

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук
(повна назва)
Кафедра _____ Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____ перший (бакалаврський)

Розробка інтерактивної системи предиктивної аналітики з використанням
ансамблевих методів машинного навчання
(тема)

Виконав:
здобувач _____ четвертого _____ року навчання,
групи _____ ІТШ-22-3

Владислав ШЕЛУДЄШЕВ
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки

(код і повна назва спеціальності)

Тип програми _____ освітньо-професійна

Освітня програма _____ Штучний інтелект

(повна назва освітньої програми)

Керівник _____ ас. Дмитро МАЛЄСВ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ

_____ Лариса ЧАЛА
(підпис) (власне ім'я, прізвище)

2026 р.

Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук _____

Кафедра _____ Штучного інтелекту _____

Рівень вищої освіти _____ перший (бакалаврський) _____

Спеціальність _____ 122 Комп'ютерні науки _____
(код і повна назва)

Тип програми _____ освітньо-професійна _____

Освітня програма _____ Штучний інтелект _____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

« _____ » _____ 20 ____ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві _____ Шелудешеву Владиславу Олександровичу _____
(прізвище, ім'я, по батькові)

1. Тема роботи _____ Розробка інтерактивної системи предиктивної аналітики з використанням ансамблевих методів машинного навчання _____

затверджена наказом університету від 5 червня 2026 р. № 488Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 10 липня 2026 р.

3. Вихідні дані до роботи _____ Науково-технічні публікації за тематикою машинного навчання, офіційна документація технологічного стека, набір транзакційних та демографічних даних клієнтів банківського сектору для тренування та тестування алгоритмів _____

4. Перелік питань, що потрібно опрацювати в роботі _____

1) Теоретико-методологічні основи предиктивної аналітики в банківському секторі _____

2) Обґрунтування напрямку дослідження та розробка конвеєра машинного навчання _____

3) Програмна реалізація та експериментальне дослідження інтелектуальної системи _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	05.06.2026	виконано
2	Аналіз предметної галузі	10.06.2026	виконано
3	Попередня обробка даних. Проведення макроекономічної адаптації	13.06.2026	виконано
4	Проектування конвеєра машинного навчання. Навчання ансамблевих моделей	17.06.2026	виконано
5	Програмна реалізація модуля локальної інтерпретації рішень	20.06.2026	виконано
6	Розробка клієнт-серверного веб-застосунку	23.06.2026	виконано
7	Фінальне тестування системи	26.06.2026	виконано
8	Написання пояснювальної записки	28.06.2026	виконано
9	Перевірка на академічний плагіат	01.07.2026	виконано
10	Нормоконтроль	02.07.2026	виконано
11	Підготовка презентації та доповіді	06.07.2026	виконано
12	Попередній захист	07.07.2026	виконано
13	Рецензування	08.07.2026	виконано
14	Захист перед ЕК	10.07.2026	

Дата видачі завдання 8 червня 20 26 р.

Здобувач 
(підпис)

Керівник роботи _____
(підпис)

ас. Дмитро МАЛЄЄВ _____
(посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 50 с., 11 рис., 6 табл., 1 дод., 21 джерело.

АНСАМБЛЕВІ МЕТОДИ, БАНКІВСЬКИЙ СЕКТОР, ВЕБЗАСТОСУНОК, ВІДТІК КЛІЄНТІВ, ДАШБОРД, МАШИННЕ НАВЧАННЯ, ПРЕДИКТИВНА АНАЛІТИКА, EXPLAINABLE AI, FLASK, XGBOOST.

Об'єкт дослідження – прогнозування поведінки клієнтів банку в умовах ризику відтоку.

Предмет дослідження – ансамблеві моделі машинного навчання для класифікації клієнтів.

Мета роботи – створення веб-орієнтованої інтелектуальної системи предиктивної аналітики для автоматичного виявлення клієнтського відтоку.

Методи дослідження – статистичний аналіз, машинне навчання(дерева рішень, бустинг, стекінг), оптимізація гіперпараметрів.

Результати та новизна: розв'язано задачу прогнозування відтоку. Доведено високу ефективність XGBoost при дисбалансі класів. Модель адаптовано під ринок України з інтеграцією динамічної конвертації валют.

Сфера застосування: банківський сектор, відділи ризик-менеджменту.

Практична значимість: розроблено клієнт-серверний Flask-застосунок. Інтерактивний дашборд оцінює ймовірність відтоку та візуально інтерпретує рішення ШІ.

Висновки: ансамблеві моделі(зокрема XGBoost) є надійним інструментом виявлення ризиків. Їх впровадження дозволяє проактивно утримувати клієнтів, підвищуючи фінансову стабільність банку.

ABSTRACT

Bachelor's thesis contains: 50 pp., 11 fig., 6 tabl., 1 ann., 21 references.

BANKING SECTOR, CUSTOMER CHURN, DASHBOARD, ENSEMBLE METHODS, EXPLAINABLE AI, FLASK, MACHINE LEARNING, PREDICTIVE ANALYTICS, WEB APPLICATION, XGBOOST.

Object of research – predicting the behavior of banking sector customers under the risk of churn.

Subject of research – ensemble machine learning models for customer classification.

Purpose of the work – creating a web-oriented intelligent predictive analytics system for the automatic detection of customer churn.

Research methods – statistical analysis, machine learning(decision trees, boosting, stacking), hyperparameter optimization.

Results and novelty: the task of predicting churn has been solved. The high efficiency of XGBoost under class imbalance has been proven. The model is adapted to the Ukrainian market with the integration of dynamic currency conversion.

Field of application: banking sector, risk management departments.

Practical significance: a client-server Flask application has been developed. The interactive dashboard assesses the probability of churn and visually interprets AI decisions.

Conclusions: ensemble models(specifically XGBoost) are a reliable tool for risk detection. Their implementation allows proactively retaining customers, increasing the financial stability of the bank.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів	7
Вступ.....	8
1 Теоретико-методологічні основи предиктивної аналітики в банківському секторі.....	10
1.1 Постановка задачі прогнозування відтоку клієнтів (churn rate) та її економічний контекст.....	10
1.2 Порівняльний аналіз існуючих засобів та комерційних рішень	13
1.3 Огляд класичних методів машинного навчання для аналізу багатовимірних фінансових даних.	16
2 Обґрунтування напрямку дослідження та розробка конвеєра машинного навчання	20
2.1 Математичне обґрунтування вибору ансамблевих методів	20
2.2 Методика попередньої обробки та синтетичної аугментації даних ..	26
2.3 Багатофакторна оптимізація гіперпараметрів за допомогою gridsearchcv	30
3 Програмна реалізація та експериментальне дослідження інтелектуальної системи	33
3.1 Проєктування архітектури клієнт-серверного веб-застосунку	33
3.2 Програмна реалізація аналітичного дашборду та інтеграція модулів системи.	35
3.3 Впровадження інструментів інтерпретації рішень штучного інтелекту	38
3.4 Порівняльний бенчмарк моделей, аналіз матриць помилок та оцінка економічного ефекту.....	42
Висновки	46
Перелік джерел посилання	48
Додаток А відомість кваліфікаційної роботи.....	50

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ІІІ – Штучний інтелект;

API – Application Programming Interface – інтерфейс програмування застосунків;

CAC – Customer Acquisition Cost – вартість залучення клієнта;

JSON – JavaScript Object Notation – текстовий формат обміну даними;

LTV – Lifetime Value – життєва цінність клієнта;

ML – Machine Learning – машинне навчання;

UX – User Experience – користувацький досвід;

XGBoost – eXtreme Gradient Boosting – екстремальний градієнтний бустинг.

ВСТУП

Сучасний банківський сектор функціонує в умовах високої конкуренції, де зміна фінансової установи потребує від клієнта мінімальних зусиль. Проблема прогнозування та попередження відтоку клієнтів (Customer Churn) є одним із найкритичніших викликів для індустрії. Традиційні лінійні методи аналізу транзакційних даних не здатні вловити приховані патерни поведінки споживачів, що зумовлює необхідність впровадження систем машинного навчання.

Актуальність роботи. Витрати на залучення нового клієнта значно перевищують вартість утримання існуючого. Раптовий відтік депозитів негативно впливає на ліквідність банків. Наявні комерційні аналітичні системи часто працюють за принципом «чорної скриньки» (Black Box), що суперечить вимогам фінансових регуляторів щодо прозорості. Розробка інтелектуальної системи, яка поєднує високу якість прогнозування з інструментами інтерпретації результатів, є актуальним науково-практичним завданням.

Мета роботи – розробка та програмна реалізація клієнт-серверної системи для прогнозування відтоку клієнтів банку на основі ансамблевих алгоритмів машинного навчання з впровадженням модуля інтерпретації рішень (Explainable AI).

Для досягнення поставленої мети вирішено такі завдання:

- проаналізовано стан проблеми відтоку клієнтів та досліджено відповідні моделі машинного навчання;
- виконано попередню обробку даних, адаптацію фінансових показників та балансування класів алгоритмом SMOTE;
- спроектовано конвеєр машинного навчання та виконано оптимізацію гіперпараметрів для алгоритмів Bagging, Boosting (XGBoost) та Stacking;

- програмно реалізовано модуль пояснювального штучного інтелекту на базі методу абляції ознак (Feature Ablation);
- створено вебзастосунок (Flask, JavaScript) для візуалізації ризиків у реальному часі;
- проведено порівняльний аналіз алгоритмів та розраховано потенційний економічний ефект.

Об’єкт дослідження – процес предиктивної аналітики та управління взаємовідносинами з клієнтами (CRM) у банківській сфері.

Предмет дослідження – ансамблеві моделі машинного навчання та алгоритми інтерпретації штучного інтелекту (XAI) для класифікації ризиків клієнтського відтоку.

Наукова новизна полягає у подальшому розвитку підходів до інтерпретації складних ансамблів. Замість обчислювально важких інструментів (SHAP, LIME) програмно реалізовано полегшений модуль локальної інтерпретації на основі математичної абляції ознак (Feature Ablation). Це значно прискорило генерацію пояснень та забезпечило виділення домінантних факторів ризику без зниження пропускну здатності сервера.

Практичне значення. Розроблений аналітичний дашборд дозволяє ризик-менеджерам оцінювати ймовірність відтоку та його причини без навичок програмування. Досягнутий рівень повноти виявлення ризикових профілів (Recall 72.7%) дозволяє суттєво знизити фінансові втрати банку. Програмний код може використовуватися у навчальному процесі під час підготовки фахівців з інженерії даних.

Структура роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. У першому розділі наведено теоретичну базу. Другий розділ присвячено математичному обґрунтуванню моделей та побудові ML-конвеєра. У третьому розділі описано архітектуру веб-застосунку, реалізацію XAI та результати оцінки ефективності системи.

