентів. Програмний продукт повинен бути сумісним зі стандартними комп'ютерними конфігураціями та операційними системами, що використовуються на багатьох персональних комп'ютерах. Це дозволить забезпечити доступність продукту для широкого кола користувачів без необхідності використання спеціалізованого обладнання.

2. Зручність та зрозумілість для користувача. Програмний продукт повинен мати інтуїтивний та легкий у використанні інтерфейс, що дозволить користувачам швидко оволодіти його функціоналом та виконувати необхідні завдання без зайвих зусиль. Це забезпечить зручність та задоволення користувачів в процесі використання програмного продукту.

3. Швидкість обробки даних та доступ до інформації в реальному часі: Програмний продукт повинен забезпечувати швидку обробку великого обсягу даних та миттєвий доступ до актуальної інформації. Це особливо важливо в контексті прогнозування кредитних ризиків, де

швидка реакція на зміни та актуальні дані є критичними для прийняття вірних рішень.

4. Можливість зручного масштабування та обслуговування. Програмний продукт повинен бути гнучким і легко масштабовуватись відповідно до зростаючих потреб бізнесу. Крім того, він повинен бути легко обслуговуватись, включаючи можливість внесення змін, виправлення помилок та покращення функціоналу без значних труднощів.

5.2 Обґрунтування функцій програмного продукту

Основна мета розділу полягає у розробці програмного продукту, який здатний аналізувати різні характеристики, що впливають на стійкість підприємства. В цьому контексті розглядаються наступні функції:

1. Вибір мови програмування (F1). Передбачається вибір самої програми для аналізу. Варіантами реалізації є використання Python або R, що забезпечують потрібні інструменти та функціонал для проведення аналізу.
2. Якісний аналіз даних та впровадження нових атрибутів (F2). Ця функція передбачає проведення детального аналізу даних та відкриття нових атрибутів, що допоможуть із передбаченнями. Це може бути реалізовано шляхом використання вбудованих функцій програмного продукту або створення власних обчислень та методів для отримання значень, проведення додаткових досліджень.
3. Методи реалізації моделі (F3). В рамках цієї функції розглядається моделювання систем, що будуть здатні оцінювати ризики в сфері кредитування клієнтів. Можливими

варіантами реалізації є використання готових бібліотек, що надаються програмним продуктом, або створення власних прототипів з урахуванням конкретних потреб та особливостей наявних даних.

Результатом виконання цих функцій є розроблений програмний продукт, який дозволяє здійснювати аналіз стійкості підприємства на основі вхідних даних та надає зручний інтерфейс для роботи з результатами аналізу.

Функція F1 включає такі варіанти:

А) Використання мови програмування Python

Б) Використання мови програмування R

Функція F2 включає такі варіанти:

А) Проведення детального аналізу даних та дослідження нових атрибутів

Б) Використання наявних даних без проведення додаткових досліджень

Функція F3 включає такі варіанти:

А) Використання готових модулів

Б) Побудова власних алгоритмів з нуля

Варіанти реалізації функцій наведені у морфологічній карті (Рис. 5.1).

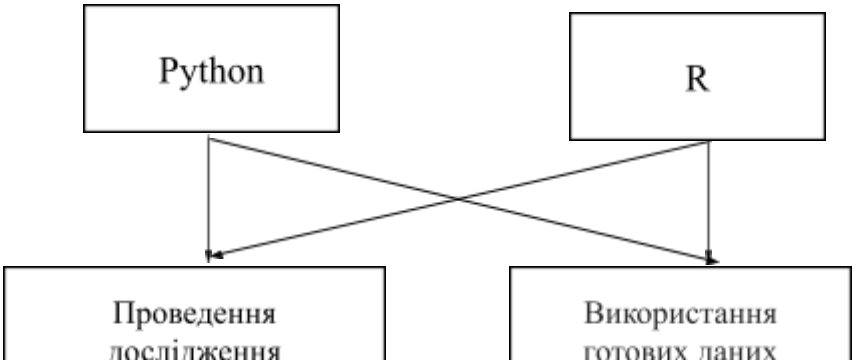


Рисунок 5.1 – Морфологічна карта

Морфологічна карта відображає різні комбінації основних функцій, що можуть бути реалізовані в програмному продукті. Кожна комбінація представляє собою унікальний спосіб використання функцій для досягнення певних цілей. Морфологічна карта допомагає визначити набір функцій, які найкраще відповідають потребам проекту.

Позитивно-негативна матриця, показана в таблиці 5.1, відображає оцінку кожної комбінації функцій за позитивними та негативними аспектами.

Таблиця 5.1 – Позитивно-негативна таблиця для функцій та вірантів реалізації

Функція	Варіанти реалізації	Переваги	Недоліки
F1	А	Простота використання, Високий рівень інтегрованості	Менший функціонал
	Б	Більш потужний та широкий функціонал	Менший рівень інтегрованості
F2	А	Шанс збільшити точність системи	Великі фінансові та часові вкладання
	Б	Більша швидкість використання Достатньо високий рівень поточної системи	Потенційні втрати від більших помилок системи
F3	А	Швидкість використання Нижчий необхідний рівень компетенції	Менший простір до кастомізації
	Б	Більша пристосованість до поточної задачі	Значно більші розходи на розробку

Функція F1 має два варіанти реалізації. Варіант А відрізняється простотою використання та високим рівнем інтегрованості. Це дозволяє легко оволодіти програмою і використовувати її без зайвих зусиль. Варіант Б, натомість, має більш потужний та широкий функціонал, але менший рівень інтегрованості з іншими системами. Вибір між цими варіантами залежить від потреб користувача, деякі можуть віддати перевагу простоті використання, тоді як інші шукають більший функціонал.

Функція F2 також має два варіанти реалізації. Варіант А дає можливість збільшити точність системи, але вимагає значних фінансових та часових вкладень для досягнення цієї мети. Варіант Б, натомість, пропонує більшу швидкість використання, але забезпечує достатньо високий рівень поточної системи. Вибір між цими варіантами залежить від балансу між точністю і ресурсами, які можуть бути витрачені на поліпшення системи.

Функція F3 також має два варіанти реалізації. Варіант А відрізняється швидкістю використання та вимагає нижчого рівня компетенції для його використання. Однак, цей варіант має менший простір для кастомізації під конкретні потреби користувача. Варіант Б, натомість, є більш пристосованим до поточної задачі, але вимагає значно більші розходи на розробку. Вибір між цими варіантами залежить від специфічних вимог та можливостей фінансування проекту.

Таким чином, вибір конкретного варіанту залежатиме від масштабу проєкту, доступних ресурсів та вимог до гнучкості та контролю над розробкою. У нашому випадку, після уважного аналізу, деякі варіанти реалізації були відкинуті з розгляду. Це може бути пов'язано з обмеженими фінансовими ресурсами, недостатньою гнучкістю для наших конкретних потреб або недостатньою можливістю контролювати та підтримувати розробку цих варіантів. Остаточний вибір варіантів реалізації буде здійснено на основі зважених критеріїв, що враховують наші поточні потреби та ресурси, а саме:

1. F1 – Обрано варіант А, Python оскільки дана мова пропонує більшу інтегрованість, що є необхідною для розробки значних систем.
2. F2 – Може бути використано як А, так і Б варіанти.
3. F3 – Обрано варіант із використанням готових модулів, А, оскільки завдання є доволі класичним для них

Отже, в результаті було утворено два основні варіанти:

1. F1(А) - F2(А) - F3(А)
2. F1(А) - F2(Б) - F3(А)

5.3 Вибір параметрів для програмного продукту

На підставі проведеного аналізу, визначаються основні параметри вибору, які використовуватимуться для обчислення коефіцієнта технічного рівня програмного продукту. При оцінці програмного продукту будуть враховуватися наступні параметри:

X1 – швидкість мови програмування;

X2 – обсяг часу витраченого на обробку даних;

X3 – час, необхідний для навчання моделі, включаючи тестування різних архітектур та методів;

X4 – очікуваний обсяг програмного коду.

Значення цих параметрів (погано, посередньо та добре) визначаються з урахуванням вимог замовника та умов експлуатації програмного продукту. Для кожного параметра встановлюються критерії оцінювання, які допомагають визначити, наскільки виконується вимога та який рівень якості може бути досягнутий. Детальніше про ці параметри та їх оцінку можна знайти у таблиці 5.2.

