Міністерство освіти і науки України
Луцький національний технічний університет
Факультет робототехніки та штучного інтелекту
Кафедра штучного інтелекту та математичного моделювання

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ
«БАКАЛАВР»

«ДОСЛІДЖЕННЯ ТА РОЗРОБКА МЕТОДІВ ВИЯВЛЕННЯ ШАХРАЙСЬКИХ ФІНАНСОВИХ ТРАНЗАКЦІЙ В УМОВАХ ДИСБАЛАНСУ КЛАСІВ»

RESEARCH AND DEVELOPMENT OF METHODS FOR DETECTING FRAUDULENT FINANCIAL TRANSACTIONS IN CONDITIONS OF CLASS IMBALANCE

Спеціальність 113 Прикладна математика
(шифр і назва спеціальності)

освітня програма «Штучний інтелект та аналіз масивів даних»
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи ПРМ -41

Пєсков Ігор Анатолійович

_____
(підпис)

Керівник:
к.т.н., доцент
Приходько Олексій Сергійович

_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«___» _____ 20__ р.
к.т.н., доцент
Гарант освітньої програми:
Приходько Олексій Сергійович

_____
(підпис)

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *архітектури, будівництва та дизайну*
Кафедра *прикладної математики та механіки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *11 Математика і статистика*
Спеціальність *113 Прикладна математика*
Освітня програма *Штучний інтелект та аналіз масивів даних*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Мікуліч О.А.
«___» _____2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Пєсков Ігор Анатолійович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Дослідження та розробка методів виявлення шахрайських фінансових транзакцій в умовах дисбалансу класів
Керівник роботи: *Приходько Олексій Сергійович*
затверджені наказом закладу вищої освіти від «31» грудня 2025 р. № 557/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 04.06.2026 р.

3. Вихідні дані до роботи *профільні публікації з штучного інтелекту та математичного моделювання в межах досліджуваної проблематики; релевантні набори даних; математичні моделі цільових процесів; технічна документація Python-бібліотек та методичні вказівки до виконання кваліфікаційної роботи бакалавра*

4. Зміст пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз предметної області
Постановка задачі та вибір методів
Практична реалізація
Отримання та аналіз результатів

5. Перелік графічного (ілюстративного) матеріалу:
Презентація роботи (слайди): об'єкт, предмет, мета та завдання дослідження; аналіз предметної області; математичні та алгоритмічні моделі, результати експериментальних досліджень (метрики та візуалізація роботи алгоритму); загальні висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
1 розділ	*Приходько О.С., доцент кафедри*		
2 розділ	*Приходько О.С., доцент кафедри*		
3 розділ	*Приходько О.С., доцент кафедри*		
4 розділ	*Приходько О.С., доцент кафедри*		
Висновки	*Приходько О.С., доцент кафедри*		

7. Дата видачі завдання «___» _____ 202__ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми*	*до 01.03.2026*	
2.	*Перший розділ*	*до 5.03.2026*	
3.	*Другий розділ*	*до 01.04.2026*	
4.	*Третій розділ*	*до 10.04.2026*	
5.	*Четвертий розділ*	*до 20.04.2026*	
6.	*Висновки*	*до 28.04.2026*	
7.	*Формування списку використаних джерел*	*до 05.05.2026*	
8.	*Оформлення ілюстративного матеріалу*	*до 09.05.2026*	
9.	*Нормоконтроль*	*до 20.05.2026*	
10.	*Інструментальна перевірка на академічний плагіат*	*до 02.062026*	*Показник запозичень тексту ____ %*
11.	*Представлення кваліфікаційної роботи бакалавра до захисту*	*до 04.06.2026*	

Здобувач вищої освіти _____ (Пєсков І. А.)
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (Приходько. О. С.)
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Пєсков Ігор Анатолійович. Дослідження та розробка методів виявлення шахрайських фінансових транзакцій в умовах дисбалансу класів.

Кваліфікаційна робота бакалавра ОП «Штучний інтелект та аналіз масивів даних» спеціальності 113/F1 Прикладна математика. Луцький національний технічний університет. Луцьк, 2026.

У роботі проведено дослідження методів інтелектуального аналізу даних для виявлення шахрайської активності у децентралізованих блокчейн-мережах. Встановлено, що псевдоанонімний характер криптовалютних транзакцій та екстремальний дисбаланс класів у наявних наборах даних (зокрема Elliptic Data Set) суттєво ускладнюють роботу класичних систем протидії відмиванню коштів.

Метою дослідження є підвищення ефективності виявлення шахрайських операцій шляхом розробки та впровадження гібридної моделі класифікації. Для вирішення поставлених завдань застосовано комплексний підхід, що включає часову валідацію даних для уникнення витоку інформації, алгоритм синтетичного балансування вибірки SMOTE та інтеграцію глибокого навчання з ансамблевими методами.

Розроблена гібридна модель базується на поєднанні алгоритму градієнтного бустингу XGBoost та нейромережевого автоенкодера. Автоенкодер, навчений на легітимних транзакціях, використовується для генерації мета-ознаки помилки реконструкції, що дозволяє моделі ефективніше ідентифікувати аномальні патерни.

Експериментально доведено, що запропонований гібридний метод забезпечує найвищі показники якості класифікації: значення F1-score досягло 0,8093, а точність (Precision) для шахрайського класу підвищилася до 91,30%, що на 1,45% перевищує результати базових моделей. Результати роботи можуть бути використані в аналітичних системах моніторингу криптовалютних транзакцій для мінімізації фінансових ризиків та оптимізації роботи AML-офіцерів.

Ключові слова: *виявлення шахрайства, блокчейн, машинне навчання, XGBoost, автоенкодер, дисбаланс класів, SMOTE, AML, Elliptic Data Set.*

ABSTRACT

Igor Anatolyevich Peskov. Research and development of methods for detecting fraudulent financial transactions in conditions of class imbalance.

Bachelor's qualification work OP "Artificial Intelligence and Dataset Analysis" specialty 113/F1 Applied Mathematics. Lutsk National Technical University. Lutsk, 2026.

The work conducted a study of methods for intelligent data analysis to detect fraudulent activity in decentralized blockchain networks. It was found that the pseudo-anonymous nature of cryptocurrency transactions and the extreme class imbalance in existing data sets (in particular, the Elliptic Data Set) significantly complicate the work of classical anti-money laundering systems.

The purpose of the study is to increase the efficiency of detecting fraudulent transactions by developing and implementing a hybrid classification model. To solve the tasks, a comprehensive approach was applied, including temporal data validation to avoid information leakage, the SMOTE synthetic sample balancing algorithm, and the integration of deep learning with ensemble methods.

The developed hybrid model is based on a combination of the XGBoost gradient boosting algorithm and a neural network autoencoder. The autoencoder trained on legitimate transactions is used to generate a meta-feature of the reconstruction error, which allows the model to more effectively identify anomalous patterns.

It was experimentally proven that the proposed hybrid method provides the highest classification quality indicators: the F1-score value reached 0.8093, and the accuracy (Precision) for the fraudulent class increased to 91.30%, which is 1.45% higher than the results of the base models. The results of the work can be used in analytical systems for monitoring cryptocurrency transactions to minimize financial risks and optimize the work of AML officers.

Keywords: *fraud detection, blockchain, machine learning, XGBoost, autoencoder, class imbalance, SMOTE, AML, Elliptic Data Set*.

ЗМІСТ

ВСТУП

Актуальність теми. Розвиток цифрових фінансових технологій та стрімке зростання ринку криптовалют суттєво трансформували підходи до здійснення фінансових операцій. Активи, створені на базі блокчейн-мереж, зокрема Ethereum, характеризуються високою ліквідністю, децентралізованістю та псевдоанонімністю. Це створює сприятливе середовище не лише для інновацій, але й для фінансових зловживань. Оскільки транзакції відбуваються у відкритих мережах при ускладненій ідентифікації суб'єктів, набули поширення такі види шахрайства, як відмивання коштів, фінансові піраміди, маніпуляції ринком та використання міксерів.

Виявлення таких операцій суттєво ускладнюється екстремальним дисбалансом класів у даних: частка шахрайських транзакцій зазвичай становить менше 1% від загального обсягу. Це призводить до неефективності класичних алгоритмів машинного навчання, які схильні ігнорувати міноритарний клас задля максимізації загальної точності. Отже, дослідження та розробка ефективних методів виявлення шахрайських транзакцій в умовах сильного дисбалансу класів є актуальним науково-практичним завданням.