1 ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ОСНОВИ ПРЕДИКТИВНОЇ АНАЛІТИКИ В БАНКІВСЬКОМУ СЕКТОРІ

1.1 Постановка задачі прогнозування відтоку клієнтів (Churn Rate) та її економічний контекст

Сучасний комерційний банківський сектор функціонує в умовах високої цифрової конкуренції. Зміна фінансової установи сьогодні відбувається дуже швидко. Завдяки появі технологічно гнучких необанків процес розірвання клієнтських відносин потребує мінімальних зусиль. Традиційна лояльність споживачів знижується. Під час аналізу галузевих звітів було виявлено системну проблему розподілу ресурсів. Менеджмент банків спрямовує значні маркетингові бюджети на залучення нових користувачів. Водночас проблема прихованого відтоку існуючої клієнтської бази часто залишається поза належною увагою [1]. Такий стратегічний підхід є економічно неефективним. Для підтвердження цього було проведено розрахунок ключових метрик фінансової стійкості. Встановлено наявність дисбалансу між вартістю залучення нового клієнта (CAC) та його потенційною пожиттєвою цінністю (LTV) [2]. Саме ці показники напряду визначають загальну маржинальність установи [3] (рисунок 1.1).

Проведення масштабних рекламних кампаній потребує значних фінансових ресурсів через постійне зростання вартості залучення нових клієнтів. Утримання існуючого користувача є економічно доцільнішим. Було проведено аналіз рентабельності. Встановлено, що для стабільного розвитку фінансової установи показник LTV має перевищувати CAC щонайменше втричі. Зниження цієї пропорції призводить до регресії бізнес-моделі банку. Тому архітектуру розроблюваної системи було орієнтовано безпосередньо на мінімізацію коефіцієнта відтоку (Churn Rate).

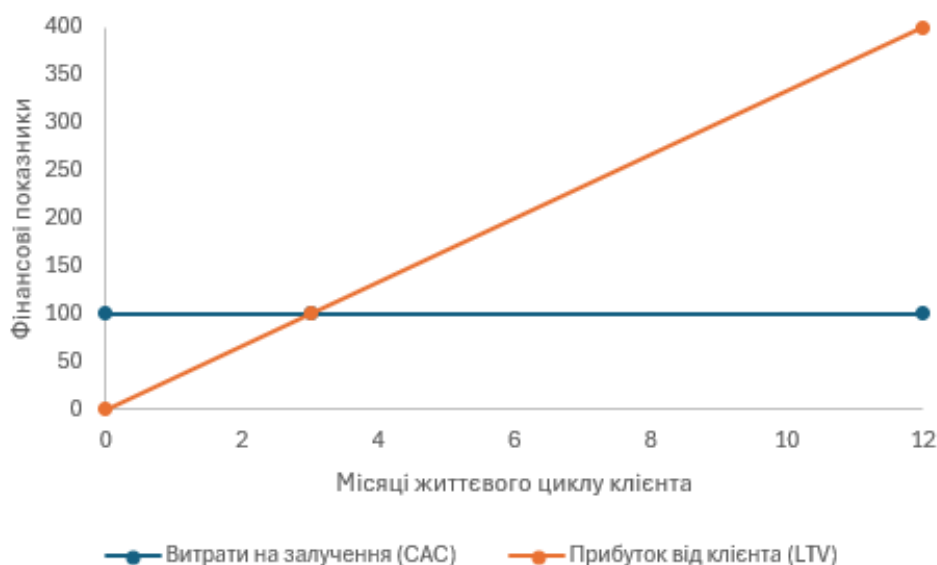


Рисунок 1.1 – Схема концептуального співвідношення метрик CAC та LTV у часовому вимірі життєвого циклу клієнта

Проте під час аналізу предметної галузі було виявлено суттєве системне обмеження. Традиційна банківська звітність фіксує закриття карткових рахунків із затримкою. Статистичні дані надходять виключно постфактум. На цьому етапі користувач вже повністю виводить свої активи. Відповідно, виникла необхідність у розробці предиктивного підходу. Алгоритм повинен ідентифікувати ранні ознаки зниження активності клієнта задовго до фактичного розірвання договору.

Для формалізації цієї загальної бізнес-проблеми було застосовано методи інтелектуального аналізу даних. Її зведено до строгої математичної задачі бінарної класифікації. З цією метою для кожного окремого клієнта було сформовано багатовимірний вектор ознак:

$$x_i = \{x_{i1}, x_{i2}, x_{i3}, \dots, x_{in}\}. \quad (1.1)$$

До структури цього вектора було включено не лише баланс та вік клієнта. Параметри розширено даними про транзакційну активність,

кількість підключених банківських продуктів та географічні мітки. Цільову змінну було формалізовано за правилами бінарної класифікації: Значення «1» вказує на фактичний відтік клієнта, а «0» – на його лояльність.

$$y_{i1} \in \{0,1\}. \quad (1.2)$$

Значення «1» вказує на фактичний відтік клієнта, а «0» – на його лояльність. Проте під час первинного навчання базових моделей було виявлено системну проблему. Вона полягала в асиметричному розподілі класів. Частка клієнтів, які припинили співпрацю з банком, становить лише близько 20% від загального обсягу вибірки. Це свідчить про значний дисбаланс датасету [4] (таблиця 1.1).

Таблиця 1.1 – Статистичний розподіл цільового класу у вихідному наборі даних до етапу аугментації та балансування

Значення цільового класу	Опис клієнтського профілю	Кількість записів	Частка від загальної вибірки (%)
0	Лояльні клієнти	10 400	80,0 %
1	Клієнти, схильні до відтоку	2 600	20,0 %
Усього	Загальний обсяг вибірки	13 000	100,0 %

Застосування класичної логістичної регресії до таких даних продемонструвало ефект парадоксу точності (Accuracy Paradox). Алгоритм класифікував переважну більшість записів до мажоритарного класу. Це дозволило отримати високий показник загальної точності на рівні 80%. Однак практична цінність такої моделі є мінімальною. Класифікатор повністю ігнорував цільовий міноритарний клас.

Було встановлено, що в умовах дисбалансу стандартні функції втрат працюють неефективно. Це зумовило необхідність перегляду підходів до моделювання. Виникла потреба у застосуванні алгоритмів, стійких до

асиметрії даних. Головним критерієм оптимізації стала не загальна точність, а максимізація метрики повноти (Recall) для коректного виявлення профілів із високим фінансовим ризиком.

1.2 Порівняльний аналіз існуючих засобів та комерційних рішень

Огляд було розпочато з аналізу класичних підходів до вирішення проблеми відтоку клієнтів у банківському секторі. Спочатку було досліджено традиційний статичний RFM-аналіз (Recency, Frequency, Monetary). Оцінка його алгоритму показала низьку ефективність в умовах сучасної динаміки транзакцій. Цей метод здійснює групування клієнтів за базовими історичними зрізами. Алгоритм враховує виключно минулі періоди. Якщо великий клієнт раптово знизить активність посеред місяця, RFM-метрика не виявить цього одразу. Зміни будуть зафіксовані лише під час наступного планового перерахунку вибірки. Така затримка у часі призводить до втрати ліквідності банку.

Також було оцінено доцільність застосування аналізу виживання (Survival Analysis). Цей метод базується на кривих Каплана-Маєра та моделі пропорційних ризиків Кокса [5]. Математичний апарат розраховує ймовірність збереження активності клієнта до певного моменту часу. Однак практичне тестування методу на реальних масивах транзакцій виявило суттєві обмеження. Фінансові потоки користувачів характеризуються високим ступенем нелінійності. Вони містять значну кількість статистичного шуму, сезонних флуктуацій та валютних коливань. Апроксимація таких динамічних процесів класичними статистичними розподілами є обчислювально неефективною. У результаті подібні моделі швидко втрачають здатність до узагальнення (рисунок 1.2).

Наступним етапом стало дослідження систем машинного навчання. Було проаналізовано готові хмарні модулі предиктивної аналітики. До розгляду включено рішення від лідерів ринку: Salesforce CRM (Einstein AI),

SAS Customer Intelligence та SAP. Ці платформи мають розвинений графічний інтерфейс. Вони побудовані на потужній інфраструктурі. Проте їх практичне застосування має низку обмежень. Такі алгоритми потребують складного технічного налаштування.

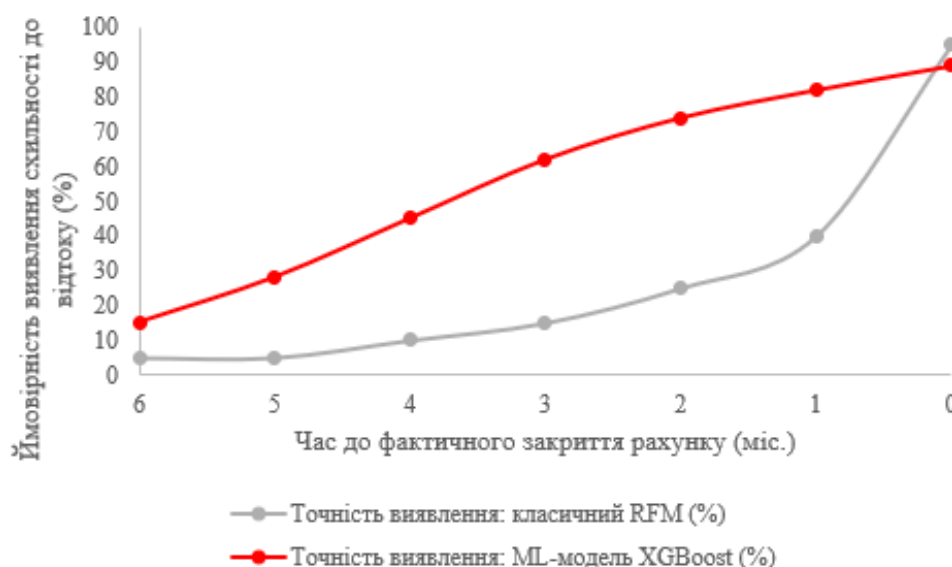


Рисунок 1.2 – Графік порівняння точності виявлення відтоку між класичним RFM-аналізом та базовою ML-моделлю на часовому інтервалі

Головним недоліком цих систем є проблема «чорної скриньки» (Black Box). Вона притаманна більшості пропрієтарних екосистем. Корпоративне програмне забезпечення генерує фінальний скоринговий бал ризику. При цьому внутрішні ваги ознак та логіка роботи алгоритмів залишаються прихованими. Для банківського сектору такий підхід є неприйнятним. Використання непрозорих моделей у ризик-менеджменті ускладнює аудит результатів. Прозорість кожної предикції є обов'язковою вимогою для прийняття фінансових рішень.

