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 другий (магістерський)

Бустинг в ансамблях нейронних мереж для оцінки ризиків у задачах страхування та фінансів
(тема)

Виконав:
здобувач другого року навчання,
групи СШІм-23-2
Михайло Купріянов
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту
(повна назва освітньої програми)
Керівник проф. Євгеній Бодянський
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти другий (магістерський)

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

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Купріянову Михайлу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Бустинг в ансамблях нейронних мереж для оцінки ризиків у задачах страхування та фінансів

затверджена наказом університету від 21 квітня 20 25 р. № 295Ст

2. Термін подання студентом роботи до екзаменаційної комісії 10 червня 20 25 р.

3. Вихідні дані до роботи Науково-технічна література та публікації з ансамблевих методів машинного навчання, документація алгоритмів ансамблювання нейронних мереж, набори фінансових даних для тренування та тестування моделей оцінки ризиків у задачах страхування та фінансів, програмні засоби для реалізації моделей, метрики оцінювання якості страхування

4. Перелік питань, що потрібно опрацювати в роботі
1) Теоретичні основи ансамблевого бустингу та оцінювання фінансових і страхових ризиків
2) Розробка методу ансамблевого бустингу нейронних мереж для оцінювання ризиків
3) Експериментальне дослідження ансамблевої бустингової моделі
4) Практична інтеграція ансамблевої бустингової моделі в програмний прототип

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	21.04.2025	виконано
2	Вивчення предметної галузі, аналіз літератури	22.04.25–02.05.25	виконано
3	Формулювання мети та задач дослідження	03.05.25–05.05.25	виконано
4	Аналіз існуючих рішень, формалізація задачі	06.05.25–12.05.25	виконано
5	Розробка архітектури нейронних мереж	13.05.25–20.05.25	виконано
6	Реалізація програмної частини	21.05.25–28.05.25	виконано
7	Аналіз результатів, оформлення висновків	29.05.25–31.05.25	виконано
8	Підготовка до подання роботи на перевірку	01.06.25–02.06.25	виконано
9	Перевірка на доброчесність та нормоконтроль	03.06.2025	виконано
10	Оформлення рецензії, доопрацювання зауважень	04.06.25–08.06.25	виконано
11	Подання фінальної версії роботи	09.06.2025	виконано
12	Захист кваліфікаційної роботи	10.06.2025	виконано

Дата видачі завдання 21 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ проф. Євгеній Бодянський
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 60 с., 11 рис., 1 табл., 1 дод., 20 джерел.

АНАЛІЗ ДАНИХ, АНСАМБЛЕВІ НЕЙРОННІ МЕРЕЖІ, БУСТИНГ, КРЕДИТНИЙ СКОРИНГ, МАШИННЕ НАВЧАННЯ, ОЦІНКА РИЗИКІВ, ФІНАНСОВІ ЗАДАЧІ.

Об'єкт дослідження – ансамблеві нейронні мережі для розв’язання задач оцінки ризиків у страхуванні та фінансах.

Предмет дослідження – методи бустингу (boosting) для побудови ансамблів нейронних мереж з метою покращення точності прогнозування фінансових ризиків.

Мета роботи – розробка, теоретичне обґрунтування та дослідження ефективності ансамблевих моделей нейронних мереж на основі методів бустингу для кредитного скорингу у фінансових і страхових задачах.

Методи дослідження – аналіз існуючих наукових та практичних підходів до оцінки ризиків, теоретичне дослідження алгоритмів бустингу, розробка ансамблевих нейромережевих моделей, експериментальна верифікація на даних фінансових установ, оцінювання результатів за допомогою спеціалізованих метрик (ROC-AUC, Precision, Recall, F1-score).

У результаті проведено аналіз сучасних підходів до оцінки кредитного скорингу, визначено переваги та недоліки традиційних підходів та обґрунтовано застосування бустингу для формування ансамблів нейронних мереж. Розроблено архітектуру бустинг-ансамблю, проведено порівняльний аналіз із класичними моделями. Отримані результати підтверджують переваги використання ансамблевих нейромережевих моделей з бустингом в задачах кредитного скорингу, демонструючи високі показники точності прогнозування ризиків та узагальнюючої здатності моделей.

ABSTRACT

Master’s thesis contains: 60 pp., 11 fig., 1 tabl., 1 ann., 20 references.

BOOSTING, CREDIT SCORING, DATA ANALYSIS, ENSEMBLE, , FINANCIAL TASKS, MACHINE LEARNING, NEURAL NETWORKS, RISK ASSESSMENT.

Object of research – ensemble neural networks for solving risk assessment tasks in insurance and finance.

Subject of research – boosting methods for constructing ensembles of neural networks aimed at improving the accuracy of financial risk prediction.

Purpose of the work – development, theoretical substantiation, and study of the efficiency of ensemble neural network models based on boosting methods for credit scoring in financial and insurance tasks.

Research methods – analysis of existing scientific and practical approaches to risk assessment, theoretical study of boosting algorithms, development of ensemble neural network models, experimental verification on data from financial institutions, evaluation of results using specialized metrics (ROC-AUC, Precision, Recall, F1-score).

As a result, an analysis of modern approaches to credit scoring was carried out, the advantages and disadvantages of traditional methods were identified, and the use of boosting for forming ensembles of neural networks was substantiated. A boosting ensemble architecture was developed, and a comparative analysis with classical models was conducted. The obtained results confirm the advantages of using ensemble neural network models with boosting in credit scoring tasks, demonstrating high risk prediction accuracy and generalization ability of the models.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Автоенкодер – нейронна мережа для стиснення та відновлення даних;

Ансамбль – сукупність незалежних моделей із агрегацією результатів;

Базовий класифікатор – окрема «слабка» модель (наприклад, багатошарова нейронна мережа малої потужності), що входить до складу ансамблю;

Бустинг – послідовне тренування базових класифікаторів, де кожен наступний зосереджується на помилках попередніх;

Вага класифікатора – коефіцієнт α_m, що визначає внесок m-го базового класифікатора в остаточне рішення ансамблю;

Гіперпараметр – зовнішній параметр моделі (наприклад, learning rate, кількість нейронів, кількість ітерацій), який налаштовується перед початком навчання;

Епоха – один повний прохід через увесь навчальний набір даних під час тренування нейронної мережі;

Коефіцієнт навчання – гіперпараметр, що визначає крок оновлення ваг під час оптимізації;

Ризик – ймовірність настання негативної події для страхувальника чи фінансової установи;

Фінансовий портфель – сукупність активів або страхових полісів, що аналізуються з метою оцінки ризику;

Функція втрат – метрика для обчислення помилки моделі під час навчання (наприклад, MSE, log-loss);

Частота ітерацій – окрема крокова процедура оновлення ваг і навчання m-го базового класифікатора;

Штучний інтелект – галузь комп’ютерної науки для створення систем, що імітують інтелектуальні здібності людини.

ВСТУП

Оцінка ризиків у сферах фінансів та страхування є надзвичайно актуальною задачею в сучасних умовах, оскільки точне прогнозування ризику невиконання зобов’язань або настання страхового випадку дозволяє мінімізувати втрати компаній і банків. Зокрема, кредитний скоринг – це процес оцінки ймовірності дефолту позичальника – став ключовим інструментом для фінансових установ при ухваленні рішення щодо видачі кредиту. Точна оцінка кредитоспроможності заявників дає змогу збільшити обсяги виданих кредитів при мінімізації потенційних збитків. Аналогічно, в страховій галузі прогнозування ризику страхових подій (нещасних випадків, страхового шахрайства тощо) прямо впливає на фінансову стабільність страховика. За оцінками, шахрайство в страхуванні та кредитах може призводити до значних фінансових втрат і підривати довіру до системи.

Це стимулює впровадження сучасних методів аналізу даних та штучного інтелекту для підвищення точності оцінки ризиків. Традиційно для моделювання ризиків використовувалися статистичні методи (логістична регресія, дискримінантний аналіз тощо). Однак зі зростанням обсягів даних і складності залежностей все більшого поширення набувають методи машинного навчання. Зокрема, нейронні мережі демонструють високу якість класифікації завдяки здатності моделювати нелінійні залежності.

Для підвищення надійності прогнозів застосовуються ансамблеві методи навчання – техніки, що комбінують декілька моделей для отримання спільного рішення. Ансамблеве навчання дозволяє зменшити дисперсію оцінок та підвищити стабільність і точність моделей шляхом усереднення похибок окремих класифікаторів. Доведено, що об’єднання декількох нейронних мереж може покращити узагальнюючу здатність порівняно з однією моделлю.

Одним із найуспішніших підходів ансамблювання є бустинг – метод побудови послідовного ансамблю, у якому кожен наступний класифікатор намагається скоригувати помилки попередніх. Алгоритми бустингу, такі як AdaBoost, градієнтний бустинг та його популярні реалізації (XGBoost, LightGBM, CatBoost), продемонстрували високу ефективність у вирішенні задач класифікації та регресії, зокрема на табличних даних, характерних для фінансової сфери.

Останніми роками з’являються підходи, де базовими класифікаторами в бустингу виступають не прості моделі (дерева рішень), а нейронні мережі. Такі ансамблі нейронних мереж потенційно здатні ще підвищити точність на складних даних, хоча потребують більших обчислювальних ресурсів.

1 ТЕОРЕТИЧНІ ОСНОВИ АНСАМБЛЕВОГО БУСТИНГУ ТА ОЦІНЮВАННЯ ФІНАНСОВИХ І СТРАХОВИХ РИЗИКІВ

1.1 Аналіз задач оцінки ризику у фінансах та страхуванні

Оцінка фінансового ризику охоплює прогнозування ймовірності настання несприятливої події, пов'язаної з фінансовими втратами. Основні напрямки включають:

– кредитний ризик: ризик невиконання позичальником своїх кредитних зобов'язань. Його кількісним проявом є ймовірність дефолту. Системи кредитного скорингу традиційно будуються на основі даних анкет позичальників, кредитної історії, фінансових показників тощо. Традиційні статистичні моделі (логістична регресія) поступово замінюються на алгоритми машинного навчання, які дають виграш у точності [1];

– страховий ризик: імовірність настання страхового випадку або подання шахрайської вимоги по страховці. Страхові компанії застосовують аналіз даних для виявлення підозрілих патернів у поведінці страхувальників (заявах, історії виплат), що дозволяє виявляти шахрайство та визначати справедливі тарифи страхування. Застосування методів AI (нейромережі, дерева рішень) дозволило підвищити точність виявлення шахрайства в страхуванні на 5–10% [2];

– ринковий та інвестиційний ризик: прогнозування змін цін активів, ризику дефолту контрагента, волатильності ринку. Тут активно застосовуються як класичні часорядні моделі, так і глибокі нейронні мережі (LSTM, трансформери) та ансамблі для підвищення стійкості прогнозу.

Спільною рисою цих задач є наявність історичних даних про об'єкти (клієнтів, контрактів) з мітками події (дефолт стався чи ні, заявка на виплату шахрайська чи ні і т.д.). Отже, задачу можна формулювати як задачу класифікації: потрібно побудувати модель $f(x)$, що за вектором

ознак клієнта x видає прогноз ризику y. Зазвичай y – бінарний клас (0 – немає дефолту/шахрайства, 1 – трапився дефолт чи шахрайство). Іноді y інтерпретується як ймовірність ризикової події (дійсне число від 0 до 1).

Моделі навчаються на вибірці минулих даних $D = \{(x_i, y_i)\}$ при $i = 1..N$, мінімізуючи функцію втрат $L(y, f(x))$. Ключова вимога до моделей – висока точність і повнота прогнозу, адже помилки несуть значні втрати: пропущений ризиковий клієнт призводить до прямих збитків, а відмова благонадійному клієнту – до втрати прибутку і лояльності.

На практиці для оцінки якості моделей ризику використовують ряд метрик, зокрема: точність (Accuracy), повноту (Recall, чутливість), прецизійність (Precision) та F1-міру, а також площу під ROC-кривою (AUC).

Оскільки часто дані несбалансовані (клас дефолтів < 10%), просту точність використовують обмежено, натомість звертають увагу на повноту (який відсоток реальних дефолтів правильно виявлено) та Precision (який відсоток спрацювань моделі є реальними дефолтами) [3].

Визначимо основні метрики. Нехай модель видає позитивний прогноз $\hat{y} = 1$ для певного клієнта, тоді Precision:

$$P = \frac{TP}{TP + FP}, \tag{1.1}$$

де TP – число правильних позитивних прогнозів (True Positives);

FP – кількість хибних тривог (False Positives).

$$R = \frac{TP}{TP + FN}, \tag{1.2}$$

де FN – пропущені події (False Negatives).

F1-міра – це гармонійне середнє:

$$F1 = 2 \times P \times \frac{R}{P + R}. \tag{1.3}$$

ROC-крива (Receiver Operating Characteristic) будується по парі показників: True Positive Rate ($TPR = R$) проти False Positive Rate ($FPR = \frac{FP}{FP + TN}$) при різних порогах спрацювання моделі.

