ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Методи та алгоритми машинного навчання для аналізу ризиків у фінансовому секторі»

на здобуття освітнього ступеня бакалавра
зі спеціальності 124 Системний аналіз
(код, найменування спеціальності)
освітньо-професійної програми Системний аналіз
(назва)

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

_____ *Богдан ЛИТВИНЕЦЬ*
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав:
здобувач вищої освіти
група САД-41 Богдан ЛИТВИНЕЦЬ
(Ім'я, ПРІЗВИЩЕ)

Керівник: д.т.н., проф. Олексій ШУШУРА
(Ім'я, ПРІЗВИЩЕ)

Рецензент: _____
(Ім'я, ПРІЗВИЩЕ)

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інформаційних технологій

Кафедра Інформаційних систем та технологій
Ступінь вищої освіти Бакалавр
Спеціальність 124 Системний аналіз
Освітньо-професійна програма Системний аналіз

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІСТ
_____Каміла СТОРЧАК
«_____» _____2026р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Литвинець Богдан Вікторович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи:Методи та алгоритми машинного навчання для аналізу ризиків у фінансовому секторі
керівник кваліфікаційної роботи Олексій ШУШУРА, професор кафедри інформаційних систем та технологій, доктор технічних наук
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від « 20 » лютого 2026р. № 5 1
2. Строк подання кваліфікаційної роботи « 01 » червня 2026р.
3. Вихідні дані до кваліфікаційної роботи: Відкриті датасети з платформи OpenML, інтернет-ресурси, інтернет платформи.

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Дослідження історії і проблем кредитного оцінювання включаючи вплив людського фактору на вердикт кредитора
2. Огляд сучасних методів оцінки ризиків в банківському секторі
3. Проблеми класифікації та дисбалансу класів на оцінку ризиків.
4. Розробка моделей з використанням методів машинного навчання
5. Інтерпретація результатів та аналіз метрик
6. Формування висновків

5.Перелік ілюстративного матеріалу: *презентація*
6.Дата видачі завдання « 20 » лютого 2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області і інструментів	01.03-09.03	Виконано
2	Обробка матеріалу.Дослідження методів для їх використання в роботі	10.03-25.03	Виконано
3	Аналіз предметної області.Написання першого розділу	15.03-25.03	Виконано
4	Дослідження математичного апарату для виконання завдання	25.03-09.04	Виконано
5	Реалізація практичної частини	10.04-28.04	Виконано
6	Оформлення результатів практичної частини	28.04-04.05	Виконано
7	Висновки за результатами роботи	04.05-09.05	Виконано
8	Розробка презентації	10.05-20.05	Виконано

Здобувач(ка) вищої о освіти _____ Богдан Литвинець
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Олексій Шушура
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Пояснювальна записка: 88 стор., 27 рис., 3 табл., 1 дод., 28 джерел.

Об’єктом дослідження є процес оцінювання кредитного ризику у фінансовому секторі.

Предметом дослідження є методи та алгоритми машинного навчання, що застосовуються для прогнозування дефолту клієнтів і підтримки прийняття кредитних рішень у банківських установах.

Метою кваліфікаційної роботи є дослідження, реалізація та порівняльний аналіз алгоритмів машинного навчання для оцінювання кредитного ризику на прикладі задачі кредитного скорингу.

У кваліфікаційній роботі розглянуто методи кредитного оцінювання, особливості традиційного підходу до прийняття кредитних рішень, вплив людського фактору, когнітивних упереджень і якості даних на результати оцінювання ризику. Проаналізовано сутність сучасного кредитного скорингу як автоматизованої системи підтримки прийняття рішень у фінансовому секторі.

У роботі сформульовано задачу кредитного скорингу як задачу бінарної класифікації, у межах якої модель машинного навчання прогнозує ймовірність настання дефолту клієнта. Розглянуто основні алгоритми машинного навчання, що застосовуються для аналізу кредитного ризику, зокрема логістичну регресію, випадковий ліс і градієнтний бустинг. Окрему увагу приділено проблемі дисбалансу класів, яка є характерною для фінансових даних, а також методам її вирішення, зокрема Random Oversampling, Random Undersampling, SMOTE та ADASYN.

Було розроблено програмне забезпечення для застосування розлянутих методів машинного навчання на основі відкритого набору даних default-of-credit-card-clients, який містить інформацію про клієнтів кредитних карток, їхні демографічні характеристики, кредитний ліміт, історію погашення, суми рахунків, фактичні платежі та цільову змінну default. У процесі дослідження виконано розвідувальний аналіз даних, виявлено основні закономірності у поведінці клієнтів, сформовано додаткові синтетичні ознаки, що характеризують боргове

навантаження, платіжну активність і дисципліну клієнта.

Для оцінювання якості моделей використано метрики accuracy, precision, recall, F1-score, ROC-AUC та PR-AUC. Проведено порівняльний аналіз результатів роботи моделей машинного навчання, досліджено вплив порогу класифікації на якість виявлення дефолтних клієнтів, а також розглянуто можливості практичного застосування отриманих результатів у системах підтримки прийняття кредитних рішень.

Практичне значення роботи полягає в тому, що отримані результати можуть бути використані як основа для створення або вдосконалення автоматизованої системи оцінювання кредитного ризику. Така система може допомагати банківським установам підвищувати об’єктивність кредитних рішень, зменшувати вплив людського фактору, своєчасно виявляти ризикових позичальників і оптимізувати процес управління кредитним портфелем.

Результати дипломної роботи були представлені та обговорені на науково-технічних конференціях у 2026 році: XVI Міжнародній науково-технічній конференції "Інформаційно-комп’ютерні технології" на базі Державного університету «Житомирська політехніка» та VII Міжнародній науково-технічній конференції «Сучасний стан та перспективи розвитку IoT» на базі Державного університету інформаційно-комунікаційних технологій.

КЛЮЧОВІ СЛОВА: КРЕДИТНИЙ РИЗИК, КРЕДИТНИЙ СКОРИНГ, МАШИННЕ НАВЧАННЯ, БІНАРНА КЛАСИФІКАЦІЯ, ДЕФОЛТ, ЛОГІСТИЧНА РЕГРЕСІЯ, ВИПАДКОВИЙ ЛІС, ГРАДІЄНТНИЙ БУСТИНГ, ДИСБАЛАНС КЛАСІВ, МЕТРИКИ ЯКОСТІ.

ЗМІСТ

ВСТУП 10
1 ТЕОРЕТИЧНІ ЗАСАДИ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 13
1.1 Проблематика кредитного оцінювання 13
1.2 Сутність і структура сучасного кредитного скорингу 17
1.3 Якість даних та вплив людського фактору на прийняття рішень 19
2 МЕТОДОЛОГІЧНІ ОСНОВИ ТА МАТЕМАТИЧНИЙ АПАРАТ 22
2.1 Постановка задачі кредитного скорингу як задачі класифікації 22
2.2 Аналіз існуючих методів оцінки ризиків у банківському секторі 23
2.3 Проблема дисбалансу класів у задачах фінансового скорингу 27
2.4 Метрики оцінювання якості моделей 29
2.5 Методи вирішення дисбалансу класів 32
3 РЕАЛІЗАЦІЯ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ 35
3.1 Характеристика набору даних та постановка задачі 35
3.2 Розвідувальний аналіз даних і виявлення ключових закономірностей 37
3.3 Підготовка даних і формування синтетичних ознак 41
3.4 Сегментація клієнтів за допомогою кластеризації KMeans 45
3.5 Попередня обробка та навчання моделей 46
3.6 Порівняння моделей машинного навчання та аналіз метрик 48
3.7 Аналіз порогу класифікації та матриці помилок 49
3.8 Інтерпретація моделі та практичне застосування результатів 52
ВИСНОВКИ 60
ПЕРЕЛІК ПОСИЛАНЬ 62
Додаток А. ПРОГРАМНИЙ КОД 66

ВСТУП

У сучасних умовах розвитку фінансового сектору управління ризиками є одним із ключових напрямів забезпечення стабільності, прибутковості та конкурентоспроможності банківських установ. Особливе місце серед фінансових ризиків займає кредитний ризик, оскільки саме помилки під час оцінювання платоспроможності позичальників можуть призводити до значних фінансових втрат, зростання частки проблемної заборгованості та зниження ефективності кредитної діяльності банку [1; 2; 3].

Процес прийняття кредитних рішень традиційно ґрунтувався на експертному аналізі анкетних даних клієнта, фінансової інформації, кредитної історії та інших характеристик позичальника. Проте такий підхід має низку суттєвих обмежень. До них належать вплив людського фактору, суб'єктивність оцінювання, когнітивні упередження, нестабільність рішень різних експертів, а також низька масштабованість ручної обробки заявок. Історично рішення кредиторів часто мали ознаки функціонування системи типу «чорної скриньки», де результат залежав не лише від об'єктивних фінансових показників, а й від досвіду, стану та суб'єктивного бачення аналітика [4; 5].

З розвитком інформаційних технологій, накопиченням великих обсягів фінансових даних та поширенням методів машинного навчання з'явилася можливість формалізувати задачу кредитного оцінювання як задачу бінарної класифікації. У такій постановці модель аналізує набір ознак клієнта та прогнозує ймовірність настання дефолту, тобто невиконання кредитних зобов'язань. Це дозволяє не лише автоматизувати процес оцінювання ризику, а й зробити його більш об'єктивним, відтворюваним і придатним для інтеграції в системи підтримки прийняття рішень [6; 7; 8; 9].

Актуальність теми кваліфікаційної роботи зумовлена необхідністю підвищення якості аналізу кредитного ризику за допомогою сучасних алгоритмів машинного навчання. На відміну від традиційних скорингових підходів, такі алгоритми здатні виявляти складні нелінійні залежності між фінансовими,

демографічними та поведінковими характеристиками клієнтів. Особливо важливим є те, що у задачах кредитного скорингу дані часто є незбалансованими: кількість клієнтів, які не допустили дефолт, значно перевищує кількість дефолтних клієнтів. Через це використання лише загальної точності моделі є недостатнім, адже модель може демонструвати високу accuracy, але водночас погано виявляти саме ризикових позичальників [6; 8; 9; 10; 11; 12].

Об’єктом дослідження є процес оцінювання кредитного ризику у фінансовому секторі.

Предметом дослідження є методи та алгоритми машинного навчання, що застосовуються для прогнозування дефолту клієнтів і підтримки прийняття кредитних рішень у банківських установах.

Метою кваліфікаційної роботи є дослідження, реалізація та порівняльний аналіз алгоритмів машинного навчання для оцінювання кредитного ризику на прикладі задачі кредитного скорингу.

Для досягнення поставленої мети у роботі необхідно вирішити такі завдання:

– дослідити проблематику кредитного оцінювання, вплив якості даних та людського фактору;

– формалізувати кредитний скоринг як задачу бінарної класифікації та визначити основні чинники, що впливають на прогнозування;

– провести розвідувальний аналіз даних, врахувати дисбаланс класів та сформувати синтетичні ознаки для навчання моделей машинного навчання;

– реалізувати та порівняти основні моделі для оцінювання імовірності дефолту клієнта;

– проаналізувати якість моделей за основними метриками, дослідити вплив порогу класифікації та розглянути можливість використання результатів у системах підтримки прийняття рішень.

У практичній частині роботи використано відкритий набір даних default-of-credit-card-clients, який містить інформацію про клієнтів кредитних карток, зокрема суму кредитного ліміту, демографічні характеристики, історію погашення, суми рахунків, фактичні платежі та цільову змінну default, що вказує на факт дефолту в

наступному місяці [13].

Практичне значення роботи полягає в тому, що отримані результати можуть бути використані як основа для створення або вдосконалення системи підтримки прийняття кредитних рішень у банківській установі. Така система може допомагати кредитному аналітику оцінювати ймовірність дефолту, визначати рівень кредитного ризику, формувати рекомендації щодо надання або відмови у кредиті, а також обґрунтовувати рекомендований кредитний ліміт.

1 ТЕОРЕТИЧНІ ЗАСАДИ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Проблематика кредитного оцінювання

У часи коли ще не було доступу до електронно-обчислювальних машин, структурованих баз даних та сучасних інформаційних систем, процес прийняття рішень щодо оцінки ризиків був повністю неформалізованим. Усі провідні фінансові установи покладалися на послуги так званих андеррайтерів, чия робота нагадувала функціонування системи типу «Black Box» [4].

Андеррайтером міг бути як окремий фахівець так і співробітник компанії, основна функція якого полягала у ручному зборі обмеженого обсягу неструктуризованої інформації про клієнта, на виході якого ми отримували результат: схвалення або відмова. Основою для цього слугувало так званий евристичний метод «5Cs of Credit»(рис. 1.1) [14].

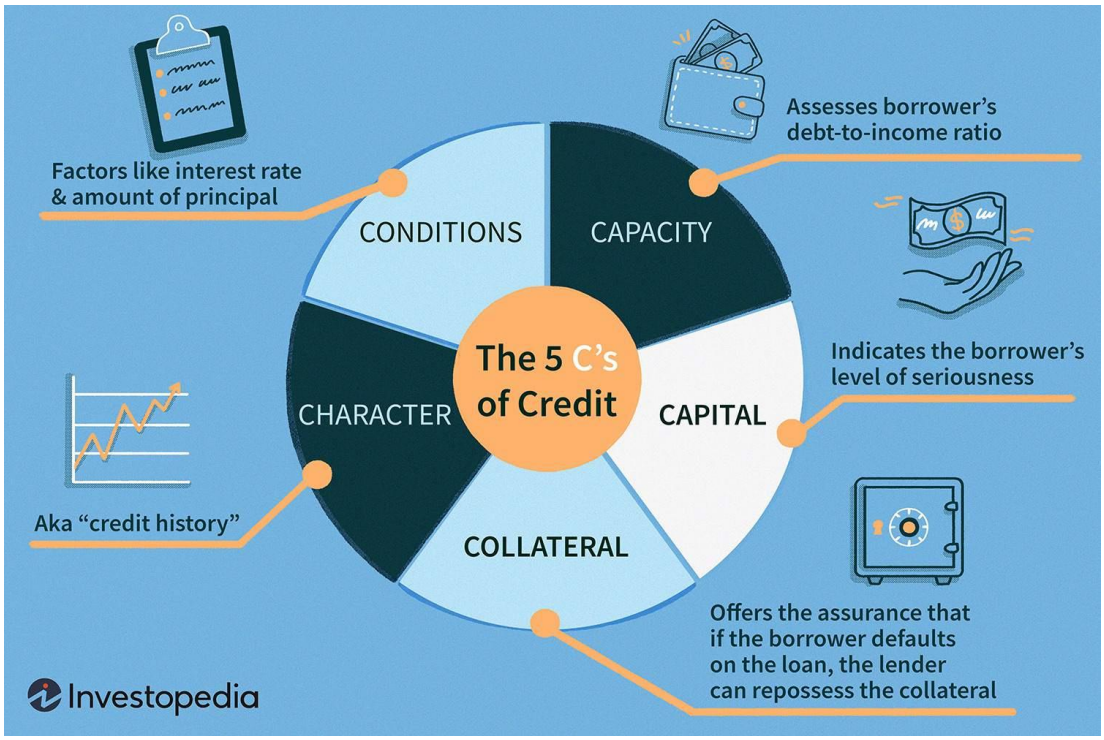


Рис. 1.1 Правило 5Cs [14]

Проте цей метод мав в собі фундаментальні недоліки для побудови хоча б якоїсь прийнятної системи. По-перше вхідні змінні (наприклад “характер” клієнта)

не мали чіткого кількісного виміру і визначалися експертом інтуїтивно. По-друге система не мала чітко заданої математичної моделі чи функції, а отже, її результати не піддавались жодній формальній верифікації чи тестуванню на історичних даних.

Основні складові методу 5Cs наведено в табл. 1.1.

Таблиця 1.1

Правило 5Cs

Критерій	**Назва в перекладі**	**Що саме оцінює банк**	**Ключові показники**
Character	Репутація	Надійність та чесність позичальника	Кредитна історія кредитний рейтинг, стаж роботи
Capacity	Спроможність	Здатність генерувати грошовий потік для виплат	Рівень доходу, співвідношення боргу до доходу(DTI), стабільність бізнесу
Capital	Капітал	Власна частка позичальника в проєкті або чисті активи	Розмір першого внеску, наявність заощаджень та інвестицій
Collateral	Застава	Додаткові гарантії у вигляді майна	Оціночна вартість нерухомості, авто або обладнання
Conditions	Умови	Вплив зовнішнього середовища на повернення коштів	Відсоткова ставка, інфляція, ринкові тренди, ціль кредиту

Також на фінальні дані впливав колосальний обсяг «когнітивного шуму» та упереджень: раса, стать, соціальний статус, особисті переконання чи навіть втома експерта мала свій вплив на результат оцінки для одного й того ж набору даних.

Крім того масштабованість такої системи була критично низькою, масштабування бізнесу вимагало лінійного збільшення штату аналітиків, що

робило процес дорогим та повільним.

Надломом та першим кроком до автоматизації процесу стало опублікування новаторської праці «Risk Elements in Consumer Installment Financing» Девідом Дюрандом у 1941 році. В ній він вперше відійшов від суб'єктивних оцінок і застосував суворий математичний апарат лінійного дискримінантного аналізу до масиву зібраних фінансових даних позичальників [15].

Він зібрав першу вибірку із 7200 виданих позик(що схоже на теперішні датасети) та спробував знайти закономірності в даних, які вирізняють класи "надійних" позичальників від тих, хто допустив дефолт. Він в своїй роботі довів можливість формалізації клієнта як багатовимірний вектор ознак(таких як вік, стаж роботи, наявність банківського рахунку, розмір початкового внеску). За допомогою математичного апарату він вирахував систему коефіцієнтів для кожної з цих змінних.

Також він вперше запропонував формулу, де характеристика клієнта перетворювалась на числовий індекс-прототип сучасного скорингового балу [15].

Це стало фундаментом і переходом від інтуїтивної експертизи до закономірностей та статистики.

З настанням 1950-х років, на тлі післявоєнної відбудови та появи перших кредитних карток сектор зіткнувся з експоненційним ростом заявок на оформлення споживчих кредитів, що в свою чергу пролило світло на проблему ручної праці андеррайтерів та їх слабкої масштабованості. Паперовий документообіг та ручна експертиза більше не могла забезпечити необхідний функціонал, це і стало драйвером розвитку для алгоритмізації.

У 1956 році відбулося створення нині відомої компанії FICO (Fair, Isaac and Company) заснованої інженером Ерлом Айзеком та математиком Біллом Фером [16].

Основним їх досягненням стало формалізування процесу оцінки позичальника як задачу класифікації, вони зрозуміли, що замість навчання безлічі людей краще аналізувати анкети, для створення системи, котра зможе самостійно генерувати рішення з передбачуваною точністю.

Вони вперше впровадили в комерційну експлуатацію скорингові карти (англ. Application Scorecards). Алгоритм працював наступним чином: інформація з анкети клієнта проходила етап категоризації, де кожній категорії присвоювалася статистично розрахована вага-кількість балів, яка відображала кореляцію цієї характеристики з імовірністю дефолту. Загальний скоринговий бал обчислювався за формулою яку нині знають як метод зважених оцінок:

$$s = \sum_{i=1}^{R} w_i x_i \tag{1.1}$$

де x_i – трансформоване значення ознаки, w_i – її вага.

Якщо фінальна сума перевищувала заданий поріг, система видавала сигнал на схвалення. Це була повноцінна система підтримки прийняття рішень, яка повністю усували людський фактор із процесу. Але на той момент обчислювальні технології ще не давали можливості повноцінно розгорнути такі системи масово.

Проте у 1960-1970-х роках, почалась експансія, впровадження на той час таких новітніх технологій як магнітні стрічки та диски дозволило запускати алгоритми скорингу в режимі пакетної обробки. Якщо раніше розрахунок скорингової карти займав сотні годин праці з калькулятором то тепер обчислювальний кластер виконував ці операції за секунди. Банківські системи перетворились з звичайних сховищ на обчислювальні центри [5; 17].

Також повпливали і другорядні фактори, наприклад Закон про рівні можливості кредитування (ECOA), який суворо забороняв використовувати для оцінки позичальника такі параметри, як раса, колір шкіри, релігія, національне походження чи стать [18].

Але повертаючись до компанії FICO, в 1989 році вона представила загальнонаціональний універсальний індекс-FICO Score, який в собі містив закритий алгоритм котрий замість того, щоб аналізувати сотні розрізнених записів з різних кредитних бюро про затримки платежів, відкриті рахунки та запити на нові кредити, він агрегуючи цю всю інформацію в стандандартизовану скалярну величину в діапазоні від 300 до 850 балів, створив галузевий стандарт та дозволив банкам автоматизувати схвалення та впровадити ризик-орієнтоване ціноутворення

(англ. risk-based pricing), де відсоткова ставка розраховувалась динамічно, як математична функція від балу FICO клієнта. Що в свою чергу створило підґрунтя для сьогодення – переходу до складних статистичних методів та логістичної регресії [19].

Починаючи з 1990-х та до сьогодення фінансова індустрія зазнала змін. Завдяки стрімкому розвитку інтернету та технологій зв'язку виріс і обсяг неструктурованої інформації. Зараз ми називаємо цей період епохою великих даних. Для банківський систем це означає кризу традиційних скорингових систем. Аналіз "цифрового сліду" людини став необхідністю, що спричинило коаліцію між банківським сектором та наукою про дані (англ. Data Science).

Першим кроком до цього стало масове впровадження дерев рішень, здатних автоматично генерувати складні ієрархічні правила, але через їх нестабільність та схильність до перенавчання(запам'ятовування тренувальної вибірки) індустрія перейшла до створення архітектури ансамблевим методів.

Поява алгоритмів паралельного навчання дозволила системам генерувати тисячі незалежних дерев і приймати рішення шляхом їхнього математичного усереднення, що кардинально знизило дисперсію моделей та підвищило їх стійкість.

Ще одне покращення яке остаточно сформувало сучасний вигляд систем скорингу, це створення високопродуктивних бібліотек (XGBoost, LightGBM), кожне наступне дерево стало оптимізовувати помилки попереднього, що в свою чергу вирішило проблему дисбалансу класів.

1.2 Сутність і структура сучасного кредитного скорингу

У контексті сьогодення сучасний кредитний скоринг це зовсім не послуга, а складна автоматизована система підтримки прийняття рішень (англ. Decision Support System), побудована на алгоритмах машинного навчання [4; 17].

Архітектурно сучасний скоринговий процес поділяють на декілька підсистем, зокрема аплікаційний скоринг, поведінковий скоринг та підсистему

виявлення аномалій

Аплікаційний скоринг (англ. Application Scoring) – алгоритм первинної фільтрації вхідного потоку даних. На цьому етапі система вирішує задачу «холодного старту», класифікуючи нові заявки на основі статичного простору ознак(дані з бюро кредитних історій). Вектор клієнта порівнюється з еталонним, після чого генерується попередній індекс довіри [4;20].

Поведінковий скоринг (англ. Behavioral Scoring) – це вже предиктивна система яка працює з часовими рядами. Аналізує динаміку змін протягом часу: частоту транзакцій, характер грошових потоків (англ. Cash flow) та своєчасне обслуговування боргу. Аналіз ризиків відбувається ітеративно, наприклад щомісяця або щокварталу, це дозволяє алгоритму завчасно виявити тригери деградації клієнта та зменшити йому для прикладу кредитний ліміт[4].

Коллекторський скоринг (англ. Collection Scoring) –вузькоспеціалізована система яка працює в разі фіксації дефолту позичальника. Вона сегментує клієнтів від тих, кому достатньо автоматичного нагадування або push-повідомлення (англ. Soft Collection), до тих, чию справу подають в юридичні департаменти з примусового стягнення боргів (англ. Hard/Legal Collection Department).

Підсистема виявлення аномалій (англ. Fraaud Scoring) – аналітичний модуль, що базується на методах пошуку аномалій (англ. Anomaly detection). Цільова функція якого полягає в наданні позичальнику індексу шахрайства, шляхом аналізу цифрового сліду та метаданих транзакції[17; 21].

Підсистема виявлення аномалій може базуватись на таких метаданих як топологічна відстань (англ. Geo-IP Distance), відповідність BIN та локації пристрою, а також детектування proxy, VPN або використання onion мереж, зокрема Tor. В свою чергу топологічна відстань передбачає порівняння координат IP-адреси, з якої здійснюється платіж, із фізичною адресою випуску картки. Відповідність BIN полягає у порівнянні країни походження картки та місця оплати без використання VPN або proxy.

Приклад градації рівнів ризику фрод-транзакцій наведено на рис. 1.2.

Very Low Risk	0 – 99	Дуже низька	Надійний клієнт. Авто-схвалення.
Low Risk	100 – 499	Низька	Безпечно. Стандартний моніторинг.
Medium Risk	500 – 699	Помірна	Невизначено. Ручна перевірка.
High Risk	700 – 899	Висока	Підозріло. Додаткова перевірка (2FA).
Very High Risk	900 – 1000	Критична	Ймовірний фрод. Блокування.

Рис. 1.2 Приклад градації рівнів ризику фрод-транзакцій

Оскільки такі системи мають доступ до чутливих даних, потрібно враховувати регуляторні обмеження, вимоги до управління ризиками та стандарти якості й безпеки інформаційних систем [1; 2; 20;22]:

а) ISO/IEC 27001 – регламентує архітектуру систем управління інформаційною безпекою (СУІБ), мінімізуючи ризики витоку даних;

б) SO/IEC 25010 – визначає критерії якості програмного забезпечення, зокрема метрики надійності, сумісності та стійкості алгоритмів;

в)ISO/IEC 38500 – формує механізм корпоративного управління IT-інфраструктурою на макрорівні організації.

1.3 Якість даних та вплив людського фактору на прийняття рішень

До того як банківський сектор почав автоматизовувати процеси, рішення про видачу кредитів приймали люди (кредитні експерти або андеррайтери). Проте людина – це нестабільний механізм. Історичні бази даних, на яких сьогодні тренуються сучасні скорингові системи, містять безліч людських помилок. Ці помилки створюють серйозну інженерну проблему – проблему якості вхідних даних[5].

Людський фактор та помилки експертів можна поділити на основні категорії,

а саме на когнітивне упередження та когнітивний шум.

Упередження – це систематична помилка. Вона виникає, коли експерт підсвідомо керується стереотипами і постійно відхиляє заявки певної групи людей(наприклад через вік, стать чи професію), ігноруючи реальні показники. Тобто він має стабільний зсув у оцінках, який не відповідає реальності[5; 18].

Відмінно від упередження, шум – це нестабільність і випадковість. На рішення людини впливає безліч зовнішніх факторів: втома, настрій, час доби або навіть складність попереднього робочого дня чи анкети. Через цей “шум” експерт може вранці схвалити, а ввечері-відхилити абсолютно ідентичні заявки[18;23]. Але на цих історичних даних навчається наша модель, це провокує проблеми Noisy Labels та GIGO.

Проблема шумливих міток (англ. Noisy Labels) – в базі даних з’являються клієнти з однаковими фінансовими показниками, але діаметрально різними рішеннями. Алгоритм намагається знайти в цьому хаосі логіку, що призводить до такого явища як “перенавчання” (англ. overfitting) моделі.