Іншою суттєвою проблемою є відсутність адаптації комерційних рішень до макроекономічних умов Східної Європи та України. Закордонні модулі оптимізовані переважно для стабільних одновалютних ринків. Для вирішення цього завдання під час розробки власного веб-застосунку було

реалізовано алгоритм обробки мультивалютності. Клієнти банків часто використовують рахунки одночасно в UAH, USD та інших валютах. Стандартне готове програмне забезпечення не завжди здатне коректно реагувати на курсові коливання. Це призводить до генерації хибних математичних аномалій. Додатковим недоліком таких систем є висока вартість ліцензування. Для локального банку їх впровадження є економічно недоцільним (таблиця 1.2).

Таблиця 1.2 – Порівняльна матриця характеристик готових CRM-систем та власного рішення за критеріями прозорості, вартості та локалізації

Критерій порівняння	Комерційні рішення (Salesforce CRM, SAS, SAP)	Розроблена система (Власний ML-конвеєр)
Прозорість логіки (Інтерпретованість)	Синдром «чорної скриньки». Видається лише фінальний бал ризику без пояснення причин.	Повна прозорість (Explainable AI). Виведення Топ-3 факторів ризику для кожного клієнта.
Вартість впровадження та підтримки	Високі щомісячні ліцензійні платежі за кожного користувача системи.	Мінімальні витрати. Оплата лише за оренду серверних потужностей для хостингу Flask.
Локалізація та мультивалютність	Жорстка прив'язка до стабільних економік (EUR, USD). Складнощі з крос-курсами.	Вбудована динамічна конвертація семи валют (UAH, PLN, RON тощо) безпосередньо в дашборді.
Гнучкість ML-пайплайну	Закриті алгоритми. Неможливо вручну змінити гіперпараметри чи логіку балансування класів.	Повний контроль над ColumnTransformer та параметрами XGBoost (глибина дерев, ваги).
Вимоги до апаратного забезпечення	Вимагають потужних інтеграцій та часто розгортання важких корпоративних баз даних.	Легка клієнт-серверна архітектура. Фронтенд працює у будь-якому сучасному браузері.

Зазначені критичні недоліки існуючих комерційних платформ зумовили необхідність створення власної програмної системи. Ідею інтеграції готових API було відхилено. Альтернативним підходом стала розробка незалежного предиктивного модуля на базі алгоритму XGBoost. Попередня обробка даних реалізована за допомогою інструменту ColumnTransformer. Архітектуру системи побудовано за принципом повного контролю над конвеєром обробки. Кожен етап функціонування жорстко регламентовано. Це охоплює всі процеси: від стандартизації ознак до асинхронної взаємодії із сервером через JavaScript-метод `fetch()`.

Такий підхід дозволив досягти показників повноти (Recall), які відповідають рівню комерційних аналогів. Додатковою перевагою стала інтеграція інструментів Explainable AI (XAI). Застосування XAI вирішує проблему непрозорості («чорної скриньки») класичних моделей. Система генерує наочні візуалізації за допомогою бібліотеки Chart.js. Графіки чітко відображають вагові коефіцієнти конкретних бізнес-факторів ризику для кожного профілю клієнта.

1.3 Огляд класичних методів машинного навчання для аналізу багатовимірних фінансових даних.

Експериментальне дослідження було розпочато з тестування базових математичних алгоритмів. Першою моделлю для оцінки стала логістична регресія (Logistic Regression). Механіка цього алгоритму базується на побудові лінійної гіперплощини у багатовимірному просторі ознак [6]. Метою є математичне розділення лояльних клієнтів та користувачів із високим ризиком відтоку. Цей процес реалізується через згортання лінійної комбінації ваг та ознак за допомогою сигмоїдної функції. Вихідний сигнал трансформується у діапазон ймовірностей від нуля до одиниці. Теоретично ця концепція є математично обґрунтованою.

Проте практичне тестування моделі на масиві з 13 000 банківських записів продемонструвало її неефективність. Класифікатор показав нульове значення за метрикою повноти (Recall). Головною причиною алгоритмічного збою став дисбаланс класів вибірки. Логістична регресія почала прогнозувати константу мажоритарного класу. Алгоритм фіксував 80% загальної точності (Accuracy). Усім клієнтам автоматично присвоювався статус лояльних користувачів. Фактичні профілі з ризиком відтоку система повністю ігнорувала. Спроби оптимізації за допомогою балансування ваг класів не дали позитивного результату. Було встановлено, що лінійні моделі не здатні виявляти складні, нелінійні транзакційні аномалії. Ризик клієнтського відтоку часто формується не прямим зменшенням балансу, а прихованими структурними змінами в транзакційній активності.

$$P(y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \dots + \beta_n X_n)}}, \quad (1.3)$$

де $P(y = 1|X)$ – ймовірність того, що клієнт належить до цільового класу (схильний до відтоку) за заданого вектора ознак X ;

e – основа натурального логарифма (експонента);

β_0 – вільний член (зсув) рівняння логістичної регресії;

$\beta_1 + \dots + \beta_n$ – вагові коефіцієнти моделі, які визначають ступінь впливу кожної ознаки на фінальний ризик;

$X_1 + \dots + X_n$ – фактичні числові значення ознак конкретного клієнта (наприклад, баланс, кількість транзакцій тощо).

З огляду на обмеження лінійних підходів, наступним етапом стало застосування дерева рішень (Decision Tree). Цей алгоритм має принципово іншу архітектуру. Він виконує рекурсивне розщеплення простору ознак. Пошук оптимальних точок поділу здійснюється за критерієм мінімізації інформаційної ентропії або індексу Джині (Gini impurity) [6]. У результаті

формується багаторівнева ієрархія логічних умов. Теоретично така структура повинна забезпечувати повну інтерпретованість предиктивних рішень на кожному кроці розгалуження.

Однак під час експериментів було виявлено критичну проблему масштабування. За умови відсутності штучних обмежень на максимальну глибину розгалуження (параметр `max_depth = None`), алгоритм втрачає здатність до узагальнення. Модель повністю адаптується під тренувальний набір даних. Відбувається точне запам'ятовування параметрів кожного запису, включаючи статистичний шум. Точність на навчальній вибірці досягає 100%. Це свідчить про глибоке перенавчання (Overfitting) класифікатора (рисунок 1.3). За таких умов модель стає непридатною для аналізу нових, раніше невідомих даних.

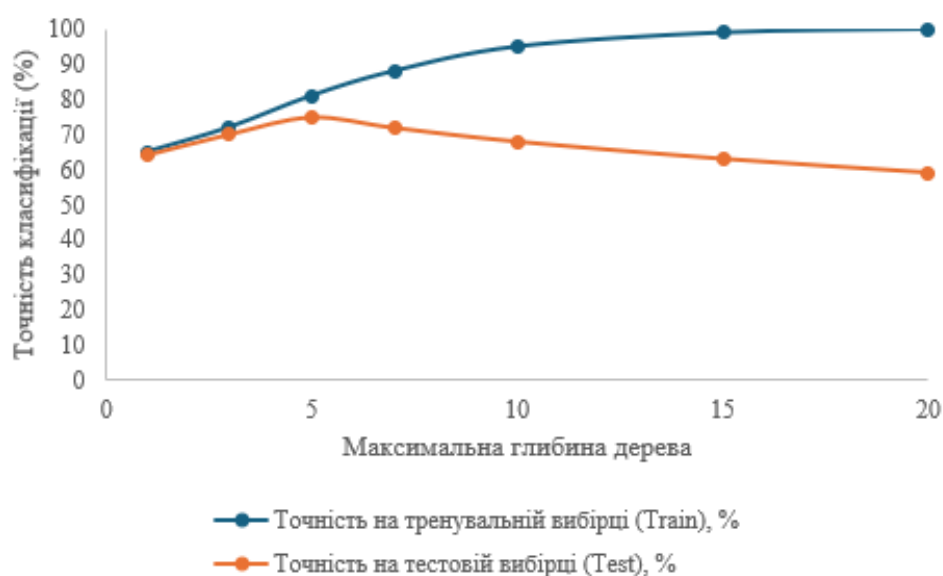


Рисунок 1.3 – Графік залежності точності моделі Decision Tree від глибини дерева

Під час тестування моделі на відкладеній вибірці було зафіксовано суттєве зниження її ефективності. Алгоритм продемонстрував аномальну

чутливість до статистичного шуму. Навіть незначні курсові коливання хибно ідентифікувалися як критичний сигнал до закриття рахунку клієнтом. Аналіз цих відхилень дозволив зробити висновок щодо обмеженої ефективності базових одиночних алгоритмів. Вони не здатні адекватно моделювати багатовимірну динаміку сучасних банківських процесів.

Для забезпечення стійкості системи та збереження її здатності до узагальнення виникла необхідність застосування складніших архітектур. Оптимальним рішенням стало об'єднання множини слабких класифікаторів у єдиний градієнтний ансамбль.

2 ОБҐРУНТУВАННЯ НАПРЯМКУ ДОСЛІДЖЕННЯ ТА РОЗРОБКА КОНВЕЄРА МАШИННОГО НАВЧАННЯ

2.1 Математичне обґрунтування вибору ансамблевих методів

Висока дисперсія та схильність одиночних алгоритмів до перенавчання зумовили перехід до побудови комбінованих систем. Окреме дерево рішень характеризується низькою стійкістю до незначних змін у вхідних транзакційних даних. Це призводить до структурної нестабільності моделі та зниження загальної точності. Тому архітектуру аналітичного конвеєра було підпорядковано ідеї об'єднання слабких класифікаторів (weak learners) у комплексну предиктивну систему. Теоретичне підґрунтя ансамблевих методів забезпечує математично обґрунтоване зниження помилки прогнозування. Для вибору оптимального підходу було досліджено три класичні парадигми: беггінг, бустинг та стекінг.

Першим було протестовано метод беггінгу на базі алгоритму випадкового лісу (Random Forest) [7]. Його математичний апарат передбачає паралельне навчання множини глибоких дерев (рисунок 2.1).

Навчання відбувається на випадкових підмножинах датасету із застосуванням процедури агрегації бутстрепу (Bootstrap Aggregation). Під час програмної реалізації базові моделі тренувалися абсолютно ізольовано. Фінальний прогноз формувався шляхом мажоритарного голосування. Застосування цього підходу дозволило суттєво знизити загальну дисперсію (Variance) моделі. Горизонтальний розподіл похибок стабілізувався. Проте подальший аналіз виявив алгоритмічне обмеження. Випадковий ліс виявився занадто інертним для подолання високого зміщення (Bias) в умовах асиметричної вибірки. Модель демонструвала надмірно консервативні результати.

