

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ МЕТОДІВ ВИЯВЛЕННЯ
ШАХРАЙСЬКИХ ДІЙ В БАНКІВСЬКІЙ СИСТЕМІ

RESEARCH AND IMPLEMENTATION OF METHODS FOR FRAUD DETECTION
IN THE BANKING SYSTEM

Виконав(ла): студент(ка) 2 курсу, групи КНТ-122м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ЄВТУШЕНКО В.В.

(ПРИЗВИЩЕ та ініціали)

Керівник ГОЛУБ Т.В.

(ПРИЗВИЩЕ та ініціали)

Рецензент ПОЛЯКОВ М.О.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ЄВТУШЕНКА Віталія Володимировича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та реалізація методів виявлення шахрайських дій в банківській системі. Research and Implementation of Method for Fraud Detection in the Banking System

керівник проєкту (роботи) к.т.н., доцент, ГОЛУБ Тетяна Василівна
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 14 ” листопада 2023 року № 437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз стратегій виявлення шахрайських транзакцій. 2. Дослідження методів обробки даних. 3. Програмна реалізація методу виявлення шахрайських дій. 4. Дослідження результатів роботи методу.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ГОЛУБ Т.В., доцент		
Нормоконтролер	ЛИПОВЕЦЬ М.В., асистент		

7. Дата видачі завдання “ 08 ” вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз стратегій виявлення шахрайських дій.	2-3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	4-5 тижні	Розділ 2
4	Дослідження методів.	6 тиждень	Розділ 3
5	Навчання моделі.	7-8 тижні	Розділ 4
6	Тестування та експериментальне дослідження моделі нейронної мережі.	9 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	10-11 тижні	Додатки
8	Нормоконтроль та рецензування.	12 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка)

(підпис) Віталій ЄВТУШЕНКО
(Імя ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Тетяна ГОЛУБ
(Імя ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра: 75 с., 7 табл., 35 рис., 3 дод., 34 джерел.

COLLAB, KERAS, TENSORFLOW, PYTHON, ДАНИХ, КЛАСТЕРИЗАЦІЯ, МАШИНЕ НАВЧАННЯ, МЕТОДИ, НАБОРИ, ТРАНЗАКЦІЇ, ШАХРАЙСЬКІ ДІЇ.

Об'єкт дослідження – програмні засоби для виявлення шахрайських дій в банківській системі.

Предмет дослідження – методи кластеризації та машинного навчання.

Мета роботи – дослідження та програмна реалізація методу виявлення шахрайських дій в банківській системі

Матеріали, методи та технічні засоби: математичні методи роботи із незбалансованими наборами даних, структурне та об'єктно-орієнтоване програмування, мови програмування Python, бібліотеки TensorFlow, Keras, програмні середовища Collab, персональний комп'ютер з процесором Intel Core 4 Duo під управлінням операційної системи Microsoft Windows 10.

Результати. Створена програмна реалізація для визначення шахрайських дій в транзакціях для малих об'ємів даних.

Висновки. У цьому дослідженні було розроблено підхід машинного навчання для виявлення шахрайських транзакцій на основі різних технологій і методологій у цій галузі. Він демонструє еволюцію методів машинного навчання та їх застосування у вирішенні таких проблем, як дисбаланс даних, коли кількість нешахрайських транзакцій значно перевищує кількість шахрайських.

Галузь використання – банківська система.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 75 pages, 7 tables, 35 figures, 3 appendices, 34 sources.

COLLAB, KERAS, TENSORFLOW, PYTHON, DATA, CLUSTERING, MACHINE LEARNING, METHODS, SETS, TRANSACTIONS, SCAMS.

Object of study is a software tools for detecting fraudulent actions in the banking system.

Subject of study methods of clustering and machine learning.

The purpose of the work is to research and software implementation of the method of detecting fraudulent actions in the banking system

Materials, methods and technical means: mathematical methods of working with unbalanced data sets, structured and object-oriented programming, Python programming languages, TensorFlow libraries, Keras, Collab software environments, a personal computer with an Intel Core 4 Duo processor under the control of the operating room Microsoft Windows 10 systems.

The results. A software implementation has been created to detect fraudulent activities in transactions for small volumes of data.

Conclusions. This study developed a machine learning approach to detect fraudulent transactions based on various technologies and methodologies in the field. It demonstrates the evolution of machine learning techniques and their application in solving problems such as data imbalance, where the number of non-fraudulent transactions far exceeds the number of fraudulent ones.

The field of use is the banking system.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ	8
1 Аналіз стратегій виявлення шахрайських транзакцій.....	10
1.1 Класифікація підходів до виявлення шахрайства	10
1.2 Підходи машинного навчання для виявлення шахрайства	14
1.3 Порівняння методів машинного навчання.....	21
1.4 Висновки за розділом.....	23
2 Дослідження методів обробки даних	25
2.1 Методи предобробки інформації	25
2.2 Нейронні мережі.....	34
2.3 Продуктивність моделі	36
2.4 Висновки за розділом.....	38
3 Програмна реалізація методу виявлення шахрайських дій	40
3.1 Алгоритм розробки	40
3.2 Програмна реалізація роботи із початковим набором даних.....	41
3.3 Застосування досліджуваних моделей	44
3.4 Висновки за розділом.....	47
4 Дослідження результатів роботи методу	48
4.1 Результати роботи нейронної мережі	48
4.2 Висновки за розділом.....	51
Висновки	52
Перелік джерел посилань	54
Додаток А Технічне завдання	58
Додаток Б Лістинги програмної реалізації методів.....	61
Додаток В Слайди презентації.....	67

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

CNN	– Convolutional Neural Network;
CVV	– Card Verification Value;
FN	– False Negative;
FNN	– Feedforward Neural Network;
FP	– False Positive;
ML	– Machine Learning;
MPL	– Multi-Layer Perceptron;
NaN	– Not a Number;
NN	– Neural Network;
OSS	– Open Source Software;
RNN	– Recurrent Neural Network;
ROC	– Receiver Operating Characteristic;
SMOTE	– Synthetic Minority Over-sampling Technique;
SVM	– Support Vector Machine;
TN	– True Negative;
TP	– True Positive.