Проблема GIGO (Garbage In, Garbage Out) – сміття на вході, сміття на виході). Алгоритм сприймає людські стереотипи як правильні правила. Якщо експерти роками дискримінували певну категорію клієнтів, алгоритм просто масштабує цю аномалію, та продовжить відмовляти людям, але робитиме це автоматично[17].

Для вирішення цих проблем та правильного функціонування моделі оцінки ризиків потрібно ретельно підготувати інформацію. Цей процес називається попередньою обробкою даних (англ. Data Preprocessing), він є одним із найважливіших у розробці і включає такі кроки[12; 17;23]:

– очищення від аномалій: видалення з вибірки нелогічних рішень та викидів, які залишились через людський фактор;
– вирішення проблеми дисбалансу класів – в датасетах ситуацій дефолту кратно менше, ніж успішно виплачених кредитів. Щоб алгоритм навчився правильно розпізнавати ризикових клієнтів, класи потрібно штучно збалансувати. Для цього існують спеціальні методи SMOTE (Synthetic Minority Over-sampling Technique) або ADASYN (Adaptive Synthetic

Sampling) [10; 11];

– нормалізація даних, тобто приведення різноманітних фінансових показників клієнта до єдиного формату, щоб алгоритми могли коректно розрахувати їх вагу.

2 МЕТОДОЛОГІЧНІ ОСНОВИ ТА МАТЕМАТИЧНИЙ АПАРАТ

2.1 Постановка задачі кредитного скорингу як задачі класифікації

Зазвичай задача кредитного скорингу розглядається як класична задача навчання з учителем, а саме – як задача бінарної класифікації. Метою такої системи є прийняття об'єктивних рішень щодо нових позичальників банку на основі ретроспективних даних про попередніх позичальників. Кожного клієнта банку можна описати як об'єкт дослідження з певними наборами характеристик: соціально-демографічними даними, фінансовими показниками, параметрами кредитної історії тощо. Математично цей набір можна представити як багатовимірний вектор ознак [17] :

$$x = \{x_1, x_2 \dots, x_n\} \in X \tag{2.1}$$

де $X \subset R^n$ – n-вимірний простір ознак, а x_i – конкретне значення характеристики клієнта.

Для кожного вектора із бази даних відомий його фактичний результат – цільова змінна. У випадку нашої задачі це множина $y \in \{0,1\}$, де $y = 1$ – означає що позичальник успішно виконав свої зобов'язання(надійний клієнт), $y = 0$ – позичальник допустив дефолт(неплатоспроможний клієнт або шахрай).

Головна задача алгоритму зводиться до пошуку такої оптимальної функції класифікації, яка б максимально точно апоксимувала(спростила) невідому цільову залежність між вектором ознак та міткою класу:

$$f: X \to \{0,1\} \tag{2.2}$$

Але потрібно враховувати і реальність, класифікація клієнтів одразу на класи є неефективною, оскільки банк не може гнучко керувати ризиками. Тому сучасні алгоритми вирішують цю задачу. Замість прогнозування класу, предиктивні моделі оцінюють імовірність того, що клієнт з певним набором ознак належить до цього класу. Для того щоб перетворити надану алгоритмом імовірність на бізнес-рішення (в нашому випадку видати кредит чи відмовити), вводять порогове значення, що визначається політикою ризик-менеджменту банку. Змінюючи це значення банк

може адаптуватися та балансувати між ризиком втратити прибуток від хорошого клієнта та ризиком зазнання збитків від видачі коштів ненадійному клієнту [17; 24; 25; 26] .

Але повертаючись до математичного підґрунтя надзвичайно важливим є згадування функції втрат (англ. Loss Function), яку і мінімізують алгоритми машинного навчання, яка математично оцінює розбіжність між фактичними мітками та спрогнозованими імовірностями на вибірці. Стандартом вважається використання логарифмічної функції втрат (англ. Binary Cross-Entropy / Log-Loss):

$$L(y,p) = -\frac{1}{N}\sum_{i=1}^{N}[y_i log(p_i) + (1-y_i) log(1-p_i)] \rightarrow \min. \quad (2.3)$$

2.2 Аналіз існуючих методів для оцінки ризиків в банківському секторі

Вибір відповідного математичного апарату є одним з найважливіших аспектів побудови автоматизованої системи підтримки прийняття рішень. Відповідно до сучасних досліджень в сфері, оцінка кредитного ризику формалізується як задача бінарної класифікації. Системі подається багатовимірний вектор ознак позичальника $x = \{x_1, x_2 \dots, x_n\}$, а цільовою функцією є прогнозування мітки класу $y \in \{0,1\}$, де 1 – позначає стан дефолту, а 0 – успішне погашення кредиту.

Еволюція цих систем демонструє перехід від лінійних статичних моделей до складних нелінійних архітектур машинного навчання. Розглянемо далі три фундаментальні алгоритми, які відображають цю еволюцію.

Останнім часом традиційним чи базовим інструменту ризик-менеджменту була Логістична регресія – алгоритм, що належить до класу узагальнених лінійних моделей (англ.Generalized Linear Models) [24].

Математичний аппарат полягає не у прямому прогнозуванні класу, а в оцінці ймовірності настання дефолту(в нашому випадку ймовірності настання цільової функції). Для цього комбінація ознак трансформується за допомогою логістичної

функції, що стискає результат у діапазон [0,1] :

$$P(y = 1 \mid x) = \frac{1}{1 + e^{-(\beta_0 + \sum_{i=1}^{n} \beta_i x_i)}} \quad (2.4)$$

де β_0 – зсув, β_i – статистична вага відповідної фінансової ознаки x_i, e – основа натурального логарифма.

Ця модель є так званою «white box» – абсолютно прозорою системою, адже коефіцієнти β_i дозволяють аналітикам обчислити співвідношення класів (англ. Odds Ratio) для кожної ознаки. Це критично важливо для регуляторів , оскільки банк може математично обґрунтувати клієнту кожну відмову [18; 24].

Але в епоху Big Data та використання альтернативних даних алгоритм виявляє суттєві недоліки. Логістична регресія будує виключно лінійну площину між класами в n-вимірному просторі. Вона не здатна самостійно виявляти складні нелінійні взаємозвязки між ознаками без ручного конструювання многочленів(поліноміальних змінних), що робить її не ефективною на сучасних масивах даних. Але і використання занадто великої кількості поліномів може призвести до перенавчання, бо модель запам’ятає шум в даних а не виявить закономірність,тому стався перехід на наступний етап.

Наступним етапом для подолання проблеми лінійності стали дерева рішень (англ. Decision trees). Вони здатні до автоматичного знаходження нелінійних закономірностей, розбиваючи простір ознак на гіперпрямокутники. Проте одиничне дерево має завелику дисперсію: найменша зміна у вхідних даних може призвести дро кардинальної зміни структури всього дерева, що викликає глибоке перенавчання [17; 25].

Рішенням цієї проблеми став перехід до ансамблевих методів. Замість однієї складної моделі система агрегує результати множини малих алгоритмів. Найвідомішим представником і є випадковий ліс [25].

Як він працює:

1) з початкового датасету розміром N генерується K нових вибірок такого ж розміру N шляхом рандомізованого вибору з поверненням;

2) вибір випадкових ознак: У кожному вузлі дерева вибирається випадкова

підмножина ознак із загального пулу. Потім визначається найкращий розподіл серед цих ознак для подальшої класифікації даних. Це допомагає запобігти залежності моделі від будь-якої ознаки, що означає що вони будуть максимально декорельованими;

3) побудова дерева: кожне дерево в лісі вирощується незалежно, використовуючи вибрані дані, отримані за допомогою вибірок з попередніх кроків, та попередніх підмножин ознак. Але не обов'язково щоб ці дерева досягали повної зрілості, їх ріст можна зупинити і на якомусь етапі, наприклад в момент досягнення необхідного рівня чистоти;

4) прогнозування та агрегація: після того як дерева виростуть, кожне окреме дерево робить прогноз на основі своїх вивчених правил прийняття рішень. Зрештою найчастіший прогноз і стає остаточним.

Перевагою цього методу є математичне усереднення, що кардинально знижує дисперсію помилки. Алгоритм надзвичайно стійкий до когнітивного шуму в даних, наявності пропусків та статистичних викидів, що робить його одним із найнадійніших методів для фінансового сектору.

Коли алгоритми дерев рішень будують дерева паралельно та незалежно, то методи градієнтного бустингу використовують послідовне навчання. На сьогодні для роботи з банківськими даними стандартом вважають бібліотеку XGBoost[26; 27].

Як він працює:

1) початково система робить найпростіший прогноз для всіх клієнтів(для прикладу середня імовірність дефолту по всьому банку);

2) наступним кроком алгоритм порівнює реальний статус клієнта з базовим прогнозом. Різниця між фактом і прогнозом називають залишком, який переходить до наступного дерева;

3) друге дерево рішень у системі навчається вже не на оригінальних даних, йому поставлена задача передбачити виключно помилки попереднього дерева;

4) далі прогнози першого і другого дерева додаються. Зрештою

розраховується нова, вже менша помилка;

5) після цього цикл знову повторюється : алгоритм обчислює залишки, навчає наступне дерево на цих залишках і додає його внесок до загального прогнозу.

Цей процес є поступовим наближенням до ідеального результату. Нехай у нас є набір клієнтів, де x_i – дані клієнта, а y_i – реальний статус клієнта (1 – дефолт, 0 – надійний позичальник). Тоді процес має такий вигляд [26]:

– поточний прогноз алгоритму що складається з $m-1$ дерев, позначається як $F_{m-1}(x_i)$;
– алгоритм розраховує помилку для кожного клієнта:

$$r_i = y_i - F_{m-1}(x_i) \tag{2.5}$$

(якщо система передбачила ризик 0.3, а дефолт дорівнює, то помилка становить 0.7);

– далі будується наступне дерево рішень яке намагається спрогнозувати саме цю помилку на основі даних клієнта;
– прогноз всієї системи оновлюється шляхом додавання нового дерева:

$$F_m(x) = F_{m-1}(x) + \eta * h_m(x) \tag{2.6}$$

де $\eta \in (0; 1]$ – швидкість навчання (англ. learning rate). Цим коефіцієнтом можна зменшувати вплив кожного нового дерева, щоб система наближалася до ідеального прогнозу плавно і не зазнавала перенавчання на випадковому шумі.

Цей метод вирішує одну з головних проблем – екстремальний дисбаланс класів. Оскільки боржників у банківських базах завжди не велика кількість, прості моделі схильні ігнорувати їх. Проте архітектура градієнтного бустингу не дає можливості алгоритму проігнорувати їх. Якщо перше дерево пропустить шахраїв, це створить велику математичну помилку, відповідно якої наступні дерева будуть концентрувати ресурси саме на них, допоки помилка не буде мінімізована. Це і робить цей алгоритм найефективнішим інструментом для розпізнавання прихованих ризиків [11; 27].

2.3 Проблема дисбалансу класів у задачах фінансового скорингу

У процесі моделювання системи аналітики неминуче стикаються з фундаментальною проблемою, яка випливає із самою природи області – екстремальним дисбалансом класів (англ. Extreme class Imbalance)[10; 11].

У реальних портфелях комерційних банків співвідношення надійних позичальників до тих, хто допускає дефолт ніколи не бувають рівномірними. Стандартний розподіл варіюється в залежності від кредитної політики банку, та становить близько 95% до 5%, а під час жорстокої фази може сягати 98% до 2%. Ця асиметрія створює перешкоди для застосування методів машинного навчання.

Оскільки мажоритарний клас у вигляді надійних клієнтів має абсолютну кількісну перевагу, то ці елементи вибірки мають визначальний вплив на формування градієнта функції втрат. Моделі стає математично «вигідніше» повністю проігнорувати маленький відсоток шахраїв і віднести 100% клієнтів до переважаючого класу. У такому випадку формальна метрика загальної точності accuracy буде відповідати близько 95-98% що є відмінним результатом, проте практична цінність такої системи дорівнюватиме нулю.

З точки зору аналізу фінансових ризиків класи повинні розглядатися не лише через призму кількості в датасеті, але й через економічну вартість помилок. Фундаментом для розрахунку всіх сучасних метрик якості є матриця помилок (англ. Confusion Matrix) – таблиця контингентності, зображена на рис. 2.1, яка декомпозує результати класифікації шляхом порівняння фактичних значень цільової змінної зі спрогнозованими моделлю [12].

Кожен з елементів матриці має чітку економічну інтерпретацію та бізнесову важливість для фінансової установи [12]:

– True Positive (істинно позитивні): кількість випадків коли модель правильно ідентифікувала клієнта як такого, що допустить дефолт. Банк може обґрунтовано відмовити такому позичальнику, тим самим уникнувши прямих збитків;
– True Negative (істино негативні): кількість випадків, коли модель

правильно розпізнала надійного клієнта. Банк видав кредит надійній особі, та отримав запланований прибуток у вигляді відсотків та комісій;

– False Positive (хибно позитивні, так звана «Помилка 1 роду»): кількість випадків, коли надійного клієнта було помилково класифіковано як ненадійного. Банк безпідставно відхилив заявку, що в свою чергу призводить до втраченої вигоди та погіршення репутації установи;
– False Negative (хибно негативні, так звана «Помилка 2 роду»): кількість випадків, коли ненадійного клієнта було кваліфіковано як надійного. В випадку кредитної класифікації це найбільш критична помилка. Банк видає кошти шахраю або неплатоспроможній особі, що призводить до втрати всього тіла кредиту та додаткових витрат на роботу колекторських служб.

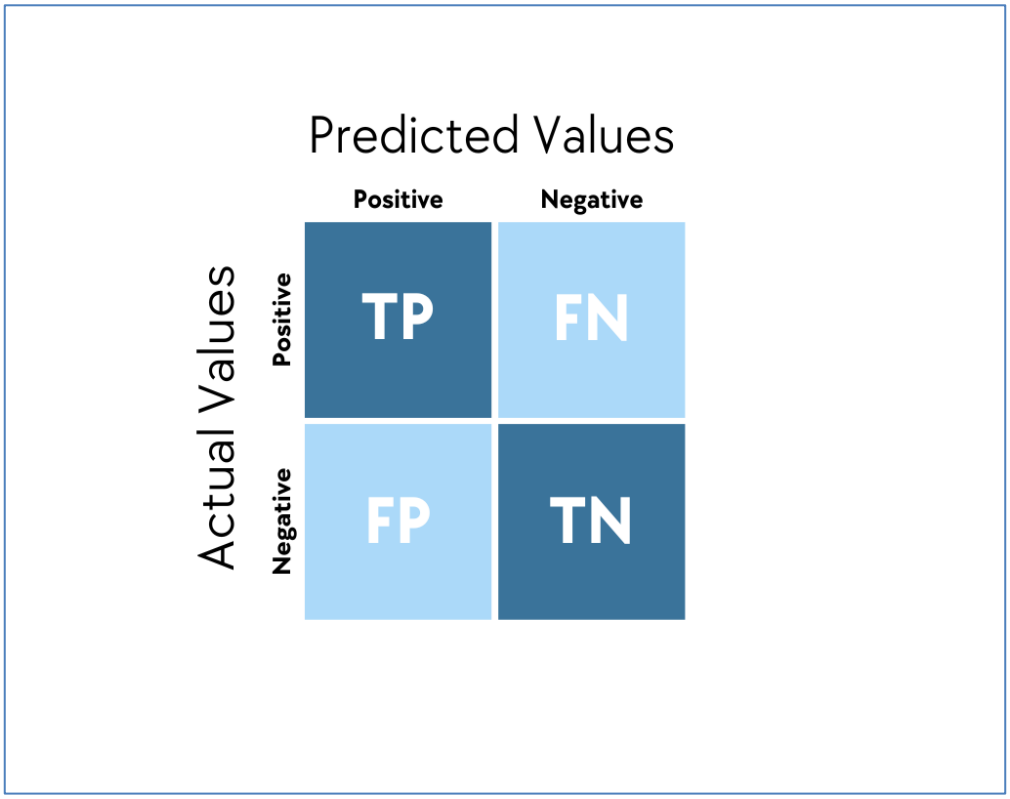


Рис. 2.1 Матриця помилок

З економічної точки зору, головна мета скорингової системи полягає у мінімізації елемента False Negative, оскільки вартість помилки другого роду є в десятки разів вищою за вартість помилки першого роду.

2.4 Метрики оцінювання якості моделей

На основі елементів матриці і розраховується найвідоміша статистична метрика – accuracy. Вона визначається як відношення кількості правильних прогнозів до загальної кількості класифікованих об’єктів [12]:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.7)$$

Вона доволі інтуїтивно зрозуміла, якщо, це означає, що модель приймає правильне рішення у 85% випадків. Однак у контексті нашого домену використання лише цієї метрики є не просто неінформативним, але й небезпечним, так як в умовах екстремального дисбалансу має місце такий феномен як «парадокс точності»(англ. accuracy paradox).

Припустимо, банк має тестову вибірку з 1000 клієнтів, серед яких 950 є надійними, а 50 – шахраями або клієнтами з визнаним дефолтом.Використавши в цьому кейсі примітивну модель,яка схвалює всі 1000 заявок,отримаємо матрицю помилок, наведену на рис 2.2.

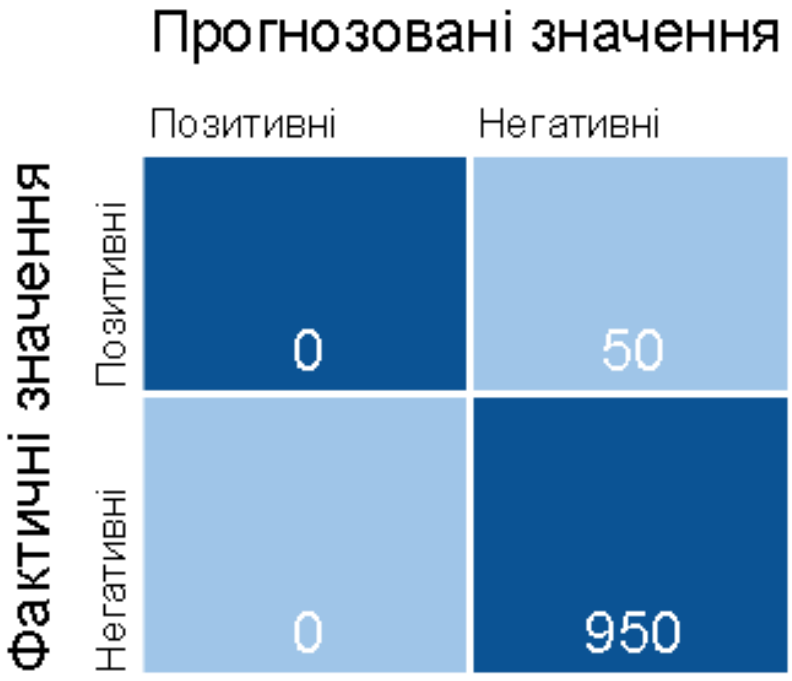


Рис. 2.2 Приклад використання матриці помилок для обрахунку accuracy

Розрахуємо точність такої системи для наочності:

$$Accuracy = \frac{0 + 950}{0 + 950 + 0 + 50} = \frac{950}{1000} = 0.95(95\%) \quad (2.8)$$

Формально алгоритм демонструє відмінну точність. Проте його практична та економічна доцільність нульова, оскільки він пропустив усі 100% дефолтів, що призведе до колосальних збитків установі. Саме тому були створені інші метрики.

Метрика влучності (англ. precision) показує, наскільки система заслуговує довіри в момент сигналізування про ризик. Вона визначає частку реальних дефолтів серед усіх клієнтів, яких алгоритм класифікував як ризикових [12]:

$$Precision = \frac{TP}{TP + FP} \quad (2.9)$$

Висока влучність означає, що банк мінімізує кількість помилкових відмов надійним клієнтам (FP), зберігаючи авторитет та отримуючи заплановані відсоткові доходи. Проте, максимізація лише цієї метрики може призвести до пропуску шахраїв через перенавчання моделі.

Метрика recall описує здатність моделі загалом знаходити міноритарний клас. Вона показує частку від усіх фактичних боржників у базі даних, яку алгоритм зміг виявити [12]:

$$Recall = \frac{TP}{TP + FN} \quad (2.10)$$

Оскільки помилка другого роду(пропуск дефолту) несе загрозу втрати всього тіла кредиту, саме максимізація recall є абсолютним бізнес-пріоритетом у задачах кредитного скорингу. Банку вигідніше відхилити кілька надійних заявок(зменшити precision), аніж видати велику суму шахраю (зменшити recall).

Між попередніми двома показниками влучності та повноти існує зворотна залежність: спроба покращити один показник майже завжди призводить до погіршення іншого. Для комплексної оцінки стабільності моделі використовують F1-міру, яка є середнім гармонійним між precision та recall [12]:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (2.11)$$

На відміну від середнього арифметичного, середнє гармонійне суворо «штрафує» модель, якщо одна з метрик наближається до нуля. Високий F1-score можливий лише за умови успішної підтримки балансу алгоритмом між пошуком дефолтів та уникненням хибних тривог.

Більшість новітніх алгоритмів які ми розглядали повертають не мітку приналежності до класу, а неперервну ймовірність настання дефолту клієнта. Перетворення цієї імовірності на остаточне рішення залежить від встановленого порогу відсікання [12].

Для порівняння предиктивної здатності моделей незалежно від обраного порогу використовують ROC-криву (Receiver Operating Characteristic), яка відображає ефективність моделі за всіма пороговими значеннями [12]. Приклад ROC-кривої наведено на рис. 2.3.

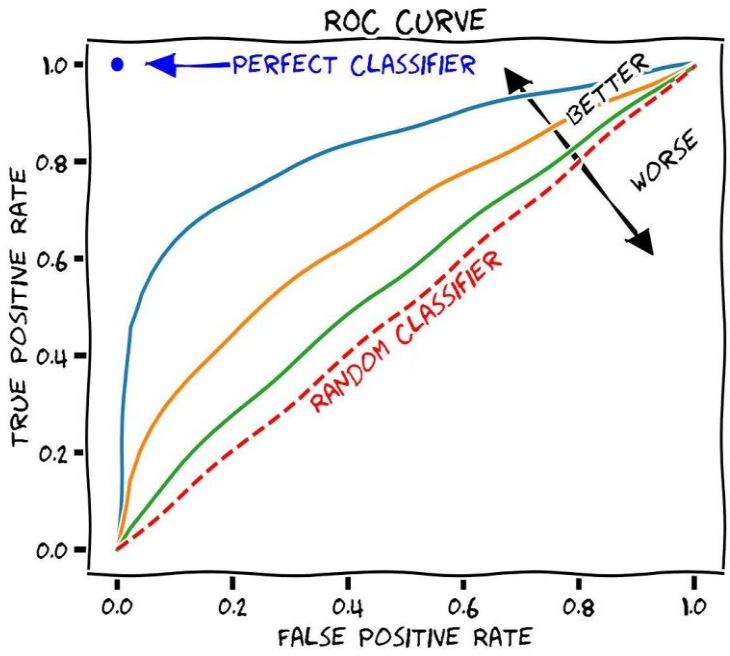


Рис. 2.3 ROC-крива

На рисунку 2.3 вісь Y позначає TPR (True Positive Rate), що є математичним еквівалентом метрики recall, а вісь X позначає FPR (False Positive Rate) – частку хибно позитивних класифікацій.

Крива будується шляхом перебору всіх можливих значень порогу від 0 до 1,де значення, а показником якості класифікатора виступає площа під ROC-кривою (Area Under the Curve, AUC), де AUC = 0.5 означає що модель працює на рівні випадкового підкидання монети та взагалі не має розділювальної здатності, а значення на рівні одиниці позначає ідеальну модель, яка безпомилково ранжує боржників вище за всіх надійних клієнтів.

У банківській сфері стандартом є використання коефіцієнта Джині, який лінійно виводиться з показника AUC [4; 12]:

$$Gini = 2 * AUC - 1 \quad (2.12)$$

Значення Gini вище 0.6 (або 60%) свідчить про високу комерційну придатність моделі.

Перевагою метрик AUC та Gini є їхня інваріантність до апріорного розподілу класів(можливість стабільно працювати в не залежності від перекосу набору даних). Навіть якщо дисбаланс зміниться під час економічного зростання, форма ROC- кривої залишиться стабільною. Це робить її єдиним рішенням для вибору найкращої алгоритмічної архітектури на етапі тестування.

2.5 Методи вирішення дисбалансу класів

Оскільки частка дефолтів у загальному портфелі становить критичну меншість, стандартні алгоритми оптимізації схильні ігнорувати цей клас заради мінімізації глобальної функції втрат, що призводить до парадоксу точності, який був описаний раніше. Для подолання цієї проблеми та зміщення розділювальної гіперплощини в бік справедливої класифікації застосовують рівні трансформації даних. До простих відносять Random undersampling та Random oversampling. Вони характеризуються збільшенням або зменшенням кількості записів одного з класів [10].

Random undersampling передбачає штучне зменшення обсягу записів мажоритарного класу шляхом випадкового вилучення доти, доки його розмір не

зрівняється з міноритарним класом. Він має в собі і перевагу в вигляді зменшення об’ємів даних, що зменшує час навчання алгоритму, але має в собі критичний архітектурний недолік, а саме – втрата інформації. Відкидаючи велику кількість записів даних мажоритарного класу алгоритм втрачає статистичні патерни та дисперсію, що описують різноманітність вибірки, що різко збільшує кількість помилок першого роду (False Positive) та відмовляє надійним клієнтам.

Random oversampling є протилежністю попереднього методу. Замість того щоб видаляти дані, він штучно збільшує міноритарний клас шляхом дублювання існуючих записів до досягнення балансу з мажоритарним класом. Також має в собі як переваги так і недоліки. Не створює жодної нової інформації, проте в контексті використання в більш складний моделях на основі дерев рішень це багаторазове синтетичне розмноження даних призведе до перенавчання алгоритму, який просто запам’ятає паттерн замість вивчення узагальнених правил виявлення подібних.

Просте дублювання існуючих записів про боржників призведе до сильного перенавчання, а зменшення записів не дасть змоги ефективно виявляти паттерни, тому для цього використовують методи ефективніші методи синтетичної інтерполяції.

Як вже було зазначено раніше, для уникнення перенавчання необхідно відмовитись від сліпого копіювання записів. Індустріальним стандартом вирішення цієї проблеми стає SMOTE (Synthetic Minority Over-sampling Technique) [10]. Логіку роботи алгоритму можна побачити на рис. 2.4.