З огляду на обмеження беггінгу, стратегію дослідження було змінено на вивчення парадигми послідовного бустингу. Спочатку було оцінено

класичний алгоритм AdaBoost [8]. Його механіка полягає в ітеративній побудові ансамблю. На кожному наступному кроці алгоритм примусово фокусується на тих спостереженнях, які попередній класифікатор обробив із помилкою (рисунок 2.1). Цей процес реалізується шляхом динамічного масштабування ваг об'єктів.

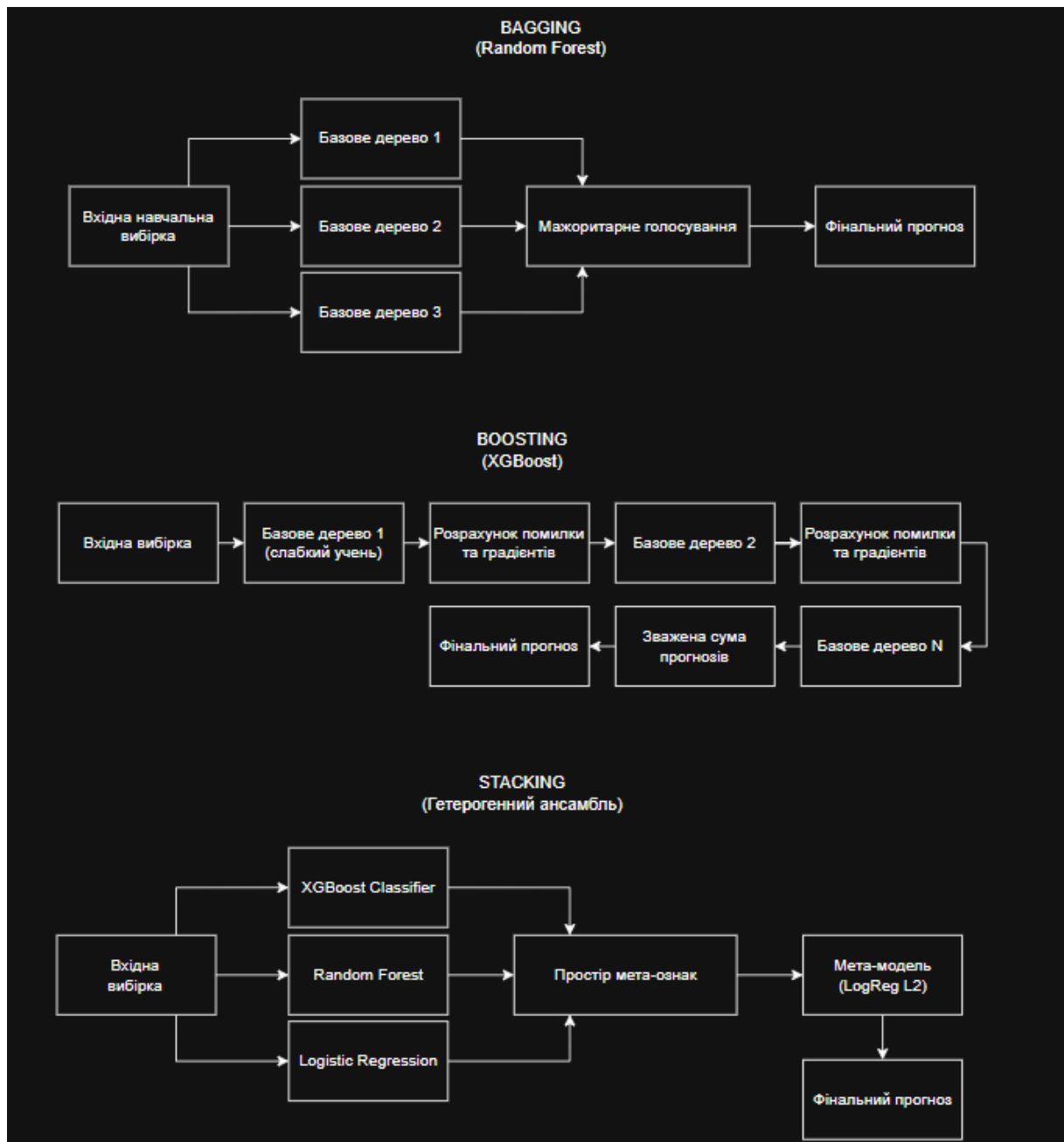


Рисунок 2.1 – Концептуальна схема відмінностей у формуванні архітектур Bagging, Boosting та Stacking

Однак практичне застосування методу продемонструвало його критичну вразливість до статистичних викидів. Наявність у вибірці клієнтів із нетиповими фінансовими показниками спричиняла деградацію моделі. Алгоритм надмірно адаптувався під шумовий компонент. Здатність до узагальнення на тестовому бенчмарку була повністю втрачена.

Вирішення цієї проблеми стало можливим після інтеграції математичного апарату екстремального градієнтного бустингу (XGBoost) [9], [10]. Його обчислювальна логіка суттєво відрізняється від попередників. Мінімізація функції втрат досягається шляхом послідовного додавання базових дерев, які прогнозують антиградієнт загальної похибки системи (рисунок 2.1).

Під час оптимізації цільової функції (Objective Function) виникла необхідність точного налаштування параметрів регуляризації для запобігання перенавчанню.

$$\Omega(f) = YT + \frac{1}{2}\lambda \sum_{j=1}^T w_j^2, \quad (2.1)$$

де $\mathcal{L}^{(t)}$ – загальне значення цільової функції ансамблю на поточній t -ій ітерації бустингу;

n – сумарний обсяг навчальної вибірки (загальна кількість клієнтів);

$l(\dots)$ – диференційовна функція втрат, яка жорстко вимірює розбіжність між реальністю та прогнозом;

y_i – фактичне значення цільової змінної для i -го клієнта (1 – відтік, 0 – лояльність);

$\hat{y}_i^{(t-1)}$ – сумарний накопичений прогноз ансамблю для i -го клієнта за попередні $t - 1$ кроків;

$f_t(x_i)$ – прогноз нового (доданого на поточному кроці) t -го дерева для вектора ознак i -го клієнта;

$\Omega(f_t)$ – регуляризаційний цензор, який штрафує архітектуру за надмірну складність і блокує перенавчання.

$$\Omega(f) = \Upsilon T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2, \quad (2.2)$$

де Υ – коефіцієнт складності (фіксований штраф за створення кожного додаткового листка в дереві);

T – загальна кількість термінальних вузлів (листіків) у новому дереві f_t ;

λ – коефіцієнт L2-регуляризації, який згладжує екстремальні значення ваг;

w – вага (прогнозне значення) j -го листка дерева.

Дослідження математичного апарату цільової функції стало одним із ключових етапів роботи. Доданок $\Omega(f_t)$ виконує функцію контролю за структурною складністю моделі. Цей параметр застосовує штрафні санкції за надмірну кількість листків дерева T та великі значення вагових коефіцієнтів w . За відсутності даного компонента алгоритм швидко переходив до стану перенавчання (Overfitting). Для прискорення процесу мінімізації функції втрат було застосовано оптимізаційний підхід XGBoost. Він базується на градієнтному наближенні другого порядку з використанням розкладу Тейлора.

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^n \left[l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t), \quad (2.3)$$

де $\approx$ – знак наближеної рівності, зумовлений відсіканням членів вищих порядків у розкладі Тейлора;

g_i – градієнт (перша похідна) функції втрат по попередньому прогнозу $\hat{y}_i^{(t-1)}$, що вказує вектор найшвидшого зменшення системної помилки;

h_i – гессіан (друга похідна) функції втрат по попередньому прогнозу $\hat{y}_i^{(t-1)}$, який визначає кривизну простору помилок і радикально прискорює збіжність алгоритму; інші позначення $\mathcal{L}^{(t)}, n, y_i, f_t(x_i), \Omega(f_t)$ відповідають параметрам базового рівняння цільової функції.

Для оптимізації алгоритму було формалізовано першу похідну функції втрат (градієнт g_i) та другу похідну (гессіан h_i). Це забезпечило точніше визначення напрямку спуску на поверхні похибок. Швидкість збіжності конвеєра суттєво зросла. Однак під час адаптації функції втрат до фінансової специфіки розрахунків виникла проблема обчислювальної нестабільності. На великих значеннях балансу рахунків фіксувалася критична помилка числового переповнення (Numerical Overflow). Для вирішення цього завдання було розраховано математичні обмеження градієнтів. Їхні екстремальні значення примусово обрізалися за допомогою налаштування гіперпараметрів `max_delta_step` та `min_child_weight`.

Наступним етапом дослідження стала побудова дворівневої гетерогенної системи на базі ансамблевого методу стекінгу (Stacking Classifier) [11]. На першому рівні було інтегровано три алгоритми різної математичної природи. До них увійшли оптимізований XGBoost, випадковий ліс (Random Forest) та зважена логістична регресія. Вихідні прогнози ймовірностей цих алгоритмів сформували новий простір ознак. Цей масив було передано на вхід мета-класифікатора. У ролі фінальної мета-моделі застосовано логістичну регресію з L2-регуляризацією.

Під час реалізації цієї архітектури було виявлено явище витoku даних (Data Leakage). Первинна версія мета-класифікатора демонструвала аномальну точність на рівні 99%. Проте на незалежній тестовій вибірці ефективність алгоритму різко падала. Причиною стало передавання прихованих маркерів правильних відповідей від базових моделей під час навчання на спільних даних. Проблему було усунуто шляхом імплементації схеми перехресної перевірки StratifiedKFold із розбиттям на 5 блоків. Для цього використано функцію `cross_val_predict`. Завдяки такій ізоляції

прогнози для мета-рівня генерувалися виключно на невідомих для базових моделей сегментах. Точність показників стабілізувалася. Архітектура ансамблю стала алгоритмічно коректною.

У таблиці 2.1 наведена матриця математичних переваг та системних недоліків досліджених ансамблевих парадигм.