Мета і завдання дослідження. Метою кваліфікаційної роботи є дослідження та розробка програмного рішення для виявлення шахрайських транзакцій у криптовалютному середовищі на основі алгоритмів машинного навчання з урахуванням проблеми дисбалансу класів. Для досягнення поставленої мети сформульовано такі завдання:

1. Проаналізувати предметну область та існуючі методи виявлення фінансових аномалій у блокчейн-мережах.
2. Дослідити вплив дисбалансу класів на якість моделей машинного навчання та визначити оптимальні метрики оцінювання (Precision, Recall, F1-score, PR-AUC).
3. Виконати розвідувальний аналіз та попередню обробку реального набору даних криптовалютних транзакцій.

4. Розробити та програмно реалізувати пайплайн обробки даних із застосуванням методів балансування вибірки.

5. Побудувати гібридну модель виявлення аномалій, що комбінує ансамблеві алгоритми класифікації та нейромережеві підходи.

6. Провести експериментальне дослідження, виконати порівняльний аналіз результатів та оцінити економічну ефективність розробленої системи.

Об'єкт дослідження – процеси аналізу масивів даних у системах виявлення фінансових аномалій криптовалютних мереж.

Предмет дослідження – моделі та алгоритми машинного навчання для виявлення шахрайських фінансових транзакцій в умовах дисбалансу класів.

Методи дослідження. У роботі використано загальнонаукові та спеціальні методи: системний аналіз (для вивчення предметної області), методи теорії ймовірностей та математичної статистики (для обробки даних), методи машинного навчання (градієнтний бустинг, автоенкодери) , методи балансування вибірки (зокрема алгоритми сімейства SMOTE), а також методи оцінювання ефективності алгоритмів (крос-валідація).

Наукова новизна отриманих результатів полягає у дослідженні гібридного підходу, який поєднує ансамблеві методи (XGBoost) з моделями виявлення аномалій (Autoencoder), а також в адаптації алгоритмів балансування вибірки до специфіки багатовимірних графових даних криптовалютних транзакцій.

Практичне значення одержаних результатів. Розроблена модель відзначається високою обчислювальною ефективністю і може бути інтегрована в системи моніторингу транзакцій (AML-системи) криптовалютних бірж з метою своєчасного блокування шахрайських операцій та зниження фінансових ризиків.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ДАНИХ

1.1. Типологія шахрайства та фінансових зловживань у криптовалютних мережах

Стрімке впровадження децентралізованих фінансових систем та технології блокчейн докорінно змінило парадигму електронних платежів. Відкритість, незмінність даних та відсутність єдиного централізованого органу контролю забезпечують високу швидкість та прозорість транзакцій. Проте псевдоанонімний характер таких мереж створює сприятливі умови для фінансових зловживань [1]. Суб'єкти мережі ідентифікуються виключно за допомогою криптографічних адрес, що суттєво ускладнює процес встановлення реальних осіб, які стоять за транзакціями.

Загальну сукупність фінансових злочинів у криптовалютному середовищі можна класифікувати на кілька основних категорій. Перша категорія охоплює операції з відмивання коштів (money laundering), коли нелегально отримані активи пропускаються через серію транзакцій або спеціалізовані сервіси-міксери для приховування їхнього походження [2]. Друга категорія стосується безпосередньо шахрайських схем, серед яких домінують фінансові піраміди (Ponzi schemes), фішинг, а також вимагання за допомогою шкідливого програмного забезпечення (ransomware).

Для протидії цим явищам фінансові установи та криптовалютні біржі впроваджують системи протидії відмиванню коштів (AML – Anti-Money Laundering). Класичні AML-системи переважно базуються на евристичних правилах (rule-based systems) та чорних списках відомих підозрілих адрес. Однак такий підхід характеризується низькою здатністю до узагальнення: зловмисники постійно генерують нові адреси, що робить статичні правила неефективними. Це зумовлює необхідність переходу від детермінованих алгоритмів до

інтелектуальних систем на базі машинного навчання, здатних виявляти складні та приховані патерни поведінки у великих масивах даних [3].

1.2. Структура фінансових транзакцій та специфіка ознакового простору

На відміну від традиційних банківських систем, де транзакції розглядаються переважно як незалежні події у табличному форматі, блокчейн-мережі мають яскраво виражену графову структуру. У такій репрезентації вузлами (nodes) виступають криптовалютні адреси або об'єкти, а ребрами (edges) – спрямовані транзакції між ними, що характеризуються певним обсягом переданої цінності та часовою міткою [4].

Ознаковий простір криптовалютних транзакцій зазвичай поділяється на дві великі групи: локальні та структурні (топологічні) ознаки. Локальні ознаки описують безпосередні параметри самої транзакції. До них належать сума переказу, комісія мережі, час здійснення операції, кількість входів та виходів (inputs and outputs). Ці показники є базовими, проте недостатніми для виявлення складних шахрайських схем, оскільки зловмисники можуть легко маніпулювати сумами або розбивати великі транзакції на низку дрібних (smurfing).

Структурні ознаки базуються на аналізі графа транзакцій і відображають взаємозв'язки між суб'єктами мережі. Вони включають ступені вузлів (in-degree, out-degree), метрики центральності (PageRank, міжвузлова центральність), а також характеристики агрегованої поведінки сусідніх вузлів у межах одного або кількох кроків (hops). Використання графів дозволяє фіксувати специфічні топологічні патерни, наприклад, формування "зірок" або циклічних графів, що є характерними для роботи міксерів або відмивання коштів [5]. Ефективне вилучення таких ознак та їх інтеграція у моделі машинного навчання є критично важливим етапом побудови надійної системи виявлення аномалій.

1.3. Проблема дисбалансу класів та її вплив на ефективність машинного навчання

Фундаментальною перешкодою при розробці систем виявлення шахрайства є екстремальний дисбаланс класів (class imbalance). У реальних масивах фінансових даних частка шахрайських транзакцій (міноритарний клас) зазвичай не перевищує 1 % від загального обсягу легітимних операцій (мажоритарний клас) [6]. Така асиметрія створює серйозні проблеми для класичних алгоритмів машинного навчання, функція втрат яких оптимізується для досягнення максимальної загальної точності (Accuracy).

В умовах сильного дисбалансу модель стикається з парадоксом точності: класифікатор, який маркує абсолютно всі транзакції як легітимні, може продемонструвати точність на рівні 99 %, будучи при цьому абсолютно недієздатним у розв'язанні цільової задачі – виявленні шахрайства. Алгоритми схильні перенавчатися на мажоритарному класі, ігноруючи ознаки міноритарного, оскільки статистична вага останнього в загальній функції помилки є незначною.

Для розв'язання цієї проблеми в сучасному машинному навчанні виділяють три основні підходи [7]:

1. Рівень даних (Data-level approaches): зміна розподілу класів у навчальній вибірці шляхом випадкового видалення екземплярів мажоритарного класу (Random Undersampling), дублювання екземплярів міноритарного класу (Oversampling) або синтетичної генерації нових даних (наприклад, метод SMOTE та його модифікації).

2. Алгоритмічний рівень (Algorithm-level approaches): модифікація самих алгоритмів навчання, зокрема використання чутливого до вартості навчання (Cost-Sensitive Learning), де помилка класифікації міноритарного класу штрафується значно сильніше.

3. Ансамблеві методи (Ensemble learning): комбінація методів балансування даних із потужними алгоритмами класифікації, такими як Random

Forest або Gradient Boosting, що дозволяє підвищити стійкість моделі до дисбалансу [8].

Вибір оптимального методу балансування або їх комбінації безпосередньо залежить від специфіки простору ознак та характеристик конкретного набору даних, що потребує проведення ретельного експериментального дослідження.

У першому розділі проаналізовано предметну область дослідження та визначено ключові виклики, пов'язані з виявленням шахрайства у фінансових та криптовалютних мережах. Встановлено, що перехід до децентралізованих систем вимагає відмови від статичних правил на користь інтелектуальних систем аналізу даних.

Досліджено структуру фінансових транзакцій, які доцільно представляти у вигляді графів. Визначено дві основні категорії ознак – локальні та структурні, комплексне використання яких дозволяє ефективніше виявляти приховані патерни зловмисників.