Таблиця 5.2 - Параметри програмного продукту

Параметр	Позначення	Одиниці виміру	Значення		
			Погано	Посередньо	Добре
Швидкодія	X1	Операція/мс	50	90	200
Обсяг часу	X2	Год	90	70	40
Час для навчання	X3	Год	40	30	20
Очікуваний обсяг продукту	X4	Кількість рядків коду	1300	700	400

На основі таблиці вище, було побудовано графіки для кожного параметра. (Рис. 5.2)

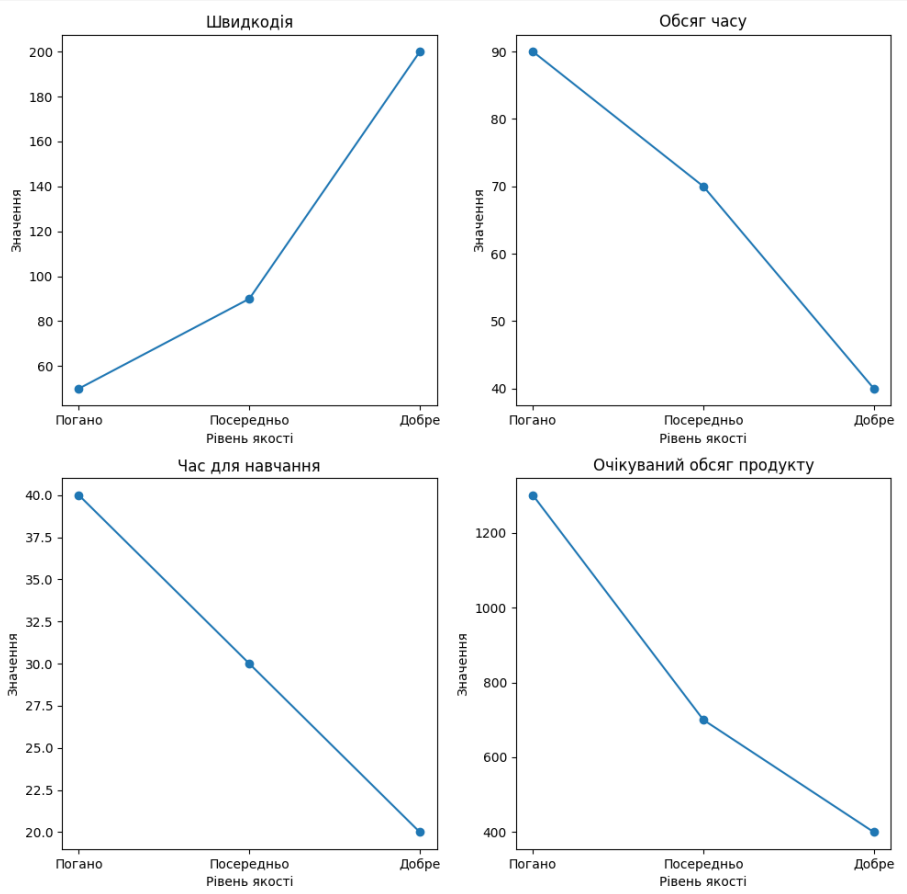


Рисунок 5.2 – Візуалізація для кожного параметра

5.4 Експертний аналіз параметрів

Після детального обговорення та аналізу кожен експерт оцінює ступінь важливості кожного параметра для розробки програмного продукту, спрямованого на досягнення найточніших результатів у моделюванні адаптивного прогнозування та обчисленні прогнозних значень. Визначення значимості кожного параметра здійснюється за допомогою методу попарного порівняння. Оцінку проводить експертна комісія, що складається з 7 осіб. Процес визначення коефіцієнтів значущості включає кілька етапів, включаючи встановлення рівня значимості параметра, перевірку придатності експертних оцінок, визначення попарного пріоритету параметрів та обробку результатів для визначення коефіцієнта значимості. Цей підхід систематизує процес оцінки важливості кожного параметра, надаючи наочну інформацію, яка є цінною для подальшого прийняття рішень та налаштування моделей адаптивного прогнозування.

Таблиця 5.3 – Результати оцінки параметрів від експертів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія	Операція/мс	3	4	3	4	4	3	4	25	-7,5	56,25
X2	Обсяг часу	Год	1	1	2	2	1	1	2	10	7,5	56,25
X3	Час для навчання	Год	2	2	1	1	2	2	1	11	6,5	42,25

Продовження таблиці 5.3												
X4	Очікуваний обсяг продукту	Кількість рядків коду	4	3	4	3	3	4	3	24	-6,5	42,25
	Сума		10	10	10	10	10	10	10	70	0	197

Для перевірки достовірності експертних оцінок визначимо наступні параметри:

Обчислимо суму рангів кожного з параметрів. Для цього додамо всі ранги, які були присвоєні експертами кожному параметру.

Обчислимо загальну суму рангів, використовуючи формулу (5.1), яка дозволяє знайти суму рангів для всіх параметрів разом.

Формула (5.1):

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70, \ (5.1)$$

де N – кількість експертів,

n – кількість параметрів.

Обчислимо середню суму рангів за формулою (5.2):

$$T = \frac{1}{n} R_{ij} = 17,5 \ (5.2)$$

Обчислимо відхилення суми рангів кожного параметра від середньої суми рангів за формулою (5.3):

$$\Delta_i = R_i - T \ (5.3)$$

Обчислимо загальну суму квадратів відхилень за формулою (5.4):

$$S = \sum_{i=1}^{N} \Delta_i^2 = 197. \ (5.4)$$

Порахуємо коефіцієнт узгодженості за формулою (5.5):

$$W = \frac{12S}{N^2\left(n^3-n\right)} = \frac{12 \cdot 197}{7^2\left(4^3-4\right)} = 0,698 > W_k = 0,80. \ (5.5),$$

де n – кількість параметрів. В даному випадку, n = 4.

Wk = 0.67 – нормативне значення коефіцієнта узгодженості.

Знайдений коефіцієнт узгодженості (W = 0.80) перевищує нормативне значення (Wk = 0.67), тому ранжування можна вважати достовірним. З використанням результатів ранжування проведемо попарне порівняння всіх параметрів, а результати занесемо у таблицю 5.4.

Таблиця 5.4 – Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	>	>	>	>	>	>	>	1,5
X1 і X3	>	>	>	>	>	>	>	>	1,5
X1 і X4	<	>	<	>	>	<	>	>	1,5
X2 і X3	<	<	>	>	<	<	>	<	0,5
X2 і X4	<	<	<	<	<	<	<	<	0,5
X3 і X4	<	<	<	<	<	<	<	<	0,5

Для визначення ступеня переваги одного параметра над іншим, числове значення aij визначається згідно з формулою (5.6):

$$a_{ij} = \{1.5 \text{ при } X_i > X_j \; 1.0 \text{ при } X_i = X_j \; 0.5 \text{ при } X_i < X_j . \quad (5.6)$$

З отриманих числових оцінок переваги складається матриця A=$\| a_{ij} \|$.

Для кожного параметра розраховується вагомість Кві згідно з наступними формулами:

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i} \quad (5.7)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \quad (5.8)$$

Відносні оцінки розраховуються декілька разів, поки наступні значення не відрізняються від попередніх менше ніж на 2%. На наступних кроках відносні оцінки розраховуються згідно з наступними формулами і запишемо дані в таблицю 5.5:

$$K_{\text{ві}} = \frac{b_i^{'}}{\sum_{i=1}^{n} b_i^{'}}, \quad (5.9)$$

$$b_i^{'} = \sum_{i=1}^{N} a_{ij} b_j \quad (5.10)$$

Таблиця 5.5 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер		Четверта ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$	b_i^3	$K_{\text{ві}}^3$
X1	1	1,5	1,5	1,5	5,5	0,36	30,25	0,48	166,375	0,57	915,0625	0,59
X2	0,5	1	0,5	0,5	2,5	0,16	6,25	0,33	15,625	0,05	40,625	0,048
Продовження таблиці 5.5												

X3	0,5	1,5	1	0,5	2,5	0,16	6,25	0,51	15,625	0,05	40,625	0,048
X4	0,5	1,5	1,5	1	4,5	0,3	20,25	0,32	91,125	0,31	410,0625	0,33
Всього:					15	1	63	1	288,75	1	1406,375	1

З таблиці 5.5 видно, що різниця між значеннями коефіцієнтів вагомості не перевищує 2%, тому додаткові ітерації не потрібні.