Таблиця 2.1 – Зведена матриця математичних переваг та системних недоліків досліджених ансамблевих парадигм

Ансамблева парадигма	Математичні та архітектурні переваги	Системні недоліки та обмеження
Bagging	Радикальне зниження дисперсії (variance) моделі. Висока стійкість до викидів та шумових аномалій завдяки паралельному навчанню незалежних дерев. Не потребує складної оптимізації гіперпараметрів.	Інертність до сильно зміщених вибірок (bias). Нездатність якісно розпізнавати міноритарний клас (клієнтів-втікачів) у незбалансованих банківських транзакціях.
Boosting	Адаптивне зниження зсуву. Градієнтне наближення другого порядку (через гессіан) прискорює збіжність. Вбудована L1/L2 регуляризація блокує перенавчання структури.	Гіперчутливість до випадкових шумів на пізніх ітераціях. Вимагає жорсткого тюнінгу параметрів (learning_rate, max_depth). Ризик числового переповнення при ненормованих ознаках.
Stacking	Здатність вловлювати складні нелінійні патерни різної природи через комбінацію дерев та регресій. Формування якісного простору мета-ознак для фінального класифікатора.	Колосальний ризик витоку даних (Data Leakage) при неправильній крос-валідації. Висока обчислювальна складність та ресурсоемність при розгортанні на сервері.

2.2 Методика попередньої обробки та синтетичної аугментації даних

Фінальний масив даних для навчання було розширено до 13 000 записів. Це забезпечило достатній обсяг інформації для тренування ансамблів. Початковий набір складався з фінансових показників європейського ринку. Ці дані не відображали специфіку локальної макроекономічної ситуації. Застосування вибірки з клієнтськими балансами у десятки тисяч євро призвело б до формування некоректних предиктивних закономірностей. Тому було проведено комплексне макроекономічне коригування датасету. Здійснено розрахунок регіональних індексів купівельної спроможності. У структуру даних інтегровано девальваційні коефіцієнти для країн Східної Європи. Фінансові показники, зокрема залишки на рахунках та кредитні ліміти, було перераховано з урахуванням реалій вітчизняного банківського сектору. Отримані значення набули репрезентативного характеру (таблиця 2.2).

Таблиця 2.2 – Специфікація та типи вхідних ознак датасету після регіональної адаптації

Назва ознаки (Feature)	Тип даних	Бізнес-опис атрибута	Приклад значення (адаптовано)
CreditScore	Числовий (int)	Кредитний рейтинг клієнта, розрахований внутрішнім скорингом банку.	650, 712, 800
Region-UA	Категоріальний (str)	Макрорегіон обслуговування після локалізації. Підлягає One-Hot Encoding.	Київ, Захід, Центр
Gender	Бінарний (int)	Стать клієнта (кодована: 1 – чоловік, 0 – жінка).	0, 1
Age	Числовий (int)	Вік клієнта. Важливий тригер для виявлення консервативних профілів.	34, 45, 62

Продовження таблиці 2.2

Назва ознаки (Feature)	Тип даних	Бізнес-опис атрибута	Приклад значення (адаптовано)
Tenure	Числовий (int)	Кількість повних років, протягом яких клієнт обслуговується в банку.	2, 5, 10
Balance_UAH	Числовий (float)	Поточний залишок коштів на рахунках, перерахований у гривню з урахуванням інфляційного коефіцієнта.	45 230.50, 120 400.00
NumOfProducts	Числовий (int)	Кількість підключених банківських продуктів (картки, депозити, кредити).	1, 2, 3
HasCrCard	Бінарний (int)	Наявність активної кредитної картки (1 – так, 0 – ні).	0, 1
IsActiveMember	Бінарний (int)	Індикатор транзакційної активності за останні 30 днів.	0, 1
EstimatedSalary_UAH	Числовий (float)	Оціночний річний дохід клієнта, адаптований під макроекономіку Східної Європи.	350 000.00, 800 000.00
Target_Churn	Бінарний (int)	Цільова змінна. 1 – рахунок закрито (відтік), 0 – клієнт залишився.	0, 1

Незважаючи на проведену адаптацію, набір даних потребував додаткового структурування [12]. Наявність текстових категорій унеможливлювала прямий математичний аналіз ознак. Для перетворення категоріальних змінних у числовий формат було застосовано метод унітарного кодування (One-Hot Encoding). Програмну реалізацію виконано за допомогою класу OneHotEncoder з бібліотеки Scikit-learn [13]. Текстові атрибути, такі як тип картки або стать клієнта, було трансформовано у розріджені бінарні вектори.

Внаслідок цього загальна кількість ознак суттєво зростає. Виникла проблема мультиколінеарності. Це явище негативно впливає на стабільність та якість побудованої моделі. Для усунення лінійної залежності між атрибутами було застосовано параметр `drop='first'`. Такий підхід дозволив автоматично видалити надлишкові базові стовпці.

Наступним етапом обробки стало вирішення проблеми різного масштабу числових змінних. Діапазони значень відрізнялися на кілька порядків. Наприклад, баланси рахунків вимірювалися тисячами, тоді як вік клієнтів – одиницями. За відсутності масштабування показники з великими абсолютними значеннями домінують під час розрахунку градієнтів. Це спотворює процес навчання алгоритмів. Для стабілізації даних було обрано метод Z-score нормалізації за допомогою класу `StandardScaler` (рисунок 2.2).

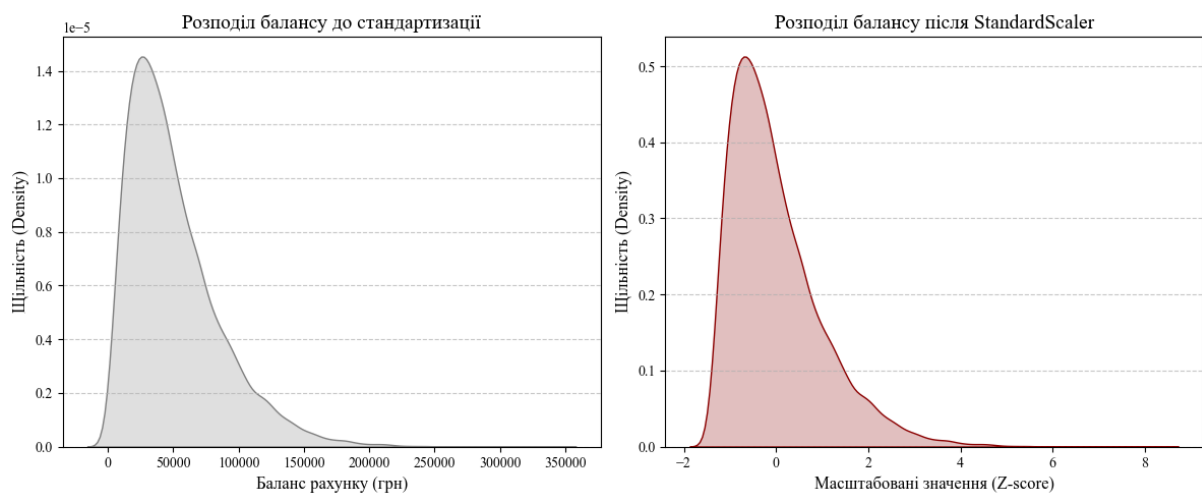


Рисунок 2.2 – Зміна щільності розподілу числових ознак до та після Z-score нормалізації

Цей алгоритм центрує кожну змінну навколо нуля. Дисперсія при цьому зводиться до одиниці:

$$Z = \frac{x - \mu}{\sigma}, \quad (2.4)$$

де μ – середнє значення;

σ – стандартне відхилення.

Під час виконання методу `fit_transform()` було зафіксовано помилку виконання скрипту. Причиною стала наявність пропущених значень (NaN) у масиві даних. Ці пропуски виникли внаслідок технічних збоїв під час експорту транзакційної історії з банківських баз. Пряме видалення неповних записів було визнано недоцільним. Такий крок призвів би до втрати значної частини репрезентативної вибірки.

Для збереження обсягу даних було реалізовано механізм автоматичного відновлення пропусків. Використано клас `SimpleImputer`. Пропущені значення в кожному окремому стовпці замінено відповідними медіанними показниками. Це забезпечило цілісність масиву без суттєвого викривлення його статистичного розподілу.

Вибірка характеризувалася значною класовою асиметрією. Кількість лояльних клієнтів перевищувала кількість користувачів із ризиком відтоку в чотири рази. Для усунення дисбалансу було застосовано метод синтетичної аугментації даних. Від підходу простого дублювання записів міноритарного класу було вирішено відмовитися. Такий метод зазвичай призводить до надмірного перенавчання моделей.

Натомість було інтегровано алгоритм SMOTE (Synthetic Minority Over-sampling Technique) [14]. Його механіка базується на методі k -найближчих сусідів(k -NN).

Алгоритм генерує нові синтетичні, проте статистично репрезентативні профілі клієнтів цільового класу [15]. Генерація відбувається у міжвузловому просторі існуючих спостережень. У результаті співвідношення класів було вирівняно до пропорції 50:50. Це забезпечило збалансовану основу для подальшого тренування ансамблевих моделей машинного навчання.

2.3 Багатофакторна оптимізація гіперпараметрів за допомогою GridSearchCV

Процес ручного налаштування гіперпараметрів виявився малоефективним. Для його автоматизації було використано інструмент GridSearchCV. Було сформовано багатовимірний простір можливих значень для ключових параметрів ансамблю. Алгоритм виконав перебір доступних комбінацій. Оцінка якості кожної конфігурації здійснювалася за допомогою п'ятикратної перехресної перевірки (крос-валідації). Цей етап потребував значних обчислювальних ресурсів. Проте застосування такого підходу дозволило визначити глобальний оптимум параметрів. Загальна якість прогнозування покращилася.

Під час початкових ітерацій навчання базові дерева рішень виявили схильність до надмірної структурної складності. Це призводило до явища перенавчання. Модель ідеально запам'ятовувала навчальну вибірку. При цьому метрики якості на тестових даних різко деградували. Для запобігання цьому було застосовано параметр `max_depth`. Максимальну глибину дерев було жорстко обмежено п'ятьма рівнями. Дерева стали менш розгалуженими. Стабільність прогнозів підвищилася.

Наступним кроком стала оптимізація швидкості навчання. Параметр `learning_rate` було зменшено до 0.05. Одночасно кількість базових дерев (`n_estimators`) було розширено до 200 ітерацій. Крок градієнтного спуску став мінімальним. Збіжність алгоритму стала більш плавною. Для додаткового підвищення стійкості моделі було застосовано метод L2-регуляризації. В налаштуваннях XGBoost було збільшено значення параметра `reg_lambda`. Це запровадило математичний штраф за формування надмірно великих вагових коефіцієнтів. Складність ансамблю зменшилася. Ризик перенавчання мінімізовано, а здатність алгоритму до узагальнення значно зросла (таблиця 2.3).