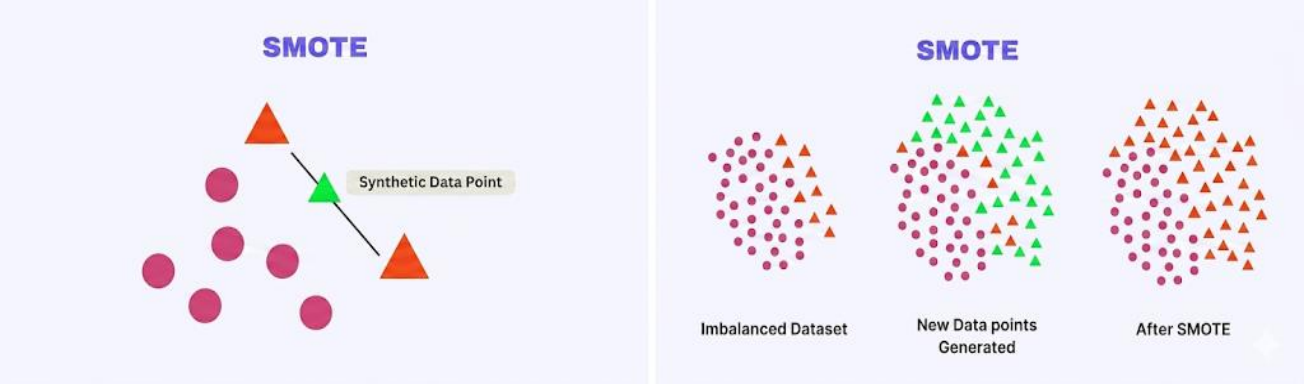


Рис. 2.4 Візуалізація синтетичних даних при використанні SMOTE

Для кожного базового вектора міноритарного класу алгоритм знаходить його k найближчих сусідів у багатовимірному просторі ознак за найменшою відстанню (Евклідова відстань), з цих знайдених сусідів випадкового обирається один і на відрізку між базовим вектором та обраним і створюється синтетичний.

Але зважаючи на ефективність алгоритм має недолік в вигляді сліпоти до мажоритарного класу. Він генерує однакову кількість даних для кожного боржника, але в випадку коли цей запис є аномалією і знаходиться всередині кластера надійних позичальників, то SMOTE почне генерувати синтетичні міноритарні класи на території мажоритарних, створюючи перекриття класів (англ. Overlapping), що в свою чергу призведе до алгоритмічного хаосу.

ADASYN (Adaptive Synthetic Sampling Approach) [10; 11] діє трохи «розумніше», очевидну ситуацію де міноритарний та мажоритарний клас чітко розмежовані і алгоритм виявить їх без проблем він пропускає. Натомість переходить до найскладнішої зони – межі прийняття рішень (англ. Decision Boundary), де межа між боржником та надійним клієнтом не така очевидна.

Алгоритм аналізує кожного боржника та оцінює його рівень складності, який вимірюється в кількості надійних клієнтів навколо нього. Якщо навколо боржника тільки аналогічні йому, то він майже не генерує навколо них даних, якщо ж боржник оточений надійними клієнтами то система фокусується на цьому випадку і генерує навколо нього максимальну кількість штучних екземплярів.

Перевагою цього алгоритму є концентрація на “проблемних зонах” між двома класами і доповнює метод градієнтного бустингу, максимізуючи результат на виході.

3 РЕАЛІЗАЦІЯ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Характеристика набору даних та постановка задачі

Для практичної реалізації було використано набір даних «default-of-credit-card-clients». Датасет містить інформацію про клієнтів кредитних карток у Тайвані та описує їхні демографічні характеристики, суму наданого кредиту, історію погашення за попередні місяць, розміри рахунків і фактичні суми платежів. Також цільова змінну default, яка приймає два значення: 0 – клієнт не допустив дефолт, 1 – клієнт допустив дефолт [13]. У табл. 3.1 подано основні ознаки датасету.

Таблиця 3.1

Основні ознаки датасету

Група ознак	**Змінні**	**Зміст**
Цільова змінна	default	Факт настання дефолту: 1–дефолт,0 – відсутність дефолту
Сума наданого кредиту	LIMIT_BAL	Сума виданого кредиту
Демографічні ознаки	SEX,EDUCATION, MARRIAGE,AGE	Стать,освіта,сімейний стан,вік клієнта.
Історія погашення	PAY_1 – PAY_6	Статус погашення за попередні місяці,включно з простроченням
Суми рахунків	BILL_AMT1 – BILL_AMT6	Розмір нарахованої заборгованості за місяцями
Суми платежів	PAY_AMT1-PAY_AMT6	Фактичні платежі,здійснені клієнтом за місяцями

Цей набір даних описує не один окремий кредитний договір, а поведінку

клієнта у межах використання кредитної картки. Це важливо для інтерпретації ознак: змінна LIMIT_BAL у роботі розглядається як сума наданого кредиту, тобто погоджений обсяг кредитного ресурсу для клієнта. Ознаки BILL_AMT1-BILL_AMT6 відображають суми рахунків за шість місяців, а PAY_AMT1-PAY_AMT6 - суми фактичних платежів. Ознаки PAY_1-PAY_6 характеризують статус погашення в кожному з попередніх місяців і є поведінковими індикаторами платіжної дисципліни.

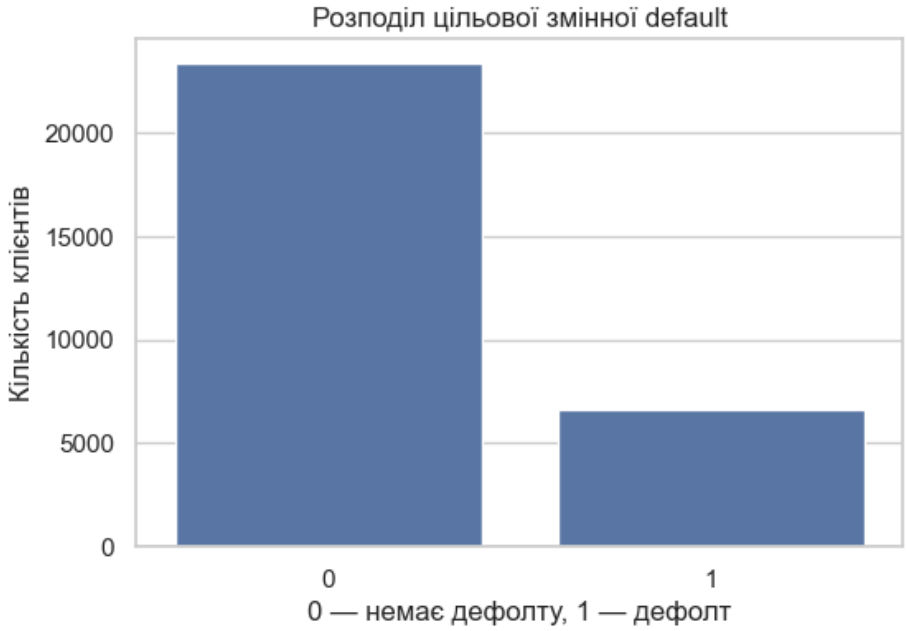


Рис. 3.1 Розподіл цільової змінної default

Первинна перевірка показала, що датасет містить 30 000 спостережень та 24 початкові змінні. У базових полях не виявлено пропущених значень, що дозволило перейти до аналізу змісту змінних без попереднього відновлення пропусків. Разом з тим було встановлено суттєву особливість задачі – дисбаланс класів(рис. 3.1). Частка клієнтів із дефолтом становить приблизно 22,1 %, тоді як клієнтів без дефолту – близько 77,9 %. Така структура є доволі типовою для задач такого типу, оскільки дефолтні випадки зазвичай трапляються рідше, ніж нормальне виконання зобов’язань [13].

Наявність дисбалансу класів одразу впливає на вибір підходів до оцінювання моделей. Якщо використовувати лише accuracy, модель може демонструвати прийнятний результат навіть тоді, коли вона погано знаходить дефолтних клієнтів.

Саме тому в подальшому аналізі акцент зроблено на precision, recall, F1-score, ROC-AUC та PR-AUC. Ці метрики дозволяють оцінити не лише загальну правильність прогнозів, а й якість виявлення менш чисельного, але найбільш важливого для задачі класу default.

3.2 Розвідувальний аналіз даних і виявлення ключових закономірностей

Розвідувальний аналіз даних проводився з метою не лише побачити розподіли окремих змінних, а й зрозуміти, які групи ознак можуть бути корисними для прогнозування дефолту. У межах цього етапу було проаналізовано описову статистику, категоріальні групи клієнтів, фінансові змінні, історію прострочень та кореляції з цільовою змінною[13].

За результатами описової статистики видно, що фінансові ознаки мають значний розкид. Сума наданого кредиту варіюється від 10 000 до 1 000 000 доларів, а суми рахунків і платежів містять як дуже малі, так і дуже великі значення. Для фінансових даних така ситуація є очікуваною: клієнти мають різний рівень кредитного навантаження, різні ліміти та різну активність користування кредитом. Окремо варто зазначити, що у змінних BILL_AMT можуть траплятися від’ємні значення. Вони не обов’язково є помилкою, оскільки у контексті кредитної картки можуть відображати переплату, коригування рахунку або залишок на користь клієнта.

На рисунку 3.2 порівняно розподіли суми наданого кредиту та віку клієнтів для двох класів. За сумою наданого кредиту видно, що клієнти з меншими значеннями LIMIT_BAL частіше потрапляють до групи дефолту. Це не означає прямої причинності, однак може свідчити про те, що клієнтам із нижчим фінансовим профілем або меншою довірою банку частіше надавалися менші кредитні суми, і вони мали вищий ризик невиконання зобов’язань. Вік клієнта має менш виражений зв’язок із дефолтом, тому ця ознака розглядається як допоміжна

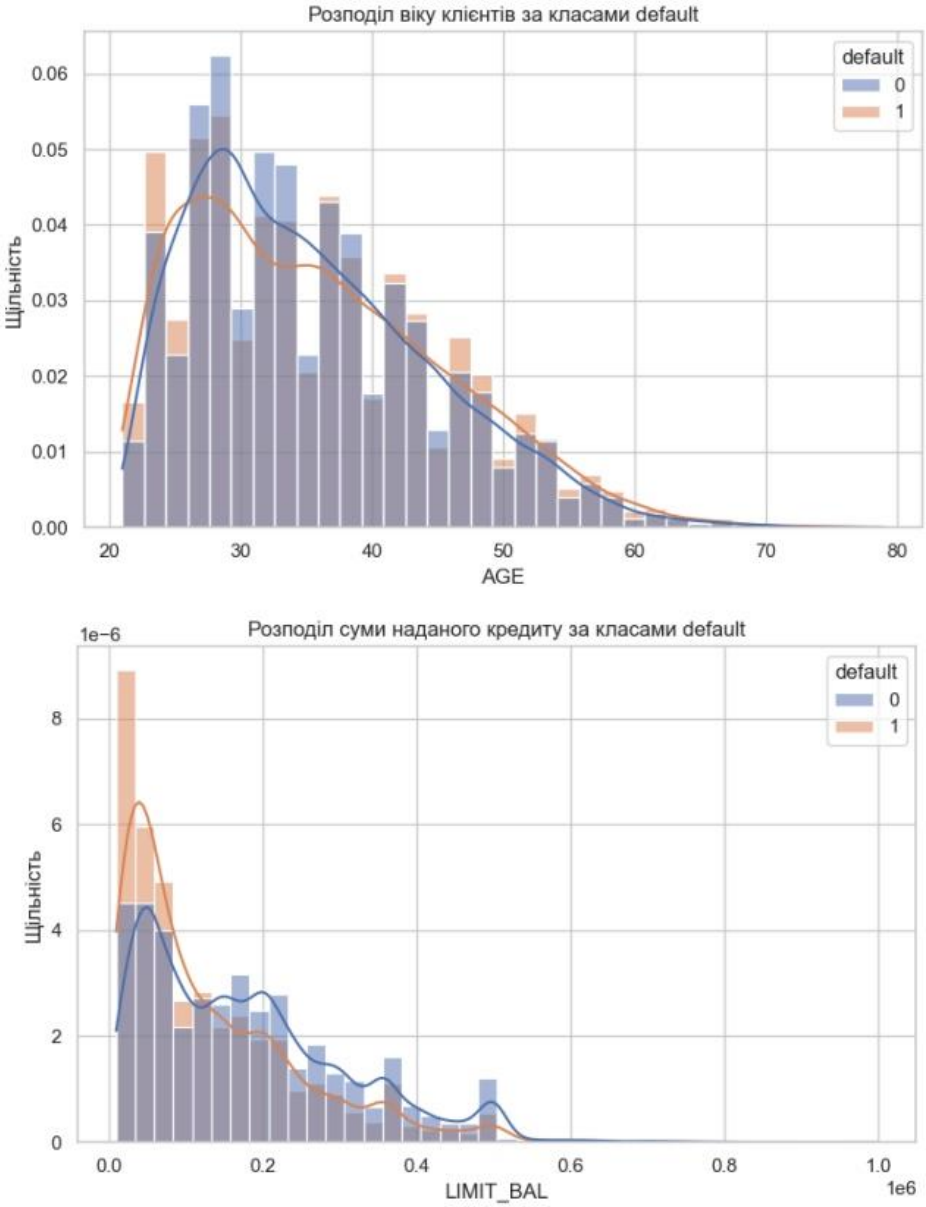


Рис. 3.2 Розподіл LIMIT_BAL та AGE за класами default

Аналіз демографічних ознак на рис. 3.3 показує, що стать, освіта та сімейний стан не формують настільки сильного сигналу, як історія платежів, проте певні відмінності між групами присутні. Такі ознаки не варто вилучати лише через слабший прямий зв’язок, оскільки в поєднанні з іншими змінними вони можуть допомагати моделі точніше формувати ризиковий профіль клієнта. При цьому інтерпретувати демографічні змінні потрібно обережно: вони не мають бути єдиною підставою для кредитного рішення, а повинні використовуватися лише як частина комплексної оцінки для оцінювання профілю клієнта.

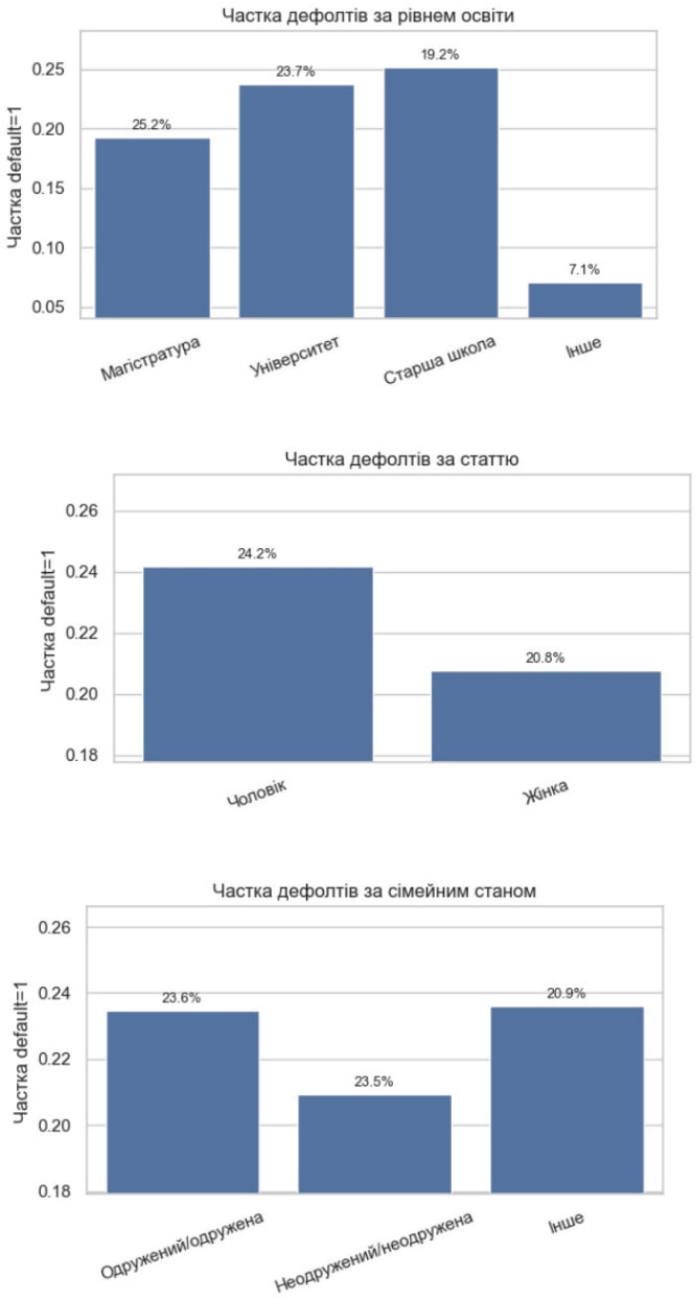


Рис. 3.3 Частка дефолтів у групах

Найбільш показовим результатом EDA є аналіз ознак PAY_1-PAY_6 (рис. 3.4). Ці змінні описують історію погашення за попередні місяці, тому безпосередньо відображають поведінкову дисципліну клієнта. На графіках видно, що зі зростанням статусу прострочення частка дефолтів помітно підвищується. Це є логічним результатом: клієнт, який уже мав прострочення в минулому, з більшою ймовірністю може мати проблеми з виконанням зобов’язань у майбутньому.

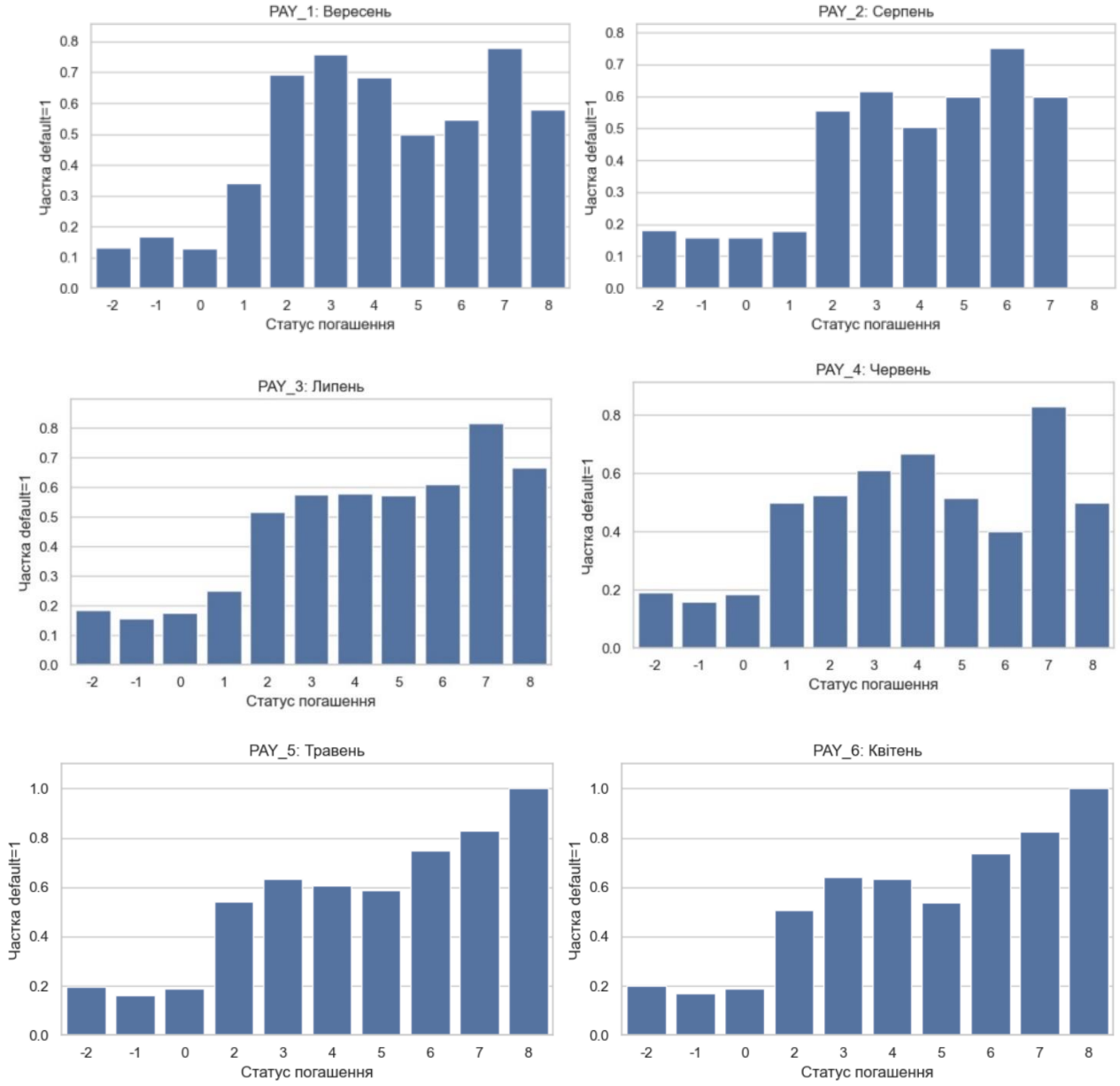


Рис. 3.4 Частка дефолтів залежно від статусу погашення PAY_1-PAY_6

Саме поведінкові ознаки стали основою для подальшого створення синтетичних показників. Замість того щоб використовувати лише окремі значення PAY_1-PAY_6, доцільно сформувати узагальнені характеристики: максимальний рівень прострочення, середній статус погашення, кількість місяців із простроченнями та поведінковий індекс ризику. Такий підхід робить вхідний простір ознак більш змістовним для моделей машинного навчання. Кореляції фінансових і поведінкових ознак із цільовою змінною default наведено на рис. 3.5.

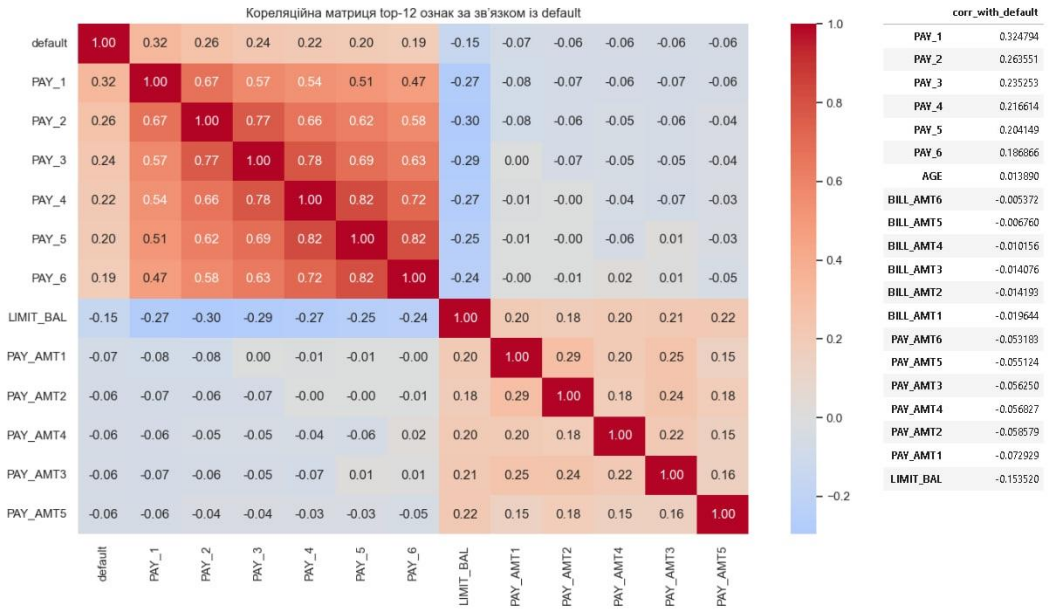


Рис. 3.5 Кореляції фінансових і поведінкових ознак з default

Аналіз підтвердив висновки візуального EDA. Найбільший додатний зв'язок із default мають саме ознаки PAY_1-PAY_6, тобто історія прострочень. Ознака LIMIT_BAL має від'ємну кореляцію з дефолтом: у середньому клієнти з більшою сумою наданого кредиту рідше переходять до дефолтного класу. Суми платежів PAY_AMT мають слабку від'ємну кореляцію, що свідчить: активніше погашення заборгованості пов'язане з нижчим ризиком дефолту.

Водночас абсолютні суми рахунків BILL_AMT мають слабкий прямий зв'язок із цільовою змінною. Це пояснюється тим, що велика сума рахунку сама по собі не є однозначним індикатором ризику. Якщо клієнт має високий кредитний ресурс і регулярно здійснює платежі, велика сума рахунку може бути нормальною частиною фінансової поведінки. Тому для аналізу потрібні не лише абсолютні значення, а й відносні показники: використання кредиту, співвідношення платежів до рахунків, зміна боргу між місяцями.

3.3 Підготовка даних і формування синтетичних ознак

Після первинного аналізу було виконано створення синтетичних ознак на основі початкових змінних (табл. 3.2). Мета цього етапу полягає в тому, щоб

перетворити сирі значення датасету на показники, які краще відображають економічний і поведінковий зміст кредитного ризику. У початковому наборі даних є багато місячних ознак, але модель не завжди може самостійно виявити всі важливі співвідношення між ними. Синтетичні ознаки допомагають явно передати моделі узагальнену інформацію про боргове навантаження, платіжну активність і дисципліну клієнта [13].

Таблиця 3.2

Сформовані синтетичні ознаки

Синтетична ознака	**Призначення**	**Логіка формування**
TOTAL_BILL	Оцінка загального боргового навантаження	Сума BILL_AMT1-BILL_AMT6.
TOTAL_PAY	Оцінка загальної платіжної активності	Сума PAY_AMT1-PAY_AMT6.
PAY_TO_BILL_RATIO	Показник здатності погашати борг	Відношення загальних платежів до загальної суми рахунків.
UTIL_1-UTIL_6	Використання наданого кредиту за місяцями	BILL_AMT за місяць / LIMIT_BAL.
AVG_UTIL, MAX_UTIL, STD_UTIL	Узагальнення кредитного навантаження	Середнє, максимальне та мінливість використання кредиту.
MEAN_PAY_DELAY	Середній рівень платіжної дисципліни	Середнє значення PAY_1-PAY_6.
MAX_PAY_DELAY	Найгірший зафіксований стан погашення	Максимальне значення PAY_1-PAY_6.

Продовження таблиці 3.2

Синтетична ознака	**Призначення**	**Логіка формування**
NUM_DELAY_MONTHS	Повторюваність прострочень	Кількість місяців, де PAY > 0.
NUM_SEVERE_DELAY_MONTHS	Кількість суттєвих прострочень	Кількість місяців, де прострочення є значним.
BEHAVIOR_RISK_SCORE	Узагальнений поведінковий ризик	Комбінація середнього, максимального та повторюваного прострочення.
BILL_CHANGE_1_6	Зміна боргу між першим і шостим місяцем	Різниця між актуальною та давнішою сумою рахунку.
BILL_GROWTH_RATE	Темп зміни боргового навантаження	Відносна зміна рахунків.
LOG_LIMIT_BAL	Стабілізація масштабу суми кредиту	Логарифмічне перетворення LIMIT_BAL.
LIMIT_PER_AGE	Умовний рівень кредитного ресурсу на рік віку	LIMIT_BAL / AGE.