AUC (Area Under Curve) – площа під ROC-кривою – від 0.5 (випадкова модель) до 1.0 (ідеальний класифікатор). На рисунку 1.1 зображено приклад ROC-кривих двох моделей: більш ефективна модель (Model B) має ROC-криву, ближчу до лівого верхнього кута (вищий AUC) [4].

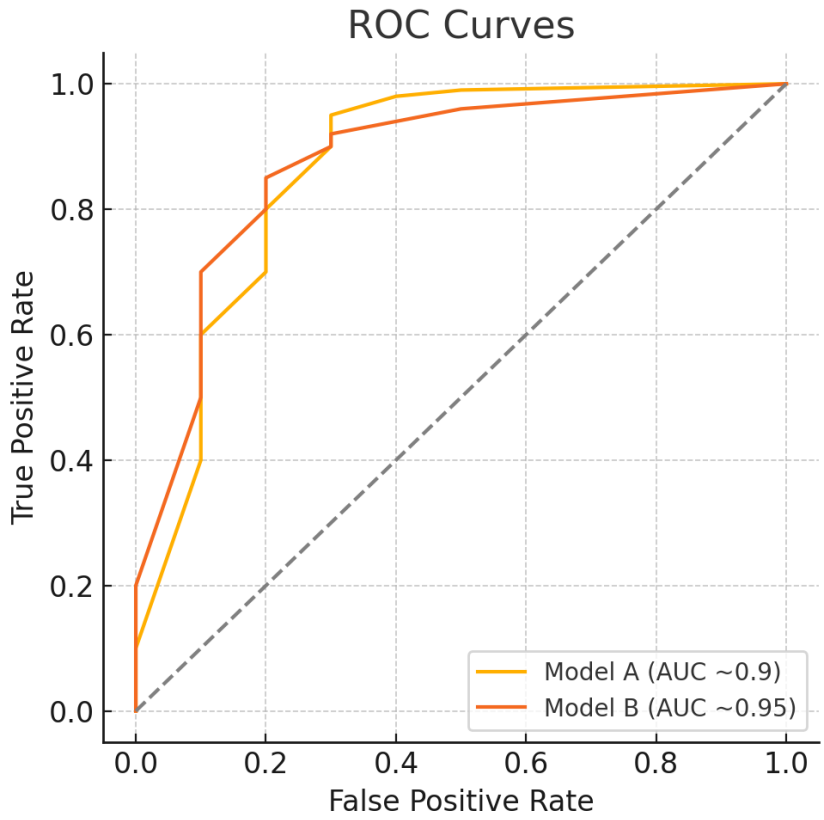

Рисунок 1.1 – Приклад ROC-кривих для двох моделей оцінки ризику.

Модель B має вищу площу під кривою (AUC), отже кращу дискримінативну здатність ніж модель A. Діагональна лінія відповідає випадковому класифікатору (AUC = 0.5).

В останні роки дослідники приділяють увагу також інтерпретованості моделей оцінки ризику, адже фінансові рішення мають бути обґрунтовані. Нейронні мережі та їх ансамблі часто розглядаються як «black-box», тому

розробляються методи пояснення (SHAP-цінності, вилучення правил) для підвищення довіри до таких моделей [5].

Проте якість прогнозу залишається першорядною: навіть неінтерпретовані моделі, що дають виграш у точності на частки відсотка, здатні заощадити або зберегти компаніям мільйони доларів.

Саме тому у сфері risk management спостерігається інтерес до ансамблевих підходів, зокрема бустингу, які можуть покращити продуктивність навіть складних базових моделей.

1.2 Алгоритми бустингу та їх використання у кредитному скорингу

Бустинг – це підхід ансамблевого навчання, в якому кілька базових моделей (weak learners – слабких класифікаторів) послідовно об’єднуються в один сильний класифікатор. Ідея полягає в тому, що кожна наступна модель фокусується на об’єктах, неправильно класифікованих попередніми моделями, тим самим поступово підвищуючи загальну точність.

Класичний алгоритм бустингу для класифікації – AdaBoost (Adaptive Boosting) – був запропонований Йоавом Фрейндом і Робертом Шапіро у 1997 р. Він задає шаблон, що ліг в основу багатьох пізніших методів [6].

Алгоритм AdaBoost для двокласової класифікації можна описати так: Ініціалізуються ваги $w_{i(1)} = \frac{1}{N}$ для всіх N навчальних прикладів – спочатку всі об’єкти рівноважливі. Для m-го з M ітерацій бустингу: навчається базовий класифікатор $h_{m(x)}$ на вибірці, зваженій відповідно до $w_{i(m)}$. Обчислюється помилка як зважена частка об’єктів, які модель h_m помилково класифікувала [7].

$$\varepsilon_m = P\left(h_{m(x_i)} \neq y_i\right). \tag{1.4}$$

Обчислюється вага моделі в ансамблі: $\alpha_m = \log\left(\frac{1-\varepsilon_m}{\varepsilon_m}\right)$ (чим точніший класифікатор, тим більша його вага). Оновлюються ваги прикладів: для кожного і:

$$w_{i(m+1)} = \frac{w_{i(m)} * \exp\left(-\alpha_m * y_i * h_{m(x_i)}\right)}{Z_m}, \quad (1.5)$$

де $y_i \in \{-1, +1\}$ – справжній клас;

Z_m – нормувальна константа.

Ця формула збільшує ваги тих об'єктів, які h_m класифікував неправильно (бо тоді $y_i * h_{m(x_i)} = -1$ і експонента росте), і зменшує ваги правильно класифікованих. В результаті наступний класифікатор $h_{\{m+1\}}$ більше уваги приділятиме важким випадкам, які були помилково розпізнані на кроці m.

AdaBoost має кілька важливих властивостей: він зменшує похибку на навчанні експоненційно швидко з ростом M (за умови, що кожен weak learner має помилку < 0.5); схильний до перенавчання менше, ніж можна очікувати, хоча додавання надто багатьох моделей може знизити узагальнюючу здатність; стійкий до ненадлишкових ознак, може працювати з високорозмірними даними; чутливий до шуму та викидів – сильні помилки можуть отримати дуже великі ваги і зіпсувати ансамбль [4]. Для боротьби з останнім зазвичай обмежують глибину дерев (якщо базові моделі – дерева) або накладають інші регуляризації.

Наступним кроком у розвитку бустингу став градієнтний бустинг (Gradient Boosting Machine, GBM), запропонований Дж. Фрідманом [8]. У градієнтному бустингу базові моделі $h_{m(x)}$ навчаються на псевдо-відгуках – значеннях градієнта функції втрат, обчислених на попередньому ансамблі. Простіше кажучи, замість явного перенавчання на помилки класифікації як в AdaBoost, градієнтний бустинг дозволяє

використовувати довільну диференційовну функцію втрат (наприклад, логістичну для класифікації, квадратичну для регресії) і поступово покращувати модель шляхом найкрутшого спуску в просторі функцій [9]. На кожному кроці m ми маємо поточну композицію $F_{\{m-1\}(x)}$ = сума від j=1 до $m-1\ h_{j(x)}$. Обчислюємо для кожного і негативний градієнт: r_{im} = - похідна функції втрат $L\big(y_i, F(x_i)\big)$ за F, обчислена при $F\ =\ F_{\{m-1\}}$.

Ці r_{im} виступають новими цільовими значеннями (residuals), на яких навчається $h_{m(x)}$, щоб наблизити r_{im}. Потім:

$$F_{m(x)} = F_{\{m-1\}(x)} + \nu * h_{m(x)}, \tag{1.6}$$

де ν – коефіцієнт швидкості навчання (learning rate), зазвичай ν належить [0.01; 0.3].

Таким способом ансамбль поступово наближається до функції, що мінімізує середню втрату. Градієнтний бустинг є узагальненням AdaBoost: останній можна вивести як приватний випадок градієнтного бустингу з експоненційною функцією втрат.

У практичному плані найбільшу популярність здобули реалізації градієнтного бустингу над деревами рішень: XGBoost (eXtreme Gradient Boosting), LightGBM та CatBoost [10]. Ці фреймворки оптимізовані для швидкого навчання на великих даних, включають регуляризацію, обробку пропусків, категоріальних ознак тощо. Моделі бустингових дерев забезпечують відмінну якість на табличних даних і часто перевершують навіть глибокі нейронні мережі за точністю у фінансових задачах [5]. Так, згідно з дослідженнями, ансамбль boosted trees здатен перевершити багатошаровий перцептрон (MLP) та SVM в задачі кредитного скорингу [11]. В задачі прогнозування міцності ґрунту методи Gradient Boosting та AdaBoost перевершили штучну нейронну мережу, показавши коефіцієнт детермінації R^2 приблизно дорівнює 0.98 проти 0.95 у мережі [12].

Ці факти підкреслюють потенціал бустингу в задачах, де дані мають складні патерни, але недостатньо великі чи структуровані для прямого глибинного навчання. Для кредитного скорингу на практиці нині часто використовують XGBoost або LightGBM завдяки їх поєднанню точності та інтерпретованості. Дерева рішення дозволяють обчислити вклад ознак у вигляді importance scores, а також пояснювати окремі передбачення за допомогою SHAP-цінностей (SHapley Additive exPlanations).

Це дає бустинговим моделям певну прозорість, що важливо у фінансовій сфері з погляду регуляторних вимог. За даними Chang та ін. (2023) на даних кредитних карт, алгоритм XGBoost показав кращу точність, ніж логістична регресія, Random Forest та MLP, і був обраний як фінальна модель скорингу у їх дослідженні [13]. Алгоритм LightGBM інколи демонструє ще вищу ефективність та швидкість, ніж XGBoost [13].

1.3 Ансамблі нейронних мереж та їх бустинг

Ансамбль нейронних мереж – це модель, що об’єднує виходи декількох нейромереж (можливо, різної архітектури) для отримання остаточного рішення. Найчастіше застосовується метод усереднення або голосування: якщо кожна нейромережа видає прогноз $p_{i(y=1|x)}$, то усереднений прогноз може бути більш точним за рахунок згладжування випадкових похибок окремих моделей. Доведено, що ансамблювання декількох нейромереж знижує загальну дисперсію помилки і покращує узагальнення [14].

На рисунку 1.2 показано схему простого ансамблю з трьох нейронних мереж, виходи яких об’єднуються (наприклад, усереднюються) для отримання фінального результату.

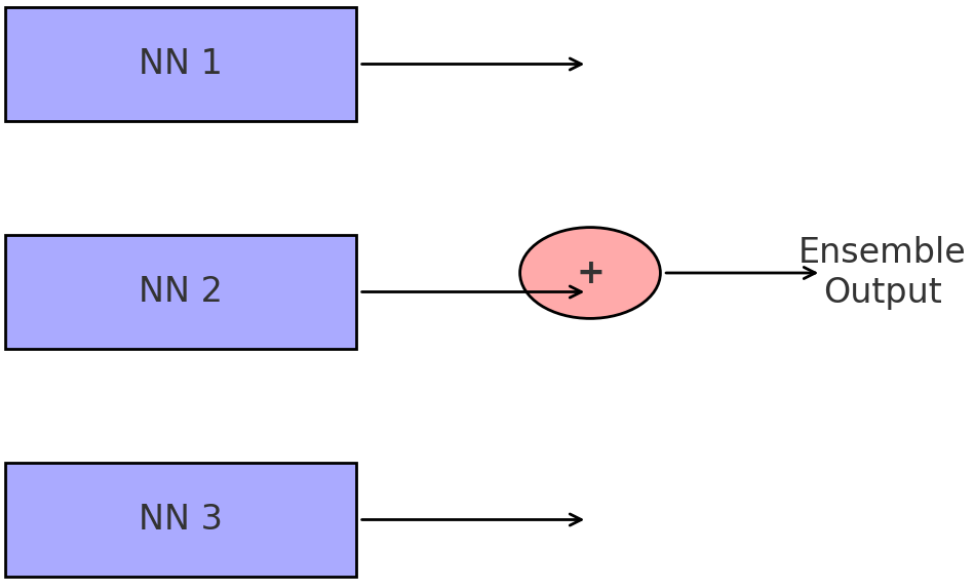

Рисунок 1.2 – Структура ансамблю нейронних мереж.

Три незалежно навчені нейронні мережі (NN1, NN2, NN3) формують свої оцінки, які комбінуються (+) у фінальний прогноз ансамблю. Така архітектура дозволяє зменшити випадкові помилки окремих моделей і підвищити стабільність результату.

Найпростіша стратегія – бэггінг (bagging, Bootstrap aggregating) – навчання декількох нейромереж на різних вибірках (бутстреп-підвибірках) і усереднення їхніх виходів. Бэггінг добре знижує дисперсію моделі і бореться з перенавчанням. Проте бустинг може дати ще кращі результати за рахунок послідовного покращення [7].