ВСТУП

Зараз проблема шахрайства в банківському секторі залишається серйозною загрозою. Існує багато методів отримання конфіденційної банківської інформації від нічого не підозрюючих осіб. Ці методи включають отримання «дампів» із баз даних, оманливих фішингових вебсайтів, де жертви несвідомо розголошують номер своєї картки, CVV-код і дату закінчення терміну дії картки, троянські програми, призначені для вилучення паролів і особистих даних із комп'ютерів, а також відстеження дій користувачів, наприклад, як введення з клавіатури. Крім того, злочинці часто використовують більш прості тактики, як-от фізичне викрадення картки користувача або використання спеціальних пристроїв, підключених до банкоматів, для «зчитування» даних картки.

Ці проблеми відображені в щорічній статистиці, зібраній на основі банківських звітів про збитки з різних причин. Згідно зі статистикою 2022 року, збитки банків через шахрайство склали приблизно 35 мільярдів доларів [1]. Це збільшення в першу чергу пов'язане з широким використанням безготівкових і онлайн-платежів і замовлень, які стали свідками експоненційного зростання після початку пандемії COVID-19.

За останнє десятиліття машинне навчання набуло все більшого поширення завдяки вдосконаленню методологій, створенню більш потужних систем і доступності величезних наборів даних, зібраних із різних джерел для навчальних моделей. Технології машинного навчання дозволяють розробляти системи, подібні до тих, які можна створити лише вручну, доклавши значних зусиль людини [2].

Застосування моделей машинного навчання пропонує альтернативу розробці спеціальних правил для виявлення шахрайства. Натомість це дає змогу фахівцям зосередитися на обробці великих обсягів даних, що є необхідністю, зумовленою потребою моделей у великій кількості даних, щоб краще розпізнавати моделі підозрілої поведінки та потенційного шахрайства.

Цей перехід віщує більш ефективний підхід до боротьби з шахрайством у банківському секторі.

Метою проєкту є дослідження та розробка методу виявлення шахрайських дій у банківській системі за допомогою машинного навчання.

Завдання, які треба вирішити:

- проаналізувати стратегії виявлення шахрайства та порівняти підходи машинного навчання;
- дослідити методи обробки даних, зокрема боротьбу з дисбалансом класів;
- розробити програмну реалізацію методу на основі градієнтного бустингу та нейронних мереж;
- протестувати та оптимізувати моделі, порівняно їх ефективності.

Проблема шахрайства залишається серйозною загрозою для банківської системи і є досить актуальною для дослідження. Машинне навчання пропонує ефективний підхід для автоматизації виявлення шахрайства. Розроблений метод може бути застосований банками для захисту від збитків та підтримки довіри клієнтів.

1 АНАЛІЗ СТРАТЕГІЙ ВИЯВЛЕННЯ ШАХРАЙСЬКИХ ТРАНЗАКЦІЙ

1.1 Класифікація підходів до виявлення шахрайства

Виявлення та запобігання шахрайству в банківській системі мають вирішальне значення для захисту фінансових активів і збереження довіри клієнтів. Два відомих підходи до виявлення шахрайства — це експертні системи та машинне навчання [3]:

- системи, розроблені експертами з безпеки в конкретних банківських установах працюють на основі попередньо визначених наборів правил, ретельно розроблених фахівцями. Ці правила сформульовані на основі накопичення практичного досвіду роботи з випадками шахрайства, а іноді й думок, якими діляться колеги з інших банківських установ під час обміну інформацією [4];

- системи, засновані на принципах машинного навчання (ML) покладаються на аналіз обширних наборів даних інформації про транзакції. Ці набори даних охоплюють безліч деталей транзакцій, зокрема часові позначки транзакцій, грошові суми, що переказуються, а також інформацію про клієнта й одержувача, серед іншого [5]. Алгоритми машинного навчання керують цими системами, виявляючи певні правила під час навчання або аналізу, проведеного на заздалегідь визначеному наборі даних, який може бути позначений для цієї мети або не позначений, вимагаючи від алгоритму автономного визначення помітних моделей, які можуть відрізнити аномальні транзакції від переважної більшості.

Проаналізуємо та порівняємо застосування цих методологій у контексті виявлення банківського шахрайства (табл. 1.1).

Системи, засновані на правилах, є більш переважними завдяки своїй прозорості та можливості інтерпретації.

Таблиця 1.1 – Машинне навчання для виявлення шахрайства

Параметри	Опис
Переваги	
Адаптивність	Моделі машинного навчання здатні адаптуватися до мінливих моделей шахрайства. Вони можуть вивчати нові дані та виявляти складні схеми шахрайства, що розвиваються, які системи на основі правил можуть пропустити.
Масштабованість	Системи машинного навчання можуть ефективно обробляти великі обсяги даних. Вони можуть обробляти величезні набори даних для виявлення тонких закономірностей і аномалій, що має вирішальне значення в банківському середовищі з мільйонами транзакцій.
Автоматизація	Після навчання моделі машинного навчання можуть автоматизувати процес виявлення шахрайства, зменшуючи потребу в постійному ручному втручанні.
Недоліки	
Складність	Моделі машинного навчання часто розглядаються як «чорні ящики». Їхні процеси прийняття рішень можуть бути складними для тлумачення, що може спричинити труднощі з поясненням обґрунтування рішень щодо виявлення шахрайства, особливо щодо дотримання нормативних вимог
Залежність від даних	Ефективність моделей машинного навчання значною мірою залежить від якості та репрезентативності навчальних даних. Упереджені або недостатні дані можуть призвести до упереджених або неефективних моделей.
Ресурсовитратність	Розробка, навчання та підтримка моделей машинного навчання вимагає значних обчислювальних ресурсів і досвіду. Інституції повинні інвестувати в науковців з обробки даних, обчислювальну інфраструктуру та постійний моніторинг і оновлення моделей.