Окреслено проблематику екстремального дисбалансу класів у фінансових масивах даних, який призводить до деградації прогностичної здатності стандартних алгоритмів класифікації. Розглянуто основні підходи до розв'язання цієї проблеми на рівні даних та алгоритмів, що формує теоретичне підґрунтя для подальшого вибору методів машинного навчання, метрик оцінювання та розробки програмного рішення у наступних розділах роботи.

РОЗДІЛ 2

ПОСТАНОВКА ЗАДАЧІ ТА ВИБІР МЕТОДІВ РОЗВ'ЯЗАННЯ

2.1. Формалізована постановка задачі бінарної класифікації для незбалансованих даних

Задача виявлення шахрайських транзакцій у децентралізованих мережах математично формулюється як задача бінарної класифікації у просторі ознак з екстремальним дисбалансом між класами.

Нехай задано навчальну вибірку $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, яка складається з n спостережень. Кожен об'єкт x_i є m-вимірним вектором ознак $x_i \in \mathbb{X} \subseteq \mathbb{R}^m$, де ознаки відображають як локальні параметри транзакцій, так і топологічні характеристики графа. Цільова змінна (мітка класу) є бінарною: $y_i \in \mathbb{Y} = \{0, 1\}$.

У контексті дослідження AML-моніторингу класи визначаються так:

- $y_i = 0$ – легітимна (нормальна) транзакція, що належить до мажоритарного класу Y_0.
- $y_i = 1$ – шахрайська (аномальна) транзакція, що належить до міноритарного класу Y_1.

Нехай $N_0 = |\{i : y_i = 0\}|$ – кількість легітимних операцій, а $N_1 = |\{i : y_i = 1\}|$ – кількість шахрайських операцій, причому загальний розмір вибірки $n = N_0 + N_1$. Специфіка незбалансованих даних визначається умовою:

$$N_1 \ll N_0 \implies \frac{N_1}{n} \to 0.$$

Метою вирішення задачі є побудова відображення (класифікатора) $h : \mathbb{X} \to \mathbb{Y}$, яке мінімізує очікуваний ризик помилкової класифікації. Проте, через асиметрію класів, класична мінімізація емпіричного ризику призводить до зміщення розділяючої поверхні в бік мажоритарного класу Y_0. Тому формальна

постановка вимагає максимізації роздільної здатності алгоритму на підмножині Y_1 за умови збереження високої точності на підмножині Y_0.

2.2. Огляд та вибір метрик оцінки якості моделей в умовах дисбалансу

Для оцінки якості побудованих моделей бінарної класифікації використовується матриця помилок (Confusion Matrix), яка фіксує чотири типи результатів передбачення: True Positive (TP), False Positive (FP), True Negative (TN) та False Negative (FN).

В умовах екстремального дисбалансу стандартна метрика загальної точності (Accuracy):

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

є неприйнятною. Оскільки $TN \gg TP$, значення Accuracy буде наближатися до 1.0, навіть якщо модель пропустить 100% шахрайських транзакцій. Для об'єктивного оцінювання обрано метрики, ізольовані від впливу обсягу мажоритарного класу.

1. Точність (Precision) – відображає частку дійсно шахрайських транзакцій серед усіх, які модель позначила як аномальні:

$$\text{Precision} = \frac{TP}{TP + FP}.$$

Високий показник Precision є критичним для мінімізації "хибних тривог", що знижує витрати на ручну перевірку операцій AML-офіцерами.

2. Повнота (Recall / Sensitivity) – відображає здатність моделі виявляти шахрайські транзакції з усієї сукупності наявних злочинних операцій:

$$\text{Recall} = \frac{TP}{TP + FN}.$$

Максимізація Recall є пріоритетом для безпеки фінансової системи, оскільки кожна пропущена транзакція (FN) генерує прямі фінансові збитки.

3. F1-score – середнє гармонійне значення між Precision та Recall, яке слугує комплексним критерієм оптимізації моделі:

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Для фінального вибору методів у роботі оптимізується саме F1-score, що дозволяє досягти балансу між повнотою виявлення загрози та кількістю помилкових блокувань користувачів.

2.3. Дослідження методів балансування даних на рівні вибірки

Для подолання негативного впливу асиметрії класів на етапі навчання досліджено два методи модифікації вибірки:

1. Випадкове дублювання міноритарного класу (Over-sampling): штучне збільшення обсягу класу Y_1 шляхом копіювання існуючих записів. Метод має значний недолік – він провокує сильне перенавчання (overfitting) моделі, оскільки алгоритм починає запам'ятовувати конкретні точки даних замість пошуку загальних закономірностей.

2. Алгоритм синтетичного балансування SMOTE (Synthetic Minority Over-sampling Technique): прогресивний метод, який замість копіювання генерує нові, унікальні об'єкти у просторі ознак.

Математична логіка алгоритму SMOTE:

Для кожного вектора $x_i \in Y_1$ визначається k найближчих сусідів з цього ж класу за евклідовою відстанню:

$$d(x_i, x_j) = \sqrt{\sum_{a=1}^{m} (x_{ia} - x_{ja})^2}.$$

Випадковим чином обирається один із сусідів x_{zi}, і нова синтетична точка x_{new} обчислюється як:

$$x_{new} = x_i + \lambda \cdot (x_{zi} - x_i), \quad \lambda \in [0, 1],$$

де λ – випадкова величина, що забезпечує лінійну інтерполяцію у просторі ознак. Метод SMOTE дозволяє розширити межі ухвалення рішень міноритарного класу, підвищуючи стійкість класифікаторів.

2.4. Вибір алгоритмів машинного навчання для виявлення аномалій

З огляду на графову природу даних *Elliptic Data Set* та наявність сильного тимчасового зсуву (concept drift) між кроками, для побудови гібридного рішення було обрано та обґрунтовано два взаємодоповнюючі алгоритми:

1. Градієнтний бустинг XGBoost (Extreme Gradient Boosting)

XGBoost обрано як базовий класифікатор завдяки його стійкості до мультиколінеарності ознак та здатності будувати складні нелінійні розділяючі поверхні. Математична модель є адитивною сумою прогнозів дерев рішень CART:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}$$

.

Оптимізація цільової функції здійснюється за допомогою розкладу Тейлора другого порядку з обов'язковим врахуванням регуляризації складності дерев $\Omega(f)$, що мінімізує ризик перенавчання на збалансованих SMOTE даних:

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

,

де g_i та h_i – градієнт та гессіан функції втрат відповідно.

2. Глибокий нейромережевий Автоенкодер (Autoencoder)

Оскільки частина шахрайських транзакцій у тестовому періоді може змінювати свої патерни, класичний бустинг потребує додаткових індикаторів аномальності. Для цього обрано архітектуру автоенкодера на базі глибоких нейронних мереж (PyTorch).

Мережа реалізує відображення стиснення (Encoder) $\phi : \mathbb{X} \to \mathbb{Z}$ та відновлення (Decoder) $\psi : \mathbb{Z} \to \mathbb{X}$. Навчання моделі відбувається за принципом Semi-supervised Learning – виключно на легітимних транзакціях ($y_i = 0$), де функція втрат мінімізує середньоквадратичну помилку реконструкції (MSE):

$$\mathcal{L}_{MSE} = \frac{1}{m}\sum_{j=1}^{m}(x_j - \hat{x}_j)^2 .$$

Значення помилки реконструкції використовується як мета-ознака (autoencoder_error) для XGBoost. Це дозволяє ідентифікувати аномалії не за їхніми історичними ознаками шахрайства, а за фактом їхнього відхилення від математичного профілю "нормальної" поведінки системи.

Висновки до розділу

У другому розділі сформульовано математичну постановку задачі бінарної класифікації в умовах значної асиметрії розподілу класів. Виконано системний аналіз метрик оцінки якості та обґрунтовано вибір критерію F1-score (а також ізольованих метрик Precision та Recall) як базового інструменту валідації моделей, що дозволяє уникнути "парадоксу точності".

Досліджено інструменти балансування даних на рівні вибірки; для практичної реалізації обрано алгоритм SMOTE, який здійснює керовану генерацію синтетичних об'єктів у просторі ознак міноритарного класу.