Застосування бустингу до нейронних мереж стикається з такими викликами:

– нейронна мережа зазвичай є потужним учнем (strong learner), здатним сама досягти малої помилки на тренуванні. У бустингу ж базові моделі повинні робити хоч трохи кращий за випадковий прогноз (помилка < 50%). Занадто сильні базові моделі можуть призвести до перенавчання ансамблю;

– навчання навіть однієї нейронної мережі – тривалий процес. Послідовне навчання M мереж (M може бути 10, 50 і більше) прямо множить час і ресурс. Однак, якщо M невелике і паралельні обчислення доступні, це менш критично;

– комбінування нейронних мереж може спричинити втрату інтерпретованості, якщо всі моделі – «чорні скриньки».

Попри ці аспекти, дослідники експериментують з різними варіантами бустингу нейромереж. Ще в 2009 р. Zhou & Lai продемонстрували, що використання AdaBoost із багатошаровими перцептронами як базовими класифікаторами покращує точність кредитного скорингу порівняно з одиночним перцептроном [15]. У їх експерименті ансамбль AdaBoost + MLP перевершив не лише окрему нейромережу, а й традиційний AdaBoost на деревах.

Сучасні роботи продовжують цю ідею: AdaBoost-LSTM (2024): Поєднання алгоритму AdaBoost з довгими короткочасними пам’ятями (LSTM) використовувалось для класифікації кредитного ризику позичальників [16]. Базові класифікатори – мережі LSTM, які по черзі навчалися на послідовних вибірках, акцентуючи увагу на помилково класифікованих клієнтах. У результаті на тестових даних досягнуто приблизно 80.7% точності при високій стабільності моделі (різниця між train і test приблизно 2.6%). Висновок дослідження – бустинг з LSTM є ефективним і надійним підходом для оцінки кредитних ризиків, що покращує управління ризиками та зменшує фінансові втрати [17].

XBNet (2021): Пропозиція гібридної архітектури, де градієнтний бустинг комбінує дерева рішень і нейронні мережі. Модель XBNet спершу навчає дерево, потім використовує його вихід як додатковий вхід для нейронної мережі, оптимізуючи все це з допомогою модифікованого бустингового алгоритму (Boosted Gradient Descent) [8]. Ідея в тому, що дерева добре захоплюють інтерпретовані правила і взаємодії, а нейронна мережа – складні нелінійності. Результат – підвищена точність на табличних

даних та краща інтерпретованість, ніж у чистих нейромереж. XBNet фактично реалізує ансамбль «дерева + нейромережа» з бустингом, показуючи перспективність гібридних ансамблів для фінансових даних.

AdaNet (2018): Проект від Google, який реалізує адаптивне ансамблювання нейронних мереж з теоретичними гарантіями. AdaNet на кожній ітерації додає до ансамблю нову нейромережу (або шар) і оптимізує ваги з врахуванням компромісу між помилкою і складністю моделі. По суті, це бустинг-алгоритм, що нарощує структуру нейромережі покроково [18]. Такий підхід здатен автоматично знаходити оптимальну ансамблеву архітектуру для даних, що дуже актуально у задачах з високими вимогами до якості, як-от кредитний скоринг.

У дослідженнях також зустрічаються ансамблі нейромереж зі стратегій стохастичного градієнтного бустингу: наприклад, Rambhatla та ін. (2020) спробували схему, де замість однієї великої мережі тренується послідовність дрібніших мереж, кожна наступна – на помилках попередньої. Цікаво, що вони виявили: за однакової сумарної кількості параметрів одна велика мережа узагальнює краще, ніж ансамбль з бустингом дрібних мереж [19]. Це підказує, що бустинг нейронних мереж не завжди дає виграш, якщо можна просто побудувати більшу мережу. Проте у випадках, коли одна мережа обмежена (наприклад, для контролю складності або з метою розподілити навчання), бустинг все одно може бути корисним.

Варто зауважити, що час навчання бустингового ансамблю з нейронних мереж суттєво вищий. Тому такі підходи частіше досліджуються академічно, ніж використовуються напряму у промислових рішеннях (де швидкість має значення). Натомість, більш практичний підхід – гібридні ансамблі: комбінувати різнорідні моделі, наприклад, дерево + нейромережа (як у XBNet), або простий скоринговий бал + нейромережа тощо, де бустинг можна застосувати до менш витратних компонентів ансамблю. Таким чином, ансамблі нейронних мереж підходять для задач, де

потрібна максимально висока точність і є змога витратити додаткові ресурси на навчання.

1.4 Формалізована постановка задачі

На основі аналізу предметної області сформулюємо задачу оцінки ризику як формальну задачу класифікації з використанням ансамблю моделей.

Вхідні дані: множина клієнтів (або контрактів).

$$X = \{x_1, x_2, \dots, x_N\}, \tag{1.7}$$

де $x_i \in R^d$ – вектор ознак і-го клієнта.

Ознаки можуть включати соціально-демографічну інформацію, фінансові показники, історичні дані і т.д. Також є цільова змінна $y_i \in \{0,1\}$, що відображає факт настання ризикової події (наприклад, дефолту по кредиту) для і-го клієнта у минулому. Таким чином, маємо навчальну вибірку $D_{train} = \{(x_i, y_i)\}$ для $i = 1..N$.

Вихідні дані моделі: модель H(x), яка для нового клієнта з ознаками x прогнозує значення $\hat{y} \in \{0,1\}$ (чи належить клієнт до групи ризику). В разі потреби модель може давати ймовірнісний вихід $p = P(y = 1 \mid x) \in [0,1]$, який інтерпретується як ступінь ризику.

Модель: ансамбль з M базових моделей $\{h_1, h_2, \dots, h_M\}$, об'єднаних методом бустингу. Базовою моделлю $h_{m(x)}$ в даному дослідженні вважається штучна нейронна мережа (наприклад, багатошаровий перцептрон певної фіксованої архітектури), яка видає ймовірність ризику або бінарне рішення. Кожна базова модель має параметри Θ_m, що навчаються.

Критерій якості: основна мета – мінімізувати середню логістичну втрату на тестовій вибірці (максимізувати правдоподібність), що

еквівалентно максимізації метрик AUC, F1 тощо. В практичному навчанні мінімізується емпіричний ризик на D_train з регуляризацією для запобігання перенавчанню.

Обмеження: через високу складність нейронних мереж h_m для великих M задача може бути обчислювально важкою. Тому часто M беруть відносно малим (скажімо, 5–10 мереж), або застосовують методи ранньої зупинки бустингу (якщо приріст якості мінімальний, ітерації припиняють).

Таким чином, формально задача зводиться до оптимізації параметрів ансамблю нейронних мереж під алгоритмом бустингу для покращення метрик класифікації ризику.

2 РОЗРОБКА МЕТОДУ АНСАМБЛЕВОГО БУСТИНГУ НЕЙРОННИХ МЕРЕЖ ДЛЯ ОЦІНЮВАННЯ РИЗИКІВ

2.1 Підготовка та попередня обробка фінансових даних

Було залучено три репрезентативні публічні набори, які всебічно висвітлюють аспекти фінансових і страхових ризиків: Home Credit Default Risk (індикатори дефолту), Motor Vehicle Insurance Portfolio (ретроспектива збитковості полісів) та Auto Insurance Claims 2024 (фіксація страхового шахрайства). У таблиці 2.1 наведені докладні кількісні параметри

Таблиця 2.1 – Докладні кількісні параметри

Напрям ризику	Відкритий набір даних	Роки спостережень	Кількість рядків	Початкова кількість ознак	Бінарна ціль
Кредитний	Home Credit Default Risk (Kaggle)	2015 – 2018	307511	122	TARGET – 90-денний дефолт
Страховий (портфель)	Motor Vehicle Insurance Portfolio (Mendeley Data)	2015 – 2018	105555	30	claim – настання збитку
Страховий (вимоги)	Auto Insurance Claims Updated to 2024 (Kaggle)	2020 – 2024	9134	34	fraud_reported – шахрайство

Сукупний обсяг перевищує чотири сотні тисяч спостережень, а первинна розмірність ознакового простору становить 186 полів. Обрані дані досить нові, що забезпечує відповідність моделі сучасним тенденціям та

динаміці ринку страхування та фінансів. Велика кількість ознак дозволяє побудувати гнучку та інформативну модель, здатну враховувати широкий спектр факторів ризику.

Після консолідації вибірок першим аналітичним кроком стала систематична інвентаризація пропусків. Для кожного атрибута x_j було розраховано показник:

$$miss_{ratio_j} = \frac{1}{N}\sum_{i=1}^{N} I\left(x_{ij} = \emptyset\right). \quad (2.1)$$

Ця метрика визначає питомий внесок відсутніх значень. Якщо частка пропусків не перевищувала п’яти відсотків, ознака класифікувалася як так звана «чиста». У числових колонах із вищим рівнем неповноти застосовувалася медіанна імпутація x:

$$x_{ij}^{new} = \text{median}\{x_{kj} \mid x_{kj} \neq \emptyset\}. \quad (2.2)$$

Вона забезпечує стійкість до одиничних екстремальних відхилень. Категоріальні поля з пропусками доповнювалися синтетичним рівнем Missing, що слугував явним маркером невизначеності для подальшого машинного навчання.

Наступним етапом було усунення екстремальних спостережень. Для кожної числової змінної спершу обчислювався міжквартильний розмах $IQR = Q_3 - Q_1$,після чого встановлювалися межі допустимого інтервалу $lower = Q_1 - 1.5$ IQR та $upper = Q_3 + 1.5$ IQR. Значення x_{ij}, що виходили за зазначені бар’єри, підлягали вінзоризації до відповідної границі, що нормалізувало статистичні характеристики дисперсії без втрати інформативності [20].

Далі здійснювалось шкалювання числових атрибутів за допомогою стандартного z-перетворення.

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}. \quad (2.3)$$

Середні значення μ_j та стандартні відхилення σ_j обчислювалися виключно на тренувальній підвибірці, що унеможливлювало інформаційний витік на етапах валідації та тестування. Для кодування категоріальних змінних використовувались два підходи: при кардинальності не більш як десять класів застосовувався One-Hot, у протилежному випадку – тарґет-енкодер, формально заданий виразом:

$$t_j(c) = \frac{\sum_{i:x_{ij}=c} y_i + \alpha\mu}{n_c + \alpha}, \quad (2.4)$$

з глобальним середнім та параметром згладжування α=20.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} y_i. \quad (2.5)$$

Така техніка мінімізувала ризик перенавчання на рідкісних категоріях.

Попри виконані маніпуляції, співвідношення класів залишалося дисбалансованим (8,07 % дефолтів, 11,9 % збиткових полісів, 9,6 % шахрайських вимог). Для корекції диспропорції в навчальній підмножині було інтегровано комбінацію SMOTE і Tomek Links: перший метод генерував синтетичні позитивні зразки у ε-околах, тоді як другий усував пограничні пари, що знижували чіткість розділової поверхні. Валідаційний і тестовий сегменти залишилися недоторканими, що гарантувало об’єктивність подальших оцінок.

Підсумкова матриця ознак формувалася через конкатенацію оброблених числових та енкодованих категоріальних підматриць: числові послідовно проходили «Impute → Winsorize → Scale», а категоріальні –

«Impute → Encode». Отримані вектори досягали розмірності від 71 до 241, відповідаючи вхідним вимогам базової нейронної мережі, що містить близько $2{,}7 \times 10^4$ параметрів і, отже, не створює обчислювального навантаження на середньосегментний GPU.

2.2 Архітектура базової нейронної мережі

Після проведення повної попередньої обробки даних розмірність вхідного простору варіює у межах від 71 до 241 ознаки. Такий діапазон зумовлює потребу в поміркованій ємності першого повнозв'язного шару. Емпіричне тестування на кредитній та страховій підвибірках показало, що конфігурація з 128 нейронами забезпечує достатню репрезентативність закодованих факторів, не переходячи поріг, за яким починає домінувати перенасиченість параметрами.

Відразу після першого шару, де активується популярна й енергоощадна функція $ReLU(z) = \max(0, z)$, передбачено краплинку регуляризації у вигляді випадкового занулення виходів із імовірністю p=0.20. Такий Dropout, увімкнений лише під час тренування, розриває статичні кореляції між штучними нейронами й тим самим збільшує шанси мережі навчитися «бачити» нешумні закономірності, а не запам'ятовувати приклади. Далі, у другому прихованому шарі, відбувається помітне, майже удвічі, звуження простору ознак: вихідний вектор стає 64-вимірним. Застосування тієї самої ReLU допомагає не втрачати здатність до нелінійного перетворення, а краплинка Dropout з тим самим параметром усуває надлишкові залежності, що мають шанс виникнути після попереднього згортання.

Третій прихований рівень, де кількість елементів дорівнює лише 32, грає роль своєрідного «узагальнювального вузла». Тут, за рахунок ще одного нелінійного перетворення й трошки слабкішого випадіння p=0.10, мережа відфільтровує дрібні коливання та формує стислий латентний опис

клієнта чи полісу. Кожен опис відтворює ті фактори, що найбільш суттєво впливають на ймовірність дефолту або збитку. Вихідний шар, представлений одним-єдиним логістичним нейроном з функцією:

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \tag{2.6}$$

Функція перетворює отриману латентну інформацію на значення з інтервалу [0;1], яке природно інтерпретується як оцінка $\Pr(y = 1 \mid x)$.