Особливе місце серед синтетичних ознак займає behavior_risk_score (рис. 3.6). Він не є окремою початковою змінною з датасету, а формується на основі історії погашення. Його зміст полягає в тому, щоб одним числом узагальнити декілька аспектів поведінкового ризику: наскільки часто клієнт допускав

прострочення, наскільки серйозними вони були і який загальний рівень платіжної дисципліни спостерігався за кілька місяців. Такий показник є зручним для моделі, оскільки перетворює серію місячних статусів у компактний ризиковий індикатор

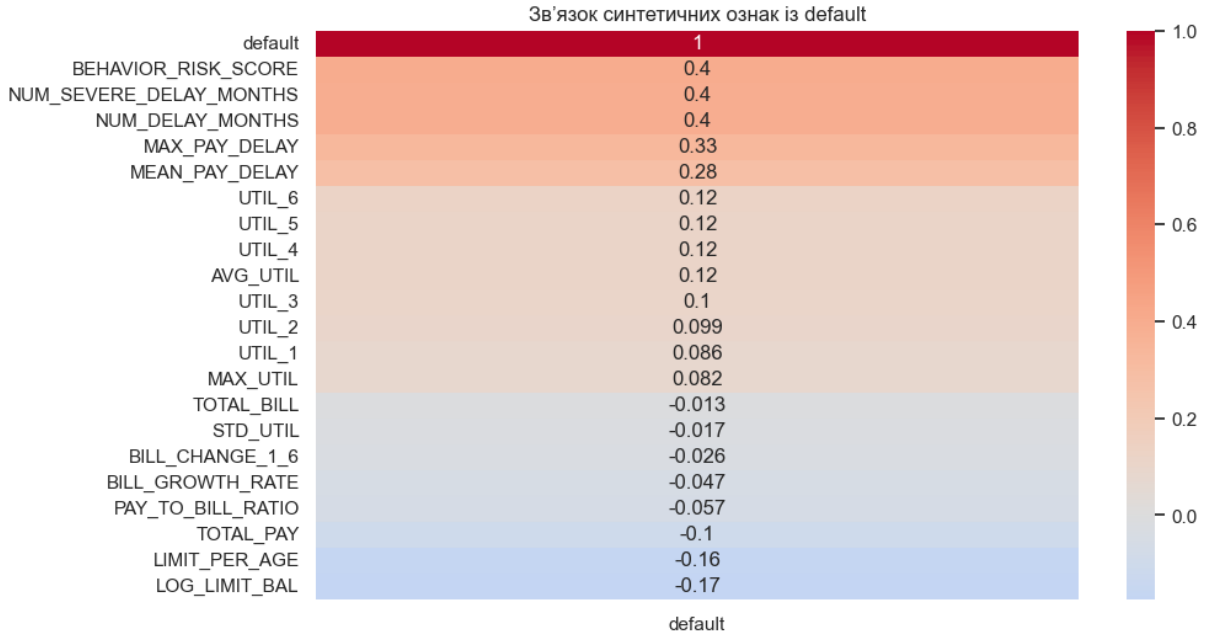


Рис. 3.6 Зв’язок синтетичних ознак із default

Після створення синтетичних ознак кількість змінних зросла з 24 до 45. Це не є механічним збільшенням розмірності, оскільки нові змінні мають аналітичний зміст. Наприклад, TOTAL_BILL і TOTAL_PAY показують загальні обсяги боргу та платежів, а PAY_TO_BILL_RATIO відображає, наскільки платежі покривають сформовану заборгованість. Ознаки NUM_DELAY_MONTHS і MAX_PAY_DELAY дають змогу моделі враховувати не лише факт прострочення, а й його тривалість і глибину.

У роботі також розглядалося питання екстремальних значень у фінансових змінних. Оскільки суми рахунків, платежів і похідні від них коефіцієнти можуть мати дуже великі значення, такі спостереження здатні впливати на масштаб ознак і на роботу окремих алгоритмів. Тому вінсоризацію доцільно розміщувати не як фінальний крок, а після розбиття даних на train та test і перед моделюванням. Це дозволяє обробляти екстремальні значення без витоку інформації з тестової вибірки.

Логіка використання вінсоризації полягає не у видаленні клієнтів із

незвичайною поведінкою, а в обмеженні надмірно великих або малих значень для тих фінансових ознак, які можуть спотворити масштаб. Такий підхід особливо важливий для похідних коефіцієнтів, наприклад UTIL або PAY_TO_BILL_RATIO, де ділення на малі значення може створювати дуже різкі викиди. При цьому поведінкові категоріальні ознаки прострочення не потрібно вінсоризувати, оскільки їхні значення мають інтерпретовану шкалу.

3.4 Сегментація клієнтів за допомогою кластеризації KMeans

Додатковим етапом дослідження стала кластеризація клієнтів методом KMeans. На відміну від класифікаційних моделей, кластеризація не використовує цільову змінну default під час навчання. Її мета полягає не в прямому прогнозуванні дефолту, а у виявленні груп клієнтів зі схожими фінансовими та поведінковими характеристиками [12]. Визначення кількості кластерів методом ліктя подано на рис. 3.7.

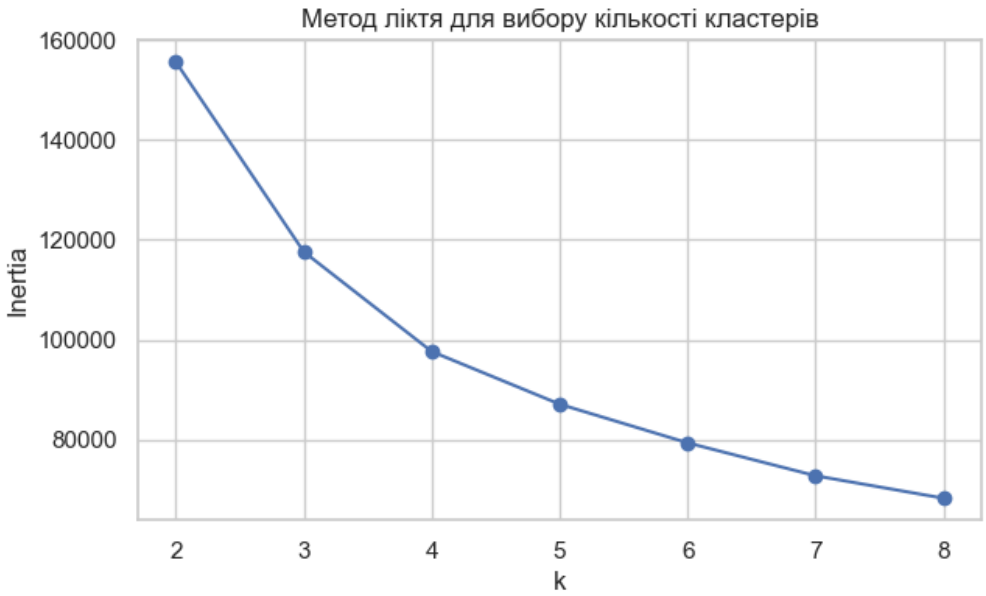


Рис. 3.7 Визначення кількості кластерів методом ліктя

Метод ліктя дозволяє оцінити, після якої кількості кластерів зменшення внутрішньокластерної відстані стає менш суттєвим. Силуетний коефіцієнт(англ. Silhouette score) доповнює цей аналіз і показує, наскільки добре об’єкти розділені

між кластерами. Разом ці два підходи дають змогу обґрунтувати вибір кількості сегментів не інтуїтивно, а на основі кількісних характеристик.

Додаткову візуалізацію якості кластеризації наведено на рис. 3.8.

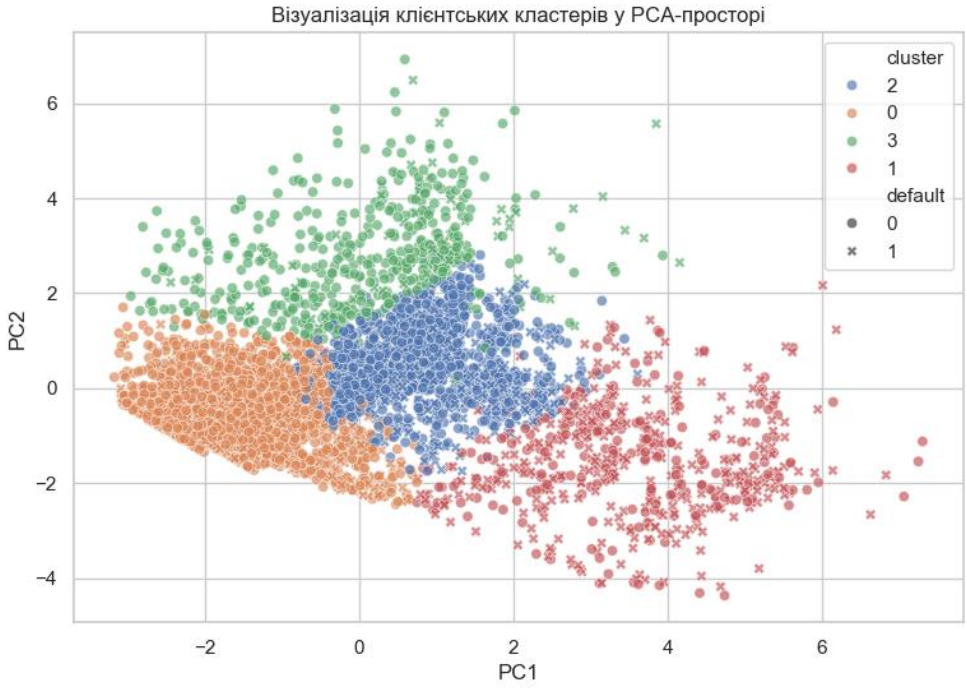


Рис. 3.8 Візуалізація кластерів

На візуалізації кластерів точки відображають надійних клієнтів, а хрести – дефолти клієнтів. Це дає змогу побачити, навколо яких умовних центрів групуються клієнти.Така сегментація корисна тим, що клієнти з однаковим прогнозованим ризиком можуть мати різну природу цього ризику: одні – через історію прострочень, інші – через високе використання кредиту або слабку платіжну активність. Отже, кластер можна використати як додаткову синтетичну ознаку , що передає моделі інформацію про тип клієнтського профілю.

3.5 Попередня обробка та навчання моделей

Після формування ознак дані було розділено на навчальну та тестову вибірки. Для коректного поділу використано стратифікацію за цільовою змінною, що дозволило зберегти однакову частку дефолтних клієнтів у train і test. У результаті навчальна вибірка містила 24 000 спостережень, а тестова – 6 000 спостережень.

Частка default у двох вибірках залишилася практично однаковою завдяки стратифікації, що є важливою умовою коректного порівняння моделей [12].

Фрагмент коду розбиття даних на навчальну та тестову вибірки наведено на рис. 3.9.

```
X_all = df_fe.drop(columns=['default'])
y_all = df_fe['default'].astype(int)

X_train, X_test, y_train, y_test = train_test_split(
    X_all, y_all, test_size=0.2, random_state=RANDOM_STATE, stratify=y_all
)

print('Train:', X_train.shape, 'Test:', X_test.shape)
print('Default rate train:', y_train.mean())
print('Default rate test:', y_test.mean())
```

```
Train: (24000, 44) Test: (6000, 44)
Default rate train: 0.22120833333333334
Default rate test: 0.22116666666666668
```

Рис. 3.9 Фрагмент коду розбиття на train і test

Далі переходимо до навчання моделей,їх список можна побачити на рис. 3.10.

```
Навчання: Dummy_most_frequent_none
Навчання: LogReg_none
Навчання: LogReg_SMOTE
Навчання: LogReg_ADASYN
Навчання: LogReg_balanced_none
Навчання: RandomForest_none
Навчання: RandomForest_SMOTE
Навчання: RandomForest_ADASYN
Навчання: RandomForest_balanced_none
Навчання: XGBoost_none
Навчання: XGBoost_SMOTE
Навчання: XGBoost_ADASYN
Навчання: XGBoost_weighted_none
```

Рис. 3.10 Список створених моделей

На цьому етапі було використано моделі різного рівня складності, модель Dummy Classifier використовувалась як контрольна модель, щоб показати чому при дисбалансі класів не можна орієнтуватись лише на accuracy. Основними моделями для порівняння стали Logistic Regression, Random Forest та XGBoost та для наочності їх версії з синтетичним балансуванням класів [12; 24; 25; 26].

Реалізація етапів попередньої обробки даних, формування навчальної та тестової вибірки, а також навчання моделей машинного навчання наведена в додатку А.

3.6 Порівняння моделей машинного навчання та аналіз метрик

На першому етапі було побудовано ширше порівняння моделей. Однак надмірна кількість варіантів робить графік менш читабельним, тому для фінального узагальнення доцільно залишити лише репрезентативні моделі які були відібрані за PR-AUC. Порівняння найкращих восьми моделей за показником PR-AUC наведено на рис. 3.11.

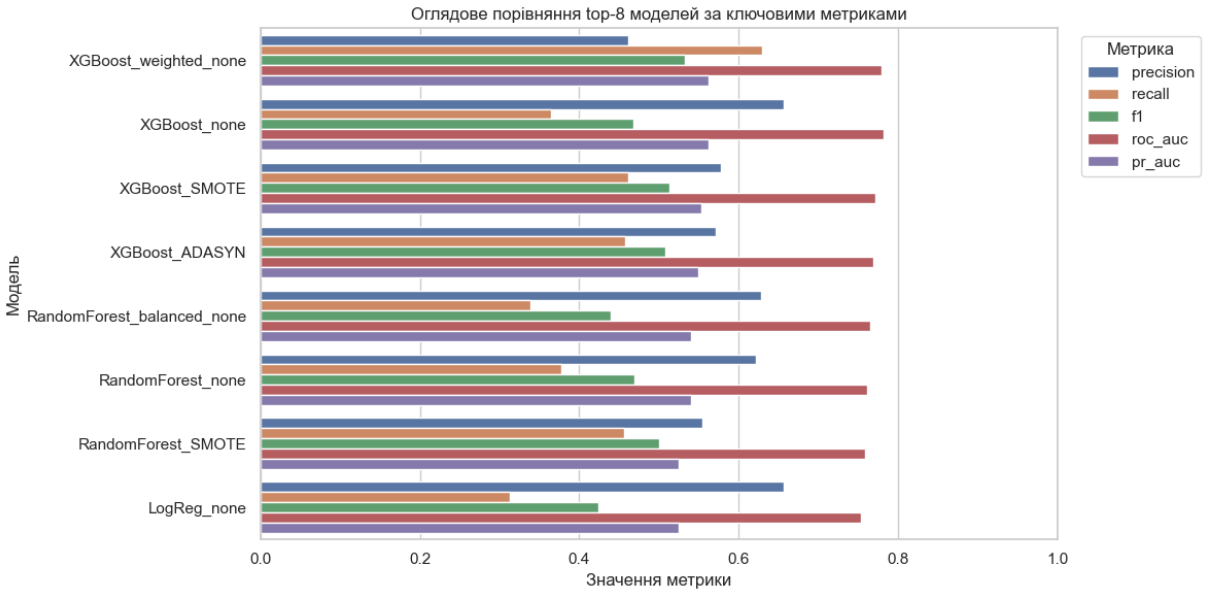
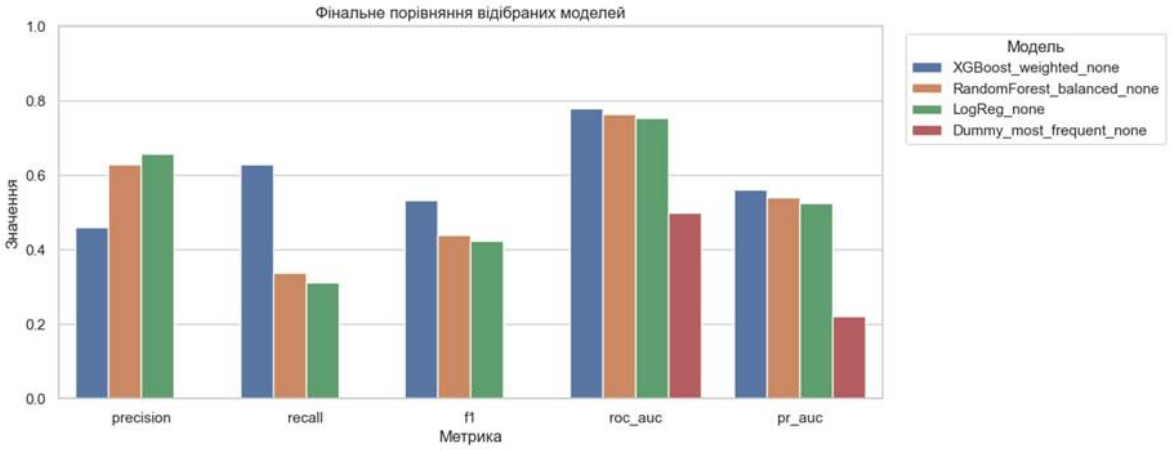


Рис. 3.11 Порівняння найкращих 8 моделей за PR-AUC

Для кожної моделі було вибрано найсильніших кандидатів за якістю роботи, тому далі розглядаємо наступні моделі, які наведені на рис. 3.12.

Результати показують, що найвища accuracy не є головним критерієм у цій задачі. Наприклад, логістична регресія має accuracy понад 0,81, однак її recall для дефолтного класу становить лише близько 0,31. Це означає, що модель пропускає значну частину клієнтів, які фактично допустили дефолт. Для кредитного ризику така ситуація є небажаною, оскільки пропущений ризиковий клієнт може бути дорожчим для бізнесу, ніж додаткова перевірка надійного клієнта [12].



Моделі, залишені для короткого фінального порівняння:
['Dummy_most_frequent_none', 'LogReg_none', 'RandomForest_balanced_none', 'XGBoost_weighted_none']

	model	accuracy	balanced_accuracy	precision	recall	f1	roc_auc	pr_auc
0	XGBoost_weighted_none	0.755333	0.710190	0.461071	0.629239	0.532186	0.779238	0.562247
4	RandomForest_balanced_none	0.809500	0.640824	0.628852	0.338357	0.439980	0.765250	0.540914
7	LogReg_none	0.812000	0.633526	0.657188	0.313489	0.424490	0.754206	0.524399
12	Dummy_most_frequent_none	0.778833	0.500000	0.000000	0.000000	0.000000	0.500000	0.221167

Рис. 3.12 Кінцеві моделі для порівняння

Найбільш збалансованим результатом серед відібраних моделей є XGBoost_weighted_none. Вона має recall близько 0,629, тобто виявляє приблизно 63 % дефолтних клієнтів при стандартному порозі 0,5. При цьому F1-score дорівнює приблизно 0,532, а PR-AUC - 0,562. Саме PR-AUC є важливою метрикою для незбалансованих даних, оскільки вона оцінює якість роботи з позитивним класом default = 1, а не лише загальну здатність розділяти класи [12; 26].

3.7 Аналіз порогу класифікації та матриці помилок

Модель класифікації фактично формує не лише клас, а й імовірність належності клієнта до дефолтного класу. Поріг класифікації threshold визначає, з якого рівня цієї ймовірності клієнт позначається як ризиковий. Стандартне значення 0,5 не завжди є оптимальним для кредитного ризику, оскільки в реальних задачах вартість помилок різна: пропустити дефолтного клієнта зазвичай небезпечніше, ніж направити на додаткову перевірку клієнта без дефолту [12].

Вплив зміни порогу класифікації на Precision, Recall та F1-score наведено на рис. 3.13.

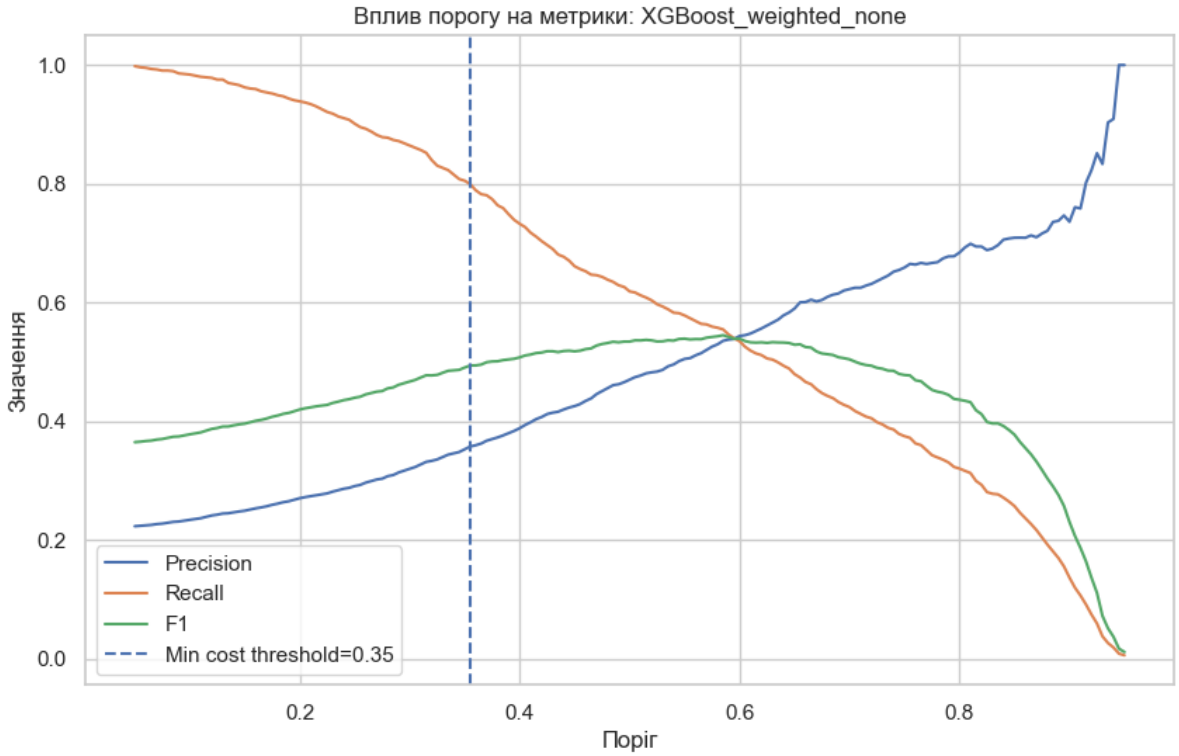


Рис. 3.13 Вплив порогу класифікації на precision, recall та F1-score

Вплив порогу класифікації на умовну бізнес-вартість помилок наведено на рис. 3.14.

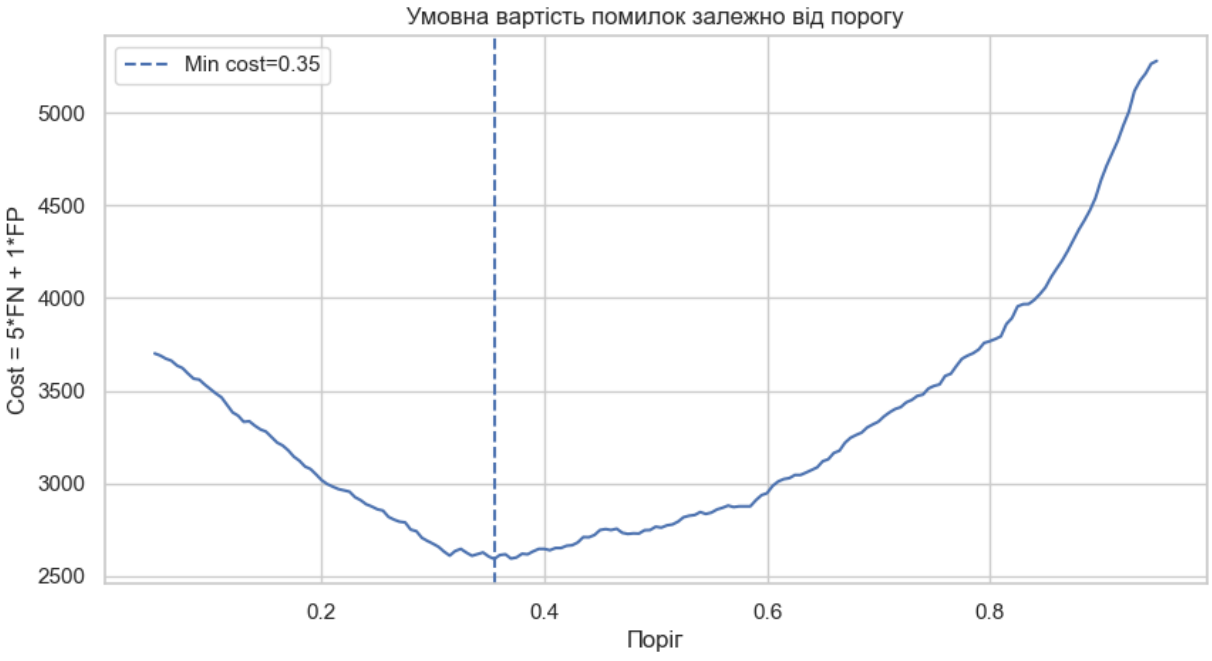


Рис. 3.14 Вплив порогу на бізнес-вартість помилок

Зниження threshold робить модель чутливішою до дефолтного класу: вона частіше позначає клієнтів як ризикових, завдяки чому зростає recall. Однак разом із цим збільшується кількість false positive – клієнтів без дефолту, яких модель помилково визначила як ризикових. Підвищення threshold, навпаки, робить модель обережнішою: precision може зростати, але recall знижується, тобто частина фактично дефолтних клієнтів буде пропущена.

Результати для різних порогів класифікації моделі XGBoost_weighted_none наведено на рис. 3.15.

	threshold	precision	recall	specificity	f1	cost	fp	fn	tp	tn
61	0.355	0.357323	0.799435	0.591493	0.493892	2592	1527	213	849	2211
64	0.370	0.367139	0.780603	0.617710	0.499398	2594	1429	233	829	2309
65	0.375	0.369937	0.774011	0.625468	0.500609	2600	1400	240	822	2338
60	0.350	0.352577	0.805085	0.579989	0.490393	2605	1570	207	855	2168
57	0.335	0.343689	0.822976	0.553505	0.484882	2609	1669	188	874	2069
53	0.315	0.331502	0.852166	0.511771	0.477321	2610	1825	157	905	1913
62	0.360	0.359502	0.789077	0.600589	0.493958	2613	1493	224	838	2245
67	0.385	0.376108	0.758945	0.642322	0.502964	2617	1337	256	806	2401
63	0.365	0.362407	0.782486	0.608882	0.495380	2617	1462	231	831	2276
58	0.340	0.345847	0.815443	0.561798	0.485698	2618	1638	196	866	2100

Рис. 3.15 Різні пороги класифікації XGBoost_weighted_none

За результатами аналізу поріг threshold = 0,355 забезпечує recall на рівні 0,799, тобто модель виявляє приблизно 79,9% клієнтів, які фактично допустили дефолт. При цьому precision становить 0,357, що означає: серед клієнтів, який модель визначила як ризикових, близько 35,7% справді належать до дефолтного класу. Значення F1– score дорівнює 0.494, що відображає компроміс між повнотою виявлення дефолтів і точністю ризикових пронозів. З цього ми можемо зробити висновок що зниження порогу вивилось доволі ефективним.Далі для аналізу була побудована матриця помилок для нашої моделі,її зображення можна побачити на рисунку 3.16.