Далі розглянемо переваги та недоліки кожного підходу, щоб отримати більш повне розуміння сфери застосування (табл. 1.2).

Таблиця 1.2 – Експертні системи для виявлення шахрайства

Параметри	Опис
Переваги	
Прозорість і можливість інтерпретації:	Експертні системи засновані на правилах і чітко визначені, що робить їхні процеси прийняття рішень прозорими та такими, що їх можна інтерпретувати. Це має вирішальне значення для відповідності, оскільки регулятори часто вимагають від установ пояснення їхніх механізмів виявлення шахрайства.
Налаштування	Експерти з безпеки можуть адаптувати експертні системи відповідно до конкретних потреб і вимог своєї банківської установи. Вони можуть використовувати специфічні для домену знання та точно налаштовувати правила для ефективного націлювання на відомі схеми шахрайства.
Швидке впровадження	Експертні системи можна розгортати відносно швидко, оскільки вони покладаються на заздалегідь визначені правила. Це робить їх привабливими варіантами для установ, яким потрібні рішення для негайного запобігання шахрайству.
Недоліки	
Відсутність адаптивності	Експертним системам може бути важко адаптуватися до нових тактик шахрайства. Вони покладаються на заздалегідь визначені правила, які можуть застаріти, оскільки шахраї постійно розробляють нові стратегії. Системи можуть не виявити нові загрози.
Помилкові спрацювання	Надто суворі набори правил можуть призвести до збільшення кількості хибних спрацювань, що створює незручності для законних клієнтів і може призвести до операційної неефективності.

Експерти з безпеки розробляють ці системи на основі свого досвіду,

завдяки чому вони здатні виявляти конкретні схеми шахрайства.

Цей підхід відносно простий у застосуванні та може бути пристосований до унікальних потреб конкретної банківської установи. Однак це також має свої обмеження. Системам, заснованим на правилах, може бути важко адаптуватися до тактики шахрайства, що розвивається, і їх продуктивність значною мірою залежить від якості попередньо визначених правил. Вони також можуть генерувати хибні спрацьовування, створюючи незручності законним клієнтам і спричиняючи робочі труднощі [6].

Системи на основі машинного навчання пропонують потенціал адаптації та масштабованості. Ці системи можуть розкривати заплутані схеми та виявляти нові схеми шахрайства. Аналізуючи великі, часто немарковані набори даних, вони можуть адаптуватися до нових загроз. Однак вони постають зі своїми проблемами. Їхні операції часто розглядаються як «чорні скриньки», що ускладнює розуміння процесу прийняття рішень. Ефективність систем машинного навчання значною мірою залежить від якості та репрезентативності навчальних даних. Крім того, вони можуть бути обчислювально інтенсивними та вимагати досвіду вибору алгоритму та налаштування моделі [7].

Таким чином, вибір між системами виявлення шахрайства на основі правил і машинного навчання потребує ретельної оцінки конкретних потреб і обмежень банківської установи. Системи, засновані на правилах, пропонують прозорість і легкість впровадження, але можуть відставати в адаптації до нових моделей шахрайства. З іншого боку, системи на основі машинного навчання забезпечують адаптивність і масштабованість, але вимагають значних даних і досвіду. Рішення залежить від досягнення правильного балансу між цими факторами для ефективного підвищення безпеки банківських операцій.

1.2 Підходи машинного навчання для виявлення шахрайства

Завдання виявлення шахрайських транзакцій можна сформулювати як завдання розрізнити відносно невелику кількість нерегулярних подій або предметів у величезному наборі подібних подій або предметів, які поведуться традиційно в конкретному контексті (рис. 1.1).