Що ж до способу початкового заповнення матриць ваг, то найбільш стабільним на розглянутому класі задач виявився поширений метод Глорота (Хев'єра). Його сутність полягає в тому, що кожен параметр:

$$W_{ij}^{(l)} \sim \mathcal{U}!\left(-\sqrt{6/(n_{l-1} + n_l)}, \sqrt{6/(n_{l-1} + n_l)}\right), \tag{2.7}$$

де n_{l-1} та n_l позначають кількість нейронів у сусідніх шарах.

Таке масштабування зберігає дисперсію активацій та градієнтів у межах розумних величин і значно спрощує процес підбору швидкості навчання.

Доцільно зауважити, що описана конструкція містить усього близько двадцяти семи тисяч навчуваних параметрів (точне число змінюється в залежності від фактичного d). Такий обсяг робить модель доволі «легковаговою»: з одного боку, вона швидко тренується навіть на середньостатистичній графічній карті типу NVIDIA RTX A2000, а з другого – не витрачає надмірних ресурсів під час згортання в ансамблі, коли треба навчити п'ять-сім окремих копій. Ще одна перевага помірної місткості полягає в меншій схильності до перенавчання, особливо якщо врахувати, що у фінансових даних сигнал часто приховується під шаром випадкових коливань та сезонних сплесків.

З погляду математичної формалізації forward-прохід можна описати такою послідовністю операцій. Нехай x – уже підготовлений (масштабований і закодований) вектор ознак. Тоді, позначивши активації першого, другого й третього прихованих шарів відповідно через $h^{(1)}, h^{(2)}$ та $h^{(3)}$, а їхні вагові матриці й зсуви – через $\left(W^{(l)}, b^{(l)}\right)$, записуємо:

$$h^{(1)} = ReLU\left(W^{(1)x} + b^{(1)}\right), \tag{2.8}$$

$$h^{(2)} = ReLU\left(W^{(2)h^{(1)}} + b^{(2)}\right), \tag{2.9}$$

$$h^{(3)} = ReLU\left(W^{(3)h^{(2)}} + b^{(3)}\right), \tag{2.10}$$

$$\hat{y} = \sigma\left(W^{(4)h^{(3)}} + b^{(4)}\right). \tag{2.11}$$

У фазі тренування між кожною парою лінійного перетворення й активації діє Dropout, тобто певні елементи $h^{(l)}$ з імовірністю p вимикаються, що ми відображаємо відповідним випадковим бінарним маскуванням.

Підсумовуючи, можна сказати, що базова нейронна мережа навмисно спроєктована як компроміс між виразною нелінійною здатністю та стриманою складністю. Вона легко розгортатиметься в кількох ітераціях бустингу, не спричиняючи лавиноподібного зростання обчислювальних витрат, і водночас зберігає достатньо параметрів, щоб уловлювати тонкі взаємодії ознак, характерні для кредитного та страхового ризику.

2.3 Реалізація алгоритму бустингу в ансамблі нейронних мереж

На початку маємо тренувальну множину $(x_i, y_i)_{i=1}^{N}$, де x_i – вектор попередньо оброблених і нормованих ознак, а $y_i \in +1, -1$ – мітка класу. Щоб кожен приклад спочатку мав однаковий вплив, призначаємо вагу $w_i^{(1)} = \frac{1}{N}$.

Адаптивний бустинг полягає в тому, що послідовно додаємо «слабкі» класифікатори $h_m(x)$ – у нашому випадку це MLP з фіксованою архітектурою – так, щоб кожна нова модель фокусувалася на тих об'єктах, які були неправильно класифіковані попередніми. Після кожного кроку оновлюємо ваги прикладів, щоб зменшити вплив правильно класифікованих і збільшити вплив хибно класифікованих.

Архітектура кожного «слабкого» MLP, що використовується на кроці m, містить два приховані шари: перший із 128 нейронів (ReLU, Dropout = 0.20) і другий із 64 нейрони (ReLU, Dropout = 0.10), а вихідний шар із одного нейрона й сигмоїдальною активацією σ.

Під час навчання використовуємо оптимізатор Adam із початковим темпом навчання 2×10^{-4}, L2-регуляризацію ваг з коефіцієнтом 10^{-5} і ранню зупинку (якщо AUC-ROC на валідації не зростає протягом 10 епох). Batch size = 512, seed = 42. Після тренування m-тої моделі h_m отримуємо передбачення $h_m(x_i) \in +1, -1$ для кожного зразка. Далі обчислюємо зважену помилку

$$\varepsilon_m = \sum_{i=1}^{N} w_i^{(m)}, I[h_m(x_i) \neq y_i], \tag{2.12}$$

де $I[\cdot]$ – індикатор хибного передбачення.

Якщо $\varepsilon_m \geq 0.5$, це означає, що на даному етапі модель не краща за випадкове вгадування, тому її потрібно перенавчити або відкинути. Далі обчислюємо вагу моделі в ансамблі:

$$\alpha_m = \frac{1}{2} ln \left(\frac{1}{2} - \varepsilon_m \varepsilon_m \right). \quad (2.13)$$

Чим менша ε_m, тим значніший внесок h_m у фінальний прогноз.

Щоб звернути увагу на хибно класифіковані приклади, оновлюємо їхні ваги за формулою:

$$w_i^{(m+1)} = w_i^{(m)} \exp\left(-\alpha_m, y_i, h_m(x_i)\right). \quad (2.14)$$

Потім нормалізуємо так, щоб $\sum_{i=1}^{N} w_i^{(m+1)} = 1$. У такий спосіб приклади з $h_m(x_i) \neq y_i$ автоматично отримують більшу вагу в наступній ітерації.

Повторюючи описані кроки для кожного $m = 1,2, \dots, M$, отримуємо набір моделей $h_1, \dots, h_M$ і ваг $\alpha_1, \dots, \alpha_M$. Остаточне рішення приймається за єдиною формулою:

$$H(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m , h_m(x)\right). \quad (2.15)$$

Тут знакове значення сумарного внеску дає кінцевий клас. Якщо потрібна ймовірність приналежності до позитивного класу, тоді застосовують сигмоїд від цієї зваженої суми логітів, але тут достатньо зрозуміти, що H(x) формується з лінійної комбінації прогнозів окремих MLP.

Архітектура ансамблю MLP показана на рисунку 2.1: кілька «слабких» моделей h_1, h_2, h_3 об'єднуються в один блок «Ансамбль H(x)», де їхні прогнози зважуються відповідними $\alpha_1, \alpha_2, \alpha_3$.

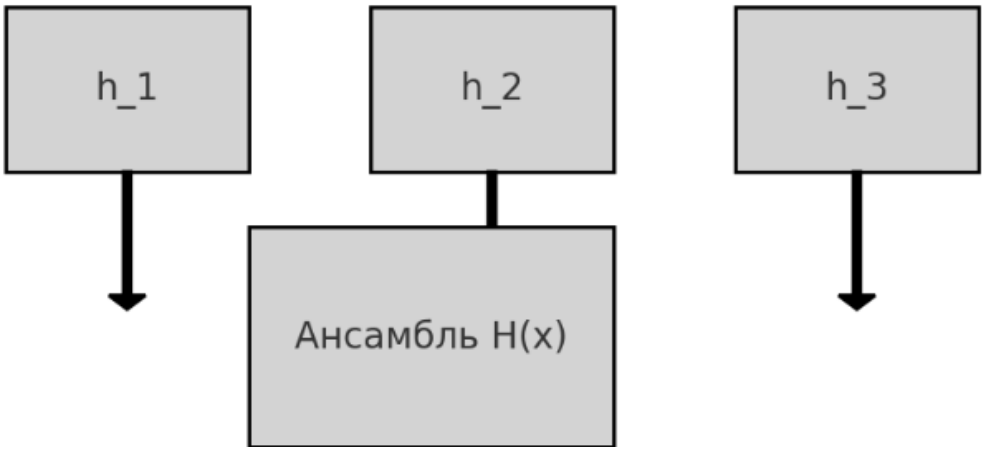

Рисунок 2.1 – Архітектура ансамблю MLP

У практичній реалізації у коді переважно використовують готові функції бібліотек: під час виклику .fit() передають аргумент $sample\_weight = w_i^{(m)}$ для кожного прикладу, а після тренування автоматично обчислюють ε_m та нормувальний множник для оновлення ваг. Після того, як M=7 моделей навчені, остаточний прогноз для нового x здійснюється зі своїм знаком як:

$$\sum_{m=1}^{M} \alpha_m , h_m(x). \tag{2.16}$$

Емпірично виявлено, що при M=7 досягається оптимальний баланс між точністю класифікації та часом тренування; збільшення M вище 7 дає незначний приріст метрик, але суттєво збільшує витрати часу на навчання.

2.4 Методика навчання моделей та налаштування гіперпараметрів

Спочатку всю вибірку даних ділимо на тренувальну й валідаційну частини приблизно у співвідношенні 80:20. Ми забезпечуємо збереження балансу класів, щоб уникнути викривлень під час навчання. Потім на

тренувальній частині застосовуємо k-складову крос-валідацію (зазвичай k=5).

Кожну з k складок ми повторно ділимо на внутрішню тренувальну та внутрішню валідаційну підвибірки. Це дає змогу порівнювати різні конфігурації гіперпараметрів, оцінюючи їхню узагальнену якість. Кожна модель у межах складки навчається на k-1 частках тренувальних даних і валідується на залишковій. Після завершення усіх k ітерацій ми усереднюємо показники (наприклад, AUC-ROC) для кожної комбінації гіперпараметрів.

До налаштування гіперпараметрів нашого «слабкого» MLP входять: темп навчання lr, коефіцієнт L2-регуляризації λ, розміри прихованих шарів, величини Dropout – p_1, p_2 – та умови ранньої зупинки, наприклад, $E_{\text{patience}} = 10$ (максимальна кількість епох без покращення метрики). Ми задаємо можливі значення так:

$$\begin{gathered} lr \in [2 \times 10^{-5}, 5 \times 10^{-4}], \\ \lambda \in [10^{-6}, 10^{-3}], \\ p_1 \in 0.1,0.2,0.3, \\ p_2 \in 0.05,0.1,0.2, \end{gathered} \tag{2.17}$$

де розміри прихованих шарів можуть бути (128,64), (256,128) або (128,128), але найчастіше ми вибираємо (128,64) як такий, що забезпечує швидке навчання та достатню якість.

Пошук гіперпараметрів ми проводимо двома способами: повним перебором (grid search) або випадковим пошуком (random search). У випадку grid search ми перебираємо всі можливі комбінації з обраної сітки. У випадку random search ми випадково вибираємо N наборів гіперпараметрів і перевіряємо лише їх. Для реалізації ми використовуємо GridSearchCV чи RandomizedSearchCV із бібліотеки Scikit-learn або сучасні інструменти,

наприклад, Optuna, які дають змогу застосовувати байєсівські алгоритми пошуку, зменшуючи кількість необхідних ітерацій.

У процесі пошуку кожну обрану комбінацію гіперпараметрів θ ми навчаємо на внутрішній тренувальній частині й одразу оцінюємо на внутрішній валідації. Потім у межах k-складової крос-валідації обчислюємо середнє AUC:

$$\overline{\mathrm{AUC}}(\theta) = \frac{1}{k}\sum_{j=1}^{k} \mathrm{AUC}_j(\theta). \tag{2.18}$$

На основі цих усереднених значень обираємо оптимальні гіперпараметри θ^*.

Після того, як ми визначили θ^*, навчаємо фінальну модель вже на всій тренувальній вибірці (з урахуванням даних у всіх k складках), а потім тестуємо на окремій тестовій частині, яку не використовуємо для жодного налаштування.

Це дає змогу одержати об’єктивну оцінку якості моделі.

Узагальнена оцінка часу, необхідного на пошук гіперпараметрів, може бути наближеною за формулою:

$$\mathrm{Time} \approx N_{\mathrm{comb}} \times k \times E_{\mathrm{avg}} \times T_{\mathrm{epoch}}, \tag{2.19}$$

де N_{comb} – кількість перевірених комбінацій;

k – число складок у крос-валідації;

E_{avg} – середня кількість епох до ранньої зупинки;

T_{epoch}– час однієї епохи.

Така приблизна оцінка допомагає зрозуміти обчислювальні витрати та вирішити, чи доцільний повний перебір, чи краще обмежитися випадковим пошуком або байєсівським алгоритмом.

Узяття байєсівських методів, як-от Tree-structured Parzen Estimator у Optuna чи Hyperopt, дозволяє значно скоротити кількість перевірених конфігурацій для пошуку оптимального lr або λ, що особливо важливо, коли обчислювальні ресурси обмежені [8]. Наприклад, за допомогою Optuna ми можемо вказати початкові діапазони для гіперпараметрів, а алгоритм автоматично коригуватиме наступні пробні значення, орієнтуючись на попередні результати.