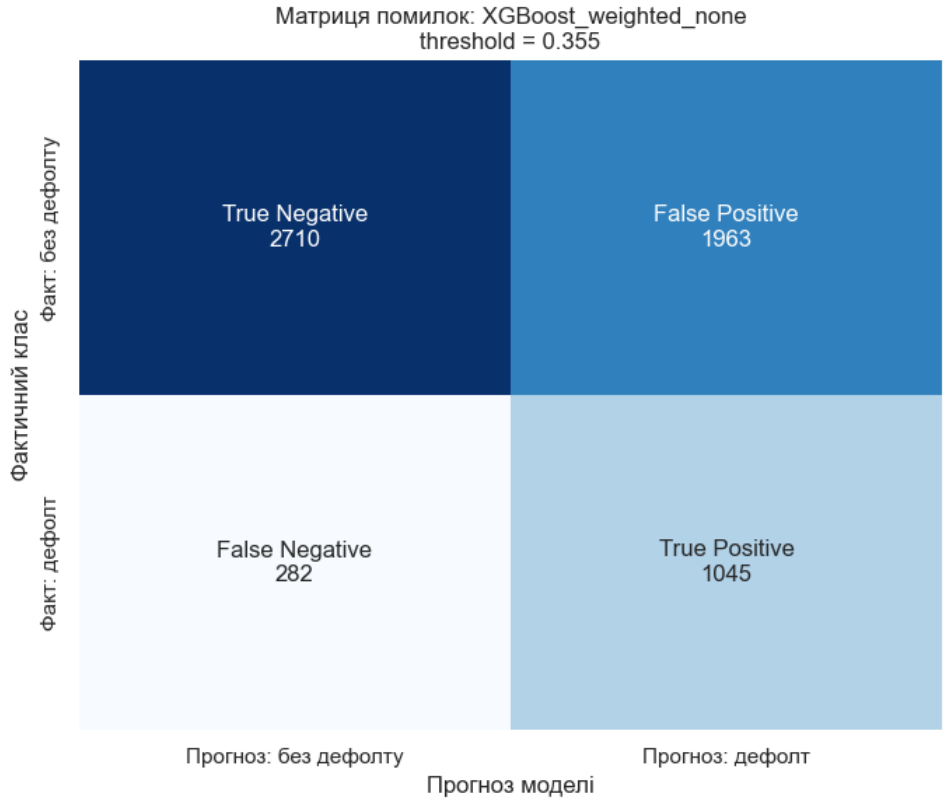


Рис. 3.16 Матриця помилок для найефективнішої моделі

Поріг 0,355 також має найменше значення умовної вартості помилок у отриманій таблиці – 2592. Це означає, що за заданих припущень про більшу вартість пропуску дефолтного клієнта порівняно з помилковим ризиковим спрацюванням саме цей поріг є найбільш доцільним. Але така стратегія супроводжується збільшенням кількості false positive: 1527 клієнтів без дефолту були віднесені до ризикових. Отже вибір цього порогу є виправданим у сценарії мінімізації та недопущення потенційно дефолтного клієнта. Повний програмний код реалізації моделей,розрахунку метрик та побудови графічних результатів наведено в додатку А.

3.8 Інтерпретація моделі та практичне застосування результатів

Після вибору найкращої моделі важливо не обмежуватися лише числовими метриками, а проаналізувати, які ознаки найбільше впливають на прогноз.

Інтерпретація потрібна для того, щоб модель могла бути зрозумілою кредитному аналітику і не виглядала як повністю непрозорий алгоритм. У роботі для цього використано важливість ознак XGBoost та permutation importance

Важливість ознак за моделлю XGBoost наведено на рис. 3.17.

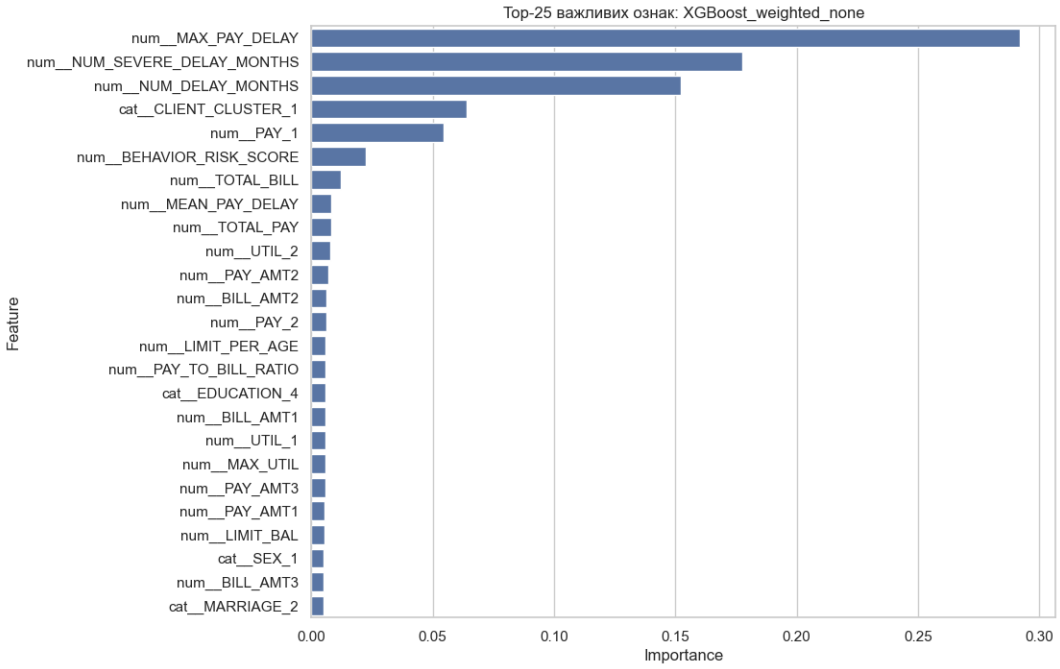


Рис. 3.17 Важливість ознак за моделлю XGBoost

Результати permutation importance для фінальної моделі наведено на рис. 3.18.

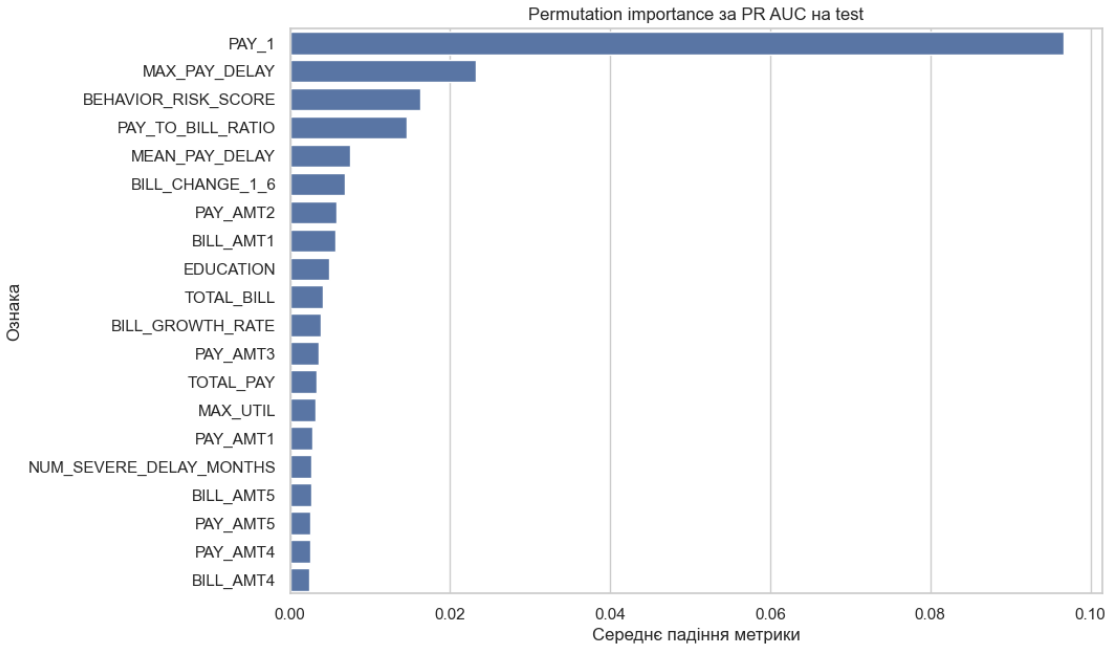


Рис. 3.18 Permutation importance для фінальної моделі

За результатами інтерпретації найважливішими виявилися поведінкові ознаки, пов’язані з історією прострочень: MAX_PAY_DELAY, NUM_SEVERE_DELAY_MONTHS, NUM_DELAY_MONTHS, PAY_1 та BEHAVIOR_RISK_SCORE. Це підтверджує логіку попереднього EDA: саме минула платіжна дисципліна є головним сигналом кредитного ризику. Фінансові ознаки, такі як TOTAL_BILL, TOTAL_PAY, PAY_TO_BILL_RATIO та UTIL, також мають значення, але переважно як додаткові фактори, що уточнюють загальний профіль клієнта.

Модель доцільно розглядати як аналітичний модуль у складі ширшої банківської системи підтримки прийняття рішень, модель якої зображена на рисунку 3.19.

Таке рішення є коректним зважаючи що в реальних умовах кредитне рішення формується не лише на основі результату машинного навчання, а також з урахуванням кредитної політики банку, нормативних обмежень, перевірки клієнта, метаданих банківського додатку клієнта та транзакцій [20].

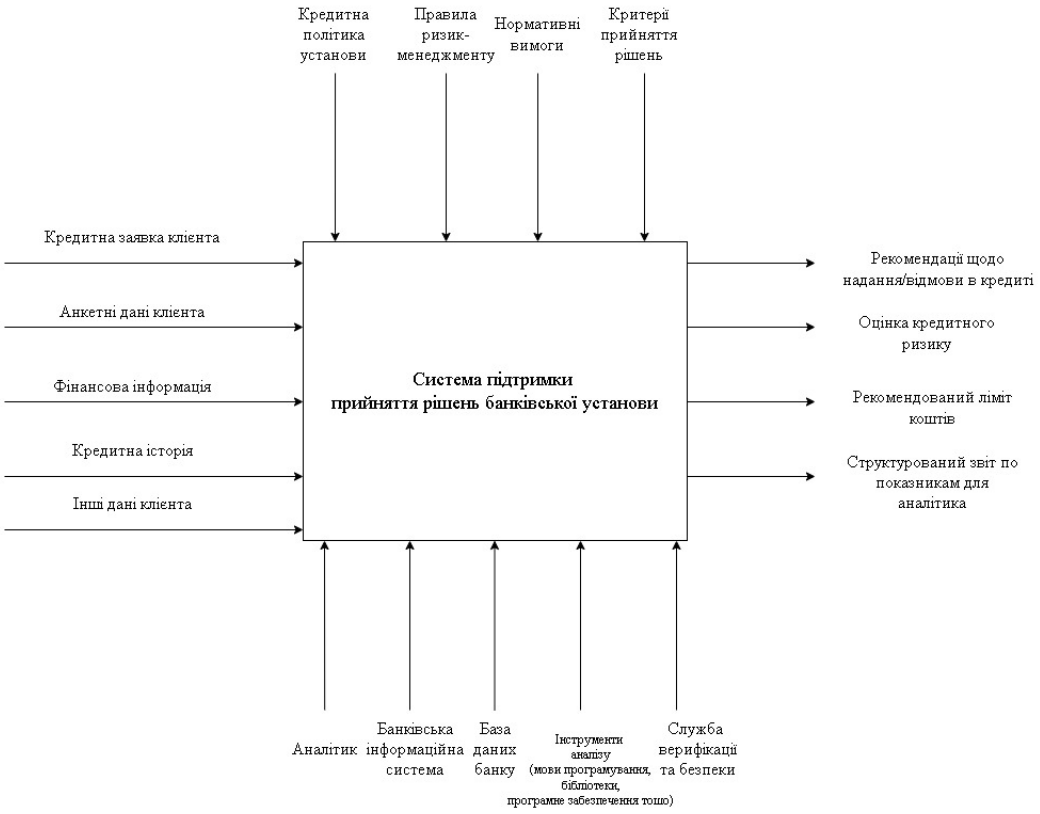


Рис. 3.19 IDEF0 СППР банку

Повна система подається як єдиний процес — підтримки прийняття кредитного рішення в банку. Входами такої системи є кредитна заявка клієнта, анкетні дані, фінансова інформація, кредитна історія та інші доступні відомості про позичальника. До керуючих впливів належать кредитна політика банку, правила ризик-менеджменту, нормативні вимоги, критерії оцінювання платоспроможності. Механізмами виконання процесу є банківська інформаційна система, база даних клієнтів, кредитний аналітик, служби перевірки та модуль машинного навчання, реалізований у межах даного дослідження. Виходами системи є оцінка кредитного ризику, рекомендація щодо кредитного рішення, можливе рішення про схвалення або відмову, аналітичний звіт для відповідального працівника банку.

Для деталізації роботи повної системи доцільно використати декомпозицію A0. У такій декомпозиції процес підтримки кредитного рішення можна поділити на кілька послідовних функціональних блоків: прийом та реєстрація кредитної заявки, збір і перевірка даних клієнта, оцінювання кредитного ризику, формування рекомендації та фіксація остаточного рішення.

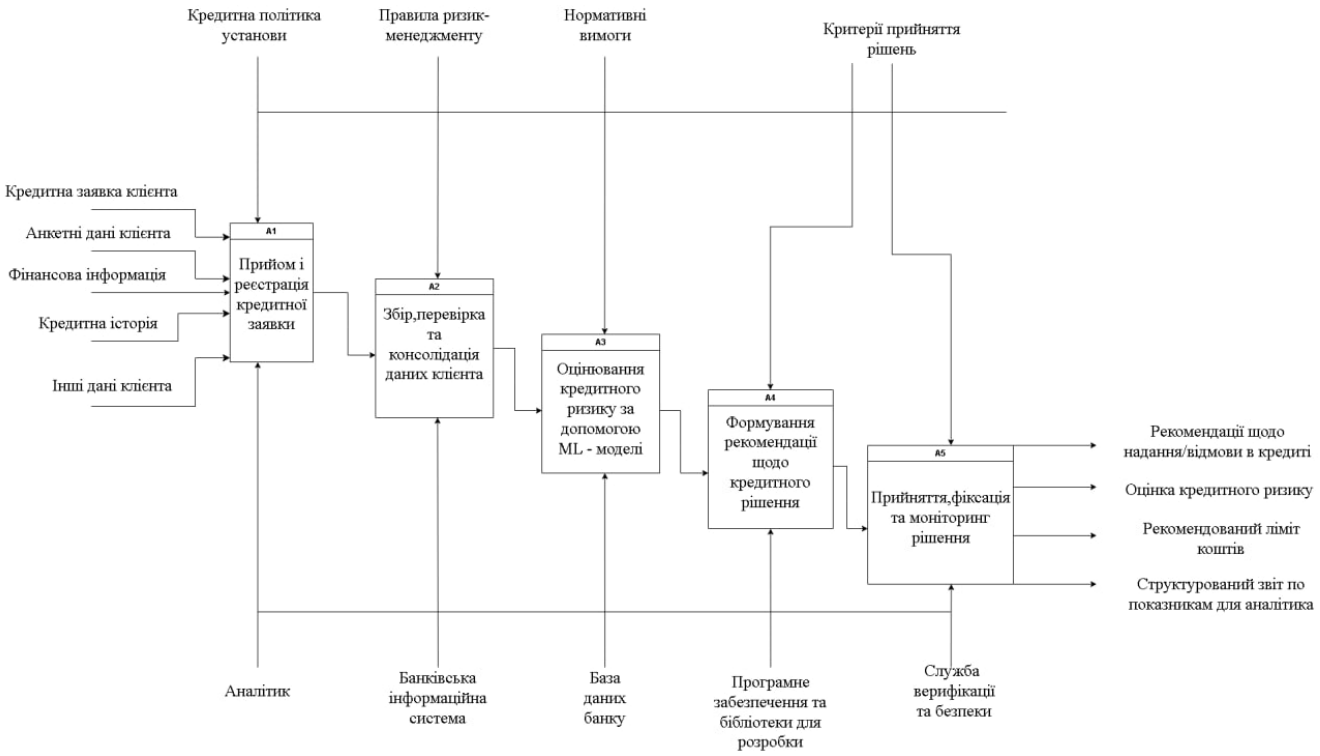


Рис. 3.20 Декомпозиція IDEF0 СППР банку

Першим етапом повної системи є прийом і реєстрація кредитної заявки. На цьому етапі клієнт подає запит на отримання кредиту, а банківська система фіксує основні параметри заявки: тип кредитного продукту, бажану суму, строк кредитування, персональні дані позичальника та іншу первинну інформацію. Результатом цього етапу є зареєстрована кредитна заявка, яка передається на подальшу обробку.

Другим етапом є збір, перевірка та консолідація даних. Його призначення полягає у тому, щоб сформувати повний інформаційний профіль клієнта. На цьому рівні можуть використовуватися внутрішні банківські дані, інформація з кредитної історії, дані про попередні платежі, доходи клієнта, поточне боргове навантаження та результати перевірки достовірності поданої інформації. Саме якість цього етапу значною мірою визначає надійність подальшого оцінювання ризику, оскільки модель машинного навчання не може сформувати коректний прогноз на основі неповних або помилкових даних.

Третім етапом є оцінювання кредитного ризику клієнта. У межах цієї функції використовується розроблений у роботі аналітичний модуль. Його завданням є перетворення підготовлених даних клієнта на числову оцінку ризику дефолту. У реалізованому дослідженні для цього було використано датасет default-of-credit-card-clients, проведено розвідувальний аналіз даних, виконано підготовку ознак, створено синтетичні показники платіжної поведінки, побудовано декілька моделей машинного навчання та обрано найбільш ефективну модель за сукупністю метрик.

Четвертий етап повної системи — формування рекомендації щодо кредитного рішення. На цьому етапі результат моделі не використовується механічно як остаточний висновок, а інтерпретується разом із правилами банку. Наприклад, якщо модель визначає високу ймовірність дефолту, система може рекомендувати відмову, зменшення кредитного ліміту, підвищення вимог до забезпечення або передачу заявки на додатковий ручний розгляд. Якщо ризик є помірним, система може запропонувати альтернативні умови кредитування. Якщо ризик є низьким, заявка може бути рекомендована до схвалення.

П’ятим етапом є прийняття, фіксація та подальший моніторинг рішення. Остаточне рішення може прийматися автоматизовано або за участю кредитного аналітика залежно від політики банку, суми кредиту та рівня ризику. Після прийняття рішення результат зберігається в інформаційній системі, що дає змогу здійснювати аудит, аналізувати якість попередніх рішень і надалі оновлювати модель на основі нових історичних даних.

Оскільки розроблена у роботі модель є частиною блоку оцінювання кредитного ризику, цей блок доцільно деталізувати окремою декомпозицією для розуміння процесу створення та навчання моделі в реальному середовищі. Така декомпозиція дозволяє показати, як саме результати дослідження можуть бути інтегровані у повну банківську систему.

Декомпозицію процесу створення ML-модуля наведено на рис. 3.21.

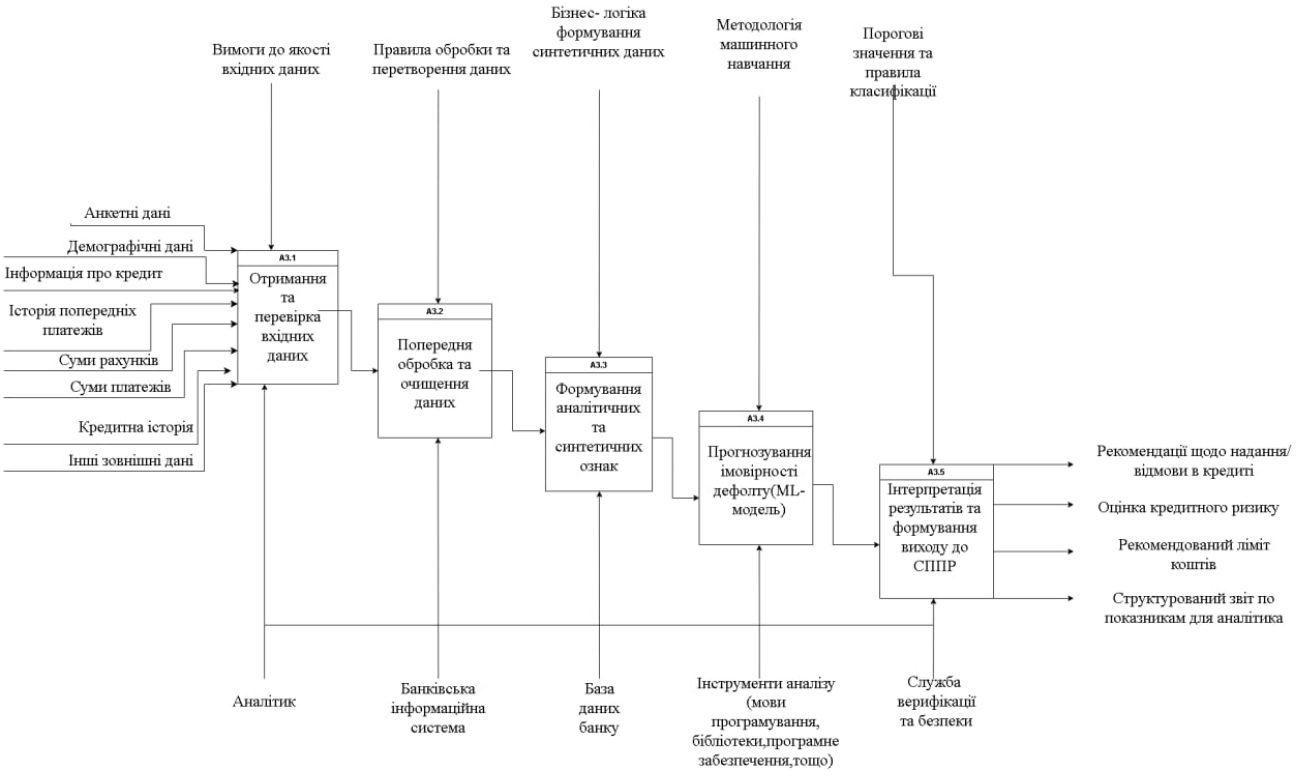


Рис. 3.21 Декомпозиція створення модуля ML

У межах декомпозиції блоку оцінювання кредитного ризику можна виділити п’ять основних підпроцесів. Перший підпроцес – отримання та перевірка вхідних

даних. Його мета полягає у тому, щоб переконатися, що для клієнта наявні всі необхідні змінні для подальшого прогнозування. У контексті виконаного дослідження такими змінними є сума наданого кредиту, демографічні характеристики, історія прострочень, суми рахунків, суми попередніх платежів та цільова ознака для навчання моделі.

Другий підпроцес – попередня обробка та очищення даних. На цьому етапі дані приводяться до форми, придатної для моделювання. Виконується перевірка типів змінних, аналіз пропущених значень, обробка категоріальних ознак, стандартизація числових змінних та підготовка даних до використання машинного навчання. Цей етап є важливим, оскільки алгоритми машинного навчання чутливі до структури вхідних даних, а некоректна підготовка може призвести до спотворення результатів.

Третій підпроцес – формування аналітичних та синтетичних ознак. У роботі було показано, що базові змінні не завжди повністю відображають поведінку клієнта, тому на їх основі були створені додаткові показники. До них належать коефіцієнти використання кредиту, загальна сума рахунків, загальна сума платежів, співвідношення платежів до рахунків, кількість місяців із простроченнями, максимальна та середня затримка платежів, а також інтегральний показник поведінкового ризику BEHAVIOR_RISK_SCORE. Такі ознаки дозволяють моделі краще враховувати не лише окремі значення, а й загальну фінансову поведінку клієнта.

Четвертий підпроцес – прогнозування ймовірності дефолту. Саме тут використовується навчена модель машинного навчання. У роботі було протестовано декілька алгоритмів, зокрема логістичну регресію, випадковий ліс та XGBoost. За результатами порівняння найбільш доцільною моделлю було визначено XGBoost_weighted_none, оскільки вона забезпечила найкращий баланс між здатністю виявляти дефолтних клієнтів і загальною якістю ранжування ризику. Особливістю цієї моделі є використання параметра scale_pos_weight, який враховує дисбаланс класів і підсилює вагу меншості, тобто клієнтів із дефолтом [24; 25; 26; 27;28].

П’ятий підпроцес – інтерпретація результату та формування рекомендації. Після отримання ймовірності дефолту система порівнює її з обраним порогом класифікації. Якщо ймовірність перевищує threshold, клієнт відноситься до ризикового класу. Якщо не перевищує – до менш ризикового. У роботі було показано, що вибір порогу суттєво впливає на співвідношення між Precision та Recall. Зменшення порогу дозволяє виявити більше потенційно дефолтних клієнтів, але водночас збільшує кількість помилкових спрацювань. Тому у практичній системі threshold має визначатися не лише статистично, а й відповідно до бізнес-логіки банку та вартості різних типів помилок.

Важливо підкреслити, що модель не замінює кредитного аналітика повністю. Її роль полягає у тому, щоб надати додаткову кількісну оцінку ризику, виявити приховані закономірності в історичних даних і зменшити суб’єктивність початкового аналізу заявки. Остаточне рішення може залишатися за банківським працівником або за автоматизованою системою, але в обох випадках результат моделі виступає важливим джерелом аналітичної інформації [4; 21].

Використання діаграм в цьому підрозділі дозволяє показати не тільки алгоритмічну частину дослідження, а й місце розробленої моделі в реальному процесі банківського кредитування. Таким чином, запропонована модель може бути використана як інтелектуальний модуль у складі банківської системи підтримки прийняття кредитних рішень. Її практична цінність полягає не лише у формуванні прогнозу дефолту клієнта, а й у можливості ранжування клієнтів за рівнем ризику, підтримці кредитного аналітика, підвищенні обґрунтованості рішень і створенні основи для подальшого розвитку автоматизованих скорингових систем.

ВИСНОВКИ

У кваліфікаційній роботі було виконано комплексне дослідження та практичну реалізацію алгоритмів машинного навчання для автоматизації аналізу кредитного ризику у фінансовому секторі. Дослідження розпочалося з аналізу еволюції кредитного скорингу від експертних евристичних методів, таких як правило 5Cs, до сучасних багаторівневих систем підтримки прийняття рішень. Показано, що традиційний ручний підхід страждає від впливу людського фактору, когнітивних упереджень та шуму, що призводить до проблем неякісних історичних даних, таких як шумливі мітки.

В рамках формалізації задачі кредитного скорингу як задачі бінарної класифікації, метою якої є прогнозування належності позичальника до одного з класів (надійного клієнта або особи, схильної до дефолту), було визначено основні чинники, що впливають на предиктивну здатність моделей. Встановлено, що найвпливовішим індикатором є платіжна дисципліна клієнта, тоді як інші показники не формують такий вплив на кінцевий результат.

Під час реалізації практичної частини роботи було розроблено програмне рішення для аналізу кредитного ризику з використанням методів машинного навчання. Реалізований підхід охоплює основні етапи роботи з даними та їх аналізом за допомогою методів машинного навчання, спираючись на метрики ефективності моделей.