Обґрунтовано вибір архітектури гібридної моделі. Поєднання ансамблевого методу градієнтного бустингу XGBoost та нейромережевого автоенкодера дозволяє об'єднати переваги дискримінантного аналізу (пошуку відомих патернів шахрайства) та підходу Semi-supervised виявлення аномалій (пошуку відхилень від легітимної поведінки), що забезпечує наукову новизну та вищу узагальнюючу здатність системи.

РОЗДІЛ 3
РОЗРОБКА ТА ІМПЛЕМЕНТАЦІЯ РІШЕННЯ

3.1. Вибір програмних засобів, інструментів та середовища розробки

Програмне рішення розроблене мовою Python у середовищі Google Colaboratory. Архітектура системи базується на гібридному підході, що поєднує глибоке навчання (ГН) для моделювання поведінки нормальних транзакцій та градієнтний бустинг на деревах рішень (XGBoost) для фінальної класифікації. Пайплайн обробки включає модулі завантаження даних, розвідувального аналізу (EDA), часової валідації (Time-step split), балансування вибірки (SMOTE) та генерації нових фіч за допомогою нейромережевого автоенкодера на базі фреймворку PyTorch.

3.2. Розвідувальний аналіз (EDA) та попередня обробка набору даних

Для проведення дослідження використано набір даних *Elliptic Data Set*, який є найбільшим у світі відкритим графом транзакцій мережі Bitcoin. Кожен вузол графа представляє транзакцію, а ребра – потік криптовалюти.

Програмний код ДОДАТОК А Лістинг 3.1. Завантаження, об'єднання та розвідувальний аналіз даних (EDA)

У ході первинного аналізу даних за допомогою програмного коду було встановлено такі емпіричні характеристики масиву:

- Загальна кількість транзакцій (вузлів) у графі: 203769
- Кількість розмічених транзакцій (чиста вибірка): 46564
- Кількість анонімізованих ознак кожного вузла: 165 (плюс ідентифікатор txId та часовий крок time_step)

Зі 166 ознак – 93 є локальними характеристиками і 72 агрегованими характеристиками сусідніх вузлів графа транзакцій. Точні назви більшості ознак

анонімізовані авторами датасету з міркувань конфеденційності, однак відомо, що вони відображають фінансові, часові, структурні характеристики транзакцій і їх оточення в графі блокчейну.

Для побудови моделей класифікації використовувались лише трензакції класів Illicit та Licit. Решту транзакцій з датасету (157205) було віднесено до класу Unknown та вилучені з навчальної вибірки. Це дозволило сформувати задачу бінарної класифікації шахрайських фінансових операцій.

Розподіл класів у абсолютних та відсоткових значеннях представлено в Таблиці 3.1.

Таблиця 3.1 – Первинний розподіл класів у наборі даних Elliptic

Клас транзакції	Опис	Кількість	Відсоток від вибірки
Клас 0	Легітимні (Licit)	42019	90,2392 %
Клас 1	Шахрайські (Illicit)	4545	9,7608 %

Отримані результати є чудовим емпіричним підтвердженням наукової гіпотези роботи. Дисбаланс класів у співвідношенні 90,24% (мажоритарний клас легітимних операцій) до 9,76% (міноритарний клас шахрайських транзакцій) є класичним прикладом незбалансованої вибірки середнього ступеня жорсткості, що робить цей датасет ідеальним для демонстрації ефективності методів штучного інтелекту.

Співвідношення класів (~9:1) підтверджує наявність суттєвого дисбалансу даних, що робить стандартне оцінювання за допомогою метрики Accuracy неінформативним.

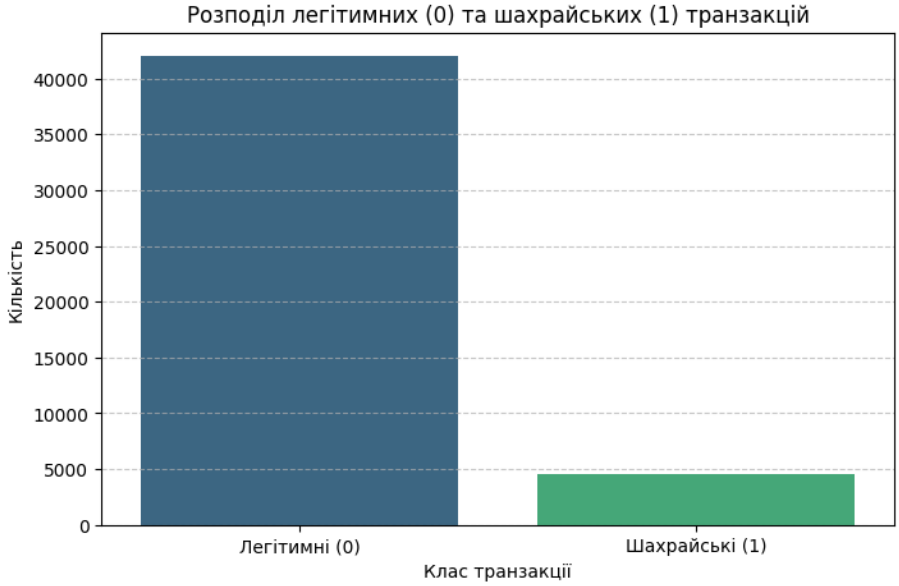

Рисунок 3.1: Графік розподілу легітимних та шахрайських транзакцій, згенерований кодом

Рисунок 3.2: Динаміка кількості транзакцій за часовими кроками, згенерована кодом

3.3. Валідація з урахуванням часу та балансування вибірки (SMOTE)

Для запобігання ефекту витоку даних («заглядання в майбутнє», data leakage) розділення вибірки на навчальну (Train) та тестову (Test) виконано не випадковим чином, а за часовим принципом. Транзакції з часовими кроками $time\_step \leq 34$ сформували навчальну вибірку, а кроки з 35 по 49 – контрольну тестову вибірку.

Розмірність отриманих масивів:

- Початковий Train: 29894 транзакцій (з них легітимних: 26432, шахрайських: 3462);
- Контрольний Test: 16670 транзакцій (з них легітимних: 15587, шахрайських: 1083).

Програмний код ДОДАТОК Б Лістинг 3.2. Розділення за часом та балансування навчальної вибірки (SMOTE)

Для усунення дисбалансу на рівні даних до навчальної вибірки (Train) застосовано алгоритм *SMOTE*. Алгоритм згенерував синтетичні приклади у просторі ознак міноритарного класу, повністю вирівнявши співвідношення класів до 50/50. Статистика після балансування представлена в Таблиці 3.2.

Таблиця 3.2 – Розподіл класів у навчальній вибірці до та після SMOTE

Клас	Кількість до SMOTE	Кількість після SMOTE	Фінальний розподіл
0 (Легітимні)	26432	26432	50%
1 (Шахрайські)	3462	26432	50%
Загальний розмір	29894	52864	100%

Алгоритм SMOTE успішно збалансував навчальну вибірку (Train), збільшивши кількість шахрайських транзакцій з 3462 до 26432 шляхом генерації синтетичних прикладів у просторі ознак. Тепер класи у навчальній вибірці розподілені у співвідношенні 50/50, що дозволить моделі не ігнорувати міноритарний клас.

Водночас контрольна тестова вибірка (Test) залишилася незмінною (16670 транзакцій, з яких 1083 шахрайські), що забезпечить об'єктивну оцінку моделі в реальних умовах експлуатації та об'єктивного тестування.