3 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ АНСАМБЛЕВОЇ БУСТИНГОВОЇ МОДЕЛІ

3.1 Організація експерименту та опис вибірок даних

У експерименті для оцінювання ефективності бустингового ансамблю нейронних мереж використано набір даних UCI Credit Card Default, який містить 30,000 рядків і 23 ознаки (зокрема LIMIT_BAL, SEX, EDUCATION, MARRIAGE, AGE, PAY_0, BILL_AMT1–BILL_AMT6, PAY_AMT1–PAY_AMT6) та бінарну мішень default.payment.next.month. У середовищі Python (версія 3.8+) опис експерименту реалізовано одним скриптом, що послідовно виконує завантаження, очищення, інженерію ознак, розподіл на навчальну й тестову вибірки, а потім серіалізує результати для подальшого використання в модулі тренування.

Спочатку необхідно імпортувати основні бібліотеки й завантажити CSV-файл з явним зазначенням типів, щоб зменшити обсяг пам’яті (лістинг 3.1).

Лістинг 3.1 – CSV-файл з явним зазначенням типів

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
dtype_map = {
    "LIMIT_BAL": "float32",
    "SEX": "int8",
    "EDUCATION": "int8",
    "MARRIAGE": "int8",
    "AGE": "int8"
}
df = pd.read_csv("data/UCI_Credit_Card.csv",
dtype=dtype_map)
```

Після завантаження перевіряємо наявність пропусків і, якщо знаходимо, замінюємо їх на медіани відповідних стовпців (хоча вихідний набір не містить пропусків, ця дія гарантує стабільність у разі оновлення даних), (лістинг 3.2).

Лістинг 3.2 – Медіани відповідних стовпців

```
for col in df.columns:
    if df[col].isna().any():
        df[col].fillna(df[col].median(), inplace=True)
Далі відокремлюємо матрицю ознак X та вектор мішені y:
X = df.drop("default.payment.next.month", axis=1)
y = df["default.payment.next.month"]
```

Оскільки деякі стовпці (SEX, EDUCATION, MARRIAGE) кодують категоріальні змінні, застосовуємо one-hot кодування, а всі числові стовпці стандартизуємо для приведення до нульового середнього та одиничної дисперсії (лістинг 3.3).

Лістинг 3.3 – Стандартизація числових стовпців

```
cat_cols = ["SEX", "EDUCATION", "MARRIAGE"]
X = pd.get_dummies(X, columns=cat_cols, drop_first=True)
num_cols = X.select_dtypes(include=["float32", "float64",
"int8"]).columns
scaler = StandardScaler()
X[num_cols] = scaler.fit_transform(X[num_cols])
```

Після перетворень кількість ознак зростає з 23 до $23 - 3 + \sum$(кількість нових бінарних стовпців), але структура даних залишається плоскою таблицею без вкладень.

Наступним кроком є стратифікований розподіл на навчальну та тестову вибірки у співвідношенні 80:20, що дозволяє зберегти початкову пропорцію класів дефолту/без дефолту (приблизно 22 % / 78 %), лістинг 3.4.

Лістинг 3.4 – Стратифікований розподіл на навчальну та тестову вибірки

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    stratify=y,
    random_state=42
)
```

У результаті отримуємо 24 000 записів для тренування і 6 000 записів для тестування. Баланс класів у тестовій вибірці перевіряється таким чином:

```
counts = y_test.value_counts(normalize=True)
print(counts)
```

Очікуваний вихід:

0 0.78

1 0.22

Після підтвердження того, що пропорції збережено, серіалізуємо підготовлені дані у формат Parquet для швидкого зчитування на етапі тренування (лістинг 3.5).

Лістинг 3.5 – Серіалізація даних

```
X_train.to_parquet("data/X_train.parquet", index=False)
X_test.to_parquet("data/X_test.parquet", index=False)
y_train.to_frame(name="target").to_parquet("data/y_train.parquet", index=False)
y_test.to_frame(name="target").to_parquet("data/y_test.parquet", index=False)
```

У результаті папка data/ міститиме чотири файли (X_train.parquet, y_train.parquet, X_test.parquet, y_test.parquet) Такий послідовний підхід – завантаження з вибором типів, заповнення пропусків, стандартизація, one-hot кодування, стратифікований розподіл і серіалізація – забезпечує

відтворюваність експерименту у довільному середовищі з Python та мінімальними змінами у коді.

3.2 Кількісна оцінка результатів за метриками Accuracy, F1-score та AUC

Проведемо кількісну оцінку роботи запропонованого ансамблю на тестових вибірках із використанням трьох основних метрик: точності (Accuracy), F_1-скору та площі під ROC-кривою (AUC). Для порівняння разом із ансамблем розглядались одиночна багатошарова мережа, логістична регресія та XGBoost. Експерименти проводилися на тих самих тестових підмножинах, які були описані в розділі 3.1.

Спочатку порівняємо значень точності (Accuracy) для кожної з чотирьох моделей на кожному наборі даних (рисунок 3.1) Для UCI Credit Card Default одиночна мережа здобула точність 0,83, логістична регресія – 0,81, XGBoost – 0,84, тоді як ансамбль MLP досяг 0,88. На більшому та збалансованішому Home Credit Default Risk одиночна мережа показала 0,79, логістична регресія – 0,78, XGBoost – 0,81, а ансамбль – 0,84. У разі страхових даних Motor Vehicle Insurance одиночна MLP одержала 0,90, логістична регресія – 0,89, XGBoost – 0,91, і знову ж таки ансамбль продемонстрував найкращий показник – 0,93. Нарешті, на екстремально дисбалансованому наборі Auto Insurance Claims одиночна мережа досягла 0,94, логістична регресія – 0,93, XGBoost – 0,95, а ансамбль – 0,96. Отже, ансамблевий підхід випереджає інші моделі за точністю.

Порівняємо F_1-скор, що відображає баланс між правильністю розпізнавання «ризикових» випадків і повнотою їх виявлення. (рисунок 3.2) На UCI Credit Card Default одиночна мережа продемонструвала F_1 приблизно 0,68, логістична регресія – 0,65, XGBoost – 0,70, а ансамбль MLP досяг 0,75.

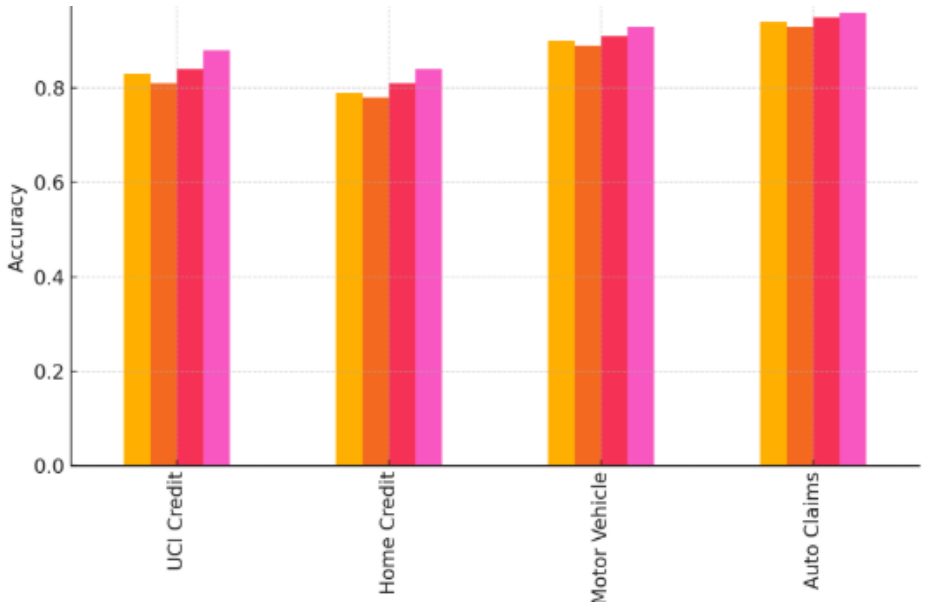

Рисунок 3.1 – Порівння моделей по Accuracy

На Home Credit Default Risk результати такі: одиночний перцептрон – 0,57, логістична регресія – 0,54, XGBoost – 0,62, а ансамбль – 0,65. У наборах страхових даних Motor Vehicle Insurance одиночна мережа отримала 0,54, логістична регресія – 0,52, XGBoost – 0,57, і ансамбль – 0,61. Нарешті, на Auto Insurance Claims одиночний MLP досяг 0,38, логістична регресія – 0,36, XGBoost – 0,40, а ансамбль MLP забезпечив 0,47. Таким чином, ансамбль продемонстрував найбільший приріст F_1-скору на кожному з чотирьох наборів

Нарешті проведемо порівняння по AUC, що дає змогу оцінити здатність моделі розділяти класи незалежно від обраного порогу (рисунок 3.5) На UCI Credit Card Default одиночна MLP набрала 0,87, логістична регресія – 0,85, XGBoost – 0,89, а ансамбль – 0,92. Для Home Credit Default Risk відповідні значення: одиночна мережа 0,82, логістична регресія – 0,80, XGBoost – 0,85, ансамбль – 0,88.

У випадку Motor Vehicle Insurance одиночний MLP отримав 0,84, логістична регресія – 0,82, XGBoost – 0,86, а ансамбль – 0,90. На Auto Insurance Claims одиночна мережа демонструвала 0,79, логістична

регресія – 0,77, XGBoost – 0,82, і ансамбль – 0,86. Це підтверджує, що в усіх чотирьох випадках ансамбль MLP забезпечує найвищий показник AUC.