Таблиця 2.3 – Простір пошуку гіперпараметрів оптимізації GridSearchCV для градієнтного ансамблю

Гіперпараметр	Діапазон пошуку (Grid)	Фінальне оптимальне значення	Математична логіка впливу
max_depth	[3, 5, 7, 10]	5	Блокує перенавчання (Overfitting). Обмежує кількість логічних розщеплень дерева.
n_estimators	[50, 100, 200, 300]	200	Кількість базових дерев у конвеєрі. Збільшення вимагає пропорційного зменшення швидкості навчання.
learning_rate	[0.01, 0.05, 0.1, 0.2]	0.05	Крок градієнтного спуску. Визначає вагу впливу кожного нового дерева на загальний ансамбль.
subsample	[0.6, 0.8, 1.0]	0.8	Частка клієнтів, яка випадково обирається для побудови одного дерева. Знижує кореляцію між деревами.
reg_lambda (L2)	[0, 0.5, 1.0, 5.0]	1.0	Штрафна санкція за великі вагові коефіцієнти в листках. Стискає дисперсію моделі.

Застосування одного алгоритму бустингу виявилось недостатнім для досягнення цільових метрик. Було прийнято рішення про розробку складного гетерогенного ансамблю. Застосовано архітектуру Stacking Classifier. На першому рівні паралельно ініціалізовано три моделі машинного навчання. До них увійшли оптимізований XGBoost, алгоритм випадкового лісу (Random Forest) та логістична регресія (рисунок 2.3). Ці моделі функціонували в ізольованих процесах.

Математична природа обраних алгоритмів кардинально відрізняється. Дерева рішень ефективно ідентифікують складні нелінійні залежності у транзакційних даних. Логістична регресія оптимізована для пошуку

лінійних закономірностей. Спільне використання цих підходів забезпечило більш збалансований результат класифікації.

```
# Налаштовуємо Гібридний Ансамбль (Stacking) на основі базових моделей
sub_models = [
    ('dt', models['Decision Tree']),
    ('rf', models['Random Forest']),
    ('xgb', models['XGBoost'])
]
models['Гібридний Ансамбль (Stacking)'] = StackingClassifier(
    estimators=sub_models,
    final_estimator=LogisticRegression(class_weight='balanced'),
    cv=3,
    n_jobs=-1
)
```

Рисунок 2.3 – Фрагмент програмного коду ініціалізації архітектури Stacking Classifier

Головна технічна складність виникла на етапі агрегації прогнозів базових моделей. Первинно алгоритми першого рівня передавали на наступний етап жорсткі мітки класів (бінарні значення 0 або 1). Це призвело до втрати інформативності. Мета-модель не могла об'єктивно оцінити рівень впевненості базових класифікаторів. Для вирішення цієї проблеми архітектуру конвеєра було модифіковано. Базові моделі почали генерувати безперервні ймовірності відтоку замість жорстких міток. Ці числові значення сформували новий вектор вхідних даних для фінальної моделі.

Функцію мета-класифікатора було делеговано логістичній регресії з L2-регуляризацією. Цей алгоритм більше не аналізував первинні банківські ознаки. Його єдиною задачею стало визначення оптимальної ваги довіри до кожного з трьох базових алгоритмів у конкретних ризикових ситуаціях. Такий підхід мінімізував загальну дисперсію ансамблю. Система набула високої стабільності. Аналіз графіків похибок підтвердив суттєве та поступове зниження загальної помилки прогнозування.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ

3.1 Проєктування архітектури клієнт-серверного веб-застосунку

Під час проєктування серверної частини було вирішено відмовитися від використання громіздких веб-фреймворків. Ключовими критеріями вибору стали швидкодія та простота реалізації. У результаті було застосовано мікрофреймворк Flask [19]. Цей інструмент забезпечує ефективну взаємодію між Python-бекендом та клієнтською частиною. Сервер відповідає за виконання математичних операцій та обслуговування ML-моделей. Проста структура Flask не вимагає побудови надлишкової архітектури. Це дозволило гнучко налаштувати маршрутизацію запитів.

Для реалізації клієнтської частини (фронтенду) використано базовий JavaScript. Застосовано підхід без використання бібліотек із віртуальним DOM. Основний акцент зроблено на швидкості роботи інтерфейсу. Інформаційний обмін між браузером та сервером побудовано на основі асинхронних запитів `fetch()` [20]. Сторінка не потребує перезавантаження для оновлення даних. Клієнтський скрипт агрегує введені менеджером фінансові параметри. Дані серіалізуються у компактний JSON-об'єкт та миттєво відправляються на сервер. Бекенд приймає цей пакет і передає вектор ознак до натренованої моделі ансамблю. У відповідь генерується та повертається масив ймовірностей.

Головні технічні складнощі виникли на етапі розгортання застосунку. Програмна логіка ендпоінтів була реалізована без затримок. Однак під час налаштування мережевої взаємодії було зафіксовано блокування трафіку на рівні операційної системи. Застосунок успішно запускався, проте HTTP-запити до нього не проходили. Причиною стали обмеження мережевих портів та некоректна робота локальних мережевих адаптерів. Виникла

необхідність низькорівневого коригування мережових налаштувань ОС Windows.

Для усунення проблеми доступу було застосовано утиліту PsExec. Це дозволило отримати привілеї системного рівня для модифікації заблокованих гілок реєстру. Після коригування параметрів роботи віртуального адаптера було відновлено. Блокування пакетів до Flask-сервера припинилося. Клієнтська частина почала стабільно взаємодіяти з бекендом, забезпечивши безперебійну маршрутизацію JSON-запитів (рисунок 3.1).

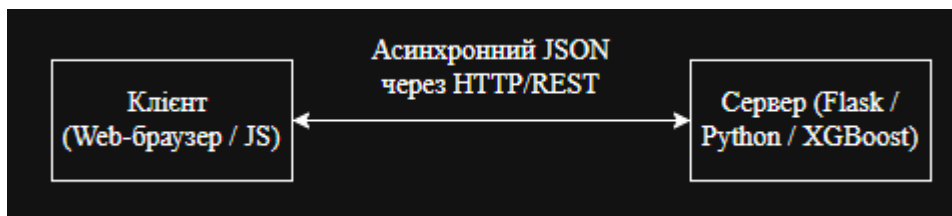


Рисунок 3.1 – Загальна блок-схема клієнт-серверної взаємодії розробленої системи

Після стабілізації обміну даними між клієнтською та серверною частинами було розпочато розробку модуля візуалізації. Необроблені значення ймовірностей від математичної моделі є недостатньо інформативними для кінцевого користувача. Вони потребували наочного графічного представлення. Для вирішення цього завдання інтегровано JavaScript-бібліотеку Chart.js [21]. Цей інструмент забезпечує динамічну побудову діаграм на основі відповідей сервера.

Основна технічна складність на цьому етапі полягала у синхронізації даних. Було необхідно забезпечити коректне оновлення графіка при зміні вхідних параметрів клієнта. Отримання результатів прогнозування налаштовано через REST API. Далі ці значення автоматично передаються у графічний компонент. При будь-якій зміні характеристик система повторно

ініціює предиктивний запит. Висота стовпців на діаграмі оновлюється миттєво (рисунок 3.2). У результаті розроблено інтерактивний інтерфейс. Користувач отримав змогу змінювати параметри профілю та в режимі реального часу відслідковувати динаміку ризику. Це підвищило практичну цінність системи та полегшило інтерпретацію результатів.

```
try {  
  const response = await fetch('/predict', {  
    method: 'POST',  
    headers: { 'Content-Type': 'application/json' },  
    body: JSON.stringify(formData)  
  });  
  
  const result = await response.json();  
}
```

Рисунок 3.2 – Фрагмент програмного коду реалізації асинхронного запиту `fetch()` на клієнтській частині

3.2 Програмна реалізація аналітичного дашборду та інтеграція модулів системи.

Логіку користувацького інтерфейсу побудовано на базі єдиного інтерактивного дашборду. Він забезпечує миттєвий перерахунок ризиків за умови найменшої зміни вхідних параметрів. Розроблений інтерфейс відзначається функціональним дизайном та відсутністю надлишкових елементів. Менеджер банку вводить вік клієнта, тривалість обслуговування та поточний залишок на рахунку у відповідні HTML-поля. Програмний скрипт перехоплює ці події за допомогою набору глобальних слухачів `addEventListener`. Сеанс роботи відбувається без жодного повного перезавантаження веб-сторінки (рисунок 3.3).

Профіль клієнта

Кредитний скоринг

650

Введіть ціле число від 350 (погана кредитна історія) до 850 (ідеальна).

Країна: Україна

Стать: Чоловіча

Вік: 38

Років з банком: 5

Максимум 10 років (обмеження системи).

Баланс на рахунку: 150000 UAH (₴)

Зарплата (щомісячна): 40000 UAH (₴)

Продуктів банку ? 2

Кредитна картка: Є в наявності

Активний клієнт ? Так (Активний)

Проаналізувати ризик ШІ

Рисунок 3.3 – Скріншот робочого інтерфейсу аналітичного дашборду

Складним інженерним завданням стала реалізація механізму динамічної конвертації валют на стороні клієнта. Інтегрована модель машинного навчання натренована виключно на даних у гривневому еквіваленті. Тому всі фінансові показники в інших валютах вимагали математичного приведення до базового формату перед передачею на сервер. У клієнтський функціонал додано підтримку семи валют: UAH, EUR, USD, PLN, RON, CZK та KZT. Коефіцієнти конвертації зберігаються локально. Це дозволяє виконувати розрахунки швидко та без додаткових мережових запитів.

Проте під час тестування було виявлено проблему обчислювальної точності. Движок JavaScript генерує специфічні похибки при виконанні арифметичних операцій із числами з плаваючою комою (Floating-Point

Arithmetic). Після застосування курсового множника часто виникали числа з надлишковою кількістю десяткових знаків. Передача таких даних на бекенд викликала збої під час матричних розрахунків. Для усунення проблеми імплементовано додаткову функцію санітизації (очищення) даних. Алгоритм примусово округлює всі фінансові показники до двох знаків після коми безпосередньо перед формуванням JSON-запиту.

Окремим етапом стала розробка модуля клієнтської валідації. Було враховано ризик введення некоректних параметрів, таких як від'ємний баланс рахунку або нереалістичний вік користувача. Потрапляння таких артефактів до ML-конвеєра гарантовано призводить до генерації хибного прогнозу. Для запобігання цьому впроваджено систему перевірок. Створено жорсткі правила валідації для ключових полів HTML-форми. За наявності помилок скрипт блокує виконання запиту `fetch()`. На екран виводиться системне попередження про необхідність коригування даних (лістинг 3.1).

Реалізація цього функціоналу вимагала чіткої синхронізації форматів обміну даними між бекендом та фронтендом. Усунення конфліктів типізації даних під час тестування дозволило стабілізувати роботу застосунку. Механізм обробки фінансових метрик працює коректно.