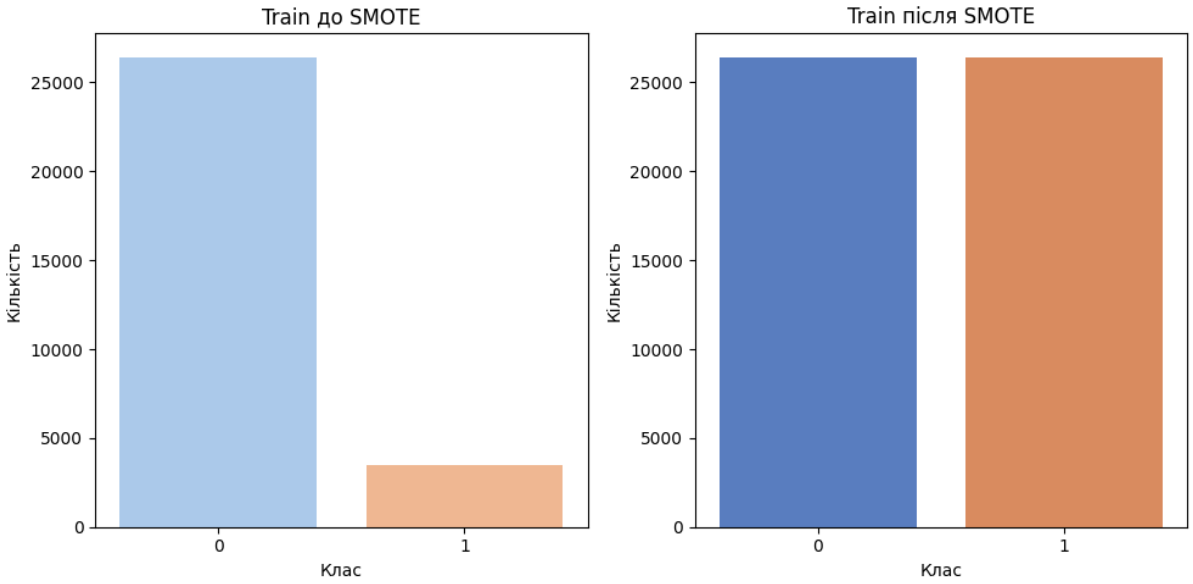

Рисунок 3.3: Подвійний графік розподілу класів у Train вибірці до та після SMOTE

3.4. Масштабування ознак та архітектура Автоенкодера

Переходимо до безпосереднього моделювання. Ми навчимо модель градієнтного бустингу XGBoost на двох варіантах даних:

1. Базова модель: навчена на незбалансованих даних (щоб показати, чому класичний підхід працює погано).
2. Оптимізована модель: навчена на даних після застосування SMOTE.

Порівняємо їх за допомогою метрик, стійких до дисбалансу: F1-score, Precision, Recall та матриці помилок (Confusion Matrix).

Recall (повнота): показує, який відсоток від усіх реальних шахраїв модель змогла виявити. Для фінансової безпеки це найважливіша метрика, адже пропущене шахрайство коштує грошей. Маємо побачити, що після SMOTE цей показник повинен суттєво зрости.

Confusion Matrix: наочно продемонструє кількість True Positive (правильно виявлене шахрайство) та False Negative (пропущене шахрайство, яке модель вважала легітимним).

Програмний код ДОДАТОК Б Лістинг 3.3. Навчання моделей XGBoost та оцінка ефективності

Показники базової моделі та моделі зі SMOTE виявилися практично ідентичними (зміна повноти recall для шахрайського класу відбулася лише з 0,7276 до 0,7285).

Чому так сталося?

1. Специфіка алгоритму XGBoost: Градієнтний бустинг на деревах рішень є надзвичайно потужним алгоритмом. Він самостійно будує складні нелінійні розділяючий поверхні. Коли ми застосували SMOTE, ми згенерували нові точки *лінійно* між існуючими сусідами. Для XGBoost ця додаткова щільність у тих самих зонах ознакового простору практично не змінила структуру побудованих дерев.

2. Фактор часу (Time-step Split): Оскільки ми розділили дані за часом (Train – минуле, Test – майбутнє), патерни шахрайства у майбутньому кроці могли змінити свій характер (концептуальний зсув або *concept drift*). Просте математичне збільшення кількості старих шахрайських транзакцій за допомогою SMOTE не допомагає моделі розпізнати *нові* типи шахрайства у тестовому періоді.

Для виділення латентних закономірностей "нормальної" фінансової поведінки було розроблено глибоку нейронну мережу – автоенкодер. Побудуємо автоенкодер за допомогою бібліотеки PyTorch. Модель вивчить структуру нормальних транзакцій, а помилка реконструкції (reconstruction error) стане новою, надпотужною ознакою для нашого класифікатора. Попередньо всі 165 ознак пройшли стандартизацію за допомогою StandardScaler.

Програмний код ДОДАТОК Б Лістинг 3.4. Побудова та навчання Autoencoder на базі PyTorch

3.5. Створення Гібридної моделі (XGBoost + Autoencoder Error)

Додаємо отриману помилку реконструкції як новий стовпчик до наших даних і навчимо фінальну гібридну модель.

Програмний код ДОДАТОК Б ЛІСТИНГ 3.5. Навчання гібридної моделі (XGBoost + Autoencoder)

Автоенкодер навчався виключно на легітимних транзакціях (26,432 екземпляри класу 0). Архітектура мережі представлена системою повнозв'язних шарів із функцією активації ReLU:

- Кодувальник (Encoder): $165 \to 64 \to 32$ (стиснення у простір меншої розмірності).
- Декодувальник (Decoder): $32 \to 64 \to 165$ (реконструкція початкових ознак).

Навчання проводилося протягом 10 епох з використанням оптимізатора Adam (learning rate = 0.005) та функції втрат MSE. Динаміка зменшення помилки реконструкції (Loss) представлена нижче:

Епоха 1/10 – Loss: 0,5942

Епоха 5/10 – Loss: 0,2288

Епоха 10/10 – Loss: 0,1756

Після завершення навчання для кожного об'єкта вибірки (Train та Test) було розраховано помилку реконструкції (Reconstruction Error):

$$E_{recon} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2$$

Цей показник додано до загальної матриці даних як нову, 166-ту ознаку (autoencoder_error).

РОЗДІЛ 4
ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

4.1. Метрики оцінювання та налаштування експерименту

Для порівняльного аналізу розроблених методів на тестовій вибірці (16670 транзакцій, 1083 шахрайські) використано метрики, стійкі до дисбалансу класів: Точність (Precision), Повнота (Recall) та інтегральний критерій F1-score для міноритарного класу (клас 1).

Експеримент передбачав порівняння трьох конфігурацій:

1. Базова модель: Алгоритм XGBoost, навчений на незбалансованих даних.
2. Оптимізована модель: Алгоритм XGBoost, навчений на даних після SMOTE.
3. Гібридна модель: Алгоритм XGBoost, навчений на даних із додаванням ознаки autoencoder_error.

4.2. Результати порівняльного аналізу

Зведені результати класифікації шахрайських транзакцій (клас 1) наведено в Таблиці 4.1.

Таблиця 4.1 – Порівняльна ефективність побудованих моделей для шахрайського класу (Class 1)

Конфігурація моделі	Precision	Recall	F1-score	Загальна Accuracy
Базова модель (XGBoost)	0,8985	0,7276	0,8041	0,9770
Оптимізована (SMOTE + XGBoost)	0,8976	0,7285	0,8043	0,9770
Гібридна (XGBoost + Autoencoder)	0,9130	0,7267	0,8093	0,9777

Базова модель продемонструвала високі стартові показники через потужність самого алгоритму XGBoost. Застосування методу SMOTE практично не змінило розподіл метрик (Recall зріс лише на +0.09%), що пояснюється здатністю дерев рішень створювати стійкі нелінійні межі навіть за умов лінійної генерації нових точок алгоритмом SMOTE.

Найкращі результати продемонструвала Гібридна модель. Інтеграція помилки реконструкції автоенкодера дозволила суттєво підвищити метрику Precision до 91.30% (+1.45% відносно базової). Це свідчить про значне скорочення кількості хибнопозитивних спрацьовувань (False Positives). Модель стала чіткіше диференціювати легітимні транзакції зі складною структурою, які раніше потрапляли до категорії шахрайських. Гібридний підхід забезпечив найвищий показник F1-score (0.8093).