5.5 Аналіз якості реалізації варіантів функцій

Для оцінки ефективності виконання основних функцій програмного продукту, розглядаються абсолютні значення різних параметрів, таких як об'єм пам'яті (X2), час для навчання (X3) та потенційний обсяг програмного коду (X4), які відповідають вимогам технічних умов для його оптимального функціонування. Додатково, оцінюється параметр швидкості роботи мови програмування (X1), який має важливе значення.

Для визначення показників якості кожної варіації реалізації програмного продукту використовується коефіцієнт технічного рівня (ККj), який враховує вагомість кожного параметра.

$$K_K(j) = \sum_{i=1}^{n} K_{\text{в}i,j} B_{i,j}, \quad (5.11)$$

де n – кількість параметрів;

Кві – коефіцієнт вагомості і-го параметра;

Ві – оцінка і-го параметра в балах.

Згідно з формулою (5.11), кожен показник якості обчислюється шляхом перемноження вагомості параметра (Кві) на його оцінку (Bi) в балах. Результати цих розрахунків представлені в таблиці 5.6, що надає оцінку якості різних варіантів реалізації основних функцій програмного продукту.

Таблиця 5.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Парамет ри	Абсолютне значення параметра	Бальна оцінка парамет ра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	А	X1	87	25	0,59	14,75
	Б	X2	100	10	0,05	0,5
F2	А	X3	25	11	0.05	0,55
F4	А	X4	25	24	0,04	7,92

$$K_K = K_{\text{ТУ}}\left[F_{1k}\right] + K_{\text{ТУ}}\left[F_{2k}\right] + \ldots + K_{\text{ТУ}}\left[F_{zk}\right], \quad (5.12)$$

$$K_{K1} = 14{,}75 + 0{,}5 + 7{,}92 = 23{,}17;$$

$$K_{K2} = 14{,}75 + 0{,}55 + 7{,}92 = 23{,}22.$$

Отже, в результаті роботи кращим є варіант 1, тому що він має незначним чином вищий рівень якості.

5.6 Економічний аналіз розробки

Для визначення вартості розробки програмного продукту, спочатку здійснюється розрахунок трудомісткості для кожного варіанта виконання основних завдань, таких як розробка проекту програмного продукту та розробка програмної оболонки.

Для обчислення загальної трудомісткості використовується формула (5.13):

$$Т_О = Т_Р \cdot К_П \cdot К_{СК} \cdot К_М \cdot К_{СТ} \cdot К_{СТ.М}, \quad (5.13),$$

де ТР – трудомісткість розробки ПП,

КП – поправочний коефіцієнт,

КСК – коефіцієнт на складність вхідної інформації,

КМ – коефіцієнт рівня мови програмування,

КСТ – коефіцієнт використання стандартних модулів і прикладних програм,

КСТ.М – коефіцієнт стандартного математичного забезпечення.

Загальна трудомісткість обчислюється шляхом сумування трудомісткості всіх завдань.

Для кожного завдання, залежно від його характеристик, обчислюється трудомісткість. Наприклад, для першого завдання, яке відноситься до розрахункового характеру і має ступінь новизни А, була обчислена трудомісткість в розмірі 30 людино-днів. Застосовуючи поправочний коефіцієнт КП = 1, який враховує вид нормативно-довідкової інформації, та коефіцієнт КСК = 1.6, який враховує складність контролю

вхідної та вихідної інформації, отримуємо загальну трудомісткість програмування першого завдання. Застосовуючи коефіцієнт КСТ = 0.9 для врахування використання стандартних модулів та КМ = 1, отримуємо:

$$T_1 = 30 \cdot 1 \cdot 1.6 \cdot 0.9 \cdot 1 = 43{,}2 \text{ людино-днів}$$

Аналогічні обчислення проводяться для другого завдання з використанням відповідних коефіцієнтів КМ та КСТ, де КСК буде нижче:

$$T_2 = 30 \cdot 1 \cdot 1.2 \cdot 0.9 \cdot 1 = 32{,}4 \text{ людино-днів}$$

Як видно з обчислень, друге завдання має більшу трудомісткість.

Для розробки програмного продукту залучається один програміст з окладом 42 000 грн та один спеціаліст з даних з окладом 52 000 грн. Середня заробітна плата обчислюється за формулою (5.14):

$$С_ч = \frac{М}{T_m \cdot t} \text{грн.}, \quad (5.14)$$

де М – місячний оклад працівників,

Tm – кількість робочих днів у місяць,

t – кількість робочих годин в день.

$$С_ч = \frac{52000+42000}{2 \cdot 21 \cdot 8} = 279,8 \text{ грн.} \quad (5.15)$$

Далі обчислюється середня заробітна плата за формулою (5.16):

$$СЗП = С_ч \cdot Т_i \cdot КД, \quad (5.16)$$

де Сч – величина погодинної оплати праці програміста,

Ті – трудомісткість відповідного завдання,

КД – норматив, що враховує додаткову заробітну плату (20%).

Заробітна плата працівників із урахуванням соціальних внесків, яка складає 22%, обчислюється:

Зарплатня робітників із соц. внеском, що становить 22% дорівнює:

1. $С_{ЗП} = 279,8 \cdot 43,2 \cdot 24 \cdot 1.22 = 353\ 917,9$ грн.`
2. $С_{ЗП} = 279,8 \cdot 32,4 \cdot 24 \cdot 1.2\ 2 = 265\ 438,4$ грн.

Один засіб обчислення вартості розробки програмного продукту передбачає розрахунок різних складових. Візьмемо увагу, що одна електронно-обчислювальна машина (ЕОМ) обслуговує одного програміста із окладом 42 000 грн. з коефіцієнтом зайнятості 0,2. Перш за все, розраховуємо сумарну годинну вартість роботи (СГ), враховуючи додаткову заробітну плату:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 42\ 000 \cdot 0,2 = 100\ 800 \text{ грн.}$$

З урахуванням додаткової заробітної плати, отримуємо загальну заробітну плату (СЗП):

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 100\ 800 \cdot (1 + 0.2) = 120\ 960 \text{ грн.}$$

Далі розраховується відрахування на соціальний внесок (СВІД), яке складає 22%:

1. $С_{ВІД}= С_{ЗП} \cdot$ 0.22 = 353 917,9 $\cdot$ 0,22 = 77 861,938 грн.
2. $С_{ВІД}= С_{ЗП} \cdot$ 0.22 = 265 438,4 $\cdot$ 0,22 = 58 396,448 грн.

Амортизаційні відрахування обчислюються з використанням амортизації на рівні 25% та вартості ЕОМ у розмірі 30 000 грн.:

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.2 \cdot 0.25 \cdot 30\,000 = 9\,000 \text{ грн.},$$

де КТМ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача,

КА – річна норма амортизації,

ЦПР – договірна ціна приладу.

Витрати на ремонт та профілактику (СР) обчислюються за допомогою коефіцієнта КР, який представляє відсоток витрат на поточні ремонти:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.2 \cdot 30000 \cdot 0.08 = 28\,800 \text{ грн.},$$

Ефективний годинний фонд часу обчислюється на основі календарної кількості днів, вихідних та святкових днів, днів планових ремонтів устаткування, кількості робочих годин в день та коефіцієнта використання приладу у часі:

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 12 – 16) \cdot 8 \cdot 0.35 =$$

= 627,2 години,

Витрати на оплату електроенергії (СЕЛ) обчислюються на основі середньо-споживчої потужності приладу, коефіцієнта зайнятості приладу та тарифу за 1 кВт-год електроенергії:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 627,2 \cdot 0,3 \cdot 0,2 \cdot 4,86889 = 183,22 \text{ грн.},$$

Накладні витрати (СН) обчислюються як 67% від заробітної плати:

$$С_Н = Ц_{ПР} \cdot 0.67 = 42000 \cdot 0,67 = 28\,140 \text{ грн}$$

Тоді річні експлуатаційні витрати (СЕКС) складаються з суми амортизаційних відрахувань, витрат на ремонт, витрат на оплату електроенергії та накладних витрат:

$$С_{ЕКС} = С_А + С_Р + С_{ЕЛ} + С_Н, (5.17)$$

$С_{ЕКС} = 9\,000 + 28\,800 + 183,226 + 28\,140 = 66\,123,226$ грн.