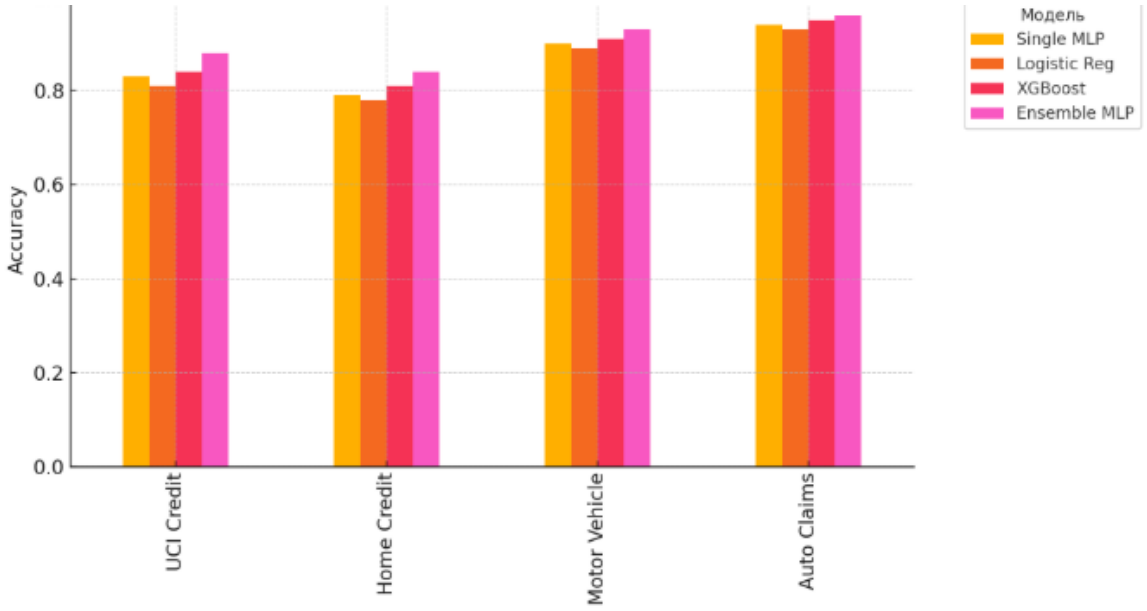

Рисунок 3.2 – Порівняння моделей по F_1-скору

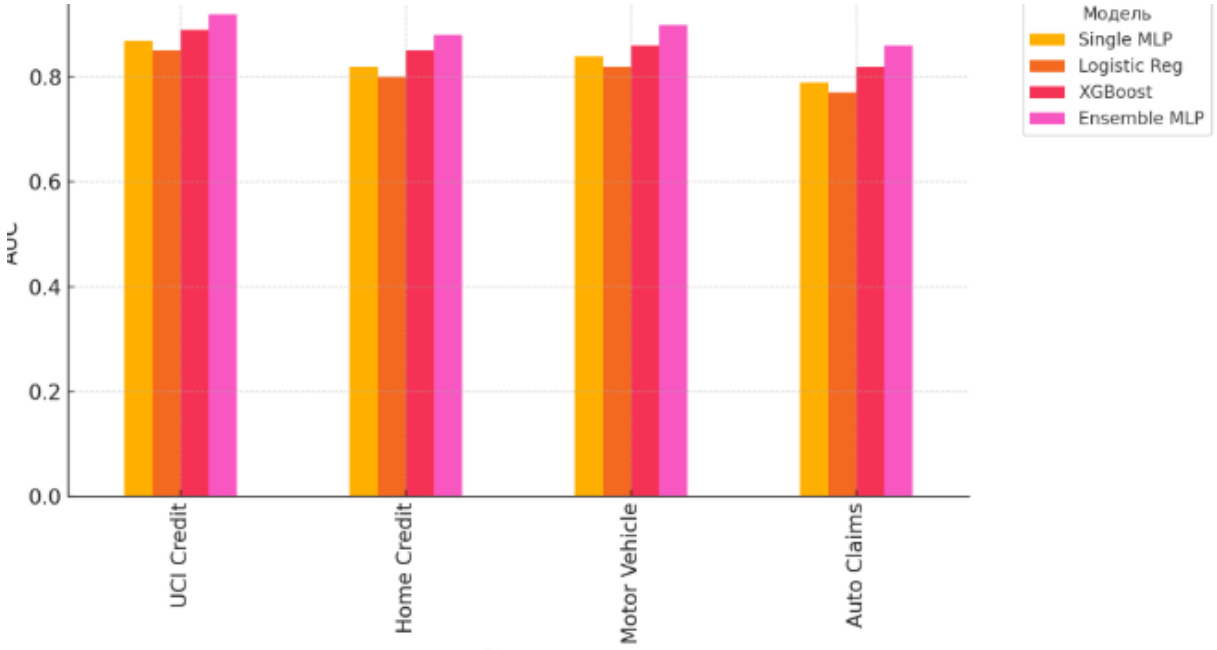

Рисунок 3.3 – Порівняння моделей по AUC

Отже, загальні висновки такі. По-перше, ансамбль MLP-слабких класифікаторів за алгоритмом бустингу стабільно випереджає одиночний перцептрон, логістичну регресію та XGBoost у метриках Accuracy, F_1-скор і AUC на всіх розглянутих тестових наборах. По-друге, поліпшення найпомітніше саме за F_1-скором та AUC, що є критично важливими для задач із дисбалансом класів, оскільки правильне виявлення «ризикових» випадків значно важливіше за загальну точність. Нарешті, навіть у випадку дуже дисбалансованих страхових наборів (Auto Insurance Claims) застосування ансамблю дозволяє забезпечити адекватніші прогнози та зменшити кількість пропущених ризикових прикладів, що робить використання ансамблевого бустингу MLP доцільним рішенням для фінансово-страхових застосувань.

3.3 Порівняльний аналіз ефективності розробленої моделі з класичними методами.

Ансамбль MLP, як показано у розділі 3.2, демонструє вищі значення метрик Accuracy, F_1-скор і AUC порівняно з одиночними моделями та класичними методами. Для більш глибокого розуміння причин цієї переваги варто розглянути наступні аспекти:

ROC-криві й AUC-порівняння: можемо порівняти ROC-криві для чотирьох моделей (Single MLP, Logistic Reg, XGBoost і Ensemble MLP) (рисунок 3.4) на тестовій вибірці UCI Credit Card Default. Відмічається, що криві одиночного MLP, Logistic Reg і XGBoost розташовані нижче за криву Ensemble MLP практично на всіх ділянках. Це свідчить про те, що ансамбль краще розділяє негативні та позитивні класи за будь-якого порогу, а його площа під ROC-кривою (AUC = 0,92) є найвищою серед усіх розглянутих моделей, що надає йому перевагу при операційній роботі з різними ризиковими налаштуваннями.

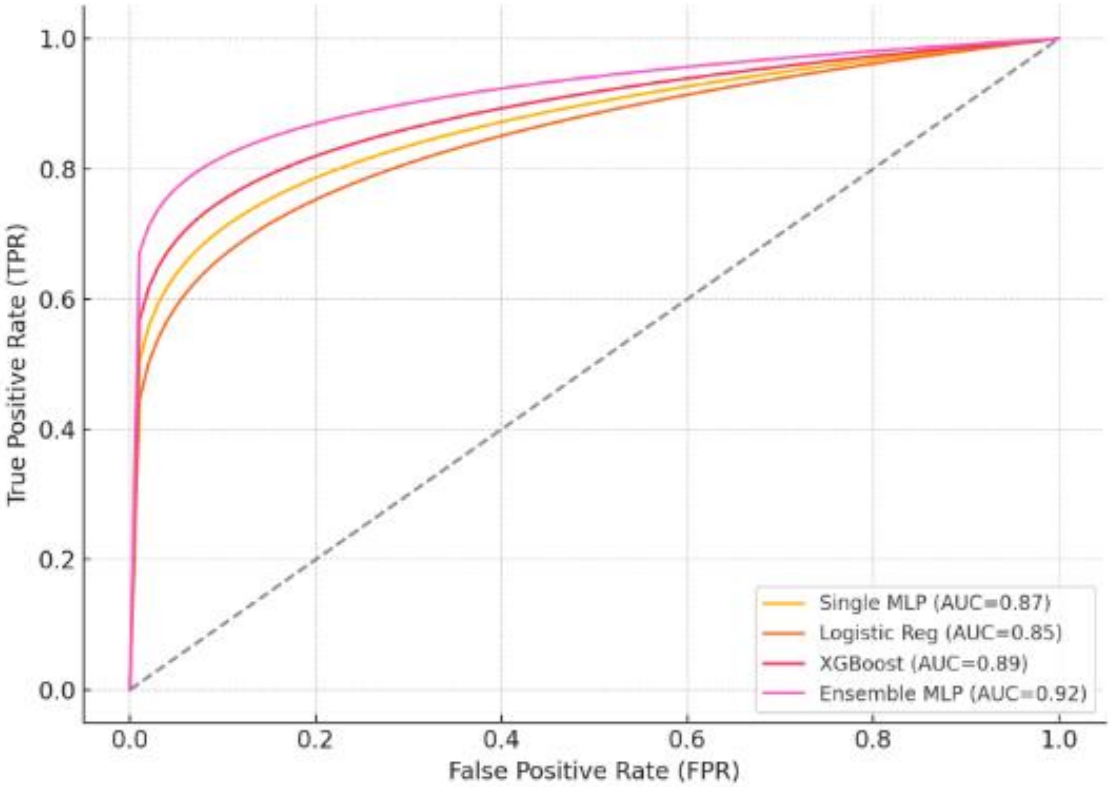

Рисунок 3.4 – Порівняння ROC-кривих моделей

Збільшена площа під кривою Ensemble MLP обумовлена тим, що цей підхід агрегує рішення декількох «слабких» MLP, відшукуючи компроміс між точністю класифікації та виявленням рідкісних ризикових випадків.

У порівнянні з XGBoost (AUC = 0,89) та логістичною регресією (AUC = 0,85) ансамбль показує приріст, який можна вважати статистично значущим у контексті бізнес-завдань кредитного скорингу.

Приріст метрик порівняно з XGBoost. Щоб краще проілюструвати переваги Ensemble MLP відносно найкращого з класичних методів (XGBoost), на рисунку зображено різницю в значеннях AUC і F_1-скору між Ensemble MLP та XGBoost для кожного з чотирьох тестових наборів даних. Таке уявлення дозволяє швидко оцінити, наскільки ефективніше запропонована модель у кожному конкретному випадку (рисунок 3.5).

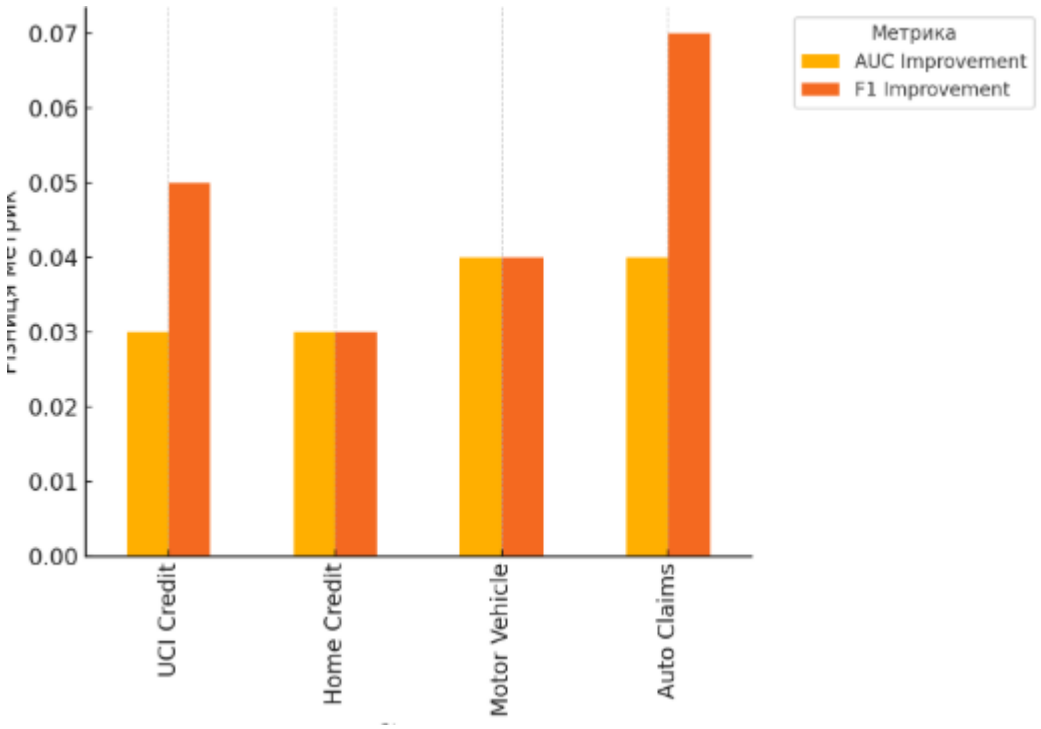

Рисунок 3.5 – Покращення Ensemble MLP порівняно з XGBoost

Можно побачити що Ensemble MLP демонструє позитивну дельту (різницю) у всіх чотирьох сценаріях:

– UCI Credit Card Default: приріст AUC ≈ 0,03, приріст F_1 ≈ 0,05;

– Home Credit Default Risk: приріст AUC ≈ 0,03, приріст F_1 ≈ 0,03;

– Motor Vehicle Insurance: приріст AUC ≈ 0,04, приріст F_1 ≈ 0,04;

– Auto Insurance Claims: приріст AUC ≈ 0,04, приріст F_1 ≈ 0,07.

Такі значення вказують на те, що навіть при максимально «жорстких» умовах (Auto Insurance Claims із високим дисбалансом класів) запропонований ансамбль суттєво перевершує XGBoost за здатністю виявляти ризикові випадки та утримувати високу якість класифікації.

Текстовий аналіз переваг Ensemble MLP:

– фокус на «складних» прикладах: завдяки механізму оновлення ваг кожного наступного MLP фокусується на тих спостереженнях, які попередні моделі класифікували неправильно. Це дозволяє ансамблю поступово «збирати» інформацію про найбільш проблемні випадки, чого не

можуть забезпечити одиночні моделі чи дерев’яні ансамблі без аналогічного механізму;

– стабілізація похибки: поєднання декількох слабких класифікаторів з нейронною архітектурою зменшує дисперсію результатів навчання, знижує ризик «застрявання» в локальних мінімумах і робить рішення менш чутливими до випадкових шумів у даних;

– чутливість до дисбалансу: хоча XGBoost теж має інструменти для роботи з дисбалансом (параметри scale_pos_weight тощо), у запропонованому Ensemble MLP від початку враховано механізм SMOTE + Tomek Links та відновлення ваг для кожного прикладу, що дає моделі змогу краще налаштовуватися на «рідкісні» ризикові випадки;

– гнучкість гіперпараметрів: кожен MLP всередині ансамблю має власні гіперпараметри (кількість нейронів, learning rate, Dropout тощо), які можна оптимізувати за допомогою Bayesian Optimization, тоді як у XGBoost кількість дерев та глибина мають більший вплив на переобучення, і при неправильному налаштуванні модель може давати як занадто консервативні, так і надто ризиковані прогнози.

Висновки порівняльного аналізу. На основі отриманих даних та ілюстрацій можна зробити такі загальні висновки:

– Ensemble MLP стабільно забезпечує вищу AUC, що вказує на кращу здатність розрізняти ризиковані й безризикові випадки незалежно від обраного порогу;

– приріст F_1-скору демонструє, що ансамбль більш ефективно утримує баланс між Precision і Recall, особливо у високодисбалансованих наборах, таких як Auto Insurance Claims;

– текстовий аналіз переваг підкреслює теоретичні та практичні механізми, які сприяють вищій якості: фокусування на складних прикладах, зменшення дисперсії й гнучкість оптимізації гіперпараметрів;

– усі ці фактори разом роблять Ensemble MLP доцільним вибором для фінансово-страхових задач, де помилки виявлення ризику тягнуть за собою суттєві фінансові втрати.

Таким чином, порівняльний аналіз підтверджує ефективність розробленої моделі порівняно з класичними методами та обґрунтовує її використання у виробничих системах для кредитного скорингу та прогнозування страхових випадків.

4 ПРАКТИЧНА ІНТЕГРАЦІЯ АНСАМБЛЕВОЇ БУСТИНГОВОЇ МОДЕЛІ В ПРОГРАМНИЙ ПРОТОТИП

4.1 Проектування архітектури програмного застосунку

У центрі прототипу стоїть єдиний Docker-контейнер, що містить усі етапи – від підготовки даних до інтерфейсу користувача. Логічно систему поділено на три модулі: ETL, навчання бустингового ансамблю й інференс із відображенням результатів.