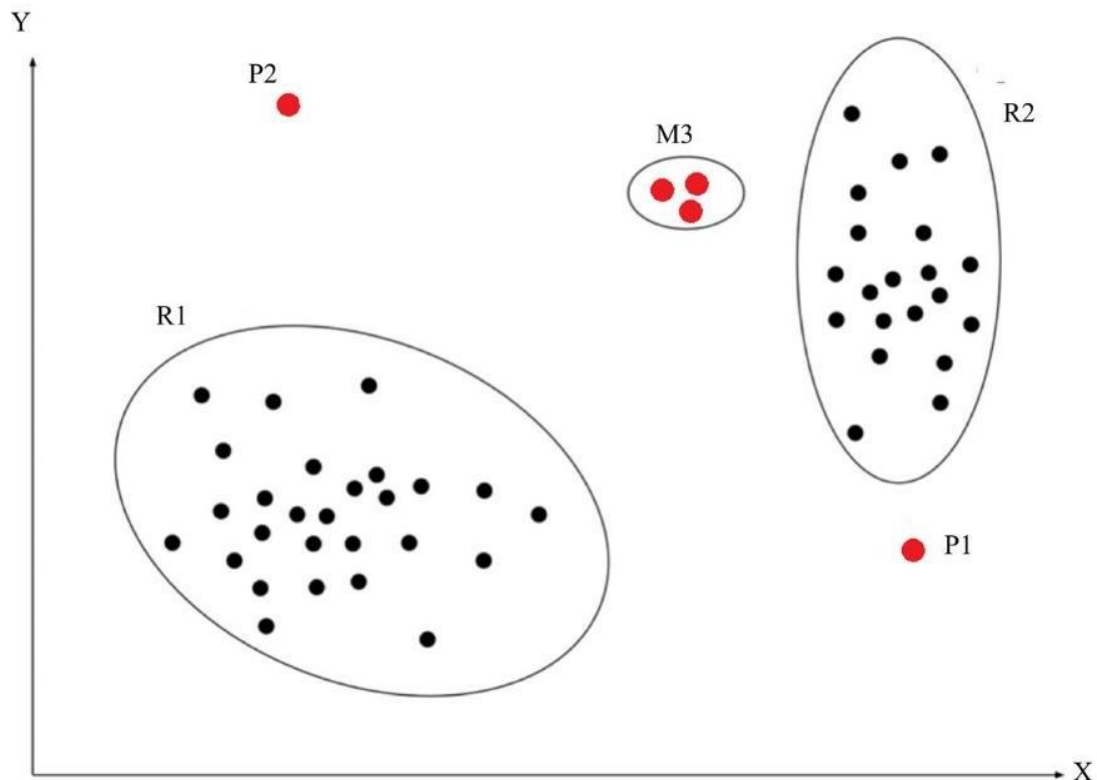


Рисунок 1.1 – Приклад аномальних точок [8]

Як показано на рисунку, більшість точок даних зосереджено в двох великих областях, позначених як R1 і R2. Однак точки P1 і P2 разом з меншим кластером M3 відхиляються від загального розподілу і, отже, розглядаються як аномалії. Ця проблема виявлення аномалії, яку іноді називають виявленням викидів, є поширеною проблемою, яка виникає в різних областях людської діяльності та наукових досліджень [8].

Як наслідок, існує безліч методів ідентифікації цих аномалій [3] з різним ступенем спеціалізації. Деякі методи вузько пристосовані до конкретних застосувань, тоді як інші мають більш універсальну сферу

застосування. На рисунку 1.2 представлено огляд методів, здатних виявляти аномальну діяльність.