Собівартість однієї машино-години ЕОМ (СМ-Г) розраховується як відношення річних експлуатаційних витрат до ефективного годинного фонду часу:

$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 66\,123,226 / 627,2 = 105,42$ грн/год.

Загальні витрати на оплату машинного часу (СМ) розраховуються як добуток собівартості однієї машино-години ЕОМ на кількість годин роботи:

$$C_М = C_{М-Г} \cdot T, \ (5.18)$$

1. $C_М$ = 105,42 · 42,2 · 24 = 106 769,376 грн.
2. $C_М$ = 105,42 · 32,4 · 24 = 81 974,592 грн.

Сумарні накладні витрати (СН) складають 67% від заробітної плати:

$$C_Н = C_{ЗП} \cdot 0,67, \ (5.19)$$

1. $C_Н$ = 353 917,9 · 0,67 = 237 124,993 грн.
2. $C_Н$ = 265 438,4 · 0,67 = 177 843,728 грн.

Таким чином, вартість розробки програмного продукту за різними варіантами складає:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_М + C_Н, \ (5.20)$$

1. $C_{ПП}$ = 353 917,9 + 77 861,938 + 106 769,376 + 237 124,993 = 775 674,207 грн.
2. $C_{ПП}$ = 265 438,4 + 58 396,448 + 81 974,592 + 177 843,728 = 583 653,168 грн.

5.7 Вибір оптимального варіанту реалізації ПП

У цьому розділі ми проведемо обчислення і визначимо найкращий варіант розробки програмного продукту за допомогою формули (5.21).

Коефіцієнт технічно-економічної рентабельності (КТЕР) обчислюється за формулою:

$$К_{ТЕР_j} = К_{К_j} / С_{Ф_j}, \quad (5.21)$$

$$К_{ТЕР1} = 23{,}17 \ / 775\,674{,}207 = 0{,}298 \cdot 10^{-4},$$
$$К_{ТЕР2} = 23{,}22 \ / 583\,653{,}168 = 0{,}397 \cdot 10^{-4}.$$

Таким чином, з обчислень видно, що кращим варіантом є другий, який має коефіцієнт розміром $0{,}397 \cdot 10^{-4}$. Після проведення функціонально-вартісного аналізу можна зробити висновок, що серед двох залишених варіантів найкращим є другий. В цьому варіанті виявився найкращий показник коефіцієнта якості.

Обраний варіант має наступні параметри:

- Використання мови програмування Python.
- Використання готових даних для тренування моделі.
- Використання готових бібліотек та функцій для моделі.

Цей варіант реалізації забезпечує швидку та надійну розробку системи і не потребує додаткових ресурсів для компанії.

5.8 Висновки до розділу

В даній частині було проведено повний функціонально-вартісний аналіз програмного продукту з метою визначення та оцінки його основних функцій. Результати аналізу дозволили знайти параметри, що характеризують програмний продукт. Крім того, на основі проведеного аналізу був обраний варіант реалізації програмного продукту. Цей етап дозволив отримати необхідну інформацію для подальшого розроблення та визначення вартості програмного комплексу.

ВИСНОВКИ

Прогнозування кредитного ризику є важливим завданням для забезпечення стабільності банківської системи. Воно вимагає використання різних моделей та методів, таких як аналіз фінансової звітності, моделі ймовірності дефолту та машинне навчання, для оцінки ризику.

Кредитний ризик може бути класифікований на кілька категорій, включаючи ризик кредитного дефолту, ризик концентрації, ризик країни, ризик контрагента та суверенний ризик. Кредитори повинні враховувати різні фактори, такі як кредитна історія позичальника, його платоспроможність та умови кредиту, при оцінці кредитного ризику.

Прогнозування кредитного ризику може бути корисним для управління ризиками в банківському секторі, але воно може мати свої обмеження та недоліки. Наприклад, методи прогнозування можуть бути неточними або непрозорими. Тому банки повинні збалансувати переваги та ризики прогнозування кредитного ризику та враховувати різні фактори для отримання найкращої оцінки ризику.

У розділі, присвяченому обробці та аналізу даних, було проведено ряд процедур, таких як обробка пропущених даних, видалення викидів, аналіз кореляції, приведення даних у числовий формат, збалансування класів та нормалізація даних. Ці процедури допомогли підготувати набір даних для подальшого моделювання та прогнозування кредитного ризику. Вся робота проходила із використанням мови Python.

Далі була проведена розробка декількох моделей, чия задача полягала в прогнозуванні дефолту позичальника. В результаті вийшло побудувати декілька моделей, одна з яких досягла результатів, що дали похибку менше 4%. Також була побудована ансамблева модель, що

поєднувала передбачення всіх розроблених моделей, вона дала результати близькі до найкращих.

Далі був проведений функціонально-вартісний аналіз розробки системи, що була описана в роботі. В ході аналізу було визначено, що найбільш оптимальним варіантом для реалізації програмного продукту є використання готових даних, що запропоновані в датасеті.

Загалом, проведені дослідження та аналіз вказують на важливість прогнозування кредитного ризику та необхідність використання різних методів та підходів для його оцінки. Отримані результати можуть бути корисними для банківських установ при управлінні ризиками та прийнятті рішень щодо кредитування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Pegah Sharifi, Vipin Jain, Mehdi Arab Poshtkohi, Erfan seyyedi, and Vahid Aghapour. Banks Credit Risk Prediction with Optimized ANN Based on Improved Owl Search Algorithm. 28 Dec 2021. URL: https://www.hindawi.com/journals/mpe/2021/8458501/
2. Olapeju Comfort Ogunmokun, Oluwasoye P. Mafimisebi & Demola Obembe. Prospect theory and bank credit risk decision-making behaviour: a systematic literature review and future research agenda. URL: https://link.springer.com/article/10.1007/s43546-023-00464-x
3. Will Kenton. Understanding Value at Risk (VaR) and How It's Computed. March 23, 2023. URL: https://www.investopedia.com/terms/v/var.asp
4. Credit Risk: Definition, Role of Ratings, and Examples. T*he investopedia team*. April 27, 2023. URL: https://www.investopedia.com/terms/c/creditrisk.asp
5. Hennie Van Greuning and Sofija-Sonja Brajovic Bratanovic. Credit Risk Management. JUN 2020. URL: https://elibrary.worldbank.org/doi/10.1596/978-1-4648-1446-4_ch7
6. Principles for the Management of Credit Risk. *BIS*. URL: https://www.bis.org/publ/bcbsc125.pdf
7. Kyle Peterdy. Credit Risk. February 14, 2023. URL: https://corporatefinanceinstitute.com/resources/commercial-lending/credit-risk/
8. Credit Risk Analysis Models. *CFI Team*. March 14, 2023. URL: https://corporatefinanceinstitute.com/resources/commercial-lending/credit-risk-analysis-models/

9. ECB Banking Supervision: Assessment of risks and vulnerabilities. *European Central Bank*. 2021. URL: https://www.bankingsupervision.europa.eu/ecb/pub/ra/html/ssm.ra2021~edbbea1f8f.en.html
10. Gabriel Lip. Credit Risk Analysis. December 22, 2022. URL: https://corporatefinanceinstitute.com/resources/commercial-lending/credit-risk-analysis/
11. Philipp Härle, András Havas, and Hamid Samandari. The future of bank risk management. 2016. URL: https://www.mckinsey.com/~/media/mckinsey/business%20functions/risk/our%20insights/the%20future%20of%20bank%20risk%20management/the-future-of-bank-risk-management.pdf
12. Bank of 2030: Transform boldly. *Deloitte*. 11 Apr 2019. URL: https://www.deloitte.com/global/en/Industries/financial-services/perspectives/bank-of-2030-the-future-of-banking.html
13. Jason Brownlee. Logistic Regression for Machine Learning. August 15, 2020. URL: https://machinelearningmastery.com/logistic-regression-for-machine-learning/
14. Gurucharan M K. Machine Learning Basics: Logistic Regression. Aug 6, 2020. URL: https://towardsdatascience.com/machine-learning-basics-logistic-regression-890ef5e3a272
15. Sanskruti Dube. Logistic Regression in Machine Learning. 9 Dec 2020. URL: https://www.scaler.com/topics/machine-learning/logistic-regression-machine-learning/
16. What is logistic regression? *IBM*. URL: https://www.ibm.com/topics/logistic-regression