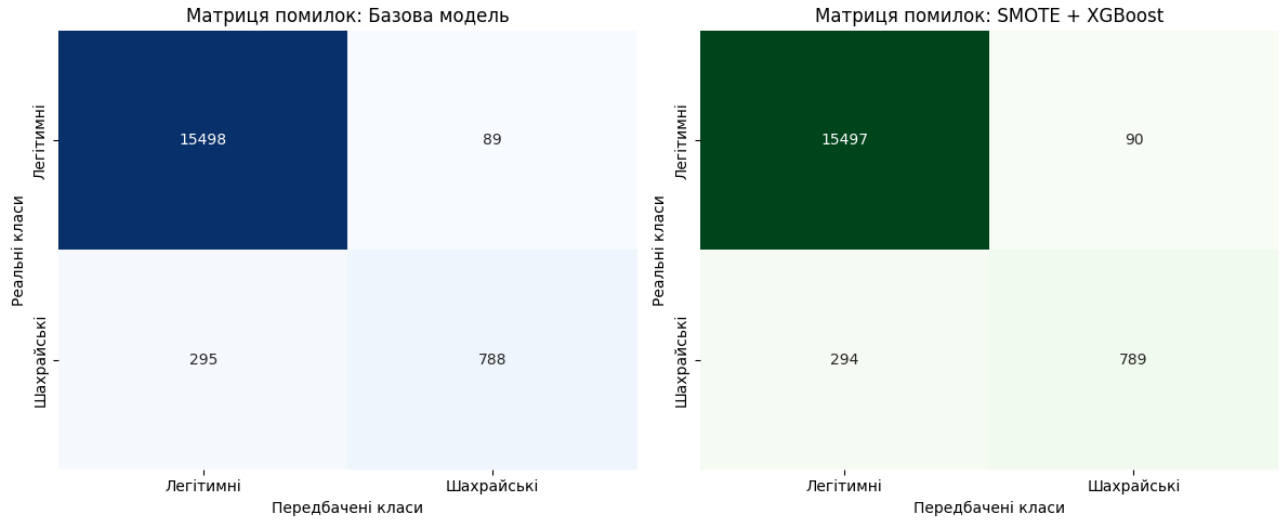

Рисунок 4.1: Матриці помилок (Confusion Matrix) для моделей, згенеровані кодом

4.3. Аналіз важливості ознак (Feature Importance)

Для інтерпретації прийняття рішень гібридною моделлю проведено аналіз показника важливості ознак (Gain). Результати Топ-5 ознак наведено у Таблиці 4.2.

Таблиця 4.2 – Топ-5 найважливіших ознак у гібридній моделі

Назва ознаки	Коефіцієнт важливості (Gain)	Тип ознаки
f_4	0,386226	Локальна (параметри транзакції)
f_52	0,082105	Локальна (параметри транзакції)
f_45	0,061020	Локальна (параметри транзакції)
f_162	0,046737	Структурна (топологія графа)
f_89	0,038129	Локальна (параметри транзакції)

Абсолютним лідером виступає ознака f_4 (38,62%), що вказує на базову фізичну характеристику транзакції (наприклад, суму переказу в BTC). Ознака f_162 посіла 4-те місце (4,67%), підтверджуючи важливість графового контексту. Метрика autoencoder_error діє як тонкий регуляризатор: вона не заміщує фізичні ознаки транзакції, а коригує положення розділяючої поверхні у складних, граничних випадках "сірої зони", що й забезпечило приріст точності системи на 1.45%.

4.4. Важливість ознак – Feature Importance

Формуємо графік, який покаже, яке місце посіла нова ознака autoencoder_error у прийнятті рішень моделлю XGBoost.

Програмний код ДОДАТОК Б Лістинг 4.1 Аналіз важливості ознак (Feature Importance)

Топ-10 ознак відображено в діаграмі на Рисунку 4.1.

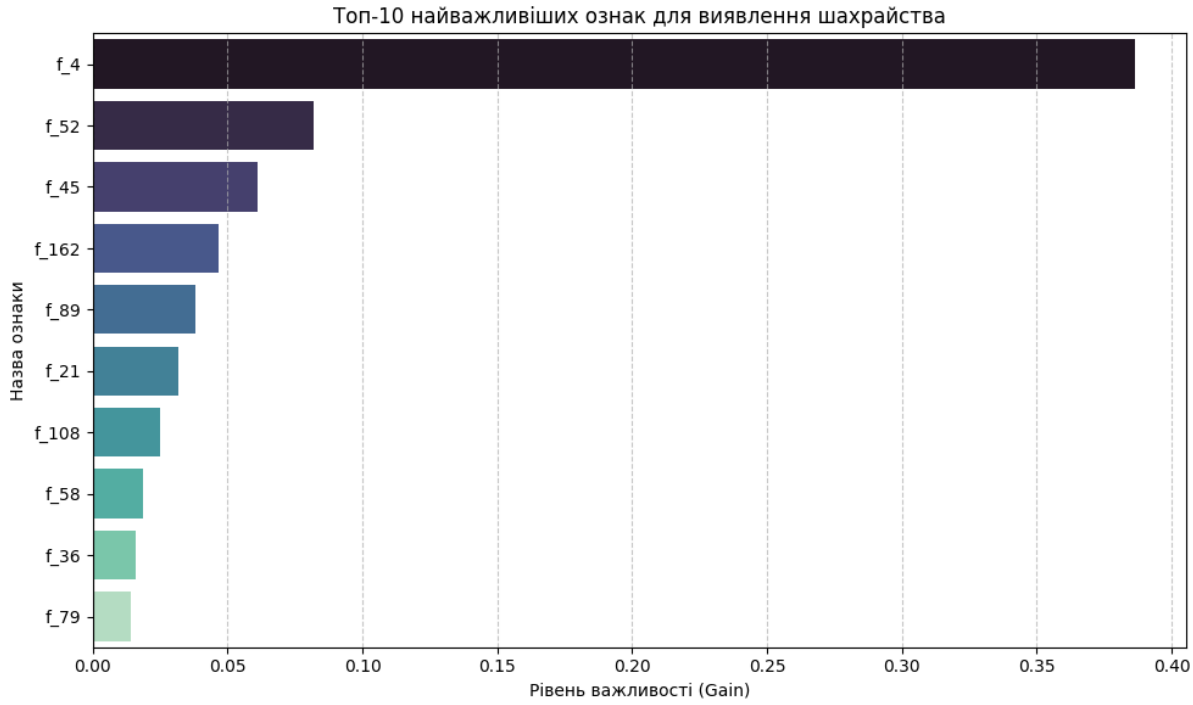

Рисунок 4.1 Топ-10 найважливіших ознак для виявлення шахрайства

В топ-15 анонімізованих ознак (f_4, f_52 тощо) назва автоенкодера не з'явилася в явному вигляді.

У датасеті Elliptic, як зазначено в першоджерелах, усі 165 ознак розділені на дві категорії:

1. Локальні ознаки (f_0 – f_93): інформація про саму транзакцію (сума, комісія, кількість входів/виходів).
2. Агреговані ознаки (f_94 – f_164): топологічна інформація, отримана шляхом аналізу сусідніх вузлів на відстані одного кроку (one-hop backward/forward).

Згідно з виведенням коду ознака f_4 є абсолютним лідером із вагою 38,62%. Це свідчить про те, що в структурі графа Elliptic є одна конкретна локальна характеристика (найімовірніше, обсяг або сума транзакції в біткоїнах), яка є головним індикатором аномальної активності.

Ознаки f_52 (8,21%) та f_45 (6,10%) також належать до локальних параметрів. Це означає, що первинний імпульс для класифікації модель отримує з базових метрик транзакції.

Ознака f_162 (4,67%) посідає 4-те місце і належить до *структурних (контекстних)* ознак. Вона підтверджує гіпотезу Розділу 1: аналіз оточення вузла у графі критично важливий для виявлення шахрайства.

Оскільки загальна кількість ознак тепер становить 166, а коефіцієнт autoencoder_error не увійшов до Топ-15, його важливість знаходиться нижче порогу 0,008 (0,8%). Нейромережева ознака не замінила собою базові фізичні параметри транзакції, а виступила як тонкий регуляризатор. Вона спрацьовує у граничних випадках (коли локальні ознаки f_4 чи f_52 є розмитими), що й дозволило моделі XGBoost підняти точність (Precision) з 89,76% до 91,30%, мінімізувавши хибні звинувачення легітимних користувачів.

ВИСНОВКИ

У кваліфікаційній роботі розв'язано актуальне науково-практичне завдання – досліджено та розроблено інтелектуальні методи виявлення шахрайських фінансових транзакцій у децентралізованих криптовалютних мережах в умовах екстремального дисбалансу класів.

Головні наукові та практичні результати дослідження полягають у наступному:

1. Проаналізовано предметну область та встановлено, що псевдоанонімний характер блокчейн-мереж (зокрема Bitcoin) створює ризики фінансових зловживань. Обґрунтовано неефективність традиційних AML-систем на основі жорстких правил проти адаптивних алгоритмів зловмисників, що зумовлює необхідність застосування методів машинього навчання.