У процесі розвідувального аналізу даних було встановлено, що найсильнішим індикатором майбутнього дефолту є історична платіжна дисципліна клієнта, тоді як демографічні показники формують значно слабший сигнал. З метою підвищення предиктивної здатності алгоритмів було проведено масштабну інженерію ознак, що дозволило розширити початковий простір із двадцяти чотирьох до сорока п'яти змінних. Були створені нові економічно обґрунтовані показники, такі як загальне боргове навантаження, платіжна активність, співвідношення платежів до рахунків, а також комплексний індекс поведінкового ризику. Під час моделювання виникла проблема екстремального дисбалансу класів,

де частка дефолтів становила близько двадцяти двох відсотків.

Серед низки реалізованих моделей, що включали базові контрольні моделі, логістичну регресію та випадковий ліс, найкращі результати продемонструвала ансамблева модель градієнтного бустингу з використанням зважування класів. Ця архітектура дозволила уникнути ігнорування міноритарного класу та забезпечила оптимальний баланс виявлення шахраїв і неплатоспроможних осіб.

Під час аналізу якості розроблених моделей оцінювання проводилось за основними метриками: ROC-AUC, Precision,Recall та F1-score. Особливу увагу було приділено здатності моделі знаходити саме проблемних позичальників.Доволі важливим етапом роботи стала бізнес-оптимізація результатів шляхом зміщення порогу класифікації. Зниження порогового значення до рівня 0,355 дозволило підвищити здатність моделі виявляти дефолтних клієнтів майже до вісімдесяти відсотків, що мінімізує найдорожчі для банківської установи помилки.

Подальший розвиток розробленого підходу передбачає його безпосереднє впровадження в інформаційну інфраструктуру фінансової установи для автоматизації процесів кредитування. Інтеграція створеної моделі дозволить побудувати комплексну систему підтримки прийняття рішень, здатну швидко, об'єктивно та без впливу людського фактору оцінювати ризики за новими заявками. На практиці це дасть банку можливість не лише майже миттєво схвалювати або відхиляти стандартні кредити, а й гнучко управляти кредитними лімітами та пропонувати індивідуальні умови для кожного позичальника. У підсумку таке впровадження суттєво оптимізує роботу аналітиків, знизить відсоток проблемних кредитів та підвищить загальну економічну ефективність діяльності установи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Про затвердження Положення про визначення банками України розміру кредитного ризику за активними банківськими операціями : постанова Правління Національного банку України від 30.06.2016 № 351. URL: https://bank.gov.ua/ua/legislation/Resolution_30062016_351 (дата звернення: 12.03.2026).
2. Про затвердження Положення про організацію системи управління ризиками в банках України та банківських групах : постанова Правління Національного банку України від 11.06.2018 № 64. URL: https://zakon.rada.gov.ua/laws/show/v0064500-18 (дата звернення: 12.03.2026).
3. Про банки і банківську діяльність : Закон України від 07.12.2000 № 2121-III. URL: https://zakon.rada.gov.ua/laws/show/2121-14 (дата звернення: 12.03.2026).
4. Thomas L. C., Crook J., Edelman D. Credit Scoring and Its Applications. 2nd ed. Philadelphia : SIAM, 2017. 387 p.
5. Kahneman D., Sibony O., Sunstein C. R. Noise: A Flaw in Human Judgment. New York : Little, Brown Spark, 2021. 464 p.
6. Клебан Ю., Горошко Н. Ідентифікація дефолтних клієнтів банку методами машинного навчання на основі біннінгу показників. *Економічний аналіз*. 2021. Т. 31, № 1. С. 133–142. URL: https://www.econa.org.ua/index.php/econa/article/download/1881/6565656963 (дата звернення: 15.03.2026).
7. Дацко М., Друщак Н. Моделювання кредитного скорингу методами машинного навчання. *Вісник Львівського університету. Серія економічна*. 2022. Вип. 63. С. 56–66. URL: https://publications.lnu.edu.ua/bulletins/index.php/economics/article/view/11898/12245 (дата звернення: 15.03.2026).
8. Стовба В. О. Статистичні та оптимізаційні методи в кредитному скорингу. *Кібернетика та комп'ютерні технології*. 2022. № 3. С. 23–36. URL:

https://cctech.org.ua/images/docs/Articles/2022/paper_22_3_3.pdf (дата звернення: 16.03.2026).

9. Семененко Ю. С. Технології машинного навчання у кредитному скорингу. *Наукові перспективи*. 2024. № 4 (46). С. 781–794. URL: https://perspectives.pp.ua/index.php/np/article/download/11010/11069/11069 (дата звернення: 18.03.2026).
10. Chawla N. V., Bowyer K. W., Hall L. O., Kegelmeyer W. P. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*. 2002. Vol. 16. P. 321–357.
11. He H., Bai Y., Garcia E. A., Li S. ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning. *Proceedings of the IEEE International Joint Conference on Neural Networks*. 2008. P. 1322–1328.
12. Pedregosa F. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011. Vol. 12. P. 2825–2830.
13. Default-of-credit-card-clients Dataset. *OpenML*. URL: https://www.openml.org/d/42477 (дата звернення: 07.04.2026).
14. Five Cs of Credit: What They Are, How They're Used, and Which Is Most Important. *Investopedia*. URL: https://www.investopedia.com/terms/f/five-c-credit.asp (дата звернення: 18.03.2026).
15. Kiviat B. Credit Scoring in the United States. *Economic Sociology*. 2019. Vol. 21, no. 1. P. 33–40. URL: https://www.econstor.eu/bitstream/10419/223110/1/Econsoc-NL-21-1-05.pdf (дата звернення: 20.03.2026)
16. FICO History. *FICO*. URL: https://www.fico.com/en/history (дата звернення: 20.03.2026).
17. Demir C. Machine Learning for Credit Risk Scoring: From Traditional Statistics to Gradient Boosting. *Medium*. 2026. URL: https://medium.com/@candemir13/machine-learning-for-credit-risk-scoring-from-traditional-statistics-to-gradient-boosting-95f056a1cc36 (дата звернення: 24.03.2026).

18. Equal Credit Opportunity Act (Regulation B). *Consumer Financial Protection Bureau*. URL: https://www.consumerfinance.gov/rules-policy/regulations/1002/ (дата звернення: 24.03.2026).
19. Understanding FICO: Credit Scores, Their Impact, and How They're Calculated. *Investopedia*. URL: https://www.investopedia.com/terms/f/fico-fair-isaac.asp (дата звернення: 19.03.2026).
20. Литвинець Б.В. Інтеграція та обробка альтернативних даних для оцінки кредитного ризику клієнтів з короткою кредитною історією. *Сучасний стан та перспективи розвитку IoT* : збірник тез VII Всеукраїнської науково-технічної конференції (м. Київ, 20 квітня 2026 р.). Київ : ДУІКТ, 2026. С. 284–285.
21. Akarlar A. Credit Default Risk Prediction: An Explainable AI Approach with XGBoost and SHAP. *Medium*. 2025. URL: https://medium.com/data-science-collective/credit-default-risk-prediction-an-explainable-ai-approach-with-xgboost-and-shap-aytug-akarlar-46355f7abc4b (дата звернення: 02.04.2026).
22. IEC 27001:2022. Information security, cybersecurity and privacy protection — Information security management systems — Requirements. Geneva : ISO, 2022 ; ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. Geneva : ISO, 2011 ; ISO/IEC 38500:2024. Information technology — Governance of IT for the organization. Geneva : ISO, 2024.
23. Литвинець Б.В. Взаємозв’язок між якістю даних та стійкістю скорингових моделей. Автоматизація конвеєрів підготовки даних. *Сучасний стан та перспективи розвитку IoT* : збірник тез VII Всеукраїнської науково-технічної конференції (м. Київ, 20 квітня 2026 р.). Київ : ДУІКТ, 2026. С. 285–287.
24. Hosmer D. W., Lemeshow S., Sturdivant R. X. Applied Logistic Regression. 3rd ed. Hoboken : John Wiley & Sons, 2013. 528 p.
25. Breiman L. Random Forests. *Machine Learning*. 2001. Vol. 45. P. 5–32.
26. Friedman J. H. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*. 2001. Vol. 29, № 5. P. 1189–1232.

27. Credit Risk Modeling with Machine Learning: A Data-Driven Comparison of Models and Pipeline. *Medium*. 2025. URL: https://medium.com/learning-data/credit-risk-modeling-with-machine-learning-a-data-driven-comparison-of-models-and-pipeline-a7ce253c8da7 (дата звернення: 26.03.2026).

28. Литвинець Б. В. Мінімізація впливу шахрайства та протидія зашумленню даних у системах кредитного скорингу. *Інформаційно-комп'ютерні технології* : тези доповідей XVI Міжнародної науково-технічної конференції (м. Житомир, 02-03 квітня 2026 р.). Житомир, 2026. С. 280–282