17. Deep Learning Overview. *Microsoft*. URL: https://learn.microsoft.com/en-us/dotnet/machine-learning/deep-learning-overview
18. Priya Pedamkar. Neural Network Machine Learning. URL: https://www.educba.com/neural-network-machine-learning/
19. Random Forest Algorithm. JavaPoint. URL: https://www.javatpoint.com/machine-learning-random-forest-algorithm
20. Gradient Boosting in ML. *GeeksForGeeks*. 31 Mar, 2023. URL: https://www.geeksforgeeks.org/ml-gradient-boosting/
21. Gradient Boosting. *DeeplAI*. URL: https://deepai.org/machine-learning-glossary-and-terms/gradient-boosting
22. Support Vector Machines. *Sklearn*. URL: https://scikit-learn.org/stable/modules/svm.html
23. Anshul Saini. Support Vector Machine(SVM): A Complete guide for beginners. October 12, 2021. URL: https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/
24. Jason Brownlee. Support Vector Machines for Machine Learning. April 20, 2016. URL: https://machinelearningmastery.com/support-vector-machines-for-machine-learning/
25. Logistic Regression: Pros and Cons. *Holy Python*.nes for Machine Learning. April 20, 2016. URL: https://holypython.com/log-reg/logistic-regression-pros-cons/
26. Amal Joby. What Is Logistic Regression? Learn When to Use It. July 29, 2021. URL: https://learn.g2.com/logistic-regression
27. Jahnavi Mahanta. Introduction to Neural Networks, Advantages and Applications. Jul 10, 2017. URL:

https://towardsdatascience.com/introduction-to-neural-networks-advantages-and-applications-96851bd1a207

28. Harshdeep Singh. Understanding Random Forests. Mar 3, 2019. URL: https://medium.com/@harshdeepsingh_35448/understanding-random-forests-aa0ccecdbbbb
29. Akash Dawari. All About Random Forest. April 25, 2022. URL: https://towardsai.net/p/l/all-about-random-forest
30. Vihar Kurama. Gradient Boosting In Classification: Not a Black Box Anymore! 2020. URL: https://blog.paperspace.com/gradient-boosting-for-classification/
31. Support vector machine in Machine Learning. *GeeksForGeeks*. URL: https://www.geeksforgeeks.org/support-vector-machine-in-machine-learning/
32. Yunfei Cao. Application of Machine Learning Algorithms in Financial Market Risk Prediction. 2021. URL: https://link.springer.com/chapter/10.1007/978-3-030-89508-2_21
33. Hua Peng, Yicheng Lin and Mingzheng Wu.AI-enabled Decision Support System: Methodologies, Applications, and Advancements 2021. 26 Feb 2022. URL: https://www.hindawi.com/journals/sp/2022/3398545/
34. Rami Ali. Predictive Modeling: Types, Benefits, and Algorithms. September 23, 2020. URL: https://www.netsuite.com/portal/resource/articles/financial-management/predictive-modeling.shtml
35. What are the benefits and drawbacks of using AI for forecasting? LinkedIn. URL: https://www.linkedin.com/advice/3/what-benefits-drawbacks-using-ai-forecasting
36. 3 Advantages and 3 Disadvantages of Forecasting. *John Galt*. May 23, 2019. URL:

https://johngalt.com/learn/blog/3-advantages-disadvantages-of-forecasting

37. Bernhard Babel, Kevin Buehler, Adam Pivonka, Bryan Richardson, and Derek Waldron. Derisking machine learning and artificial intelligence. February 19, 2019. URL: https://www.mckinsey.com/capabilities/risk-and-resilience/our-insights/derisking-machine-learning-and-artificial-intelligence (Дата зернення 11.06.2023)
38. Mandeep Mahendru, Burak Erkut, Sandhya Makkar, Simon Grima. AI in Financial Management, Risk and Forecasting: Perspectives from the COVID-19-Era. 23 June 2022. URL: https://www.frontiersin.org/research-topics/31966/ai-in-financial-management-risk-and-forecasting-perspectives-from-the-covid-19-era
39. Martin Leo, Suneel Sharma and K. Maddulety. Machine Learning in Banking Risk Management: A Literature Review. 5 March 2019. URL: https://www.mdpi.com/2227-9091/7/1/29