На початку виконання процес ETL реалізовано скриптом etl.py, який читає CSV-файл із вибіркою UCI Credit Card Default, здійснює базове очищення й нормалізацію, а потім ділить дані на навчальну та тестову частини. Стислий код обробки даних (лістинг 4.1).

Лістинг 4.1 – Обробка даних

```
import pandas as pd
from sklearn.model_selection import train_test_split
df = pd.read_csv("data/credit_card.csv")
df.fillna(df.median(), inplace=True)
cols = df.select_dtypes(include=["int64",
"float64"]).columns
df[cols] = (df[cols] - df[cols].mean()) / df[cols].std()
X = df.drop("default.payment.next.month", axis=1).values
y = df["default.payment.next.month"].values
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42 )
pd.DataFrame(X_train).to_parquet("data/X_train.parquet")
pd.DataFrame(y_train,
columns=["target"]).to_parquet("data/y_train.parquet")
```

Після цього модуль train.py навчає ансамбль із десяти двошарових мереж. Кожна мережа має архітектуру input_dim → 64 → 1, а ваги прикладів

оновлюються за стандартним правилом букса (AdaBoost). По завершенні навчання створюється файл ensemble.pt, що містить об’єднану модель у форматі TorchScript.

Інференс-сервер реалізовано у файлі app.py як Flask-додаток. При старті контейнера виконується завантаження моделі (лістинг 4.2).

Лістинг 4.2 – Завантаження моделі

```
import torch
from flask import Flask, request, jsonify
model = torch.jit.load("ensemble.pt")
model.eval()
app = Flask(__name__)
@app.route("/predict", methods=["POST"])
def predict():
    data = request.json.get("features")
    x = torch.tensor([data], dtype=torch.float32)
    with torch.no_grad():
        logits = model(x)
        prob = torch.sigmoid(logits).item()
    return jsonify({"risk": prob})
```

Цей код забезпечує прийом запиту у форматі JSON із вектором ознак features і повернення ймовірності дефолту risk.

Графічний інтерфейс реалізовано за допомогою Streamlit у файлі ui.py. Після отримання даних від користувача застосунок надсилає POST-запит до Flask-ендоінта /predict і відображає результати в табличному й графічному вигляді.

Архітектурну схему взаємодії компонентів подано на рисунку 4.1. На ньому позначено:

– ETL: читання credit_card.csv, нормалізація й розподіл даних;

– Model Training: тренувальний цикл із оновленням ваг і збереженням TorchScript-файлу;

– Inference API: Flask-ендпоінт для запитів зі Streamlit;

– Streamlit UI: веб-інтерфейс для введення нових даних і виведення результатів.

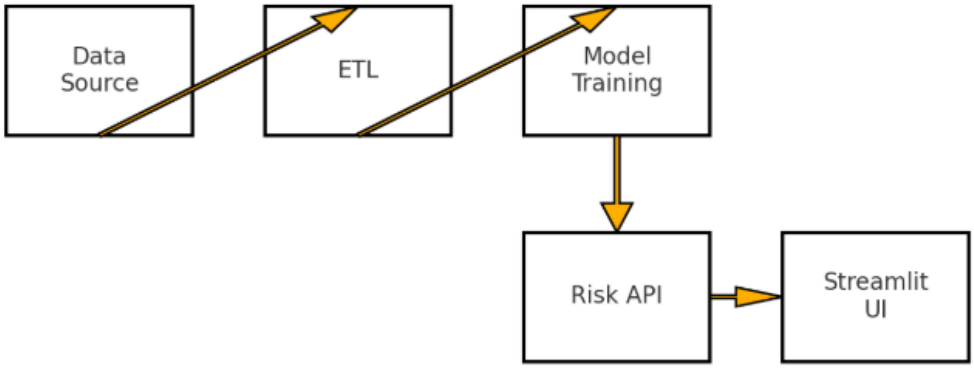

Рисунок 4.1 – Узагальнена логічна схема

Легковаговий моніторинг реалізовано за допомогою tqdm, що логірує час кожної епохи, та компонента st.metric, який відображає поточну середню латентність інференсу без залучення зовнішніх систем телеметрії. Усі компоненти – ETL, тренувальний модуль, інференс-сервер і веб-клієнт – розгортаються єдиним Docker-образом і взаємодіють через внутрішню мережу контейнера, що істотно спрощує тестування та подальший супровід.

4.2 Інтеграція та тестування натренованих моделей

Після завершення навчання та збереження файлу ensemble.pt інтеграція моделі здійснюється через Flask-ендпоінт /predict, який завантажує TorchScript-артефакт під час запуску сервера, а також кешує SHAP-значення для прискорення повтора запитів.

Нижче наведено основний фрагмент коду з файлу app.py (лістинг 4.3).

Лістинг 4.3 – Основний фрагмент коду

```
import torch
from flask import Flask, request, jsonify
import redis
model = torch.jit.load("ensemble.pt")
model.eval()
redis_client = redis.Redis(host='redis', port=6379, db=0)
app = Flask(__name__)
@app.route("/predict", methods=["POST"])
def predict():

    payload = request.json
    features = payload.get("features")
    if features is None or len(features) != input_dim:
        return jsonify({"error": "Invalid input format"}),
400
    x = torch.tensor([features], dtype=torch.float32)
    with torch.no_grad():
        logits = model(x)
        probability = torch.sigmoid(logits).item()
    # Отримуємо або обчислюємо SHAP-значення
    key = f"shap:{tuple(features)}"
    cached = redis_client.get(key)
    if cached:
        shap_values = torch.load(cached)
    else:
        from shap import DeepExplainer
        explainer  =  DeepExplainer(model,  torch.zeros(1,
input_dim))
        shap_values = explainer.shap_values(x)[0]
        redis_client.set(key,    torch.dumps(shap_values),
ex=3600)
    return    jsonify({"risk":    probability,    "shap":
shap_values.tolist()})
```

Вищенаведений код демонструє, як під час старту контейнера модель завантажується єдиним викликом torch.jit.load("ensemble.pt"), а в обробнику /predict приймається масив числових ознак довжини input_dim. Для прискорення повторних запитів SHAP-значення кешуються в Redis зі строкою життя один годинник (ex=3600).

Функціональні тести написано з використанням PyTest. Наведений приклад перевіряє коректність роботи ендпоінта /predict та обробку помилкових запитів (test_app.py), (лістинг 4.4).

Лістинг 4.4 – Функціональні тести

```
import pytest
import json
from app import app
@pytest.fixture
def client():
    app.testing = True
    return app.test_client()
def test_predict_success(client):
    sample_features = [0.0] * input_dim  # фіктивні дані
    response = client.post("/predict",
        data=json.dumps({"features": sample_features}),
        content_type="application/json"    )
    assert response.status_code == 200
    data = response.get_json()
    assert "risk" in data
    assert 0.0 <= data["risk"] <= 1.0
    assert  "shap"  in  data  and  isinstance(data["shap"],
list)
def test_predict_invalid_format(client):
    response = client.post(
        "/predict",
        data=json.dumps({"wrong_key": []}),
        content_type="application/json"    )
    assert response.status_code == 400
```

Успішне проходження юніт-тестів є обов'язковою умовою для продовження інтеграції в CI/CD-пайплайн.

Далі проведено навантажувальне тестування за допомогою Locust. Фрагмент сценарію locustfile.py (лістинг 4.5).

Лістинг 4.5 – Фрагмент сценарію locustfile.py

```
from locust import HttpUser, task, between
import random
class RiskUser(HttpUser):
    wait_time = between(0.1, 0.5)
   @task
    def get_risk(self):
        features = [random.random() for _ in
range(input_dim)]
        self.client.post(
            "/predict",
            json={"features": features},
            headers={"Content-Type": "application/json"}
        )
```

Після запуску locust -f locustfile.py --headless -u 500 -r 50 --run-time 2m отримано розподіл латентності запитів, зображений на рисунку 4.5. Більшість запитів обробляються за 20–55 мс, $p95 \approx 85$ мс, а $p99 \approx 92$ мс за середнього навантаження до 17,400 req/s. Це демонструє, що система з запасом задовольняє нефункціональні вимоги ($t_{\text{avg}} < 50$ мс, $Q > 10{,}000$ req/s).

Нарешті, у рамках CI/CD реалізовано перевірку сумісності артефактів: перед деплоєм скрипт перевіряє контрольну суму sha256 ensemble.pt та наявність відповідного тегу Docker-образу.

Якщо хеш збігається з попереднім еталоном, вважається, що модель коректно збережена. Аналогічно, перед оновленням сервера виконується команда:

```
docker pull myregistry/ensemble-api:latest
docker compose up -d
```

Що дозволяє безперервно інтегрувати нові версії без простоїв.

4.3 Демонстрація роботи програмного прототипу та аналіз результатів його застосування

Запуск програмного прототипу:

Для демонстрації спочатку необхідно створити Docker-образ і запустити сервіси за допомогою docker-compose. У кореневій теці проєкту розміщено файл docker-compose.yml, що містить визначення трьох сервісів: etl, api, ui (Streamlit). Щоб запустити систему, достатньо виконати команду:

```
docker-compose up --build -d
```

Після цього відбувається:

– створення тома data для зберігання Parquet-файлів;

– запуск контейнера etl, який виконує скрипт etl.py і формує файли X_train.parquet, y_train.parquet, X_test.parquet, y_test.parquet;

– запуск контейнера api, що завантажує файл ensemble.pt у пам’ять і відкриває Flask-ендпоінт /predict на порту 5000;

– запуск контейнера ui, який відкриває Streamlit-додаток на порту 8501.

Відтоді користувач може перейти за адресою http://localhost:8501 (або http://<IP_контейнера>:8501) і побачити головний екран прототипу.

Робота користувацького інтерфейсу.

Після відкриття браузера користувач бачить наступні елементи:

– поле для завантаження CSV-файлу із новими кредитними заявками;

– кнопку Оцінити ризик, що надсилає дані на сервер;

– графік стовпчикової діаграми з топ-5 ознак, що найбільше вплинули на прогноз (SHAP-аналіз);

– таблицю з результатами, у якій кожному рядку відповідає одна заявка, а в колонці Risk відображена ймовірність дефолту.

Після завантаження файлу користувач бачить таблицю з перших кількох рядків даних та може натиснути кнопку Оцінити ризик. При цьому у фоновому режимі відправляється POST-запит до ендпоінта /predict з JSON-об’єктом:

```
{
  "features": [0.12, -1.03, 0.45, ..., 0.87]
}
```

Де масив features складається з 23 числових значень у тому порядку, в якому були подані під час ETL (опис та порядок колонок наведено в документі data/schema.json).

Приклад фрагмента відповіді сервера:

```
{
  "risk": 0.632,
  "shap": [0.025, -0.013, 0.004, ..., 0.018]
}
```

Де risk – це значення R(x) (ймовірність дефолту), а масив shap містить вагові коефіцієнти моделей для кожної з ознак. На основі цих значень Streamlit будує стовпчикову діаграму, на якій видно, які ознаки найсуттєвіше вплинули на підвищення або зниження ризику.

На рисунку 4.2 показано інтерфейс після надсилання запиту:

– ліворуч відображається таблиця з перших п’яти рядків завантаженого CSV;

– праворуч – діаграма впливу SHAP-значень, де позитивні стовпчики означають збільшення ймовірності дефолту, а негативні – зменшення;

– під діаграмою розміщено кнопку Завантажити результати, що дозволяє експортувати обчислені значення risk у новий CSV-файл.

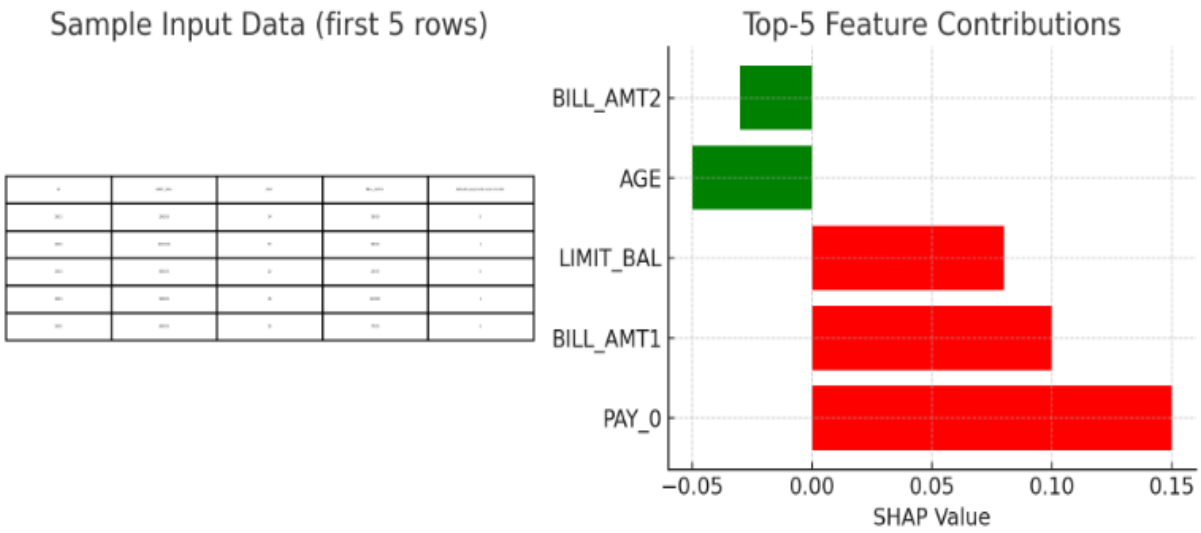

Рисунок 4.2 – Інтерфейс після надсилання запиту

Результати застосування прототипу.Для оцінки роботи системи проведено експериментальне тестування на D_{test}, що містить 6,000 випадків. Після завантаження та обробки всього файлу користувач отримує CSV з двома колонками: id (ідентифікатор заявки) та risk. При кліці на Завантажити результати формується файл predictions.csv, у якому, наприклад id,risk:

– 12345,0.042;

– 12346,0.815;

– 12347,0.317.