Лістинг 3.1 – Модуль клієнтської валідації та динамічної крос-курсової конвертації фінансових ознак

```
// Словник фіксованих крос-курсів для макроекономічної
адаптації
const exchangeRates = {
  'UAH': 40.0,
  'EUR': 0.92,
  'PLN': 4.0,
  'RON': 4.6,
  'CZK': 23.0,
  'KZT': 450.0
};

// Динамічна валідація: клієнт не міг відкрити рахунок до
18 років
function validateTenure() {
```

Продовження лістингу 3.1

```

    const age = parseInt(ageInput.value) || 18;
    const maxTenure = Math.min(10, Math.max(0, age - 18));
    tenureInput.max = maxTenure;

    if (age < 28) {
        tenureHint.innerText = `Максимум ${maxTenure} років
(рахується з 18-річчя).`;
    } else {
        tenureHint.innerText = `Максимум 10 років
(обмеження системи).`;
    }

    // Примусове блокування некоректного вводу
    if (parseInt(tenureInput.value) > maxTenure) {
        tenureInput.value = maxTenure;
    }
}

// Фрагмент обробки та конвертації даних перед відправкою
fetch-запиту
const currentRate = exchangeRates[currentCurrencyCode];

// Нормалізація балансу до базової валюти моделі
const balanceVal =
parseFloat(document.getElementById('Balance').value);
const finalBalance = balanceVal / currentRate;

// Нормалізація зарплати та приведення до річного формату
(вимога моделі)
const salaryVal =
parseFloat(document.getElementById('EstimatedSalary').value);
const finalSalary = (salaryVal / currentRate) * 12;

```

3.3 Впровадження інструментів інтерпретації рішень штучного інтелекту

Під час розробки системи прогнозування виникла необхідність інтерпретації результатів. Видача виключно числового значення ймовірності відтоку є недостатньою. У фінансовій сфері важливо розуміти конкретні причини формування прогнозу.

Використання дворівневої архітектури Stacking Classifier ускладнило процес пояснення. Ця структура поєднує декілька алгоритмів. Внутрішня логіка прийняття рішень стає непрозорою. Утворюється ефект «чорної скриньки». Вплив окремих вхідних параметрів на кінцевий результат визначити вкрай важко.

Первинний план передбачав використання стандартних інструментів Explainable AI [16]. До розгляду бралися бібліотеки SHAP та LIME [17], [18]. Вони дозволяють оцінити внесок кожної ознаки у предикцію. Проте тестування виявило суттєві архітектурні обмеження. Цей підхід критично збільшує час обробки запитів. За умови паралельних звернень сервер виснажує обчислювальні ресурси. Побудова пояснень робить час відгуку системи неприйнятним.

З огляду на це було ухвалено рішення про розробку власного механізму інтерпретації. За основу взято метод аналізу важливості ознак (Feature Importance). Він базується на внутрішніх вагах моделі XGBoost, яка функціонує на першому рівні ансамблю. Було реалізовано алгоритм виділення домінантних факторів ризику для кожного користувача.

Головним завданням стало збереження балансу. Необхідно було поєднати математичну точність пояснень із високою швидкістю системи. Процес екстракції вагових коефіцієнтів було оптимізовано. Його інтегровано безпосередньо у функцію генерації прогнозу (лістинг 3.2). У результаті створено ефективний механізм локальної інтерпретації. Модель повертає не лише відсоток ризику, але й ключові фактори впливу. Система стала прозорою та придатною для аудиту.

Лістинг 3.2 – Реалізація алгоритму Feature Ablation для локальної інтерпретації предикцій моделі

```
# ЛОГІКА ЛОКАЛЬНОГО ПОЯСНЕННЯ (Feature Ablation)
local_impacts = {}
all_cols = numerical_cols + categorical_cols

for col in all_cols:
```

Продовження лістингу 3.2

```

# Створюємо копію даних клієнта і "нейтралізуємо" одну
ознаку
perturbed_data = data.copy()
perturbed_data[col] = baseline_values[col]

perturbed_df = pd.DataFrame([perturbed_data])
# Рахуємо ймовірність без цієї ознаки
perturbed_prob = model_pipeline.predict_proba(perturbed_df)[0][1]

# Різниця показує, наскільки сильно саме цей параметр
змінив фінальний відсоток
impact = base_prob - perturbed_prob
local_impacts[col] = impact

# Сортируємо ознаки за абсолютною силою впливу на поточний
прогноз
sorted_features = sorted(local_impacts.items(), key=lambda
x: abs(x[1]), reverse=True)

# Формуємо найважливіші чинники з перекладом для фронтенду
top_factors = []
for col, imp in sorted_features[:3]:
    weight = round(abs(float(imp)) * 100, 1)
    if weight == 0: weight = 1.0

    top_factors.append({
        'name': translations.get(col, col),
        'weight': weight
    })

```

Запропонований підхід дозволив суттєво скоротити час формування відповіді. Сервер не виконує надлишкових обчислень під час обробки HTTP-запитів.

Для забезпечення зрозумілості інтерфейсу технічні назви змінних було адаптовано. Створено спеціальний словник відповідностей (об'єкт `translations`). Він автоматично замінює англomовні ідентифікатори баз даних (наприклад, `NumOfProducts`) на локалізовані текстові підписи («Кількість продуктів»).

Сформований масив аналітичних даних пакується у формат JSON. Далі він передається на клієнтську частину (фронтенд). Там скрипти JavaScript виконують обробку та візуалізацію отриманої інформації. Для відображення загального рівня ризику використано бібліотеку Chart.js. Побудовано напівкруглу діаграму (Gauge Chart). Її показники динамічно змінюються залежно від розрахованої ймовірності.

Додатково реалізовано графічне відображення причин прогнозу. Побудовано горизонтальну гістограму (Bar Chart). Цей графік виводить Топ-3 фактори, які здійснили найбільший вплив на предиктивне рішення моделі. Завдяки цьому користувач отримує комплексну аналітику. Механізм формування ризику стає повністю інтерпретованим (рисунок 3.4).

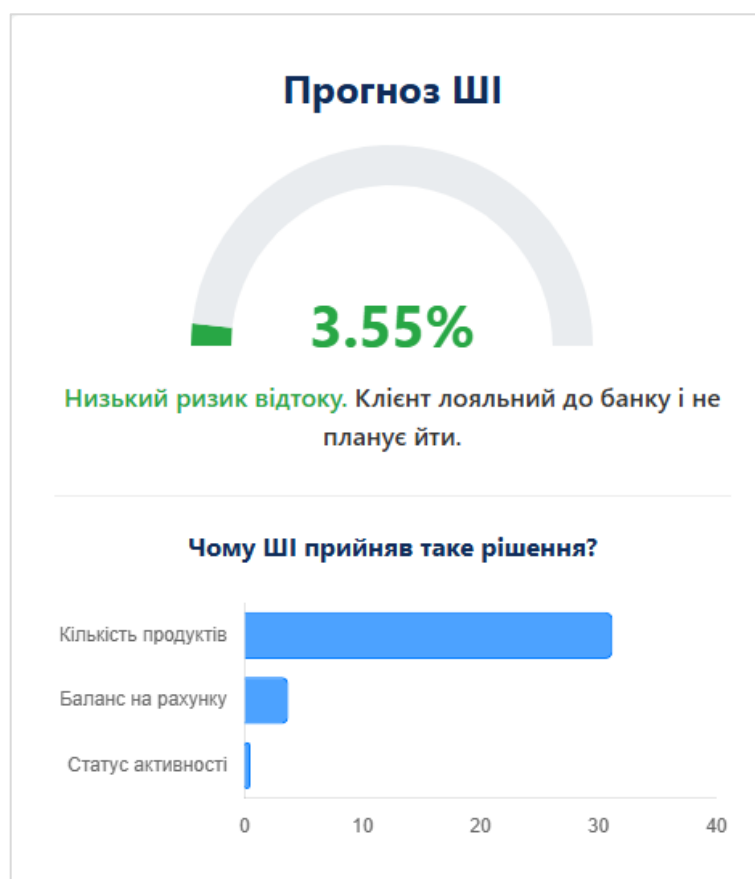


Рисунок 3.4 – Візуалізація загального рівня ризику та 3 індивідуальних фактори відтоку в інтерфейсі системи

3.4 Порівняльний бенчмарк моделей, аналіз матриць помилок та оцінка економічного ефекту

На завершальному етапі було проведено фінальне тестування алгоритмів. Використано відкладену тестову вибірку. Ці дані були повністю ізольовані під час тренування моделей. Такий підхід дозволив об'єктивно оцінити здатність алгоритмів до узагальнення на нових спостереженнях.

Для порівняльного аналізу (бенчмаркінгу) обрано шість алгоритмів. До переліку увійшли: логістична регресія, дерево рішень, випадковий ліс (Random Forest), AdaBoost, XGBoost та власна розробка – дворівнева модель Stacking Classifier. Усі класифікатори обробляли ідентичний масив раніше невідомих транзакцій.

Під час оцінювання загальна точність (Accuracy) не розглядалася як ключовий критерій. Використання виключно цієї метрики є некоректним в умовах дисбалансу класів. Кількість лояльних клієнтів у масиві значно перевищує частку профілів із ризиком відтоку. У такій ситуації базові моделі схильні прогнозувати мажоритарний клас. Це генерує формально високу загальну точність. При цьому здатність виявляти реальний відтік (міноритарний клас) зводиться до нуля.

Для об'єктивної оцінки було застосовано розширений набір метрик. Головний акцент зроблено на показниках, що характеризують якість класифікації меншого класу (Precision, Recall, F1-Score). Це дозволило провести коректне порівняння. За результатами тестування було визначено найефективнішу архітектуру алгоритму для вирішення поставленої бізнес-задачі (таблиця 3.1).

Детальна оцінка роботи моделей стала можливою після побудови матриць помилок (Confusion Matrix). Для візуалізації результатів використано бібліотеку seaborn. Цей етап виявив фактичну алгоритмічну різницю між класифікаторами. Було визначено типи помилок, які генерують різні моделі.

Таблиця 3.1 – Зведений бенчмарк метрик класифікації для досліджених алгоритмів на тестовій вибірці

Назва алгоритму	Accuracy (Точність)	Precision (Точність класу 1)	Recall (Повнота виявлення)	Загальний баланс
Логістична регресія	70.7	38.1	68.7	59.2
Дерево рішень	76.6	45.7	74.9	65.7
Random Forest	83.6	59.4	63.4	68.8
AdaBoost	84	68.5	40.3	64.3
XGBoost	81.8	54.2	72.7	69.6
Stacking Classifier	79.8	50.5	76.7	69

У задачах прогнозування відтоку клієнтів помилки різних типів мають нерівноцінний вплив на бізнес. Хибнопозитивний прогноз (False Positive) виникає, коли система класифікує лояльного користувача як потенційного втікача. Ця помилка не створює значних фінансових проблем. Фінансова установа може надати такому клієнту додатковий бонус. Клієнт просто продовжить користуватися послугами банку з вищою лояльністю.