Додаток А Лістинг

```
#!/usr/bin/env python
# coding: utf-8

# In[67]:


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import missingno as msno
from sklearn.impute import KNNImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from xgboost import XGBClassifier
from sklearn.model_selection import cross_val_score, KFold
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB

from imblearn.under_sampling import RandomUnderSampler
from sklearn.model_selection import train_test_split

import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import StackingClassifier

import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv("/kaggle/input/loan-default-dataset/Loan_Default.csv")


# In[68]:


# Convert all column names to lowercase
```

```
df.columns = df.columns.str.lower()

# Make ID string type
df['id'] = df['id'].astype(str)

# Drop year since all entries are 2019
df.drop(columns=['year'], inplace=True)

# Datatypes
df.info()


# In[69]:


# Null values
nas = df.isna().sum()
print(nas)

# Visualizing Null values (heatmap)
plt.figure(figsize=(15, 9))
sns.heatmap(df.isnull(), cmap="YlGnBu")
plt.show()


# In[70]:


# Getting dataframes by datatype
dtypes = pd.DataFrame(df.dtypes).reset_index()

cat_vars = []
num_vars = []
for i, l in zip(dtypes['index'], dtypes[0]):
    if l == 'object':
        cat_vars.append(i)
    else:
        num_vars.append(i)


# In[71]:


# Numeric Dataframe
df_num = df[num_vars]

# knn
knn = KNNImputer(n_neighbors = 3)
knn.fit(df_num)
X = knn.fit_transform(df_num)
```

```
# Check for any nas
df_num = pd.DataFrame(X, columns=num_vars)
nas_num = df_num.isna().sum()
print(nas_num)


# In[78]:


# Categorical Dataframe
df_cat = df[cat_vars]

for i in cat_vars:
    mode = df[i].mode()
    mode = mode[0]
    df_cat[i].fillna(value=mode, inplace=True)

# Check for any nas
nas_cat = df_cat.isna().sum()
print(nas_cat)

# Combining dataframes
df_full = pd.concat([df_num, df_cat], axis=1, join='inner')


# In[79]:


# Null values
nas = df_full.isna().sum()
print(nas)

# Visualizing Null values (heatmap)
plt.figure(figsize=(15, 9))
sns.heatmap(df_full.isnull(), cmap="YlGnBu")
plt.show()


# In[80]:


df_num = df_full[num_vars]

# Function for finding outliers using 3 times the IQR
def find_outliers_IQR(col):
   Q1=col.quantile(0.25)
   Q3=col.quantile(0.75)
   IQR=Q3-Q1
   outliers = col[((col<(Q1-3*IQR)) | (col>(Q3+3*IQR)))]
   return(outliers)
```

```
# Function for finding proportion of column that is null
def outlier_prop(outliers, col):
    outlier_size = len(outliers)
    return outlier_size / (len(col) + outlier_size)

# Termporary dataframe
df_temp = df_full.copy()

# Both term and status are 15-20% outliers and can be treated as categorical
variables
df_temp['term'] = df_temp['term'].astype(str)
df_temp['status'] = df_temp['status'].astype(str)


# Getting num_vars and cat_vars
dtypes = pd.DataFrame(df_temp.dtypes).reset_index()
cat_vars = []
num_vars = []
for i, l in zip(dtypes['index'], dtypes[0]):
    if l == 'object':
        cat_vars.append(i)
    else:
        num_vars.append(i)

# Getting proportion of
outlier_props = []
cols = []
for i in num_vars:
    outliers = find_outliers_IQR(df_temp[i])
    cols.append(i)
    prop = outlier_prop(outliers, df_temp[i])
    outlier_props.append(prop)

outlier_props_df   =   pd.DataFrame([cols,   outlier_props],   index=['Variable',
'OutlierProp']).transpose()

# Deleting outliers
for col in num_vars:
    q1 = df_temp[col].quantile(0.25)
    q3 = df_temp[col].quantile(0.75)
    IQR = q3-q1
    lower = q1-3*IQR
    upper = q3+3*IQR
    df_temp = df_temp[(df_temp[col] < upper) & (df_temp[col] > lower)]

df_full = df_temp

# Term
term_vals = pd.DataFrame(df_full['term'].value_counts().reset_index())

# Drop terms that have less than 10 appearences in dataset
```

```
terms_to_drop = []
for i, l in zip(term_vals['index'], term_vals['term']):
    if l < 10:
        terms_to_drop.append(i)

for i in terms_to_drop:
    df_full = df_full[df_full['term'] != i]

# Remaining Data
proportion_remaining = round(len(df_full) / len(df), 5)
proportion_dropped = round(1 - proportion_remaining, 2) * 100
dropped = len(df) - len(df_full)
print("We dropped ", dropped, ' rows from the dataset')
print("That is about", proportion_dropped, "% of the original dataset" )
print("The proportion of the original dataset remaining is: ",
proportion_remaining)



# Make term categorical dtype
df_full['term'] = df_full['term'].astype('float')


# In[81]:


# Correlation
corr = df_num.corr()

# Correlation Heatmap
plt.figure(figsize=(16, 6))
heatmap = sns.heatmap(corr, vmin=-1, vmax=1, annot=True)
heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':12}, pad=12);


# In[82]:


# Unique values in 'status'
unique_values = df['status'].unique()

# Plot distribution
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='status')
plt.xlabel('Status')
plt.ylabel('Count')
plt.title('Distribution of Status')
plt.show()


# In[83]:
```

```
# Create a 3x3 grid of subplots
fig, axes = plt.subplots(3, 3, figsize=(16, 18))
fig.tight_layout(pad=3.0)  # Adjust the spacing between subplots

# Plot histograms
for i, var in enumerate(num_vars):
    row = i // 3
    col = i % 3
     sns.histplot(data=df_full, x=var, hue="status", multiple="dodge", shrink=0.8,
bins=4, ax=axes[row, col])
    axes[row, col].set_title(var)

plt.show()


# In[84]:


def plot_hist(col):
    plt.figsize=(16,6)
    sns.set_theme(style='darkgrid')
     sns.histplot(data=df_full, x=col, hue="status", multiple="dodge", shrink=.8,
stat='count')
    plt.show()


# loan_limit
plot_hist(df_full['loan_limit'])

# Gender
plot_hist(df_full['gender'])

# approved in advance
plot_hist(df_full['approv_in_adv'])

# loan_type
plot_hist(df_full['loan_type'])

# loan_purpose
plot_hist(df_full['loan_purpose'])

# Credit worthiness
plot_hist(df_full['credit_worthiness'])

# Open credit
plot_hist(df_full['open_credit'])

# Business or commercial
plot_hist(df_full['business_or_commercial'])
```

```
# neg_ammortization
plot_hist(df_full['neg_ammortization']) # neg_amm more likely to default

# interest_only
plot_hist(df_full['interest_only'])

# Lump sum payment
plot_hist(df_full['lump_sum_payment'])
df_full['lump_sum_payment'].value_counts()
lpsm = df_full[df_full['lump_sum_payment'] == 'lpsm']
lpsm['status'].value_counts() # Vast majority of lumpsum payments default

# Construction type
plot_hist(df_full['construction_type'])
df['construction_type'].value_counts() # Only 33 values for mh
mh = df_full[df_full['construction_type'] == 'mh']
mh['status'].value_counts() # All mh vals were defaults

# occupancy_type
plot_hist(df_full['occupancy_type'])

# secured_by
plot_hist(df_full['secured_by'])
df_full['secured_by'].value_counts()
land = df_full[df_full['secured_by'] == 'land']
land['status'].value_counts() # All land security defaulted

# total_units
plot_hist(df_full['total_units'])
df_full['total_units'].value_counts()

# credit_type
plot_hist(df_full['credit_type']) # All EQUI credit type defaults

# co-applicant_credit_type
plot_hist(df_full['co-applicant_credit_type'])

# age
plot_hist(df_full['age'])

# submission_of_application
plot_hist(df_full['submission_of_application'])

# region
plot_hist(df_full['region'])

# security_type
plot_hist(df_full['security_type'])
df_full['security_type'].value_counts()
indirect = df_full[df_full['security_type'] == 'Indriect']
```

```
indirect['status'].value_counts() # All indirect security type defaults


# In[85]:


df_full.drop(columns=['id'], inplace=True)

dtypes = pd.DataFrame(df_full.dtypes).reset_index()
cat_vars = []
num_vars = []
for i, l in zip(dtypes['index'], dtypes[0]):
    if l == 'object':
        cat_vars.append(i)
    else:
        num_vars.append(i)



# Binary variables
binary_vars        =        ['security_type',        'submission_of_application',
'co-applicant_credit_type', 'secured_by',
                        'lump_sum_payment',  'interest_only',  'neg_ammortization',
'construction_type', 'business_or_commercial',
                'open_credit', 'credit_worthiness', 'approv_in_adv', 'loan_limit',
'status']


# In[86]:


# Label Encoder
label = LabelEncoder()
for i in binary_vars:
    df_full[i] = label.fit_transform(df_full[i])



# OneHotEncoding
df_cat = df_full[cat_vars]
df_cat.drop(columns=binary_vars, inplace=True)

df_cat.columns

cat_encoder = OneHotEncoder()
df_cat_1hot = cat_encoder.fit_transform(df_cat)
df_cat_encoded = pd.DataFrame(df_cat_1hot.toarray())


# Column names
cat_encoder.categories_
```