ДОДАТОК А
ПРОГРАМНИЙ КОД

```
import warnings
warnings.filterwarnings('ignore')

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split, StratifiedKFold, cross_validate, RandomizedSearchCV
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.base import clone
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
    roc_auc_score, average_precision_score, balanced_accuracy_score,
    confusion_matrix, classification_report, RocCurveDisplay, PrecisionRecallDisplay,
    ConfusionMatrixDisplay, roc_curve, precision_recall_curve
)
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.inspection import permutation_importance
from sklearn.dummy import DummyClassifier

from imblearn.pipeline import Pipeline as ImbPipeline
from imblearn.over_sampling import SMOTE, ADASYN

RANDOM_STATE = 42
pd.set_option('display.max_columns', 100)
sns.set_theme(style='whitegrid')
#%% md
## Завантаження датасету з OpenML
#%%
# Цей блок завантажує датасет з OpenML і створює початкові таблиці X_raw та y_raw.
# X_raw містить ознаки клієнтів, а y_raw містить цільову змінну - факт дефолту.

data = fetch_openml(data_id=42477, as_frame=True, parser='auto')
X_raw = data.data.copy()
y_raw = data.target.copy()

print(data.DESCR[:1200])
print('Форма X:', X_raw.shape)
print('Форма y:', y_raw.shape)
X_raw.head()
#%%
# У різних версіях датасету назви колонок відрізняються.
# Наприклад, офіційний опис використовує X1-X23, а в OpenML часто зустрічаються назви LIMIT_BAL, PAY_0,
PAY_2
# Перейменовуємо
rename_map = {
    'x1': 'LIMIT_BAL', 'limit_bal': 'LIMIT_BAL',
    'x2': 'SEX', 'sex': 'SEX',
```

```
    'x3': 'EDUCATION', 'education': 'EDUCATION',
    'x4': 'MARRIAGE', 'marriage': 'MARRIAGE',
    'x5': 'AGE', 'age': 'AGE',

    # Історія погашення
    'x6': 'PAY_1', 'pay_0': 'PAY_1', 'pay_1': 'PAY_1',
    'x7': 'PAY_2', 'pay_2': 'PAY_2',
    'x8': 'PAY_3', 'pay_3': 'PAY_3',
    'x9': 'PAY_4', 'pay_4': 'PAY_4',
    'x10': 'PAY_5', 'pay_5': 'PAY_5',
    'x11': 'PAY_6', 'pay_6': 'PAY_6',

    'x12': 'BILL_AMT1', 'bill_amt1': 'BILL_AMT1',
    'x13': 'BILL_AMT2', 'bill_amt2': 'BILL_AMT2',
    'x14': 'BILL_AMT3', 'bill_amt3': 'BILL_AMT3',
    'x15': 'BILL_AMT4', 'bill_amt4': 'BILL_AMT4',
    'x16': 'BILL_AMT5', 'bill_amt5': 'BILL_AMT5',
    'x17': 'BILL_AMT6', 'bill_amt6': 'BILL_AMT6',

    'x18': 'PAY_AMT1', 'pay_amt1': 'PAY_AMT1',
    'x19': 'PAY_AMT2', 'pay_amt2': 'PAY_AMT2',
    'x20': 'PAY_AMT3', 'pay_amt3': 'PAY_AMT3',
    'x21': 'PAY_AMT4', 'pay_amt4': 'PAY_AMT4',
    'x22': 'PAY_AMT5', 'pay_amt5': 'PAY_AMT5',
    'x23': 'PAY_AMT6', 'pay_amt6': 'PAY_AMT6',

    'default payment next month': 'default',
    'default': 'default'
}

def normalize_col_name(col):
    key = str(col).strip().lower().replace(' ', '_')
    key = key.replace('default_payment_next_month', 'default payment next month')
    return rename_map.get(key, str(col).strip())

X = X_raw.copy()
X.columns = [normalize_col_name(c) for c in X.columns]

y = pd.Series(y_raw, name='default').astype(int)

# Прибираємо службову колонку id
for id_col in ['ID', 'id']:
    if id_col in X.columns:
        X = X.drop(columns=[id_col])

required_cols = [
    'LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE',
    'PAY_1', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6',
    'BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6',
    'PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6'
]
missing_cols = [c for c in required_cols if c not in X.columns]
if missing_cols:
    raise ValueError(f'Не знайдено очікувані колонки після перейменування: {missing_cols}. Поточні колонки:
{list(X.columns)}')

df = X[required_cols].copy()
df['default'] = y
print(df.shape)
df.head()
#%% md
## Опис ознак датасету
#%%
```

```
# Створення словника ознак.
# Коротке пояснення ознак в датасетію.
feature_dictionary = pd.DataFrame({
   'feature': [
      'LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE',
      'PAY_1', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6',
      'BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6',
      'PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6',
      'default'
   ],
   'group': [
      'given_credit_amount', 'demographic', 'demographic', 'demographic', 'demographic',
      'repayment_status', 'repayment_status', 'repayment_status', 'repayment_status', 'repayment_status', 'repayment_status',
      'bill_amount', 'bill_amount', 'bill_amount', 'bill_amount', 'bill_amount', 'bill_amount',
      'previous_payment', 'previous_payment', 'previous_payment', 'previous_payment', 'previous_payment',
'previous_payment',
      'target'
   ],
   'description_uk': [
      'Сума наданого кредиту в  доларах.',
      'Стать: 1 - чоловік, 2 - жінка.',
      'Освіта: 1 - graduate school, 2 - university, 3 - high school, 4 - others; нестандартні категорії об’єднуються в
other.',
      'Сімейний стан: 1 - married, 2 - single, 3 - others; нестандартні категорії об’єднуються в other.',
      'Вік клієнта у роках.',
      'Статус погашення у вересні ; в оригінальних назвах OpenML ця змінна часто позначена як PAY_0.',
      'Статус погашення у серпні ',
      'Статус погашення у липні ',
      'Статус погашення у червні',
      'Статус погашення у травні',
      'Статус погашення у квітні',
      'Сума рахунку у вересні',
      'Сума рахунку у серпні',
      'Сума рахунку у липні.',
      'Сума рахунку у червні',
      'Сума рахунку у травні',
      'Сума рахунку у квітні',
      'Сума платежу у вересні',
      'Сума платежу у серпні',
      'Сума платежу у липні',
      'Сума платежу у червні',
      'Сума платежу у травні',
      'Сума платежу у квітні',
      'Цільова змінна: 1 - дефолт, 0 - дефолту немає.'
   ]
})

feature_dictionary
#%% md
## EDA: структура, пропуски, дублікати, баланс класів
#%%
# Групуємо ознаки за змістом
# Зручніше EDA і побудова графіків

feature_groups = {
   'demographic': ['SEX', 'EDUCATION', 'MARRIAGE', 'AGE'],
   'payment_status': ['PAY_1', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6'],
   'bill_amounts': [f'BILL_AMT{i}' for i in range(1, 7)],
   'payment_amounts': [f'PAY_AMT{i}' for i in range(1, 7)],
}

# Окремі змінні для зручності
payment_status = feature_groups['payment_status']
```

```
bill_cols = feature_groups['bill_amounts']
pay_amt_cols = feature_groups['payment_amounts']

feature_groups
#%%
# Базова перевірка даних
# типи колонок, описова статистику, пропуски, дублікати та баланс цільового класу

display(df.info())
display(df.describe().T)

missing = df.isna().sum().sort_values(ascending=False)
print('Пропуски:')
display(missing[missing > 0])
print('Кількість дублікатів:', df.duplicated().sum())

class_counts = df['default'].value_counts().sort_index()
class_share = df['default'].value_counts(normalize=True).sort_index()
display(pd.DataFrame({'count': class_counts, 'share': class_share}))
#%% md
### Аналіз описової статистики датасету
#%%
#будуємо графік розподілу цільової змінної.
#дивимось чи є дисбаланс в класах чи ні

plt.figure(figsize=(6,4))
sns.countplot(data=df, x='default')
plt.title('Розподіл цільової змінної default')
plt.xlabel('0 — немає дефолту, 1 — дефолт')
plt.ylabel('Кількість клієнтів')
plt.show()

print('Частка дефолтів: {:.2%}'.format(df['default'].mean()))
print('Imbalance ratio, non-default / default: {:.2f}'.format((df['default']==0).sum() / (df['default']==1).sum()))
#%% md
## Аналіз категоріальних ознак та очищення некоректних категорій
#%%
# перевіряємо категоріальні ознаки.
# для того щоб побачити нестандартні значення

cat_cols_base = ['SEX', 'EDUCATION', 'MARRIAGE']
for col in cat_cols_base:
    print(f'\n{col}:')
    display(df[col].value_counts().sort_index())
#%%
df_clean = df.copy()

# EDUCATION: 1 graduate school, 2 university, 3 high school, 4 others; 0,5,6 об'єднаємо всі в 4
df_clean['EDUCATION'] = df_clean['EDUCATION'].replace({0: 4, 5: 4, 6: 4}).astype(int)

# MARRIAGE: 1 married, 2 single, 3 others; 0-не вказано змінимо на 3
df_clean['MARRIAGE'] = df_clean['MARRIAGE'].replace({0: 3}).astype(int)

# SEX, EDUCATION, MARRIAGE - категоріальні змінні.
for col in cat_cols_base:
    df_clean[col] = df_clean[col].astype('category')

for col in cat_cols_base:
    print(f'\n{col} після очищення:')
    display(df_clean[col].value_counts().sort_index())
#%% md
## Візуальний EDA
#%%
```

```
fig, axes = plt.subplots(1, 2, figsize=(15, 5))

sns.histplot(
    data=df_clean,
    x='LIMIT_BAL',
    hue='default',
    bins=40,
    kde=True,
    stat='density',
    common_norm=False,
    ax=axes[0]
)
axes[0].set_title('Розподіл суми наданого кредиту за класами default')
axes[0].set_xlabel('LIMIT_BAL')
axes[0].set_ylabel('Щільність')

sns.histplot(
    data=df_clean,
    x='AGE',
    hue='default',
    bins=35,
    kde=True,
    stat='density',
    common_norm=False,
    ax=axes[1]
)
axes[1].set_title('Розподіл віку клієнтів за класами default')
axes[1].set_xlabel('AGE')
axes[1].set_ylabel('Щільність')

plt.tight_layout()
plt.show()
#%%
#Частка дефолтів по категоріальних групах
# Щоб графіки були зрозумілими, числові коди категорій замінив на назви


df_eda_labels = df_clean.copy()

df_eda_labels['SEX_LABEL'] = df_eda_labels['SEX'].map({
    1: 'Чоловік',
    2: 'Жінка'
})

df_eda_labels['EDUCATION_LABEL'] = df_eda_labels['EDUCATION'].map({
    1: 'Магістратура',
    2: 'Університет',
    3: 'Старша школа',
    4: 'Інше'
})

df_eda_labels['MARRIAGE_LABEL'] = df_eda_labels['MARRIAGE'].map({
    1: 'Одружений/одружена',
    2: 'Неодружений/неодружена',
    3: 'Інше'
})

categorical_eda_cols = {
    'SEX_LABEL': 'статтю',
    'EDUCATION_LABEL': 'рівнем освіти',
    'MARRIAGE_LABEL': 'сімейним станом'
}
```

```
fig, axes = plt.subplots(1, 3, figsize=(18, 4))

for ax, (col, title) in zip(axes, categorical_eda_cols.items()):
    default_rate = (
        df_eda_labels
        .groupby(col, observed=False)['default']
        .mean()
        .reset_index()
        .sort_values('default', ascending=False)
    )

    sns.barplot(
        data=default_rate,
        x=col,
        y='default',
        ax=ax
    )

    # Звуження вісі Y, так краще видно різницю між групами.
    # тільки візуал
    y_min = max(0, default_rate['default'].min() - 0.03)
    y_max = min(1, default_rate['default'].max() + 0.03)
    ax.set_ylim(y_min, y_max)

    # Підписи відсотків на стовпцях
    for container in ax.containers:
        ax.bar_label(
            container,
            labels=[f'{v * 100:.1f}%' for v in default_rate['default']],
            padding=3,
            fontsize=9
        )

    ax.set_title(f'Частка дефолтів за {title}')
    ax.set_xlabel('')
    ax.set_ylabel('Частка default=1')
    ax.tick_params(axis='x', rotation=20)

plt.tight_layout()
plt.show()
#%%

# частка дефолтів залежно від історії платежів
# PAY_1, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6 — це статуси платежів
# за останні 6 місяців спостереження.


pay_month_labels = {
    'PAY_1': 'Вересень',
    'PAY_2': 'Серпень',
    'PAY_3': 'Липень',
    'PAY_4': 'Червень',
    'PAY_5': 'Травень',
    'PAY_6': 'Квітень'
}

pay_status_cols = list(pay_month_labels.keys())

fig, axes = plt.subplots(2, 3, figsize=(18, 8))

for ax, col in zip(axes.ravel(), pay_status_cols):
    default_rate = (
        df_clean
```

```
        .groupby(col, observed=False)['default']
        .mean()
        .reset_index()
        .sort_values(col)
    )

    sns.barplot(data=default_rate, x=col, y='default', ax=ax)

    ax.set_title(f'{col}: {pay_month_labels[col]}')
    ax.set_xlabel('Статус погашення')
    ax.set_ylabel('Частка default=1')
    ax.set_ylim(0, max(0.7, default_rate['default'].max() * 1.1))

fig.suptitle(
    'Частка дефолтів залежно від статусу погашення за місяцями',
    fontsize=14,
    y=1.02
)

plt.tight_layout()
plt.show()
#%%
# кореляції фінансових і поведінкових ознак з default
# допомагає побачити які кореляції між змінними і класом дефолту
#
# будуємо таблицю кореляцій кожної ознаки з default.
# потім відбираємо 12 ознак із найбільшою абсолютною кореляцією
# потім heatmap

bill_cols = feature_groups['bill_amounts']
pay_amt_cols = feature_groups['payment_amounts']

corr_cols = (
    ['default', 'LIMIT_BAL', 'AGE']
    + payment_status
    + bill_cols
    + pay_amt_cols
)

corr_matrix = df_clean[corr_cols].astype(float).corr()

# Таблиця кореляцій з цільовою змінною
corr_with_default = (
    corr_matrix['default']
    .drop('default')
    .sort_values(ascending=False)
    .to_frame('corr_with_default')
)

display(corr_with_default)

# Відбір top-12 ознак за силою зв'язку з default
top_corr_cols = (
    corr_matrix['default']
    .drop('default')
    .abs()
    .sort_values(ascending=False)
    .head(12)
    .index
    .tolist()
)

focused_corr_cols = ['default'] + top_corr_cols
```

```
plt.figure(figsize=(12, 8))
sns.heatmap(
    corr_matrix.loc[focused_corr_cols, focused_corr_cols],
    cmap='coolwarm',
    center=0,
    annot=True,
    fmt='.2f'
)

plt.title('Кореляційна матриця top-12 ознак за зв’язком із default')
plt.tight_layout()
plt.show()
#%% md
##  Створення синтетичних ознак
#%%
# функція створює синтетичні ознаки на основі фінансової логіки.
# загальний борг, загальні платежі, кількість прострочень і коефіцієнти використання кредиту.

def add_domain_features(input_df: pd.DataFrame) -> pd.DataFrame:
    df_fe = input_df.copy()
    eps = 1e-6

    pay_cols = ['PAY_1', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6']
    bill_cols = [f'BILL_AMT{i}' for i in range(1, 7)]
    pay_amt_cols = [f'PAY_AMT{i}' for i in range(1, 7)]

    # Боргове навантаження відносно суми наданого кредиту.
    for i, bill_col in enumerate(bill_cols, start=1):
        df_fe[f'UTIL_{i}'] = df_fe[bill_col] / (df_fe['LIMIT_BAL'] + eps)

    util_cols = [f'UTIL_{i}' for i in range(1, 7)]
    df_fe['AVG_UTIL'] = df_fe[util_cols].mean(axis=1)
    df_fe['MAX_UTIL'] = df_fe[util_cols].max(axis=1)
    df_fe['STD_UTIL'] = df_fe[util_cols].std(axis=1)

    # Загальний борг та загальні платежі.
    df_fe['TOTAL_BILL'] = df_fe[bill_cols].sum(axis=1)
    df_fe['TOTAL_PAY'] = df_fe[pay_amt_cols].sum(axis=1)
    df_fe['PAY_TO_BILL_RATIO'] = df_fe['TOTAL_PAY'] / (df_fe['TOTAL_BILL'].abs() + eps)

    # Уникаємо надмірного впливу одиничних екстремальних значень,
    # які виникають через ділення на дуже малу суму рахунку.
    df_fe['PAY_TO_BILL_RATIO'] = df_fe['PAY_TO_BILL_RATIO'].clip(0, 10)

    # Платіжна дисципліна. Значення PAY_* > 0 означають прострочення.
    df_fe['MEAN_PAY_DELAY'] = df_fe[pay_cols].mean(axis=1)
    df_fe['MAX_PAY_DELAY'] = df_fe[pay_cols].max(axis=1)
    df_fe['NUM_DELAY_MONTHS'] = (df_fe[pay_cols] > 0).sum(axis=1)
    df_fe['NUM_SEVERE_DELAY_MONTHS'] = (df_fe[pay_cols] >= 2).sum(axis=1)

    # Динаміка заборгованості: позитивне значення може означати зростання боргу.
    df_fe['BILL_CHANGE_1_6'] = df_fe['BILL_AMT1'] - df_fe['BILL_AMT6']
    df_fe['BILL_GROWTH_RATE'] = (df_fe['BILL_AMT1'] - df_fe['BILL_AMT6']) / (df_fe['BILL_AMT6'].abs() + eps)
    df_fe['BILL_GROWTH_RATE'] = df_fe['BILL_GROWTH_RATE'].clip(-10, 10)

    # Відносна сума наданого кредиту за віком та логарифмована сума кредиту.
    df_fe['LIMIT_PER_AGE'] = df_fe['LIMIT_BAL'] / (df_fe['AGE'] + eps)
    df_fe['LOG_LIMIT_BAL'] = np.log1p(df_fe['LIMIT_BAL'])

    # Евристичний агрегований ризик-бал як пояснювальна синтетична ознака.
    # Цільова змінна default тут не використовується, тому leakage немає.
    df_fe['BEHAVIOR_RISK_SCORE'] = (
```

```
        df_fe['NUM_DELAY_MONTHS'] * 0.35 +
        df_fe['NUM_SEVERE_DELAY_MONTHS'] * 0.45 +
        df_fe['AVG_UTIL'].clip(-1, 3) * 0.20
    )

    return df_fe

df_fe = add_domain_features(df_clean)
print('Кількість колонок до FE:', df_clean.shape[1])
print('Кількість колонок після FE:', df_fe.shape[1])
df_fe.head()
#%% md
### Пояснення створення синтетичних ознак
#%%
# перевіряються нові синтетичні ознаки.
# описова статистика і зв'язок із default, щоб переконатися, що вони мають аналітичний сенс.

new_features = sorted(set(df_fe.columns) - set(df_clean.columns))
display(df_fe[new_features + ['default']].describe().T)

plt.figure(figsize=(10,6))
sns.heatmap(df_fe[new_features + ['default']].astype(float).corr()[['default']].sort_values('default', ascending=False),
annot=True, cmap='coolwarm', center=0)
plt.title('Зв’язок синтетичних ознак із default')
plt.show()
#%% md
## Розбиття train/test
#%%
# дані діляться на ознаки X та ціль y, а потім на train і test
# стратифікація зберігає однакову частку дефолтів у навчальній і тестовій вибірках

X_all = df_fe.drop(columns=['default'])
y_all = df_fe['default'].astype(int)

X_train, X_test, y_train, y_test = train_test_split(
    X_all, y_all, test_size=0.2, random_state=RANDOM_STATE, stratify=y_all
)

print('Train:', X_train.shape, 'Test:', X_test.shape)
print('Default rate train:', y_train.mean())
print('Default rate test:', y_test.mean())
#%% md
##  Обробка екстремальних значень через winsorization
#%%
#  функція winsorization обрізає надмірно низькі та високі значення.
# межі обчислюються тільки на train, а потім застосовуються і до train, і до test, щоб уникнути витоку даних
# Неперервні ознаки, для яких крайні значення можуть спотворювати моделі
# категоріальні ознаки, PAY_1-PAY_6 та цільову змінну не включаємо
bill_cols = [f'BILL_AMT{i}' for i in range(1, 7)]
pay_amt_cols = [f'PAY_AMT{i}' for i in range(1, 7)]
util_cols = [f'UTIL_{i}' for i in range(1, 7)]

winsor_columns = [
    'LIMIT_BAL',
    *bill_cols,
    *pay_amt_cols,
    *util_cols,
    'AVG_UTIL', 'MAX_UTIL', 'STD_UTIL',
    'TOTAL_BILL', 'TOTAL_PAY',
    'PAY_TO_BILL_RATIO',
    'BILL_CHANGE_1_6', 'BILL_GROWTH_RATE',
    'LIMIT_PER_AGE'
]
```

```
# залишаємо тільки ті колонки, які реально є в таблиці
winsor_columns = [col for col in winsor_columns if col in X_train.columns]


def summarize_extremes(X, columns, lower_q=0.01, upper_q=0.99):
    """Показує, наскільки широкими є діапазони значень до/після winsorization."""
    rows = []
    for col in columns:
        lower = X[col].quantile(lower_q)
        upper = X[col].quantile(upper_q)
        rows.append({
            'feature': col,
            'min': X[col].min(),
            'p01': lower,
            'median': X[col].median(),
            'p99': upper,
            'max': X[col].max(),
            'below_p01_count': int((X[col] < lower).sum()),
            'above_p99_count': int((X[col] > upper).sum())
        })
    return pd.DataFrame(rows)


print('Кандидати для winsorization:', len(winsor_columns))
print(winsor_columns)

extreme_summary_before = summarize_extremes(X_train, winsor_columns)
display(extreme_summary_before.sort_values('above_p99_count', ascending=False))


def winsorize_by_train_quantiles(X_train, X_test, columns, lower_q=0.01, upper_q=0.99):
    """
    Обрізає значення за квантилями train-вибірки.
    Межі розраховуються тільки на train, щоб уникнути data leakage.
    """
    X_train_w = X_train.copy()
    X_test_w = X_test.copy()
    bounds = {}

    for col in columns:
        lower = X_train_w[col].quantile(lower_q)
        upper = X_train_w[col].quantile(upper_q)

        X_train_w[col] = X_train_w[col].clip(lower, upper)
        X_test_w[col] = X_test_w[col].clip(lower, upper)

        bounds[col] = {
            'lower_q': lower_q,
            'upper_q': upper_q,
            'lower_bound': lower,
            'upper_bound': upper
        }

    return X_train_w, X_test_w, bounds


X_train, X_test, winsor_bounds = winsorize_by_train_quantiles(
    X_train,
    X_test,
    columns=winsor_columns,
    lower_q=0.01,
    upper_q=0.99
```

```
)

winsor_bounds_df = pd.DataFrame(winsor_bounds).T
print('Межі winsorization, розраховані на train-вибірці:')
display(winsor_bounds_df)

extreme_summary_after = summarize_extremes(X_train, winsor_columns)
print('Опис екстремальних значень після winsorization:')
display(extreme_summary_after)

#%% md
## Кластеризація KMeans як додаткова синтетична ознака
#%%
# Пояснення: обираються ознаки для кластеризації клієнтів.
# Використовуються фінансові та поведінкові показники, які описують тип клієнта.

cluster_features = ['LIMIT_BAL', 'AGE', 'AVG_UTIL', 'MAX_UTIL', 'TOTAL_BILL', 'TOTAL_PAY',
'MEAN_PAY_DELAY', 'NUM_DELAY_MONTHS', 'BEHAVIOR_RISK_SCORE']

cluster_scaler = StandardScaler()
X_train_cluster_scaled = cluster_scaler.fit_transform(X_train[cluster_features])
X_test_cluster_scaled = cluster_scaler.transform(X_test[cluster_features])

inertias = []
K_range = range(2, 9)
for k in K_range:
    km = KMeans(n_clusters=k, random_state=RANDOM_STATE, n_init=10)
    km.fit(X_train_cluster_scaled)
    inertias.append(km.inertia_)

plt.figure(figsize=(7,4))
plt.plot(list(K_range), inertias, marker='o')
plt.title('Метод ліктя для вибору кількості кластерів')
plt.xlabel('k')
plt.ylabel('Inertia')
plt.show()
#%%
# Пояснення: KMeans ділить клієнтів на групи зі схожою поведінкою.
# Номер кластера додається як нова ознака CLIENT_CLUSTER для подальшого моделювання.

N_CLUSTERS = 4
kmeans = KMeans(n_clusters=N_CLUSTERS, random_state=RANDOM_STATE, n_init=10)
X_train = X_train.copy()
X_test = X_test.copy()
X_train['CLIENT_CLUSTER'] = kmeans.fit_predict(X_train_cluster_scaled).astype(str)
X_test['CLIENT_CLUSTER'] = kmeans.predict(X_test_cluster_scaled).astype(str)

cluster_default = pd.DataFrame({'cluster': X_train['CLIENT_CLUSTER'], 'default':
y_train}).groupby('cluster')['default'].agg(['count', 'mean']).sort_values('mean', ascending=False)
display(cluster_default)

plt.figure(figsize=(7,4))
sns.barplot(data=cluster_default.reset_index(), x='cluster', y='mean')
plt.title('Частка дефолтів у кластерах train')
plt.xlabel('Кластер')
plt.ylabel('Default rate')
plt.show()
#%%
# Пояснення: PCA зменшує багатовимірні дані до двох координат для візуалізації кластерів.
# Хрестики на графіку показують центри кластерів, а точки - клієнтів.

pca = PCA(n_components=2, random_state=RANDOM_STATE)
pca_points = pca.fit_transform(X_train_cluster_scaled)
```

```
pca_df = pd.DataFrame({'PC1': pca_points[:,0], 'PC2': pca_points[:,1], 'cluster': X_train['CLIENT_CLUSTER'].values,
'default': y_train.values})

plt.figure(figsize=(9,6))
sns.scatterplot(data=pca_df.sample(min(5000, len(pca_df)), random_state=RANDOM_STATE), x='PC1', y='PC2',
hue='cluster', style='default', alpha=0.65)
plt.title('Візуалізація клієнтських кластерів у PCA-просторі')
plt.show()
#%% md
## Препроцесинг для моделей
#%%
# визначаються категоріальні та числові змінні для препроцесингу.
# числові ознаки масштабуються, категоріальні перетворюються на dummy/one-hot ознаки.

categorical_cols = ['SEX', 'EDUCATION', 'MARRIAGE', 'CLIENT_CLUSTER']
for col in categorical_cols:
    X_train[col] = X_train[col].astype('category')
    X_test[col] = X_test[col].astype('category')

numeric_cols = [c for c in X_train.columns if c not in categorical_cols]

preprocess = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_cols),
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_cols)
    ]
)

print('Кількість числових ознак:', len(numeric_cols))
print('Категоріальні ознаки:', categorical_cols)
#%% md
## Функції оцінювання моделей
#%%
# функція збирає всі основні метрики якості моделі в один словник.
# це дає можливість однаково оцінити всі моделі та порівняти їх у таблиці

def evaluate_predictions(y_true, y_pred, y_proba, model_name, threshold=0.5):
    cm = confusion_matrix(y_true, y_pred)
    tn, fp, fn, tp = cm.ravel()
    specificity = tn / (tn + fp) if (tn + fp) else 0
    gmean = np.sqrt(recall_score(y_true, y_pred, zero_division=0) * specificity)
    return {
        'model': model_name,
        'threshold': threshold,
        'accuracy': accuracy_score(y_true, y_pred),
        'balanced_accuracy': balanced_accuracy_score(y_true, y_pred),
        'precision': precision_score(y_true, y_pred, zero_division=0),
        'recall': recall_score(y_true, y_pred, zero_division=0),
        'specificity': specificity,
        'f1': f1_score(y_true, y_pred, zero_division=0),
        'roc_auc': roc_auc_score(y_true, y_proba),
        'pr_auc': average_precision_score(y_true, y_proba),
        'gmean': gmean,
        'tn': tn, 'fp': fp, 'fn': fn, 'tp': tp
    }

def get_proba(model, X):
    if hasattr(model, 'predict_proba'):
        return model.predict_proba(X)[:, 1]
    scores = model.decision_function(X)
    return (scores - scores.min()) / (scores.max() - scores.min() + 1e-9)

def plot_model_diagnostics(model, X_test, y_test, title, threshold=0.5):
```

```
    y_proba = get_proba(model, X_test)
    y_pred = (y_proba >= threshold).astype(int)

    fig, axes = plt.subplots(1, 3, figsize=(18, 5))
    ConfusionMatrixDisplay.from_predictions(y_test, y_pred, ax=axes[0], cmap='Blues')
    axes[0].set_title(f'Матриця помилок: {title}')

    RocCurveDisplay.from_predictions(y_test, y_proba, ax=axes[1])
    axes[1].set_title(f'ROC-крива: {title}')

    PrecisionRecallDisplay.from_predictions(y_test, y_proba, ax=axes[2])
    axes[2].set_title(f'PR-крива: {title}')
    plt.tight_layout()
    plt.show()

    print(classification_report(y_test, y_pred, digits=4))
#%% md
## Навчання базових моделей
#%%
# scale_pos_weight враховує дисбаланс класів для XGBoost.
# збільшує вагу рідшого класу default=1, щоб модель краще реагувала на дефолтних клієнтів

neg, pos = np.bincount(y_train)
scale_pos_weight = neg / pos

def build_models():
    models = {
        'Dummy_most_frequent': DummyClassifier(strategy='most_frequent'),
        'LogReg': LogisticRegression(max_iter=2000, random_state=RANDOM_STATE),
        'LogReg_balanced': LogisticRegression(max_iter=2000, class_weight='balanced', random_state=RANDOM_STATE),
        'RandomForest': RandomForestClassifier(n_estimators=300, random_state=RANDOM_STATE, n_jobs=-1),
        'RandomForest_balanced': RandomForestClassifier(n_estimators=300, class_weight='balanced_subsample',
random_state=RANDOM_STATE, n_jobs=-1),
    }
    if XGBOOST_AVAILABLE:
        models['XGBoost'] = XGBClassifier(
            n_estimators=350, max_depth=4, learning_rate=0.05, subsample=0.9, colsample_bytree=0.9,
            objective='binary:logistic', eval_metric='logloss', random_state=RANDOM_STATE, n_jobs=-1
        )
        models['XGBoost_weighted'] = XGBClassifier(
            n_estimators=350, max_depth=4, learning_rate=0.05, subsample=0.9, colsample_bytree=0.9,
            objective='binary:logistic', eval_metric='logloss', random_state=RANDOM_STATE, n_jobs=-1,
            scale_pos_weight=scale_pos_weight
        )
    else:
        print('XGBoost недоступний: XGBoost-моделі не додаються до експериментів.')
    return models

base_models = build_models()
base_models.keys()
#%%
# створюються варіанти балансування класів.
# SMOTE і ADASYN генерують синтетичні приклади рідшого класу, а none означає навчання без оверсемплінгу

samplers = {
    'none': None,
    'SMOTE': SMOTE(random_state=RANDOM_STATE, k_neighbors=5),
    'ADASYN': ADASYN(random_state=RANDOM_STATE, n_neighbors=5)
}

# Для моделей із власним балансуванням не дублюємо всі sampler-и, щоб не перевантажувати обчислення.
# Назви експериментів формуємо з одним підкресленням, щоб уникнути KeyError у наступних блоках.
experiments = {}
```

```
for model_name, clf in base_models.items():
    if model_name == 'Dummy_most_frequent' or 'balanced' in model_name or 'weighted' in model_name:
        experiments[f'{model_name}_none'] = ImbPipeline([('preprocess', preprocess), ('model', clf)])
    else:
        for sampler_name, sampler in samplers.items():
            steps = [('preprocess', preprocess)]
            if sampler is not None:
                steps.append(('sampler', sampler))
            steps.append(('model', clf))
            experiments[f'{model_name}_{sampler_name}'] = ImbPipeline(steps)

print('Кількість експериментів:', len(experiments))
list(experiments.keys())
#%%
# у циклі навчаються всі моделі й одразу оцінюються на test set.
# для початкового порівняння використовується стандартний threshold=0.5

results = []
fitted_models = {}

for name, pipe in experiments.items():
    print(f'Навчання: {name}')
    pipe.fit(X_train, y_train)
    y_proba = get_proba(pipe, X_test)
    y_pred = (y_proba >= 0.5).astype(int)
    results.append(evaluate_predictions(y_test, y_pred, y_proba, name, threshold=0.5))
    fitted_models[name] = pipe

results_df = pd.DataFrame(results).sort_values(['pr_auc', 'recall', 'f1'], ascending=False).reset_index(drop=True)
display(results_df)
#%% md
## Тюнинг XGBoost
#%%

# randomizedSearchcv перевіряє кілька комбінацій гіперпараметрів XGBoost.
# пошук проводиться тільки на train за крос-валідацією.
# тестова вибірка не використовується для підбору параметрів, а лише для фінальної перевірки.

# результат tuned XGBoost виводимо окремо і не додаємо до results_df щоб не змінювати попередню таблицю
порівняння моделей.

if not XGBOOST_AVAILABLE:
    print('XGBoost недоступний. Блок тюнінгу не виконується.')
else:
    tuned_xgb_pipe = ImbPipeline([
        ('preprocess', preprocess),
        ('model', XGBClassifier(
            objective='binary:logistic',
            eval_metric='logloss',
            random_state=RANDOM_STATE,
            n_jobs=-1,
            scale_pos_weight=scale_pos_weight
        ))
    ])

    param_distributions = {
        'model__n_estimators': [250, 350, 500, 700],
        'model__max_depth': [2, 3, 4, 5],
        'model__learning_rate': [0.02, 0.03, 0.05, 0.08],
        'model__subsample': [0.75, 0.85, 0.95],
        'model__colsample_bytree': [0.75, 0.85, 0.95],
        'model__min_child_weight': [1, 3, 5, 8],
        'model__gamma': [0, 0.1, 0.3],
```

```
        'model__reg_lambda': [1, 3, 5, 10]
    }

    xgb_search = RandomizedSearchCV(
        estimator=tuned_xgb_pipe,
        param_distributions=param_distributions,
        n_iter=12,
        scoring='average_precision',
        cv=3,
        random_state=RANDOM_STATE,
        n_jobs=-1,
        verbose=1
    )

    # Навчання пошуку гіперпараметрів
    xgb_search.fit(X_train, y_train)

    # best_estimator_ — це вже навчений pipeline з найкращими параметрами
    tuned_model = xgb_search.best_estimator_
    tuned_name = 'XGBoost_tuned_weighted_none'

    # Оцінювання tuned XGBoost на test при стандартному threshold = 0.5
    y_proba_tuned = get_proba(tuned_model, X_test)
    y_pred_tuned = (y_proba_tuned >= 0.5).astype(int)

    tuned_result = evaluate_predictions(
        y_test,
        y_pred_tuned,
        y_proba_tuned,
        tuned_name,
        threshold=0.5
    )

    print('Найкращі параметри tuned XGBoost:')
    display(pd.Series(xgb_search.best_params_).to_frame('value'))

    print('Найкращий CV average_precision:')
    print(round(xgb_search.best_score_, 4))

    print('Результат tuned XGBoost на test при threshold = 0.5:')
    display(pd.DataFrame([tuned_result]))

#%%
# загальний графік метрик буде перевантаженим  якщо показувати всі експерименти
# тут виводимо тільки top-8 моделей за PR-AUC
# Детальніше далі буде

top_overview_models = results_df.sort_values('pr_auc', ascending=False).head(8)['model'].tolist()
plot_overview_df = results_df[results_df['model'].isin(top_overview_models)].copy()

plot_df = plot_overview_df.melt(
    id_vars='model',
    value_vars=['precision', 'recall', 'f1', 'roc_auc', 'pr_auc'],
    var_name='metric',
    value_name='value'
)

plt.figure(figsize=(12, 6))
sns.barplot(data=plot_df, x='value', y='model', hue='metric')
plt.title('Оглядове порівняння top-8 моделей за ключовими метриками')
plt.xlabel('Значення метрики')
plt.ylabel('Модель')
plt.xlim(0, 1)
```

```
plt.legend(title='Метрика', bbox_to_anchor=(1.02, 1), loc='upper left')
plt.tight_layout()
plt.show()
#%%
# назви моделей розділяються на алгоритм і спосіб балансування.
# це зробить графіки зрозумілішими та чистими

results_plot = results_df.copy()

def parse_model_name(model_name):
    if model_name.startswith("LogReg"):
        algorithm = "Logistic Regression"
    elif model_name.startswith("RandomForest"):
        algorithm = "Random Forest"
    elif model_name.startswith("XGBoost"):
        algorithm = "XGBoost"
    elif model_name.startswith("Dummy"):
        algorithm = "Dummy"
    else:
        algorithm = "Other"

    if "_SMOTE" in model_name:
        balancing = "SMOTE"
    elif "_ADASYN" in model_name:
        balancing = "ADASYN"
    elif "_balanced_" in model_name:
        balancing = "class_weight"
    elif "_weighted_" in model_name:
        balancing = "class_weight"
    elif "_none" in model_name:
        balancing = "none"
    else:
        balancing = "other"

    return pd.Series([algorithm, balancing])

results_plot[["algorithm", "balancing"]] = results_plot["model"].apply(parse_model_name)
results_plot
#%%
# побудова окремих графіків за алгоритмами
# краще видно вплив балансування класів

metrics_to_plot = ["precision", "recall", "f1", "roc_auc", "pr_auc"]

algorithms_to_show = ["Logistic Regression", "Random Forest", "XGBoost"]

for algo in algorithms_to_show:
    subset = results_plot[results_plot["algorithm"] == algo].copy()

    plot_df = subset.melt(
        id_vars=["model", "algorithm", "balancing"],
        value_vars=metrics_to_plot,
        var_name="metric",
        value_name="value"
    )

    plt.figure(figsize=(11, 5))
    sns.barplot(data=plot_df, x="metric", y="value", hue="balancing")
    plt.title(f"{algo}: порівняння підходів до балансування")
    plt.xlabel("Метрика")
    plt.ylabel("Значення")
    plt.ylim(0, 1)
    plt.legend(title="Балансування", bbox_to_anchor=(1.02, 1), loc="upper left")
```

```
    plt.show()
#%%
# побудова окремих графіки за способами балансування
# вигляд по алгоритмам

balancing_to_show = ["none", "SMOTE", "ADASYN", "class_weight"]

for bal in balancing_to_show:
    subset = results_plot[
        (results_plot["balancing"] == bal) &
        (results_plot["algorithm"] != "Dummy")
    ].copy()

    if subset.empty:
        continue

    plot_df = subset.melt(
        id_vars=["model", "algorithm", "balancing"],
        value_vars=metrics_to_plot,
        var_name="metric",
        value_name="value"
    )

    plt.figure(figsize=(11, 5))
    sns.barplot(data=plot_df, x="metric", y="value", hue="algorithm")
    plt.title(f"Порівняння алгоритмів для підходу балансування: {bal}")
    plt.xlabel("Метрика")
    plt.ylabel("Значення")
    plt.ylim(0, 1)
    plt.legend(title="Алгоритм", bbox_to_anchor=(1.02, 1), loc="upper left")
    plt.show()
#%%
# залишаємо тільки найкращі моделі для порівняння

# Обираємо компактний набір моделей для фінального порівняння:
# 1) Dummy як базова лінія; 2) найкраща Logistic Regression;
# 3) найкращий Random Forest; 4) найкращий XGBoost/бустинг.
def best_by_prefix(prefix):
    subset = results_df[results_df['model'].str.startswith(prefix)].copy()
    if subset.empty:
        return None
    return subset.sort_values(['pr_auc', 'recall', 'f1'], ascending=False).iloc[0]['model']

selected_models = [
    best_by_prefix('Dummy'),
    best_by_prefix('LogReg'),
    best_by_prefix('RandomForest'),
    best_by_prefix('XGBoost')
]
selected_models = [m for m in selected_models if m is not None]

final_plot_df = results_df[results_df['model'].isin(selected_models)].copy()

print('Моделі, залишені для короткого фінального порівняння:')
print(selected_models)

display(final_plot_df[[
    'model', 'accuracy', 'balanced_accuracy', 'precision',
    'recall', 'f1', 'roc_auc', 'pr_auc'
]])

plot_df = final_plot_df.melt(
    id_vars='model',
```

```
    value_vars=['precision', 'recall', 'f1', 'roc_auc', 'pr_auc'],
    var_name='metric',
    value_name='value'
)

plt.figure(figsize=(11, 5))
sns.barplot(
    data=plot_df,
    x='metric',
    y='value',
    hue='model'
)

plt.title('Фінальне порівняння відібраних моделей')
plt.xlabel('Метрика')
plt.ylabel('Значення')
plt.ylim(0, 1)
plt.legend(title='Модель', bbox_to_anchor=(1.02, 1), loc='upper left')
plt.show()
#%%

#окреме порівняння фінальних моделей за кожною ключовою метрикою з кольорами
important_metrics = ['recall', 'precision', 'f1', 'roc_auc', 'pr_auc']

palette_values = sns.color_palette('Set2', n_colors=len(selected_models))
model_palette = dict(zip(selected_models, palette_values))

for metric in important_metrics:
    metric_df = final_plot_df.sort_values(metric, ascending=False)

    plt.figure(figsize=(10, 4))
    sns.barplot(
        data=metric_df,
        x=metric,
        y='model',
        hue='model',
        palette=model_palette,
        dodge=False,
        legend=False
    )

    plt.title(f'Порівняння фінальних моделей за метрикою: {metric}')
    plt.xlabel(metric)
    plt.ylabel('Модель')
    plt.xlim(0, 1)
    plt.show()
#%% md
## Крос-валідація найкращих моделей
#%%
# для крос-валідації беруться найкращі моделі з попереднього порівняння.


# Крос-валідація

candidate_model_names = results_df.head(10)['model'].tolist()
top_model_names = [name for name in candidate_model_names if name in experiments]

print('Моделі для крос-валідації:')
print(top_model_names)

cv = StratifiedKFold(
    n_splits=5,
    shuffle=True,
```

```
    random_state=RANDOM_STATE
)

scoring = {
    'precision': 'precision',
    'recall': 'recall',
    'f1': 'f1',
    'roc_auc': 'roc_auc',
    'average_precision': 'average_precision'
}

cv_rows = []

for name in top_model_names:
    print('CV:', name)

    scores = cross_validate(
        experiments[name],
        X_train,
        y_train,
        cv=cv,
        scoring=scoring,
        n_jobs=-1,
        error_score='raise'
    )

    row = {'model': name}

    for metric in scoring:
        row[f'{metric}_mean'] = scores[f'test_{metric}'].mean()
        row[f'{metric}_std'] = scores[f'test_{metric}'].std()

    cv_rows.append(row)

cv_df = pd.DataFrame(cv_rows).sort_values(
    'average_precision_mean',
    ascending=False
).reset_index(drop=True)

display(cv_df)
#%% md
## Аналіз порогу класифікації
#%%
#  обирається фінальна модель для аналізу threshold.

# Підбір оптимального порогу класифікації

available_results_df = results_df[
    results_df['model'].isin(fitted_models.keys())
].copy()

display(available_results_df[[
    'model', 'precision', 'recall', 'f1', 'roc_auc', 'pr_auc'
]].sort_values('pr_auc', ascending=False))

# Фінальну модель для аналізу порогу обираємо за PR-AUC серед реально навчених моделей.
# PR-AUC особливо корисна при дисбалансі класів, бо фокусується на якості виявлення класу 1
best_default_name = available_results_df.sort_values(
    ['pr_auc', 'recall', 'f1'],
    ascending=False
).iloc[0]['model']

best_model = fitted_models[best_default_name]
```