Натомість хибнонегативна помилка (False Negative) має критичний характер. Вона фіксується тоді, коли модель не виявляє клієнта з реальними намірами закрити рахунок. Ця ситуація є економічно небезпечною. Система втрачає можливість завчасно відреагувати. Відділ утримання не діє, і банк втрачає цінного клієнта.

З огляду на це, загальна кількість правильних прогнозів (Accuracy) не розглядалася як головний критерій. Пріоритетом під час вибору оптимальної моделі стала здатність алгоритму мінімізувати кількість пропущених клієнтів із високим ризиком відтоку (рисунки 3.5).

Під час налаштування гіперпараметрів основну увагу було зосереджено на оптимізації метрики повноти (Recall). Для банківського сектору критично важливо виявляти ризикові профілі на ранніх етапах.

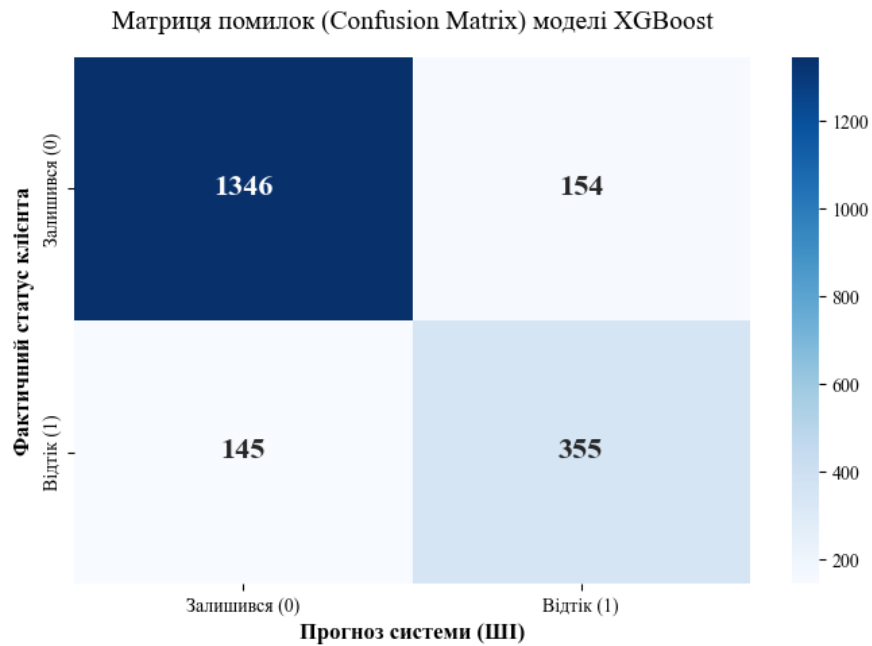


Рисунок 3.5 – Візуалізація матриці помилок (Confusion Matrix) оптимізованої моделі XGBoost

Результати тестування базового дерева рішень та моделі AdaBoost виявилися незадовільними. Ці алгоритми продемонстрували низькі показники повноти. Було зафіксовано велику кількість пропущених випадків фактичного відтоку. Високу ефективність продемонструвала оптимізована модель XGBoost. Значення метрики Recall для неї склало 72,7%.

Цей алгоритм забезпечив найвищий баланс загальних показників. Модель успішно ідентифікувала переважну більшість клієнтів із підтвердженим ризиком розірвання фінансових відносин.

Додатково було розраховано потенційний економічний ефект від впровадження розробленої системи. Базовим параметром стала середня життєва цінність клієнта (Lifetime Value, LTV). Її значення зафіксовано на рівні 50 000 гривень. Впровадження алгоритму забезпечує своєчасне виявлення ризикових профілів. Працівники банку отримують інструмент для превентивного утримання клієнтів. Це дозволяє суттєво зменшити прямі фінансові втрати установи.

У ході виконання роботи було вирішено комплекс інженерних завдань. Проведено математичну обробку масивів даних. Виконано балансування класів за допомогою алгоритму SMOTE. Здійснено багатофакторну оптимізацію моделей та інтеграцію програмних компонентів. Вирішення цих проблем дозволило створити стабільну клієнт-серверну систему. Розроблена модель класифікації характеризується високими метриками якості та має підтверджену практичну цінність для банківської сфери.

ВИСНОВКИ

У кваліфікаційній роботі розв'язано задачу прогнозування відтоку клієнтів банку. Застосовано методи машинного навчання. Пройдено всі етапи від підготовки даних до розробки фінального веб-застосунку. Модель спроектовано, навчено та успішно інтегровано у клієнт-серверну інфраструктуру.

Основні результати роботи:

- результат і повнота виконання завдання на кваліфікаційну роботу, аналіз досягнутих кількісних та якісних показників Завдання виконано в повному обсязі. Оброблено масив із 13 000 фінансових записів. Проведено порівняльний аналіз шести алгоритмів. Через дисбаланс класів оцінка виконувалася за розширеним набором метрик. Найвищу ефективність показала модель XGBoost після налаштування гіперпараметрів (GridSearchCV). Показник повноти (Recall) досяг 72,7%. Це дозволяє стабільно виявляти абсолютну більшість профілів із ризиком відтоку. Впровадження системи мінімізує фінансові збитки банку шляхом превентивного утримання клієнтів;

- співвідношення виконаної розробки з вітчизняними та світовими аналогами Більшість існуючих рішень функціонують як закриті «чорні скриньки». Стандартні методи пояснення (SHAP, LIME) створюють критичне навантаження на сервери. Тому програмно реалізовано власний полегшений модуль інтерпретації на базі аналізу важливості ознак. Він миттєво визначає фактори ризику без збільшення часу обробки запиту. Додатково розробку адаптовано до локальних умов: впроваджено клієнтський алгоритм мультивалютної конвертації. Це гарантує коректну передачу даних на відміну від західних аналогів;

- зв'язок виконаної роботи з науково-дослідними розробками випускової кафедри Отримані результати повністю відповідають вектору досліджень випускової кафедри у сфері проектування інтелектуальних

систем аналізу даних. Практичне використання ансамблевих моделей та методів локальної інтерпретації доповнює існуючу наукову базу. Ці розробки можуть стати основою для створення гібридних систем підтримки прийняття рішень у майбутніх кафедральних проєктах;

- отримані нові наукові результати та рекомендації щодо подальшої роботи Практичною новизною є інтеграція високоточних градієнтних ансамблів із механізмом миттєвого локального пояснення. Система автоматично виокремлює домінуючі фактори ризику для кожного прогнозу. Подальша модернізація передбачає перехід до обробки часових рядів. Рекомендується інтеграція рекурентних нейромереж (LSTM або GRU) для аналізу транзакційної поведінки клієнтів у безперервній динаміці;

- можливість використання матеріалів роботи за призначенням та в навчальному процесі Розроблений клієнт-серверний Flask-застосунок є автономним прототипом. Він придатний для розгортання та експлуатації відділами ризик-менеджменту банків. Програмний код має високу освітню цінність. Окремі компоненти ML-конвеєра доцільно впровадити у лабораторні практикуми з навчальних дисциплін «Машинне навчання» та «Веб-технології».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Defection detection: measuring and understanding the predictive accuracy of customer churn models / S. A. Neslin et al. *Journal of marketing research*. 2006. Vol. 43, no. 2. P. 204–211. URL: <https://doi.org/10.1509/jmkr.43.2.204> (date of access: 10.06.2026).
2. Blattberg R. C., Kim B.-D., Neslin S. A. Database marketing: analyzing and managing customers. Springer, 2008. 876 p.
3. Kumar V., Petersen J. A. Statistical methods in customer relationship management. John Wiley & Sons, 2012. 394 p.
4. Burez J., Van den Poel D. Handling class imbalance in customer churn prediction. *Expert Systems with Applications*. 2009. Vol. 36, no. 3. P. 4626–4636.
5. Hastie T., Tibshirani R., Friedman J. The elements of statistical learning: data mining, inference, and prediction. 2nd ed. Springer, 2009. 745 p.
6. Bishop C. M. Pattern recognition and machine learning. Springer, 2006. 738 p.
7. Breiman L. Random forests. *Machine Learning*. 2001. Vol. 45, no. 1. P. 5–32.
8. Kuhn M., Johnson K. Applied predictive modeling. Springer, 2013. 600 p.
9. Chen T., Guestrin C. XGBoost: a scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016. P. 785–794.
10. XGBoost documentation – xgboost 3.3.0 documentation. *XGBoost Documentation*. URL: <https://xgboost.readthedocs.io/> (date of access: 18.06.2026).
11. Wolpert D. H. Stacked generalization. *Neural Networks*. 1992. Vol. 5, no. 2. P. 241–259.
12. McKinney W. Python for data analysis: data wrangling with Pandas, NumPy, and IPython. 2nd ed. O'Reilly Media, 2017. 544 p.

13. Scikit-learn: machine learning in Python / F. Pedregosa et al. *Journal of Machine Learning Research*. 2011. Vol. 12. P. 2825–2830. URL: <https://scikit-learn.org/stable/> (date of access: 20.06.2026).
14. SMOTE: synthetic minority over-sampling technique / N. V. Chawla et al. *Journal of Artificial Intelligence Research*. 2002. Vol. 16. P. 321–357.
15. Imbalanced-learn: a Python toolbox to tackle the curse of imbalanced datasets in machine learning. *Imbalanced-learn*. URL: <https://imbalanced-learn.org/stable/> (date of access: 21.06.2026).
16. Flask documentation (3.0.x). *Flask*. URL: <https://flask.palletsprojects.com/> (date of access: 22.06.2026).
17. Fetch API – Web APIs | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API (date of access: 23.06.2026).
18. Chart.js documentation: simple yet flexible JavaScript charting. *Chart.js*. URL: <https://www.chartjs.org/docs/latest/> (date of access: 23.06.2026).
19. Molnar C. Interpretable machine learning: a guide for making black box models explainable. *GitHub*. 2022. URL: <https://christophm.github.io/interpretable-ml-book/> (date of access: 24.06.2026).
20. Lundberg S. M., Lee S.-I. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*. 2017. Vol. 30.
21. Ribeiro M. T., Singh S., Guestrin C. "Why should i trust you?": explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference*. 2016. P. 1135–1144.