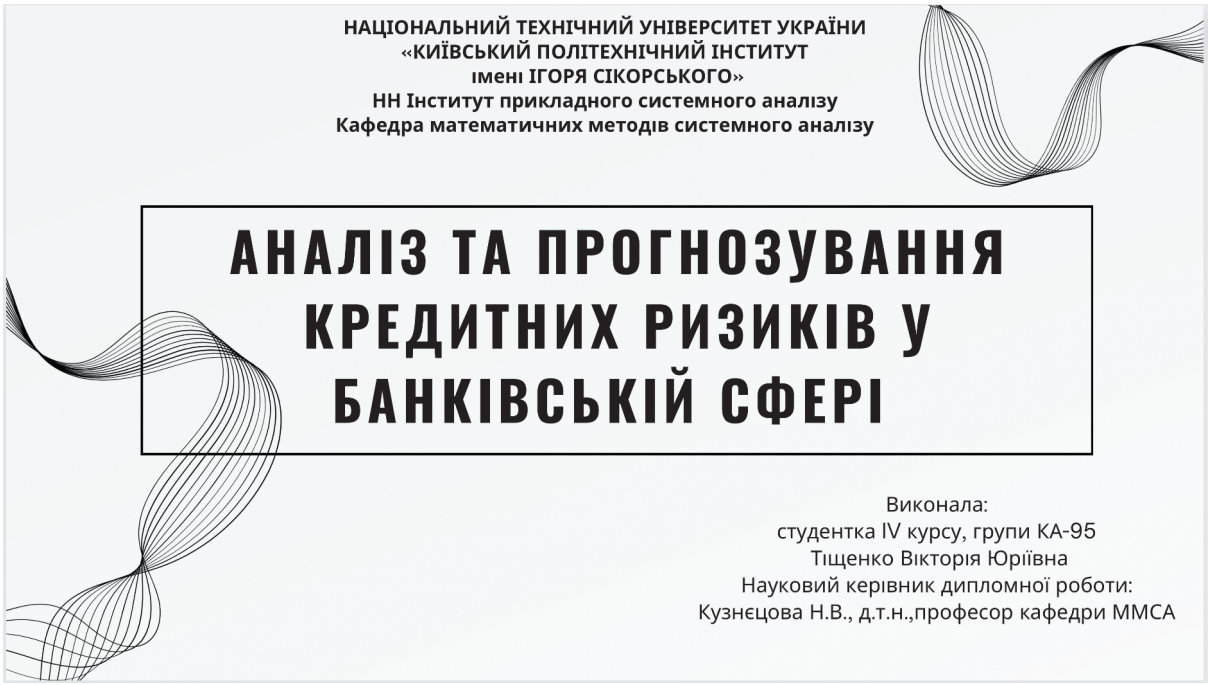
```
cat_columns = ['Female', 'Joint', 'Male', 'Sex Not Available',
               'type1', 'type2', 'type3',
               'p1', 'p2', 'p3', 'p4',
               'ir', 'pr', 'sr',
               'U1', 'U2', 'U3', 'U4',
               'CIB', 'CRIF', 'EQUI', 'EXP',
                 'age_25-34', 'age_35-44', 'age_45-54', 'age_55-64', 'age_65-74',
'under_25', 'over_74',
               'North', 'North-East', 'central', 'south']

df_cat_encoded.columns = cat_columns
df_full.drop(columns=df_cat.columns, inplace=True)
# Concat
df = pd.concat([df_full, df_cat_encoded], axis=1, join='inner')


# In[87]:


# Splitting Data
train_set, test_set = train_test_split(df, test_size=0.2, random_state=42)

y_train = train_set['status']
X_train = train_set.drop(columns=['status'])
y_test = test_set['status']
X_test = test_set.drop(columns=['status'])

# Random undersampling
rus = RandomUnderSampler(random_state=42)
X_train, y_train = rus.fit_resample(X_train, y_train)

# Check the new class distribution
print("Class distribution after undersampling:")
print(y_train.value_counts())


# In[88]:


# Plot distribution of y_train
plt.figure(figsize=(8, 6))
sns.countplot(y_train)
plt.xticks(rotation=45)
plt.xlabel('Status')
plt.ylabel('Count')
plt.title('Distribution of y_train')
plt.show()


# In[89]:
```

```
# Train models
rf = RandomForestClassifier()
rf.fit(X_train, y_train)

xgbc = XGBClassifier()
xgbc.fit(X_train, y_train)

logreg = LogisticRegression()
logreg.fit(X_train, y_train)

mlp = MLPClassifier()
mlp.fit(X_train, y_train)

# Make predictions
rf_preds = rf.predict(X_test)
xgb_preds = xgbc.predict(X_test)
lr_preds = logreg.predict(X_test)
mlp_preds = mlp.predict(X_test)

# Calculate accuracies
rf_accuracy = accuracy_score(y_test, rf_preds)
xgb_accuracy = accuracy_score(y_test, xgb_preds)
lr_accuracy = accuracy_score(y_test, lr_preds)
mlp_accuracy = accuracy_score(y_test, mlp_preds)

# Calculate confusion matrices
rf_cm = confusion_matrix(y_test, rf_preds)
xgb_cm = confusion_matrix(y_test, xgb_preds)
lr_cm = confusion_matrix(y_test, lr_preds)
mlp_cm = confusion_matrix(y_test, mlp_preds)

# Normalize confusion matrices
rf_cm_norm = rf_cm / rf_cm.sum(axis=1, keepdims=True)
xgb_cm_norm = xgb_cm / xgb_cm.sum(axis=1, keepdims=True)
lr_cm_norm = lr_cm / lr_cm.sum(axis=1, keepdims=True)
mlp_cm_norm = mlp_cm / mlp_cm.sum(axis=1, keepdims=True)

# Plot results
fig, axes = plt.subplots(2, 2, figsize=(10, 10))
fig.tight_layout(pad=5)

sns.heatmap(rf_cm_norm,  annot=True,  cmap="YlGnBu",  ax=axes[0,  0],  fmt=".2%",
cbar=False)
axes[0, 0].set_title("Random Forest\nAccuracy: {:.4f}".format(rf_accuracy))

sns.heatmap(xgb_cm_norm,  annot=True,  cmap="YlGnBu",  ax=axes[0,  1],  fmt=".2%",
cbar=False)
axes[0, 1].set_title("XGBoost\nAccuracy: {:.4f}".format(xgb_accuracy))
```

```
sns.heatmap(lr_cm_norm, annot=True, cmap="YlGnBu", ax=axes[1, 0], fmt=".2%",
cbar=False)
axes[1, 0].set_title("Logistic Regression\nAccuracy: {:.4f}".format(lr_accuracy))

sns.heatmap(mlp_cm_norm, annot=True, cmap="YlGnBu", ax=axes[1, 1], fmt=".2%",
cbar=False)
axes[1, 1].set_title("Neural Network\nAccuracy: {:.4f}".format(mlp_accuracy))

plt.show()


# In[90]:


# Define the base models
base_models = [
    ('rf', RandomForestClassifier()),
    ('xgbc', XGBClassifier()),
    ('mlp', MLPClassifier())
]

# Initialize the stacking classifier with logistic regression as the final
estimator
stacking_clf = StackingClassifier(estimators=base_models,
final_estimator=LogisticRegression())

# Fit the stacking classifier
stacking_clf.fit(X_train, y_train)

# Make predictions
stacking_preds = stacking_clf.predict(X_test)

# Calculate accuracy
stacking_accuracy = accuracy_score(y_test, stacking_preds)

# Calculate confusion matrix
stacking_cm = confusion_matrix(y_test, stacking_preds)

# Normalize the confusion matrix
normalized_cm = stacking_cm / stacking_cm.sum(axis=1, keepdims=True)

# Plot the confusion matrix
sns.heatmap(normalized_cm, annot=True, cmap="YlGnBu")
plt.title("Stacking Classifier\nAccuracy: {:.4f}".format(stacking_accuracy))
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()


# In[91]:
```

```
# Get importances of the base models
base_model_importances = stacking_clf.final_estimator_.coef_

# Plot the importances
model_names = [name for name, _ in base_models]
plt.bar(model_names, np.abs(base_model_importances[0]), color='skyblue')
plt.xlabel('Base Model')
plt.ylabel('Importance')
plt.title('Importance of Base Models in Stacking Classifier')
plt.xticks(rotation=45)
plt.show()


# In[92]:


accuracy_data = [
    ['Random Forest', rf_accuracy],
    ['XGBoost', xgb_accuracy],
    ['Logistic Regression', lr_accuracy],
    ['Neural Network', mlp_accuracy],
    ['Stacking Classifier', stacking_accuracy]
]

df = pd.DataFrame(accuracy_data, columns=['Model', 'Accuracy'])

df
```

Додаток Б Лістинг




ОБ'ЄКТ, ПРЕДМЕТ ТА МЕТА ДОСЛІДЖЕННЯ
Об'єкт дослідження
Вибірка даних про клієнтів, що брали позики у банку, на яких проводиться навчання для прогнозування ризику у банківській сфері
Предмет дослідження
Методи машинного навчання та методи їх застосування у сфері визначення ризиків
Мета дослідження
Підвищення точності автоматичного визначення ризиків у банківській сфері на основі алгоритмів машинного навчання

ВИЗНАЧЕННЯ МЕТИ РОБОТИ

Постановка задачі: розробка програмного забезпечення для автоматичного визначення ризиків у банківській сфері на основі алгоритмів машинного навчання.

Кредитний ризик – це ризик неотримання кредитором основної суми боргу та відсотків, що призводить до переривання грошових потоків та збільшення витрат на стягнення заборгованості

АКТУАЛЬНІСТЬ РОБОТИ

В сфері кредитування ключовим фактором є передбачення ризиків, пов'язаних з позичанням грошей. Точне передбачення ймовірності невиплати позики може значно знизити фінансові втрати установ, що займаються кредитуванням, а також дозволить зменшити процентну ставку для більш надійних клієнтів. Прогнозування ризиків у банківській сфері є актуальною темою, яка ще не повністю вирішена і потребує значних покращень. Це можливо завдяки використанню прогнозних моделей.

Від дослідження теми аналізу та прогнозування кредитних ризиків нього залежить фінансова стійкість та прибутковість установ, а також забезпечення більш надійного та ефективного кредитування.