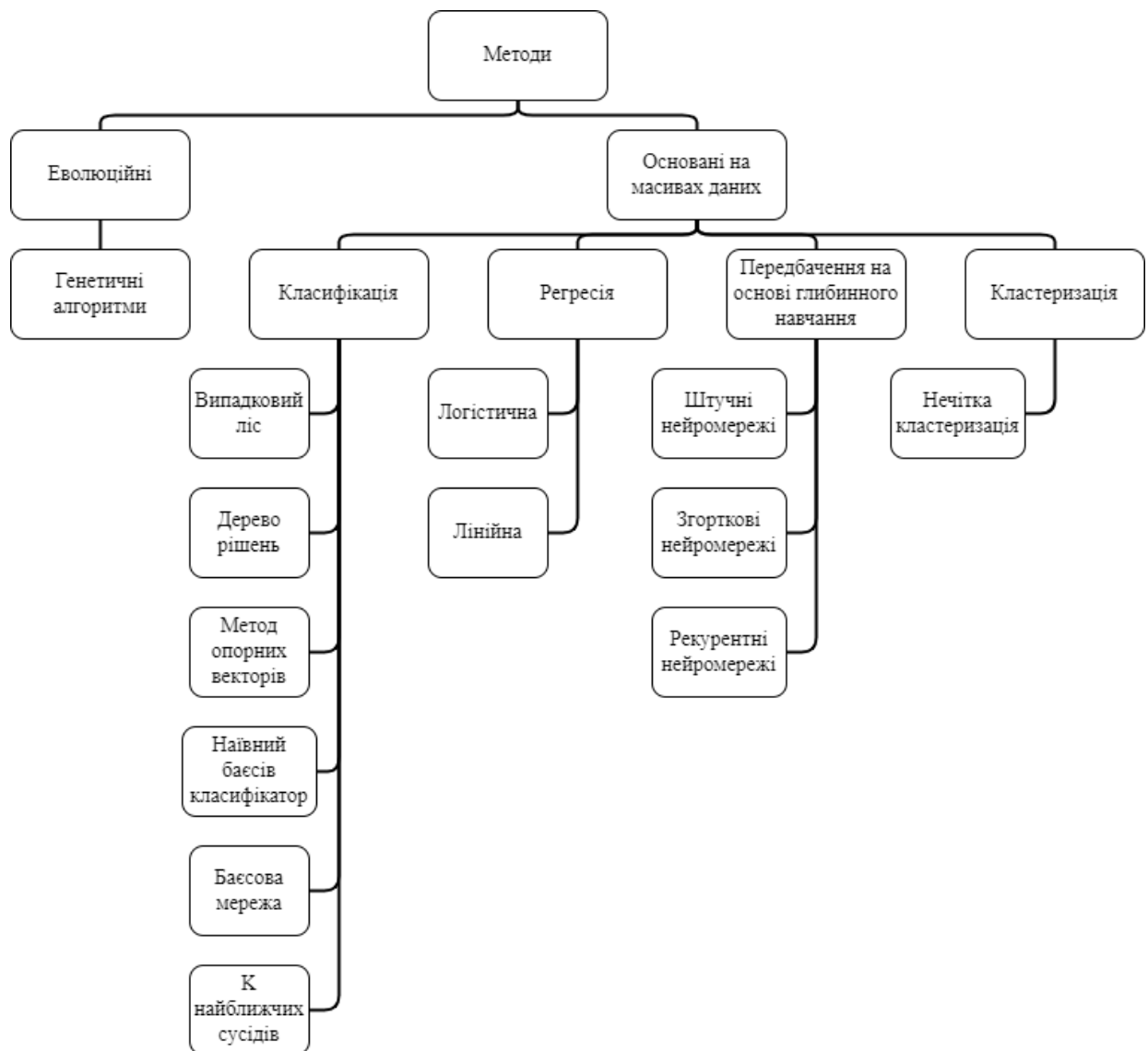


Рисунок 1.2 – Методи виявлення аномалій

У сфері виявлення шахрайства в банківському секторі ця систематика методів виявлення аномалій стає особливо актуальною. Використовуючи моделі машинного навчання, аналітики можуть використовувати ці методи, щоб відрізнити шахрайські транзакції від справжніх, які часто демонструють чіткі шаблони або характеристики, що позначають їх як аномалії в ширшому наборі даних. Машинне навчання розширює можливості цих методів, автоматизуючи процес виявлення та позначення підозрілих дій, сприяючи

підвищенню безпеки у фінансовому середовищі. Оскільки складність і різноманітність тактик шахрайства продовжує розвиватися, адаптивність і масштабованість підходів машинного навчання робить їх цінним активом у поточній боротьбі з фінансовим шахрайством [9].

Дерево рішень, як показано на рисунку 1.3, є алгоритмом навчання під наглядом. Він приймає деревоподібну структуру, що складається з кореневого вузла та різноманітних дочірніх вузлів, які розгалужуються у двох або більше напрямках. Кожна гілка керується спеціальним алгоритмом для процесу поділу даних, ефективно класифікуючи екземпляри на основі їхніх атрибутів [6]. Однак дерева рішень можуть зіткнутися з проблемами, пов'язаними з надмірним зростанням, що потенційно може призвести до перенавчання та появи аномалій або неточностей у гілках. Щоб пом'якшити ці проблеми, для оптимізації результатів класифікації дерева використовується обрізка гілок. Деревам рішень віддають перевагу через їх зручність і адаптивність при роботі з атрибутами різних типів даних.

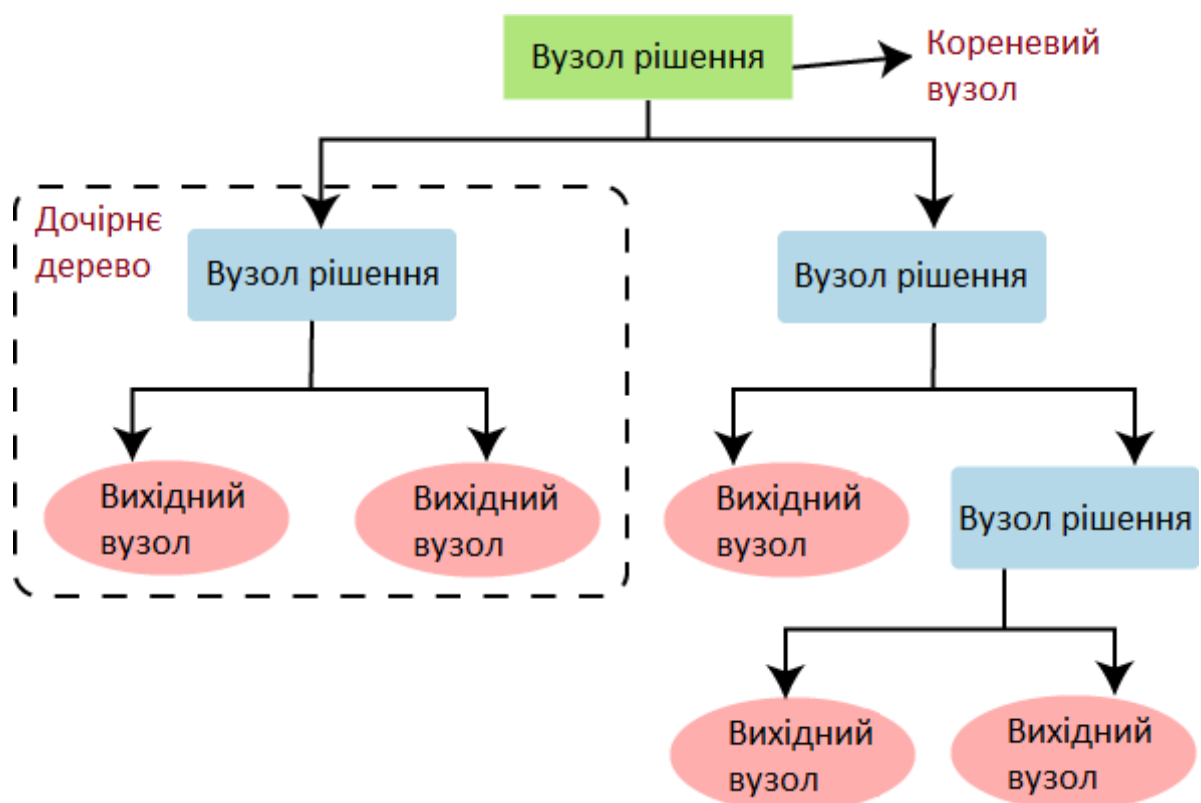


Рисунок 1.3 – Архітектура дерева рішень [10]

У вузлах рішення оцінюються певні параметри, що призводить до поділу вхідних даних на дві або більше підмножин (у випадку багатокласової класифікації) [7]. Вихідні вузли, з іншого боку, забезпечують остаточне рішення щодо класифікації екземплярів даних у певні класи.