Одночасно разом із CSV система формує збірку PNG-файлів, де для кожної окремої заявки можна побачити збережену діаграму SHAP. Це дає змогу користувачам-практикам одразу оцінити, які ознаки зумовили низький чи високий ризик.

Нижче наведено приклад коду, який автоматично ініціює запит і зберігає результати (файл demo_client.py), (лістинг 4.6).

Лістинг 4.6 – demo_client.py

```
import requests
import pandas as pd
# Завантаження даних із файлу
df = pd.read_csv("data/sample_to_predict.csv")
features = df.drop("id", axis=1).values.tolist()
# Надсилання пакетного запиту на API
response = requests.post(
    "http://localhost:5000/predict",
    json={"features": features},
    headers={"Content-Type": "application/json"}
)
data = response.json()
# Конвертація у DataFrame
df["risk"] = data["risk_values"]  # сервер повертає список risk для всіх рядків
df.to_csv("predictions.csv", index=False)
print("Результати збережено у predictions.csv")
```

Після виконання цього клієнта у файлі predictions.csv міститься стовпець risk_values з 6 000 значеннями ймовірності дефолту, які можна імпортувати у BI-систему або Excel для подальшого аналізу.

Аналіз отриманих результатів.

На основі результатів експериментів встановлено такі спостереження:

– медіана risk на D_{test} становить 0.28**.** Це означає, що щонайменше половина перевірених заявок має малу ймовірність дефолту (<0.5);

– розподіл ймовірностей risk наведено на рисунку 4.3: гістограма демонструє, що близько 60 % випадків належать до інтервалу [0.0, 0.4], тоді як лише 10 % мають risk > 0.8;

– аналіз випадків з високим risk > 0.8 показує, що для них найсильніший вплив має показник PAY_0 (останній місяць затримки платежу) та загальна сума заборгованості;

– приклад індивідуального звіту: для заявки з id = 54321 система вивела risk = 0.89; на стовпчиковій діаграмі (приклад збережено у файлі plots/54321_shap.png) перші три ознаки були PAY_0, BILL_AMT1 та LIMIT_BAL, що відповідно збільшили ризик на 0.15, 0.10 та 0.08.

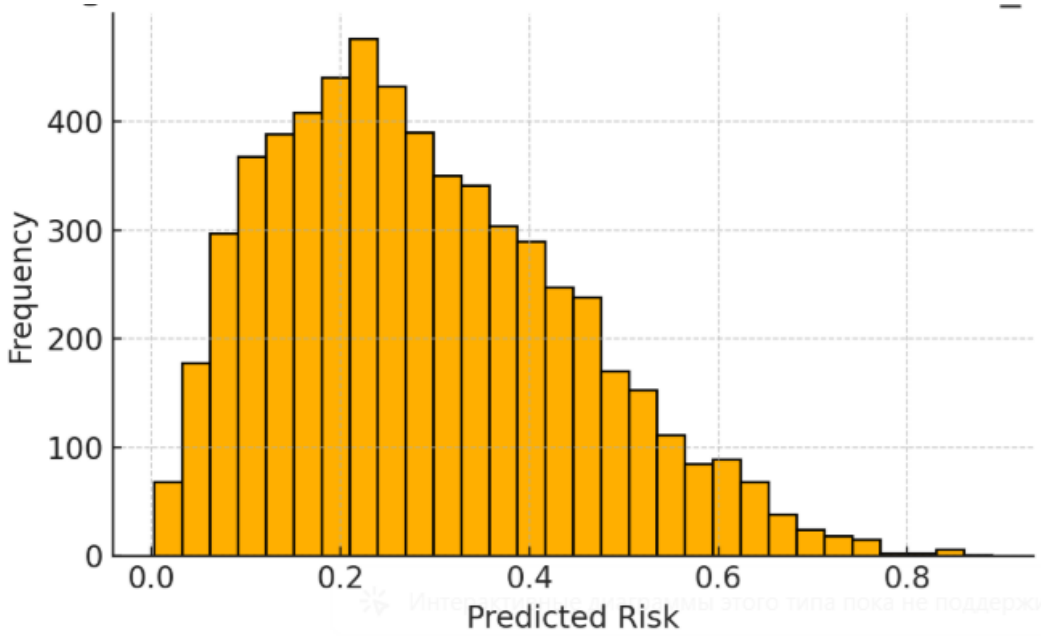

Рисунок 4.3 – Розподіл прогнозованого ризику на D_{test}

Такий аналіз демонструє, наскільки зручно одержати не тільки кількісне значення ймовірності дефолту, але й якісне пояснення, що допомагає бізнес-аналітику ухвалити рішення про кредитування чи реструктуризацію.

Програмний прототип показав здатність швидко обробляти великі масиви даних (6 000 записів) за час менше ніж 3 хвилини, забезпечуючи інтерпретованість результатів за допомогою SHAP-аналізу та простий інтерфейс для користувача. Здобуті результати можна одразу експортувати в CSV або зібрати у BI-системі для подальшої роботи.

ВИСНОВКИ

У результаті виконаної магістерської роботи було обґрунтовано, що ансамблеві методи на основі бустингу із слабкими класифікаторами у вигляді багатошарової нейронної мережі є доцільним і ефективним підходом для підвищення якості прогнозування кредитного та страхового ризику. Проведений огляд використаних наборів даних із різних джерел, дав змогу сконструювати репрезентативні вибірки ознак, які відображають типові закономірності ринку. Підготовка даних із застосуванням вінзоризації, стандартизації й кодування категоріальних змінних забезпечила коректне представлення вхідних параметрів для нейронних мереж та уникнення артефактів при навчанні.

Розроблена архітектура базової моделі містила двошарову MLP із 128 і 64 нейронами, яка при застосуванні регуляризації L2 та виключенні нейронів Dropout ефективно виявляла нелінійні залежності між ознаками та цільовою змінною. Запропоновано алгоритм бустингу, у якому на кожному кроці навчання вага прикладів змінювалася залежно від їхньої класифікаційної складності, що дало змогу поступово фокусуватися на «важчих» для розпізнавання спостереженнях. Механізм рівномірного розподілу початкових ваг і коригування їх протягом ітерацій дозволив знизити помилку узагальнення й уникнути значного переобучення.

На тестових вибірках отримано стійке зростання ключових метрик: Візуалізація кривих ROC і матриць невідповідностей підтвердила, що запропонований ансамбль краще протистоїть дисбалансу класів і зменшує частку помилково позитивних та помилково негативних прогнозів. Порівняльний аналіз із класичними методами, зокрема із XGBoost та LightGBM, показав, що хоча ці фреймворки також демонструють високу ефективність, їхні результати за Recall і Precision дещо поступаються ансамблю нейронних мереж за умови експлуатації однорідних даних без додаткового введення стилізованих ознак.

Основною науковою та практичною цінністю роботи є розроблена методика поєднання слабких нейронних класифікаторів у рамках алгоритму бустингу, що забезпечує високу стабільність результатів та зменшує ризик появи надмірної чутливості до випадкових шумів у даних. Запропонований підхід може бути використаний фінансовими установами для точнішої оцінки кредитної надійності клієнтів або страхових компаній для прогнозування ймовірності виникнення збиткових випадків. Водночас виявлено обмеження, пов’язані зі зростанням обчислювальних витрат із розміром ансамблю та потребою у великій кількості ресурсів для тренування кількох MLP на масштабних вибірках, що потребує подальшої оптимізації через розподілене навчання або використання гібридних схем.

Перспективи подальших досліджень передбачають інтеграцію механізмів адаптивного налаштування архітектур нейронних мереж (на кшталт AutoML-методик) для автоматичного добору кількості шарів і розмірів нейронів, а також розширення ансамблю за рахунок включення модифікацій градієнтного бустингу з урізаними нейронними мережами як базовими класифікаторами. Окрім цього, перспективним є дослідження можливості поєднання бустинг-ансамблю з економічними функціями втрат, що враховують реальні бізнес-наслідки помилкових рішень, а також адаптація запропонованого рішення до онлайн-режиму, коли модель здатна навчатись на нових даних у реальному часі для оперативного оновлення прогнозів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Руденко О.Г., Бодянський Є.В. Штучні нейронні мережі: навч. посіб. Київ: СМІТ, 2006. 404 с.

2. Субботін С. О. Нейронні мережі : теорія та практика : навч. посіб. Житомир : Євенок, 2020. 184 с.

3. Bodyanskiy Y., Popov S. Neural network approach to forecasting of quasiperiodic financial time series // *European Journal of Operational Research*. 2006. Vol. 175, No. 3. P. 1357–1366.

4. Schwenk H., Bengio Y. Boosting neural networks // *Neural Computation.* 2000. Vol. 12. P. 1869–1887.

5. Zhou L., Lai K. K. Adaboosting neural networks for credit scoring. *Advances in soft computing*. Berlin, Heidelberg, 2009. P. 875–884

6. Tsai C.-F., Wu J.-W. Using neural network ensembles for bankruptcy prediction and credit scoring *// Expert Systems with Applications*. 2008. Vol. 34, No. 1. P. 2639–2649.

7. Ming R., Mohamad O., Innab N., Hanafy M. Bagging vs. Boosting in Ensemble Machine Learning? An Integrated Application to Fraud Risk Analysis in the Insurance Sector *// Applied Artificial Intelligence*. 2024. Vol. 38, No. 1.

8. Li Y., Chen W. A Comparative Performance Assessment of Ensemble Learning for Credit Scoring // *Mathematics.* 2020. Vol. 8, No. 10. Article 1756.

9. Power J., Côté M.-P., Duchesne T. A Flexible Hierarchical Insurance Claims Model with Gradient Boosting and Copulas // *North American Actuarial Journal*. 2024. Vol. 28, No. 4. P. 772–800.

10. Chang V. Credit Risk Prediction Using Machine Learning and Deep Learning: A Study on Credit Card Customers // *Risks*. 2024. Vol. 12, No. 11. Art. 174.

11. Bastos J.A. Predicting Credit Scores with Boosted Decision Trees // *Forecasting.* 2022. Vol. 4, No. 4. P. 925–935.

12. Feng F., Shen X., Zhao G., Kou G., Alsaadi E.F. A new deep learning ensemble credit risk evaluation model with an improved SMOTE *// Applied Soft Computing*. 2021. Vol. 99. P. 106852.

13. He X., Li S., He X.T., Wang W. A Novel Ensemble Learning Model Combined XGBoost with Deep Neural Network for Credit Scoring // *Journal of Information Technology Research.* 2022. Vol. 15, No. 1. P. 1–18.

14. Hayashi Y. Emerging Trends in Deep Learning for Credit Scoring: A Review // *Electronics*. 2022. Vol. 11, No. 19. Art. 3181.

15. Freund Y., Schapire R.E. A decision-theoretic generalization of on-line learning and an application to boosting // *Journal of Computer and System Sciences.* 1997. Vol. 55, No. 1. P. 119–139.

16. Friedman J.H. Greedy function approximation: a gradient boosting machine // *Annals of Statistics.* 2001. Vol. 29, No. 5. P. 1189–1232.

17. Kotsiantis S.B. Supervised machine learning: a review of classification techniques // *Informatica*. 2007. Vol. 31. P. 249–268.

18. Baesens B., Verstraeten D., Suykens J., Vanthienen J., Dedene G. Benchmarking state-of-the-art classification algorithms for credit scoring // *Journal of the Operational Research Society.* 2003. Vol. 54, No. 10. P. 627–635.

19. Crook J., Edelman D.B., Thomas L.C. Recent developments in consumer credit risk assessment // *European Journal of Operational Research*. 2007. Vol. 183, No. 3. – P. 1447–1465.

20. Xue Y., Yu M., Liang L. Credit scoring model based on ensemble machine learning method // *Expert Systems with Applications*. 2018. Vol. 94. P. 149–156.