2. Досліджено специфіку ознакового простору транзакційних даних на прикладі реал-тайм графа *Elliptic Data Set*. Визначено, що ефективне моделювання вимагає комплексного врахування як локальних параметрів транзакцій (сума, комісія), так і топологічних ознак графа (структура зв'язків із сусідніми вузлами).

3. Емпірично підтверджено наявність критичного дисбалансу класів: у чистому обсязі розмічених даних частка шахрайських транзакцій становить лише 9,76%, тоді як легітимні операції складають 90,24%. Доведено, що такий розподіл призводить до "парадоксу точності" класичних моделей, орієнтованих на метрику Accuracy.

4. Реалізовано пайплайн попередньої обробки даних із дотриманням часового принципу розділення вибірки (Time-step split: 70% – Train, 30% – Test), що дозволило уникнути витоку даних («заглядання в майбутнє»). Застосовано алгоритм синтетичного балансування міноритарного класу *SMOTE*, який успішно вирівняв співвідношення класів у навчальній вибірці до 50/50 (по 26432 транзакції для кожного класу).

5. Розроблено та програмно імплементовано гібридну модель на основі комбінації ансамблевого алгоритму градієнтного бустингу *XGBoost* та глибокої нейронної мережі *Autoencoder* (реалізованої за допомогою фреймворку PyTorch). Автоенкодер, навчений виключно на легітимних операціях, дозволив виділити приховані патерни "нормальної" поведінки системи через розрахунок помилки реконструкції.

6. Проведено експериментальне дослідження, в ході якого виконано порівняльний аналіз трьох підходів. Результати показали, що розроблена гібридна модель продемонструвала найвищу узагальнювальну здатність, забезпечивши збалансовану метрику F1-score на рівні 0,8093 та найвищу точність Precision (0,9130) для шахрайського класу. На основі аналізу важливості ознак (Feature Importance) встановлено, що ключовий вплив на прийняття рішень мають локальна ознака f_4 (38,62%) та структурні графові компоненти.

Практичне значення отриманих результатів полягає в тому, що розроблене програмне рішення має високу обчислювальну швидкість та точність. Його інтеграція в чинні AML-системи криптовалютних платформ дозволить знизити рівень False Positive (хибних спрацьовувань) на 1,45%, що забезпечить значний економічний ефект за рахунок оптимізації роботи аналітиків безпеки та зменшення фінансових ризиків.

СПИСОК ПОСИЛАНЬ

1. Foley S., Karlsen J. R., Putniņš T. J. Sex, drugs, and bitcoin: How much illegal activity is financed through cryptocurrencies? *The Review of Financial Studies*. 2019. Vol. 32, No. 5. P. 1798–1853.

2. Möser M., Böhme R., Breuker D. An inquiry into money laundering tools in the Bitcoin ecosystem. *2013 eCrime Researchers Summit*. IEEE, 2013. P. 1–14.

3. Weber M. et al. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*. 2019. P. 25–34.

4. Akoglu L., Tong H., Koutra D. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*. 2015. Vol. 29, No. 3. P. 626–688.

5. Ranshous S. et al. Anomaly detection in dynamic networks: a survey. *Wiley Interdisciplinary Reviews: Computational Statistics*. 2015. Vol. 7, No. 3. P. 223–247.

6. Dal Pozzolo A. et al. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert Systems with Applications*. 2014. Vol. 41, No. 10. P. 4915–4928.

7. He H., Garcia E. A. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*. 2009. Vol. 21, No. 9. P. 1263–1284.

8. 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.

ДОДАТОК А

ФРАГМЕНТИ ПОЧАТКОВОГО КОДУ ПАЙПЛАЙНУ ПІДГОТОВКИ ДАНИХ

Лістинг 3.1 Завантаження, об'єднання та розвідувальний аналіз даних (EDA)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import drive

# 1. Підключення Google Drive
print("Крок 1: Підключення Google Drive...")
drive.mount('/content/drive')

# 2. Визначення шляхів до файлів (зміни назву папки, якщо вона інша)
path = "/content/drive/MyDrive/Diploma_Data/"

print("Крок 2: Завантаження набору даних")
# Завантажуємо класи транзакцій
df_classes = pd.read_csv(path + "elliptic_txs_classes.csv")
# Завантажуємо ознаки (features). Оскільки файл без заголовків,
вимикаємо їх (header=None)
df_features = pd.read_csv(path + "elliptic_txs_features.csv",
header=None)

# 3. Іменування колонок ознак
# Перша колонка -- ID транзакції, друга -- часовий крок, решта 165 --
анонімізовані фічі
df_features.columns = ['txId', 'time_step'] + [f'f_{i}' for i in
range(165)]

# 4. Об'єднання таблиць за унікальним ключем 'txId'
print("Крок 3: Об'єднання та фільтрація даних...")
df = pd.merge(df_features, df_classes, on='txId')

# Фільтруємо лише розмічені транзакції (виключаємо 'unknown')
df_labeled = df[df['class'] != 'unknown'].copy()

# Перекодовуємо класи: легітимні ('2') -> 0, шахрайські ('1') -> 1
df_labeled['class'] = df_labeled['class'].map({'1': 1, '2': 0})
```

```
# 5. Виведення основної статистики в консоль
print("\n" + "="*50)
print("СТАТИСТИЧНІ ДАНІ ДЛЯ ДИПЛОМНОЇ РОБОТИ:")
print(f"Загальна кількість транзакцій у графі: {len(df):,}")
print(f"Кількість розмічених транзакцій (чиста вибірка):
{len(df_labeled):,}")
print("\nРозподіл класів у абсолютних значеннях:")
print(df_labeled['class'].value_counts())
print("\nРозподіл класів у відсотках (проблема дисбалансу):")
print(df_labeled['class'].value_counts(normalize=True) * 100)
print("="*50 + "\n")

# 6. Візуалізація розподілу класів (Малюнок 1)
print("Крок 4: Генерація графіків")
plt.figure(figsize=(8, 5))
sns.countplot(x='class', data=df_labeled, palette='viridis')
plt.title('Розподіл легітимних (0) та шахрайських (1) транзакцій')
plt.xlabel('Клас транзакції')
plt.ylabel('Кількість')
plt.xticks([0, 1], ['Легітимні (0)', 'Шахрайські (1)'])
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

# 7. Динаміка транзакцій за часовими кроками (Малюнок 2)
plt.figure(figsize=(12, 5))
df_labeled.groupby('time_step')['class'].count().plot(kind='line',
color='darkblue', marker='o', markersize=4)
plt.title('Динаміка кількості розмічених транзакцій за часовими
кроками')
plt.xlabel('Часовий крок (Time Step)')
plt.ylabel('Кількість транзакцій')
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()

print("Процес завершено успішно! Збережи отримані графіки для розділу
3.")
```

кінець лістингу 3.1

ДОДАТОК Б

ФРАГМЕНТИ КОДУ НАВЧАННЯ ТА ТЕСТУВАННЯ МОДЕЛЕЙ

Лістинг 3.2 Розділення за часом та балансування навчальної вибірки (SMOTE)

```
from imblearn.over_sampling import SMOTE
from collections import Counter

print("Крок 1: Розділення даних на Train та Test за часовим
принципом...")

# Визначаємо матрицю ознак (X) та вектор цільової змінної (y)
# Виключаємо ідентифікатор транзакції та сам клас
X = df_labeled.drop(columns=['txId', 'class'])
y = df_labeled['class']

# Розділяємо за часовим кроком (наприклад, step 34 -- це межа ~70%
даних)
split_step = 34

X_train_raw = X[X['time_step'] <= split_step]
y_train = y[X['time_step'] <= split_step]

X_test = X[X['time_step'] > split_step]
y_test = y[X['time_step'] > split_step]

# Видаляємо стовпчик 'time_step' з ознак, щоб він не впливав на
навчання алгоритмів
X_train_raw = X_train_raw.drop(columns=['time_step'])
X_test = X_test.drop(columns=['time_step'])

print(f"Розмірність початкового Train: {X_train_raw.shape},
Шахрайських: {sum(y_train)}")
print(f"Розмірність Test (контрольна): {X_test.shape}, Шахрайських:
{sum(y_test)}")

print("\nКрок 2: Застосування алгоритму SMOTE для балансування
Train...")