Випадкові ліси, як показано на рисунку 1.4, виникли як відповідь на чутливість окремих дерев рішень до конкретних навчальних даних і їх невід’ємну нестабільність. У випадковому лісі кілька дерев рішень будуються незалежно, що призводить до значного підвищення ефективності обчислень [8]. Цей підхід ефективно формує колектив дерев, призначених для завдань регресії або класифікації. Відхилення розподіляються по деревах у лісі, що спрощує процес побудови моделі. Кожне дерево генерується з використанням початкових зразків усіх даних і випадкової підмножини атрибутів.

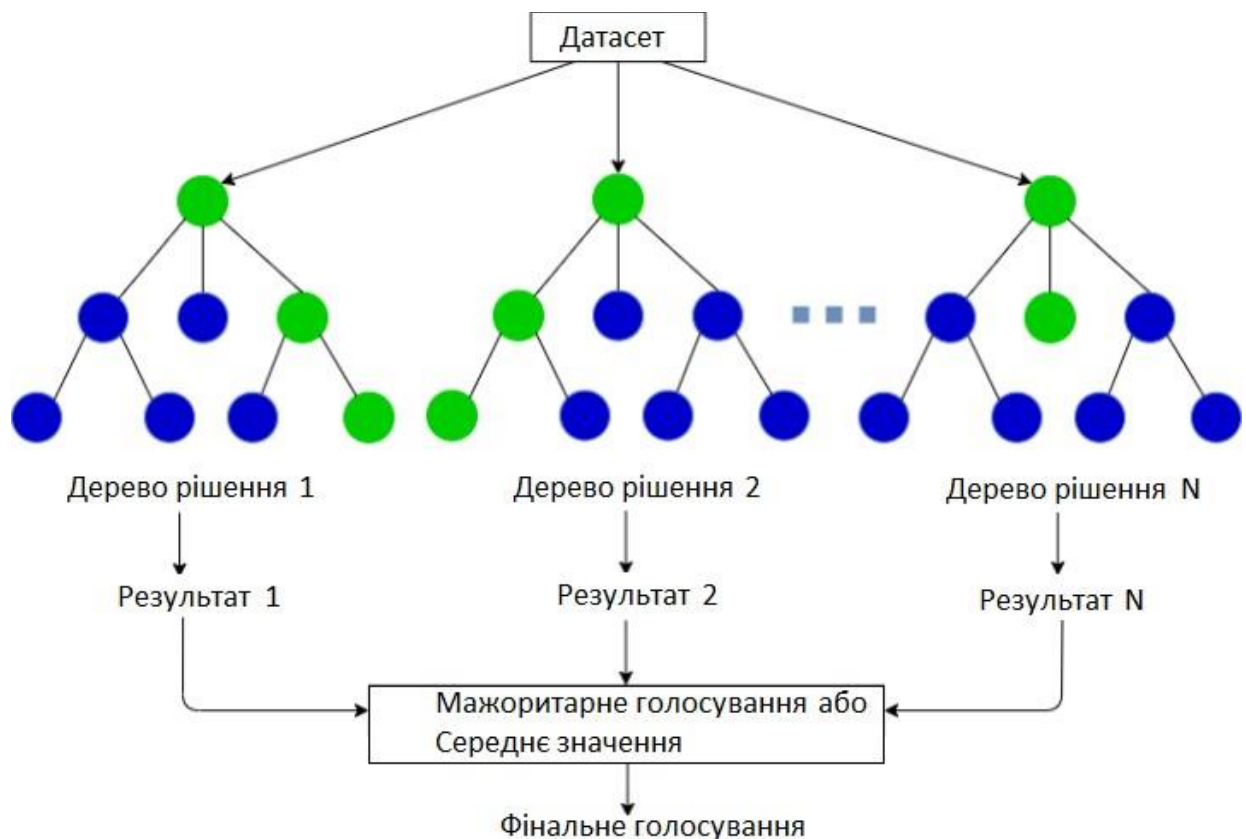


Рисунок 1.4 – Структура випадкового ліса

На рисунку 1.4 кожне дерево унікально підходить до проблеми

класифікації. Згодом більшістю голосів визначається приналежність екземпляра до класу, або у випадку регресійного завдання обчислюється середнє значення, яке вказує на ймовірність належності до певного класу.

Байєсовські мережі використовують теорему Байєса для обчислення ймовірності гіпотези та визначення її правдивості. Класифікатор використовується для обчислення умовних ймовірностей для всіх класів, присутніх у конкретному виклику класифікації, і визначення класу з найвищим значенням умовної ймовірності для даного параметра X [10]. Графічно це можна візуалізувати як ациклічний орієнтований граф, як показано на рисунку 1.5, де навчальні приклади зображені як вузли графа, а залежності представлені орієнтованими ребрами. Дві змінні вважаються незалежними, якщо між ними немає сполучних ребер. Байєсовські мережі полегшують факторизацію та специфікацію спільного розподілу ймовірностей [11].