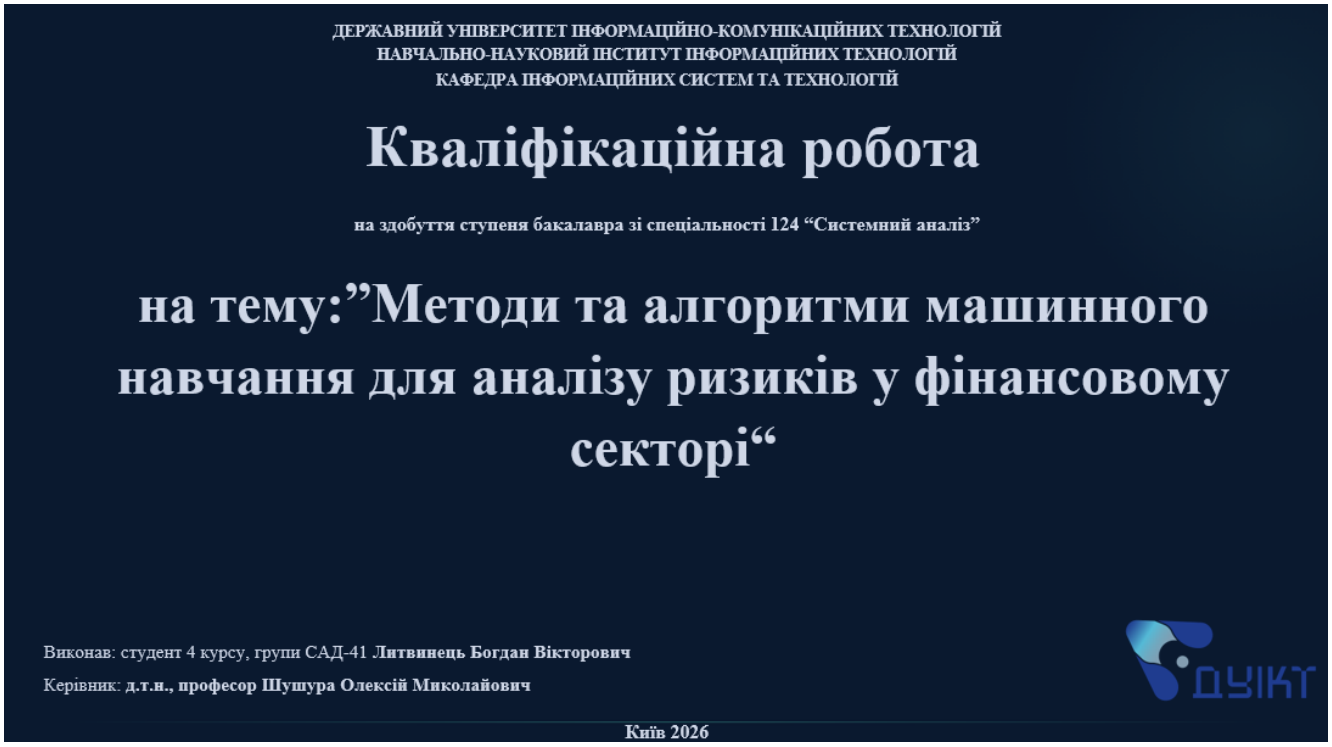
```
y_proba_best = get_proba(best_model, X_test)

print('Найкраща доступна модель за PR-AUC:', best_default_name)


def threshold_table(y_true, y_proba, fp_cost=1, fn_cost=5):
    thresholds = np.linspace(0.05, 0.95, 181)
    rows = []

    for t in thresholds:
        y_pred = (y_proba >= t).astype(int)
        tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()

        rows.append({
            'threshold': t,
            'precision': precision_score(y_true, y_pred, zero_division=0),
            'recall': recall_score(y_true, y_pred, zero_division=0),
            'specificity': tn / (tn + fp) if (tn + fp) else 0,
            'f1': f1_score(y_true, y_pred, zero_division=0),
            'cost': fn_cost * fn + fp_cost * fp,
            'fp': fp,
            'fn': fn,
            'tp': tp,
            'tn': tn
        })

    return pd.DataFrame(rows)


# Додатково виділяємо validation-частину з train для підбору threshold.
X_train_inner, X_val_thr, y_train_inner, y_val_thr = train_test_split(
    X_train,
    y_train,
    test_size=0.20,
    stratify=y_train,
    random_state=RANDOM_STATE
)

threshold_model = clone(experiments[best_default_name])
threshold_model.fit(X_train_inner, y_train_inner)
y_val_proba = get_proba(threshold_model, X_val_thr)

# У кредитному скорингу FN дорожчий за FP:
# FN = пропущений дефолт, FP = зайва перевірка або помилкова відмова.
thr_df = threshold_table(
    y_val_thr,
    y_val_proba,
    fp_cost=1,
    fn_cost=5
)

best_f1_thr = thr_df.loc[thr_df['f1'].idxmax(), 'threshold']
best_cost_thr = thr_df.loc[thr_df['cost'].idxmin(), 'threshold']
best_youden_thr = thr_df.loc[
    (thr_df['recall'] + thr_df['specificity'] - 1).idxmax(),
    'threshold'
]

print('Поріг для max F1 на validation:', round(best_f1_thr, 3))
print('Поріг для min cost на validation:', round(best_cost_thr, 3))
print('Поріг для max Youden на validation:', round(best_youden_thr, 3))

display(thr_df.sort_values('cost').head(10))
```

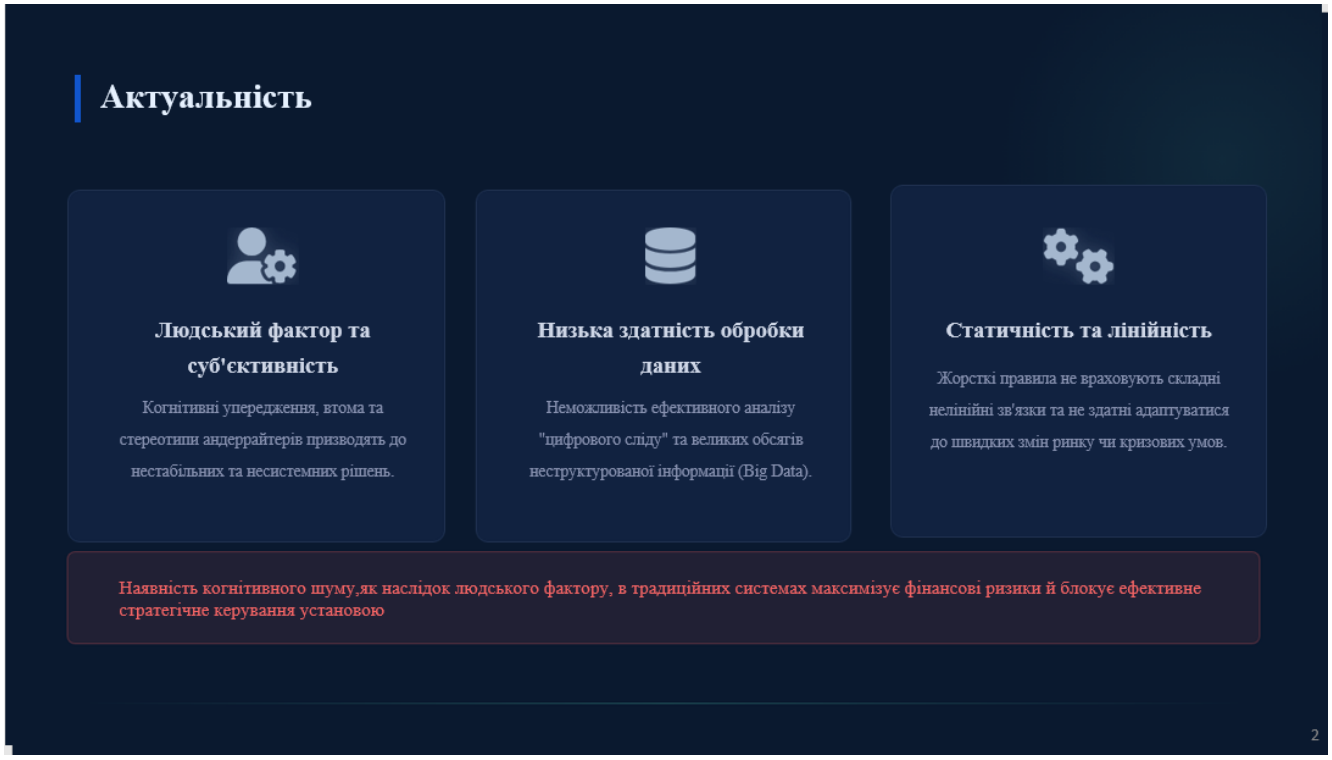
```
#%%
# графіки показують, як зміна threshold впливає на Precision, Recall, F1 і умовну вартість помилок на validation-частині.
# Можна побачити поріг який краще відповідає ціні

plt.figure(figsize=(10,6))
plt.plot(thr_df['threshold'], thr_df['precision'], label='Precision')
plt.plot(thr_df['threshold'], thr_df['recall'], label='Recall')
plt.plot(thr_df['threshold'], thr_df['f1'], label='F1')
plt.axvline(best_cost_thr, linestyle='--', label=f'Min cost threshold={best_cost_thr:.2f}')
plt.title(f'Вплив порогу на метрики: {best_default_name}')
plt.xlabel('Поріг')
plt.ylabel('Значення')
plt.legend()
plt.show()

plt.figure(figsize=(10,5))
plt.plot(thr_df['threshold'], thr_df['cost'])
plt.axvline(best_cost_thr, linestyle='--', label=f'Min cost={best_cost_thr:.2f}')
plt.title('Умовна вартість помилок залежно від порогу')
plt.xlabel('Поріг')
plt.ylabel('Cost = 5*FN + 1*FP')
plt.legend()
plt.show()
#%%
# перевірка різних порогів класифікації
# пороги відбирали на валідаційнійній вибірці тут чекаєм на тестовій

compare_thresholds = []
for t in [0.5, best_f1_thr, best_youden_thr, best_cost_thr]:
    yp = (y_proba_best >= t).astype(int)
    compare_thresholds.append(
        evaluate_predictions(y_test, yp, y_proba_best, best_default_name, threshold=float(t))
    )

threshold_results = pd.DataFrame(compare_thresholds).drop_duplicates(subset=['threshold'])
display(threshold_results)

plot_model_diagnostics(
    best_model,
    X_test,
    y_test,
    f'{best_default_name}, validation threshold={best_cost_thr:.2f}',
    threshold=best_cost_thr
)
#%% md
## Інтерпретація моделі та важливість ознак
#%%
# отримуються назви ознак після препроцесингу.
# це потрібно, щоб коректно інтерпретувати важливість факторів у моделі

# Отримання назв ознак після препроцесингу
preprocessor = best_model.named_steps['preprocess']
try:
    feature_names = preprocessor.get_feature_names_out()
except Exception:
    feature_names = np.array([f'f{i}' for i in range(preprocessor.transform(X_train).shape[1])])

model_step = best_model.named_steps['model']

if hasattr(model_step, 'feature_importances_'):
    importances = pd.Series(model_step.feature_importances_,
index=feature_names).sort_values(ascending=False).head(25)
```

```
    plt.figure(figsize=(10,8))
    sns.barplot(x=importances.values, y=importances.index)
    plt.title(f'Top-25 важливих ознак: {best_default_name}')
    plt.xlabel('Importance')
    plt.ylabel('Feature')
    plt.show()
    display(importances.to_frame('importance'))
elif hasattr(model_step, 'coef_'):
    coefs = pd.Series(model_step.coef_.ravel(), index=feature_names).sort_values(key=np.abs, ascending=False).head(25)
    plt.figure(figsize=(10,8))
    sns.barplot(x=coefs.values, y=coefs.index)
    plt.title(f'Top-25 коефіцієнтів: {best_default_name}')
    plt.xlabel('Coefficient')
    plt.ylabel('Feature')
    plt.show()
    display(coefs.to_frame('coefficient'))
else:
    print('Для цієї моделі немає прямого атрибуту важливості. Використайте permutation importance нижче.')
#%%
# permutation importance оцінює, наскільки погіршується якість моделі при випадковому перемішуванні окремої ознаки.
# кщо якість сильно падає, ознака є важливою для прогнозу

# Permutation importance для моделі
perm = permutation_importance(best_model, X_test, y_test, n_repeats=5, random_state=RANDOM_STATE,
scoring='average_precision', n_jobs=-1)
perm_imp = pd.Series(perm.importances_mean, index=X_test.columns).sort_values(ascending=False).head(20)

plt.figure(figsize=(10,7))
sns.barplot(x=perm_imp.values, y=perm_imp.index)
plt.title('Permutation importance за PR AUC на test')
plt.xlabel('Середнє падіння метрики')
plt.ylabel('Ознака')
plt.show()

display(perm_imp.to_frame('permutation_importance'))
#%% md
## Бізнес-інтерпретація матриці помилок
#%%
# матриця помилок

chosen_threshold = float(best_cost_thr)
y_pred_chosen = (y_proba_best >= chosen_threshold).astype(int)
cm = confusion_matrix(y_test, y_pred_chosen)

true_negative, false_positive, false_negative, true_positive = cm.ravel()

business_summary = pd.DataFrame({
    'Тип': [
        'True Negative: правильно визначені клієнти без дефолту',
        'False Positive: помилково визначені дефолтні клієнти',
        'False Negative: пропущені дефолтні клієнти',
        'True Positive: правильно визначені дефолтні клієнти'
    ],
    'Кількість': [true_negative, false_positive, false_negative, true_positive],
    'Бізнес-сенс': [
        'Клієнт не дефолтний і модель не позначила його ризиковим',
        'Можлива зайва відмова або додаткова перевірка надійного клієнта',
        'Найбільш небезпечна помилка: ризиковий клієнт може отримати кредит',
        'Ризиковий клієнт правильно виявлений моделлю'
    ]
})
```

```
display(business_summary)

# Візуалізація
fig, ax = plt.subplots(figsize=(7, 6))

sns.heatmap(
    cm,
    annot=np.array([
        [f'True Negative\n{true_negative}', f'False Positive\n{false_positive}'],
        [f'False Negative\n{false_negative}', f'True Positive\n{true_positive}']
    ]),
    fmt='',
    cmap='Blues',
    cbar=False,
    xticklabels=['Прогноз: без дефолту', 'Прогноз: дефолт'],
    yticklabels=['Факт: без дефолту', 'Факт: дефолт'],
    ax=ax
)

plt.title(f'Матриця помилок: {best_default_name}\nthreshold = {chosen_threshold:.3f}')
plt.xlabel('Прогноз моделі')
plt.ylabel('Фактичний клас')
plt.tight_layout()
plt.show()
```

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ





Постановка завдання

Об'єкт дослідження

Процес оцінювання кредитного ризику у фінансовому секторі в умовах невизначеності та великих обсягів даних.

Предмет дослідження

Методи та алгоритми машинного навчання (Logistic Regression, Random Forest, XGBoost) для прогнозування дефолту.

Мета роботи

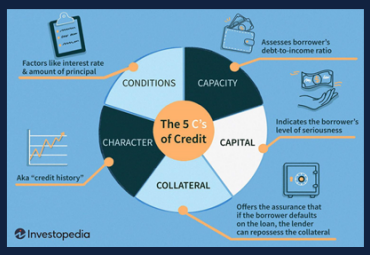
Дослідження, реалізація та порівняльний аналіз ML-алгоритмів для задачі кредитного скорингу.

Завдання роботи

- Дослідження та реалізація методів машинного навчання для задачі кредитного скорингу,
- аналіз дисбалансу класів,
- порівняння якості моделей та оцінювання ефективності прогнозування дефолту клієнтів.

3

Трансформація кредитного скорингу



Традиційний підхід: 5Cs

- Character (Репутація)
- Capacity (Спроможність)
- Capital (Активи)
- Collateral (Застава)
- Conditions (Умови)

Базується на інтуїції та експертній оцінці.

Обмеженість

- **Експертна суб'єктивність**
- **Низька швидкість обробки**
- **Ігнорування нелінійних зв'язків**

ML Моделі та Алгоритми

- Логістична регресія
- Випадковий ліс (Random Forest)
- Градієнтний бустинг (XGBoost)
- Інженерія ознак (Feature Eng.)
- Автоматизований скоринг

Забезпечує об'єктивність та прогнозну точність.

Обмеженість традиційного підходу збільшує частку проблемних активів та гальмує темпи розвитку установи, що робить впровадження ML моделей необхідністю.

4

Методологія дослідження методів машинного навчання

Підготовка даних

Очищення від викидів,вінсоризація фінансових показників (квантилі 1% та 99%)

Інженерія ознак

Створення синтетичних ознак для збільшення предиктивної здатності моделей

Балансування класів

Вирішення дисбалансу класів за допомогою SMOTE та ADASYN та зважуванням ваг класів

Аналітичні інструменти

Метрики: accuracy,recall,f1-score,ROC-AUC,PR-AUC.

Графіки: стовпчасті діаграми, теплові карти, діаграма розсіювання для кластерів..

5

Програмна реалізація

```
X_all = df_fe.drop(columns=['default'])
y_all = df_fe['default'].astype(int)

X_train, X_test, y_train, y_test = train_test_split(
    X_all, y_all, test_size=0.2, random_state=RANDOM_STATE, stratify=y_all
)

print('Train:', X_train.shape, 'Test:', X_test.shape)
print('Default rate train:', y_train.mean())
print('Default rate test:', y_test.mean())
```

```
Train: (24000, 44) Test: (6000, 44)
Default rate train: 0.2212083333333334
Default rate test: 0.22116666666666668
```

Фрагмент підготовки та розділення даних

Мова та бібліотеки

Python,pandas,matplotlib,imbalanced-learn,numpy,seaborn,scikit-learn,xgboost.

```
Навчання: Dummy_most_frequent_none
Навчання: LogReg_none
Навчання: LogReg_SMOTE
Навчання: LogReg_ADASYN
Навчання: LogReg_balanced_none
Навчання: RandomForest_none
Навчання: RandomForest_SMOTE
Навчання: RandomForest_ADASYN
Навчання: RandomForest_balanced_none
Навчання: XGBoost_none
Навчання: XGBoost_SMOTE
Навчання: XGBoost_ADASYN
Навчання: XGBoost_weighted_none
```

Навчання моделей

Моделі

Dummy baseline, Logistic Regression, Random Forest, XGBoost

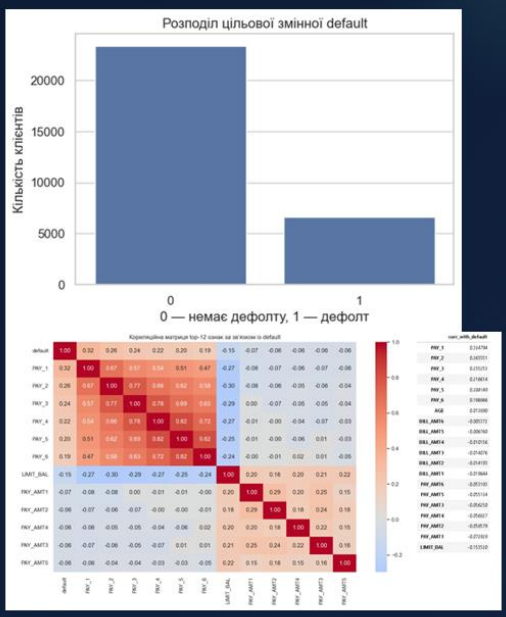
6

Розвідувальний аналіз даних

Ключові закономірності датасету:

- **Дисбаланс класів:** 22.1% дефолтних випадків проти 77.9% надійних.
- **Платіжна дисципліна:** Ознаки PAY_1-PAY_6 мають найвищу позитивну кореляцію з дефолтом.
- **Кредитний ліміт:** Виявлено обернену залежність — клієнти з нижчим LIMIT_BAL мають вищу схильність до дефолту.
- **Якість даних:** Відсутність пропусків у 30,000 записах, проте значний розкид у сумах рахунків.

30 тисяч записів | 24 ознаки



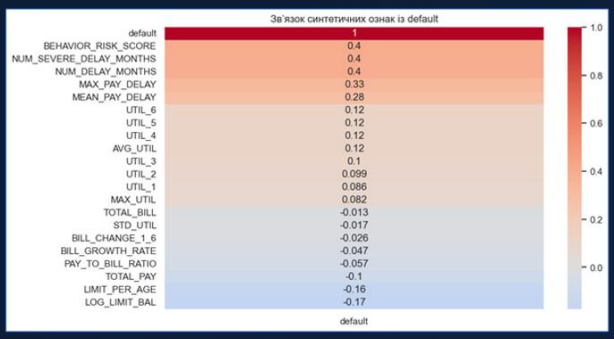
7

Синтетичні ознаки та їхній вплив

Створено 21 нову синтетичну ознаку (разом 45). Ключові нововведення:

- **BEHAVIOR_RISK_SCORE:** Агрегований показник на основі глибини прострочень.
- **PAY_TO_BILL_RATIO:** Відношення платежів до рахунків (здатність до погашення).
- **AVG_UTIL:** Рівень використання кредитного ліміту.

Результат: Синтетичні ознаки мають значно вищу кореляцію з цільовою змінною, ніж демографічні параметри.



8

Порівняльний аналіз найефективніших моделей

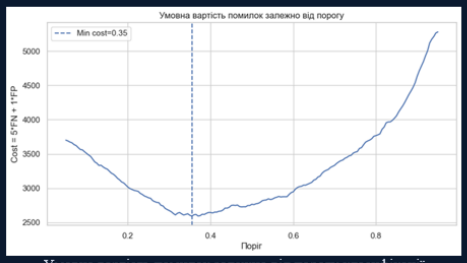
Оцінка ефективності моделей на тестовій вибірці (стандартний поріг 0.5):

Моделі, залишені для короткого фінального порівняння:
['Dummy_most_frequent_none', 'LogReg_none', 'RandomForest_balanced_none', 'XGBoost_weighted_none']

	model	accuracy	balanced_accuracy	precision	recall	f1	roc_auc	pr_auc
0	XGBoost_weighted_none	0.755333	0.710190	0.461071	0.629239	0.532186	0.779238	0.562247
4	RandomForest_balanced_none	0.809500	0.640824	0.628852	0.338357	0.439980	0.765250	0.540914
7	LogReg_none	0.812000	0.633526	0.657188	0.313489	0.424490	0.754206	0.524399
12	Dummy_most_frequent_none	0.778833	0.500000	0.000000	0.000000	0.000000	0.500000	0.221167

Хоча логістична регресія має вищу Accuracy, вона пропускає 69% дефолтів. XGBoost є найбільш збалансованим інструментом для виявлення ризиків.

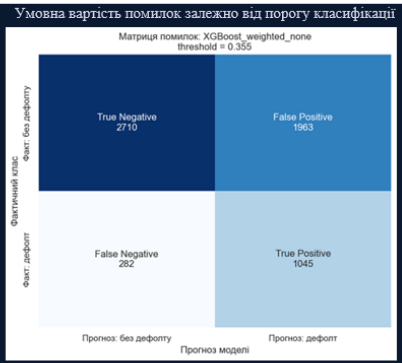
Оптимізація та вартість помилок



Умовна вартість помилок залежно від порогу класифікації

	threshold	precision	recall	specificity	f1	cost	fp	fn	tp	tn
61	0.355	0.357323	0.799435	0.591493	0.493892	2592	1527	213	849	2211
64	0.370	0.367139	0.780603	0.617710	0.499398	2594	1429	233	829	2309
65	0.375	0.369937	0.774011	0.625468	0.500609	2600	1400	240	822	2338
60	0.350	0.352577	0.805085	0.579989	0.490393	2605	1570	207	855	2168
57	0.335	0.343689	0.822976	0.553505	0.484882	2609	1669	188	874	2069
53	0.315	0.331502	0.852166	0.511771	0.477321	2610	1825	157	905	1913
62	0.360	0.359502	0.789077	0.600589	0.493958	2613	1493	224	838	2245
67	0.385	0.376108	0.758945	0.642322	0.502964	2617	1337	256	806	2401
63	0.365	0.362407	0.782486	0.608882	0.495380	2617	1462	231	831	2276
58	0.340	0.345847	0.815443	0.561798	0.485698	2618	1638	196	866	2100

Метрики в залежності від порогу класифікації



Матриця помилок для найбільш ефективної моделі зі зміненим порогом

Бізнес-орієнтований підхід

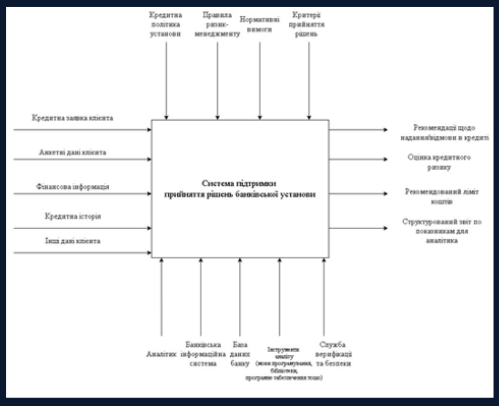
Стандартний поріг (0.5) не враховує ціну помилки.

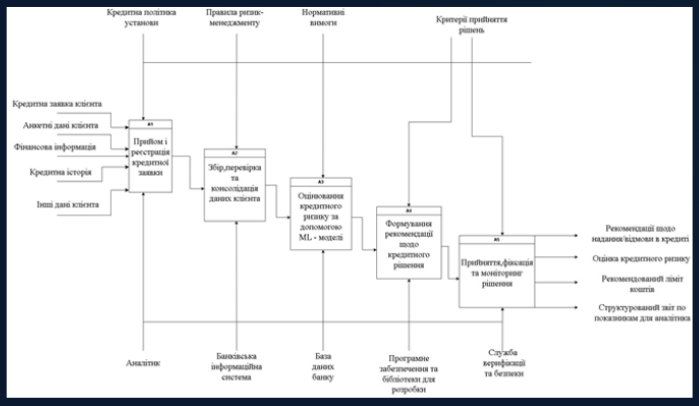
Зниження порогу до **0.355** забезпечує:

- Зростання Recall до **79.9%** (виявлення дефолтів).
- Мінімізацію сумарної бізнес-вартості помилок (Cost = 5*FN + 1*FP).

Оптимізація порогу дозволила врятувати на **17%** більше тіла кредитів порівняно зі стандартним налаштуванням.

Інтерпретація моделі як модуля СППР





Інтерпретація моделі машинного навчання як модуля системи підтримки прийняття рішень (СППР) у кредитному скорингу забезпечує трансформацію складних алгоритмів (зокрема ансамблевих методів на зразок XGBoost) у прозорий та прикладний інструмент для ризик-менеджменту банку. Такий підхід дозволяє фінансовим установам нівелювати проблему «чорної скриньки», забезпечувати відповідність нормативним вимогам регуляторів щодо прозорості оцінювання та трансформувати вихідні дані у зрозумілі бізнес-рекомендації для прийняття об'єктивних рішень у реальному часі.

11

Висновки

У роботі досліджено методи машинного навчання для задачі оцінювання кредитного ризику та кредитного скорингу

Проведено аналіз факторів, що впливають на прогнозування дефолту, а також досліджено проблему дисбалансу класів у фінансових даних

Реалізовано та порівняно моделі Logistic regression, Random forest і XGBoost для задачі бінарної класифікації позичальників

Отримані результати підтверджують доцільність використання ML-моделей для автоматизації кредитного скорингу та підтримки прийняття рішень у банківському секторі

12

Апробація результатів дослідження

Литвинець Б. В. Мінімізація впливу шахрайства та протидія зашумленню даних у системах кредитного скорингу. *Інформаційно-комп'ютерні технології* : тези доповідей XVI Міжнародної науково-технічної конференції (м. Житомир, 02-03 квітня 2026 р.). Житомир, 2026.

Литвинець Б.В. Взаємозв'язок між якістю даних та стійкістю скорингових моделей. Автоматизація конвеєрів підготовки даних. *Сучасний стан та перспективи розвитку IoT* : збірник тез VII Всеукраїнської науково-технічної конференції (м. Київ, 20 квітня 2026 р.). Київ : ДУІКТ, 2026. С. 285–287.

Литвинець Б.В. Інтеграція та обробка альтернативних даних для оцінки кредитного ризику клієнтів з короткою кредитною історією. *Сучасний стан та перспективи розвитку IoT* : збірник тез VII Всеукраїнської науково-технічної конференції (м. Київ, 20 квітня 2026 р.). Київ : ДУІКТ, 2026. С. 284–285.