01
02
03
04
05
АНАЛІЗ ПОПЕРЕДНІХ ДОСЛІДЖЕНЬ В ОБЛАСТІ ПРОГНОЗУВАННЯ БАНКІВСЬКИХ РИЗИКІВ
АНАЛІЗ АЛГОРИТМІВ ПРОГНОЗУВАННЯ РИЗИКІВ У БАНКІВСЬКІЙ СФЕРІ
АНАЛІЗ ТА ОБРОБКА НАБОРУ ДАНИХ ДЛЯ ПРОГНОЗУВАННЯ
СТВОРЕННЯ СИСТЕМИ ПРОГНОЗУВАННЯ БАНКІВСЬКИХ РИЗИКІВ
ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ РОЗРОБКИ ЗАПРОПОНОВАНОЇ СИСТЕМИ

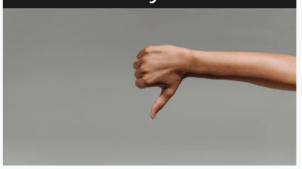
ПЕРЕВАГИ ТА НЕДОЛІКИ ВИКОРИСТАННЯ ПРОГНОЗУВАННЯ КРЕДИТНИХ РИЗИКІВ У БАНКІВСЬКІЙ СФЕРІ

Плюси

- Покращення управління ризиками
- Підвищення ефективності
- Краще прийняття рішень

- Надмірна залежність від моделей
- Неточність
- Брак прозорості

Мінуси




ПОРІВНЯННЯ АЛГОРИТМІВ ПРОГНОЗУВАННЯ РИЗИКІВ
ЛОГІСТИЧНА РЕГРЕСІЯ
Плюси: простота, легкість реалізації, ефективність у навчанні та здатність надавати ймовірності.
Мінуси: схильність до надмірної підгонки, якщо кількість ознак перевищує кількість спостережень, а також складність отримання складних взаємозв'язків.
НЕЙРОНІ МЕРЕЖІ
Плюси: здатність моделювати складні взаємозв'язки та адаптивність до різних типів даних
Мінуси:обчислювальні витрати, чутливість до перенавчання та залежність від великих обсягів даних.
ВИПАДКОВИЙ ЛІС
Плюси: Стійкість до викидів, може обробляти зашумлені дані і добре узагальнює нові дані.
Мінуси: має певні обмеження з точки зору обчислювальних витрат, інтерпретованості та надмірного пристосування.
ГРАДІЄНТНИЙ БУСТИНГ
Плюси: гнучкість, добре працює як з категоріальними, так і з числовими значеннями, не вимагає попередньої обробки даних.
Мінуси: схильність до надмірної адаптації, особливо якщо дані є зашумленим, дорогий в обчислювальному плані і потребує багато часу для навчання, особливо на центральних процесорах.

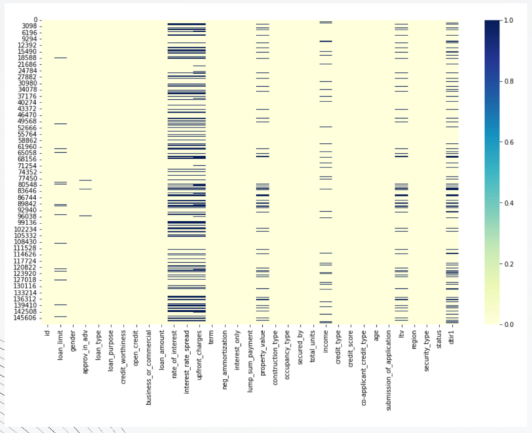
ОГЛЯД ДАТА СЕТУ

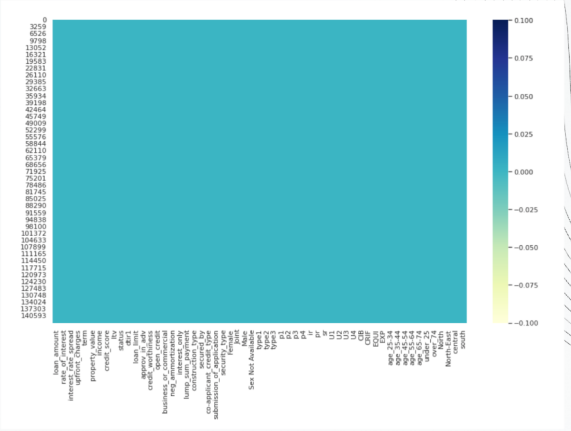
Для роботи був використаний відкритий датасет із сайту Kaggle. Датасет включає в себе дані про позичальників і їх позик. Моєю метою було розробити модель ML, щоб класифікувати, чи новий позичальник виконає зобов'язання.

Атрибут	Опис
id	Унікальний ідентифікатор
loan_limit	Ліміт позики
gender	Стать позичальника
approv_in_adv	Одобрення вперед
loan_type	Тип позики
loan_purpose	Мета позики
credit_worthiness	Кредитна здатність
open_credit	Відкритий кредит
business_or_commercial	Бізнес або комерційна ціль
loan_amount	Сума позики
rate_of_interest	Процентна ставка

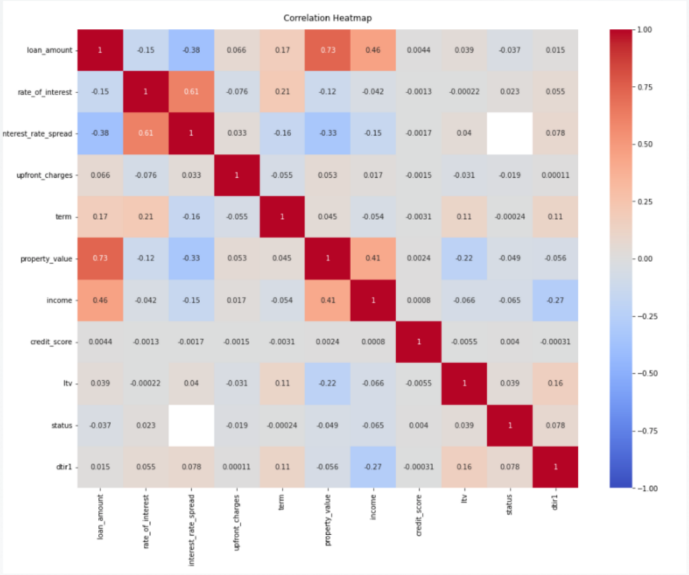
interest_rate_spread	Різниця процентних ставок
upfront_charges	Передоплата
term	Терміи позики
neg_ammortization	Негативна амортизація
interest_only	Тільки відсотки
lump_sum_payment	Одноразовий платіж
property_value	Вартість нерухомості
construction_type	Тип будівництва
occupancy_type	Тип зайнятості
secured_by	Забезпечено чим
total_units	Загальна кількість одиниць
income	Дохід позичальника
credit_type	Тип кредиту
credit_score	Кредитний бал
co-applicant_credit_type	Тип кредиту співзаявника
age	Вік позичальника
submission_of_application	Подання заявки
ltv	Відношення вартості кредиту до вартості нерухомості
region	Регіон
security_type	Тип забезпечення
status	Статус позики
dtir1	Відношення боргу до доходу перед одержанням позики

Heatmap для пропущених значень до та після їх заповнення

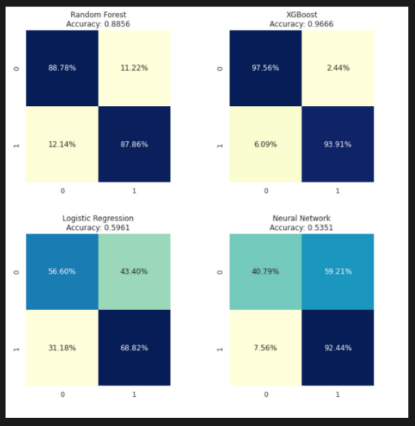


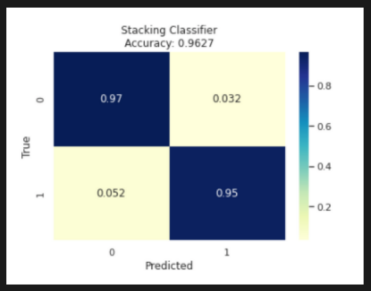


Кореляційна матриця для даних з датасету



РЕЗУЛЬТАТИ ОЦІНКИ РОБОТИ МОДЕЛЕЙ





ВИСНОВОК

Під час роботи була проведена розробка декількох моделей, чия задача полягала в прогнозуванні дефолту позичальника. В результаті вийшло побудувати декілька моделей, одна з яких досягла результатів, що дали похибку менше 4%. Також була побудована ансамблева модель, що поєднувала передбачення всіх розроблених моделей, вона дала результати близькі до найкращих.

Загалом, проведені дослідження та аналіз вказують на важливість прогнозування кредитного ризику та необхідність використання різних методів та підходів для його оцінки. Отримані результати можуть бути корисними для банківських установ при управлінні ризиками та прийнятті рішень щодо кредитування.

Новизною моєї роботи я вважаю створення моделей прогнозування ризику, використовуючи комбінацію декількох різних моделей в ансамбль, за допомогою логістичної регресії. Результати порівняння стандартного та альтернативного підходів демонструють перевагу розробленої моделі.


ДЯКУЮ ЗА
УВАГУ!