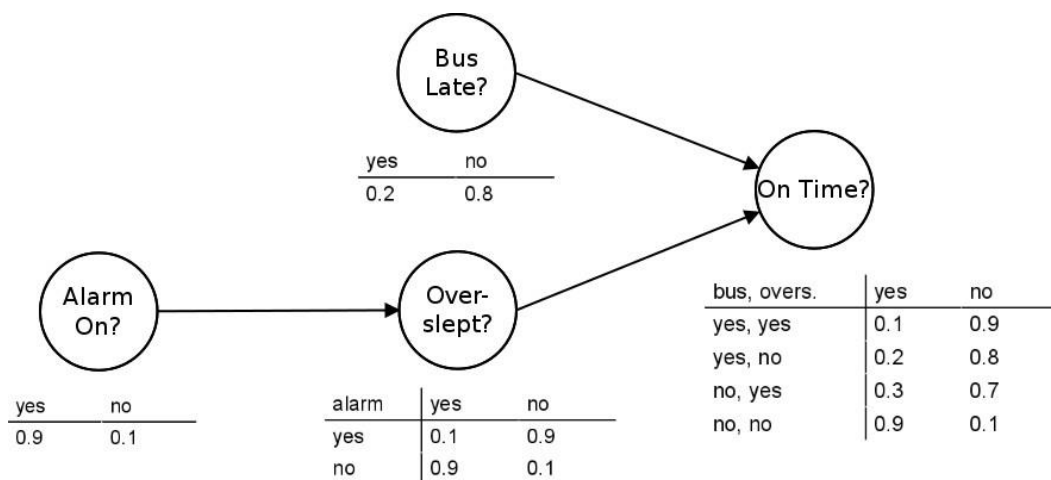


Рисунок 1.5 – Приклад мережі Байєса [11]

Генетичні алгоритми черпають натхнення з принципів природної еволюції, де перспективи виживання здоровіших і сильніших індивідів у популяції є кращими над перспективами слабших [12]. Алгоритм імітує цей процес, відбираючи найбільш придатних особин для розмноження, що в кінцевому підсумку підвищує середню придатність популяції. Найкращі

представники попереднього покоління вибираються батьками для наступного покоління, а менш придатні кандидати відкидаються. Цей еволюційний процес залежить від численних параметрів, таких як кількість батьків, кількість нащадків у наступному поколінні, функції відтворення та пристосованості, а також критерії зупинки розвитку популяції [13].

Логістична регресія є підходящою технікою для прогнозного аналізу, коли залежна змінна демонструє бінарні характеристики [14]. З огляду на те, що виявлення шахрайських транзакцій за своєю суттю передбачає двійкову класифікацію, цей метод добре підходить. Логістична регресія використовує ймовірнісну статистичну модель класифікації, яка використовує логістичну криву для виявлення шахрайства. Оскільки значення логістичної кривої коливаються від 0 до 1 (рис. 1.6), вона підходить для інтерпретації ймовірностей приналежності до певного класу. Модель навчається та перевіряється з використанням попередньо класифікованого набору даних, а також вибирається мінімальний поріг для прийняття рішень щодо членства в класі. Крім того, процес передбачає визначення найбільш значущих змінних і відповідне коригування моделі. Логістична регресія використовує функцію натурального логарифма для обчислення ймовірностей і визначення класу [15].

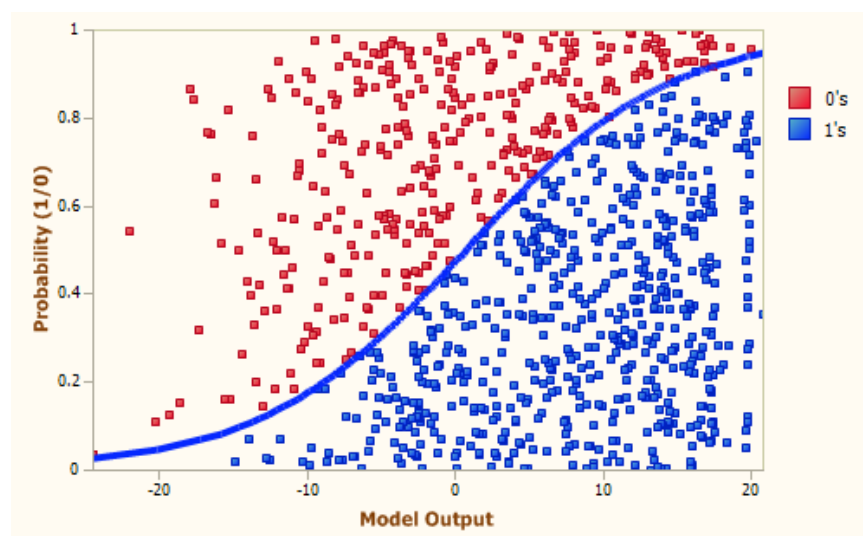


Рисунок 1.6 – Використання генетичного алгоритма

Метод опорного вектора, або SVM, служить лінійним класифікатором, особливо придатним для вирішення умов великого параметра, де нелінійні вхідні завдання перетворюються на лінійні, що робить SVM придатними для виявлення шахрайства [16]. Двома основними атрибутами SVM є його функції ядра, які представляють функцію класифікації як скалярний добуток проєкцій точок вхідних даних, і мета методу ідентифікувати гіперплощину (рис. 1.7), яка максимізує поділ класів, мінімізуючи потребу в інтенсивному перенавчанні даних навчання. Ця властивість забезпечує високий потенціал узагальнення [17].