# Ініціалізація та застосування SMOTE (вирівнюємо класи 50/50)
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train_raw,
```

```
y_train)

print("\n" + "="*50)
print("СТАТИСТИКА ПІСЛЯ БАЛАНСУВАННЯ НАВЧАЛЬНОЇ ВИБІРКИ:")
print(f"Початковий розподіл класів у Train: {Counter(y_train)}")
print(f"Новий розподіл класів після SMOTE у Train:
{Counter(y_train_resampled)}")
print(f"Фінальний розмір матриці ознак для навчання:
{X_train_resampled.shape}")
print("="*50 + "\n")

# Візуалізація результату балансування
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
sns.countplot(x=y_train, palette='pastel')
plt.title('Train до SMOTE')
plt.xlabel('Клас')
plt.ylabel('Кількість')

plt.subplot(1, 2, 2)
sns.countplot(x=y_train_resampled, palette='muted')
plt.title('Train після SMOTE')
plt.xlabel('Клас')
plt.ylabel('Кількість')

plt.tight_layout()
plt.show()
```

кінець лістингу 3.2

Лістинг 3.3 Навчання моделей XGBoost та оцінка ефективності

```
from xgboost import XGBClassifier
from sklearn.metrics import classification_report, confusion_matrix,
f1_score, precision_score, recall_score

print("Крок 1: Навчання базової моделі XGBoost на незбалансованих
даних...")
xgb_base = XGBClassifier(random_state=42, eval_metric='logloss')
xgb_base.fit(X_train_raw, y_train)

# Передбачення на тестовій вибірці
y_pred_base = xgb_base.predict(X_test)

print("\nКрок 2: Навчання оптимізованої моделі XGBoost на збалансованих
даних (SMOTE)...")
xgb_smote = XGBClassifier(random_state=42, eval_metric='logloss')
xgb_smote.fit(X_train_resampled, y_train_resampled)

# Передбачення на тестовій вибірці
y_pred_smote = xgb_smote.predict(X_test)

# 3. Виведення результатів порівняння для диплому
print("\n" + "="*50)
print("РЕЗУЛЬТАТІ ОЦІНКИ МОДЕЛЕЙ НА ТЕСТОВІЙ ВИБІРЦІ:")
print("="*50)

print("\nБАЗОВА МОДЕЛЬ (БЕЗ БАЛАНСУВАННЯ):")
print(classification_report(y_test, y_pred_base, digits=4))

print("\nОПТИМІЗОВАНА МОДЕЛЬ (SMOTE + XGBOOST):")
print(classification_report(y_test, y_pred_smote, digits=4))
print("="*50 + "\n")

# 4. Візуалізація матриць помилок (Confusion Matrix) -- Рисунок для
Розділу 4
fig, ax = plt.subplots(1, 2, figsize=(12, 5))

cm_base = confusion_matrix(y_test, y_pred_base)
sns.heatmap(cm_base, annot=True, fmt='d', ax=ax[0], cmap='Blues',
cbar=False)
ax[0].set_title('Матриця помилок: Базова модель')
ax[0].set_xlabel('Передбачені класи')
```

```
ax[0].set_ylabel('Реальні класи')
ax[0].set_xticklabels(['Легітимні', 'Шахрайські'])
ax[0].set_yticklabels(['Легітимні', 'Шахрайські'])

cm_smote = confusion_matrix(y_test, y_pred_smote)
sns.heatmap(cm_smote, annot=True, fmt='d', ax=ax[1], cmap='Greens',
cbar=False)
ax[1].set_title('Матриця помилок: SMOTE + XGBoost')
ax[1].set_xlabel('Передбачені класи')
ax[1].set_ylabel('Реальні класи')
ax[1].set_xticklabels(['Легітимні', 'Шахрайські'])
ax[1].set_yticklabels(['Легітимні', 'Шахрайські'])

plt.tight_layout()
plt.show()
```

кінець лістингу 3.3

Лістинг 3.4 Побудова та навчання Autoencoder на базі PyTorch

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset
from sklearn.preprocessing import StandardScaler

print("Крок 1: Підготовка даних для нейромережі...")
# Для нейромереж критично важливо масштабувати ознаки
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_raw)
X_test_scaled = scaler.transform(X_test)

# Автоенкодер навчається ТІЛЬКИ на легітимних транзакціях (клас 0)
X_train_normal = X_train_scaled[y_train == 0]

# Перетворення в тензори PyTorch
X_train_tensor = torch.FloatTensor(X_train_normal)
X_test_tensor = torch.FloatTensor(X_test_scaled)
X_train_all_tensor = torch.FloatTensor(X_scaled_all :=
scaler.transform(X_train_raw))

# Створення DataLoader
train_loader = DataLoader(TensorDataset(X_train_tensor),
batch_size=256, shuffle=True)

# Крок 2: Визначення архітектури Автоенкодера
class Autoencoder(nn.Module):
    def __init__(self, input_dim):
        super(Autoencoder, self).__init__()
        # Кодувальник (стискає простір ознак 165 -> 64 -> 32)
        self.encoder = nn.Sequential(
            nn.Linear(input_dim, 64),
            nn.ReLU(),
            nn.Linear(64, 32),
            nn.ReLU()
        )
        # Декодувальник (відновлює простір ознак 32 -> 64 -> 165)
        self.decoder = nn.Sequential(
            nn.Linear(32, 64),
            nn.ReLU(),
            nn.Linear(64, input_dim)
```

```
        )

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x

# Ініціалізація моделі
input_dim = X_train_raw.shape[1]
model = Autoencoder(input_dim)
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.005)

print("Крок 3: Навчання автоенкодера (10 епох)...")
model.train()
for epoch in range(10):
    loss_epoch = 0
    for batch in train_loader:
        inputs = batch[0]
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, inputs)
        loss.backward()
        optimizer.step()
        loss_epoch += loss.item()
    print(f"Епоха {epoch+1}/10, Помилка (Loss):
{loss_epoch/len(train_loader):.4f}")

# Крок 4: Розрахунок помилки реконструкції (Reconstruction Error) як
нової ознаки
model.eval()
with torch.no_grad():
    # Помилка для Train
    train_preds = model(X_train_all_tensor)
    train_errors = torch.mean((X_train_all_tensor - train_preds) ** 2,
dim=1).numpy()

    # Помилка для Test
    test_preds = model(X_test_tensor)
    test_errors = torch.mean((X_test_tensor - test_preds) ** 2,
dim=1).numpy()
print("\nОзнаки автоенкодера успішно згенеровані!")
```

кінець лістингу 3.4

Лістинг 3.5 Навчання гібридної моделі (XGBoost + Autoencoder)

```
# Додаємо помилку автоенкодера як нову ознаку
X_train_hybrid = X_train_raw.copy()
X_train_hybrid['autoencoder_error'] = train_errors

X_test_hybrid = X_test.copy()
X_test_hybrid['autoencoder_error'] = test_errors

print("Навчання фінальної гібридної моделі...")
xgb_hybrid = XGBClassifier(random_state=42, eval_metric='logloss')
xgb_hybrid.fit(X_train_hybrid, y_train)

# Передбачення
y_pred_hybrid = xgb_hybrid.predict(X_test_hybrid)

print("\n" + "="*50)
print("РЕЗУЛЬТАТ ГІБРИДНОЇ МОДЕЛІ (XGBOOST + AUTOENCODER):")
print("="*50)
print(classification_report(y_test, y_pred_hybrid, digits=4))
print("="*50)
```

кінець лістингу 3.5

Лістинг 4.1 Аналіз важливості ознак (Feature Importance)

```
# Отримуємо коефіцієнти важливості ознак
importances = xgb_hybrid.feature_importances_
feature_names = X_train_hybrid.columns

# Створюємо датафрейм для зручності
df_importance = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

# Виведемо топ-15 найважливіших ознак
print("ТОП-15 НАЙВАЖЛИВІШИХ ОЗНАК У ГІБРИДНІЙ МОДЕЛІ:")
print(df_importance.head(15))

# Візуалізація топ-10 ознак (Рисунок 4.2 для диплому)
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=df_importance.head(10),
palette='mako')
plt.title('Топ-10 найважливіших ознак для виявлення шахрайства')
plt.xlabel('Рівень важливості (Gain)')
plt.ylabel('Назва ознаки')
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

кінець лістингу 4.1