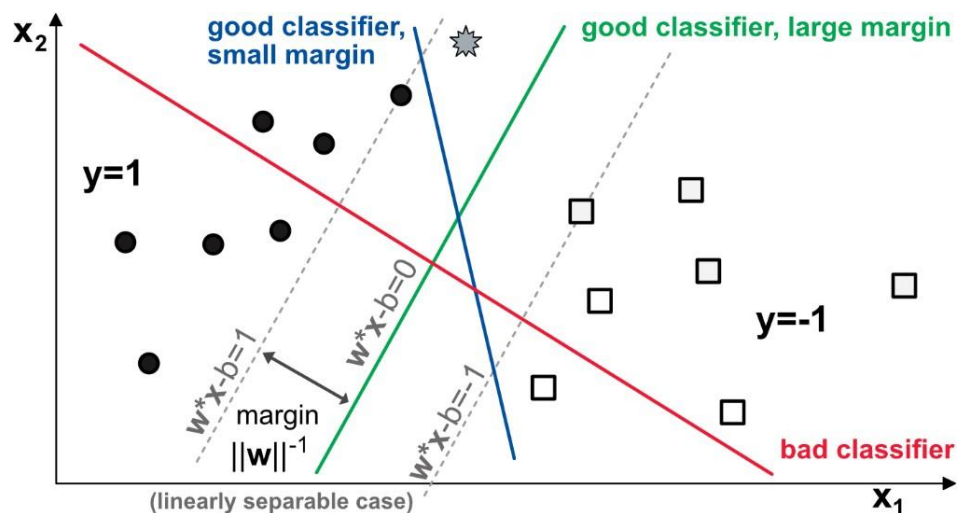


Рисунок 1.7 – Приклад методи опорних векторів [7]

Нейронна мережа — це складна система, що складається з нейронів із ключовим компонентом, відомим як перцептрон, який служить основним лінійним двійковим класифікатором для вхідних даних. Кожен перцептрон структурований на чотири послідовні частини: вхідний рівень, ваги та переміщення, зважене підсумовування та функція активації. Існує три основних типи нейронних мереж, кожна з яких має різні характеристики та застосування.

Одним із таких типів нейронної мережі є штучна нейронна мережа (ШНМ), яка складається з трьох рівнів: вхідного, прихованого та вихідного. Дані проходять процес прямого поширення, проходячи від вхідного рівня

через проміжний прихований рівень і, нарешті, до вихідного рівня. Навчання починається з того, що мережа вивчає типову поведінку власника картки, а потім класифікує транзакції як звичайні або потенційно шахрайські. Ефективність цієї методики полягає в її адаптованості без необхідності перепрограмування, що забезпечує швидку обробку вхідних даних [18].

Згорткові нейронні мережі (CNN) містять нейрони з вивченими значеннями ваги та зміщення, отриманими під час навчання мережі. Кожен нейрон отримує різні вхідні вектори, обчислює зважену суму, обробляє її за допомогою функції активації та видає рішення щодо класифікації транзакції як шахрайської чи ні [19].

Повторювані нейронні мережі (RNN) представляють циклічну структуру, яка дозволяє зберігати інформацію. По суті, його можна розглядати як багат шаровий персептрон, де кожен рівень передає інформацію наступному. RNN підтримують внутрішні стани, оновлені на кожному кроці часу, але вони стикаються з проблемами під час навчання. Двома поширеними проблемами є зникнення градієнта та вибух градієнта, особливо зі збільшенням довжини вхідної послідовності. Щоб вирішити ці проблеми, були розроблені такі методи, як усунення залежності послідовності від вагових матриць і впровадження нових комірок пам'яті, відомих як мережі довготривалої короткочасної пам'яті (LSTM) [20] – [21]. Мережі LSTM розроблені для вирішення цих проблем, забезпечуючи покращену продуктивність і стабільність послідовностей обробки.

1.3 Порівняння методів машинного навчання

Порівняємо алгоритми машинного навчання, описані в попередньому розділі: дерева рішень, випадкові ліси, логістичну регресію, генетичні алгоритми та опорні векторні машини (SVM), а також нейронні мережі (табл. 1.3).

Таблиця 1.3 – Порівняння методів машинного навчання до знаходження шахрайський дій

Назва методу	Переваги	Недоліки
Дерева рішень [22]	Прозорість і можливість інтерпретації, зручний для користувача, гнучкість з різними типами даних	Схильність до переобладнання та може погано адаптуватися до нових тактик шахрайства
Випадковий ліс [23]	Покращена продуктивність у порівнянні з окремими деревами рішень, адаптованість до змінних даних, відмінна масштабованість і надійність	Недоліки: у деяких випадках все ще може перевищувати, складна модель з обмеженою інтерпретацією
Логістична регресія [24]	Підходить для задач бінарної класифікації, інтерпретується як ймовірності, ефективний в обробці вхідних даних.	Обмежені можливості обробки складних нелінійних зв'язків у даних.
Генетичні алгоритми [25]	Може оптимізувати складні моделі, підходить для вибору складних функцій	Сильно залежить від параметрів, складний у налаштуванні та реалізації.
Підтримка векторних машин (SVM) [26]	Лінійний класифікатор здатний обробляти дані великої розмірності, хороше узагальнення, ефективний для максимального розділення класів.	Може потребувати інтенсивних обчислень, особливо для великих наборів даних, потребує налаштування параметрів, менш піддається інтерпретації.
Нейронні мережі [27]	Надзвичайно гнучкий і потужний, здатний обробляти різні типи даних, варіанти глибокого навчання можуть виявляти складні шаблони, добре вилучати ознаки, висока швидкість обробки	Схильність до переобладнання (особливо для глибоких архітектур), складна в налаштуванні, може вимагати значних обчислювальних ресурсів, їх можна розглядати як «чорні ящики» з обмеженою інтерпретацією

Таким чином, вибір алгоритму машинного навчання для виявлення шахрайства залежить від конкретних вимог проблеми, доступних даних і компромісів між можливістю інтерпретації, адаптивністю та обчислювальними ресурсами. Дерева рішень і випадкові ліси прозорі та зручні для користувача, але можуть відставати в адаптації. Логістична регресія ефективна для завдань бінарної класифікації, але може мати проблеми зі складними даними. Генетичні алгоритми пропонують можливості оптимізації, але вимагають ретельного налаштування параметрів. SVM — це лінійні класифікатори з хорошим узагальненням, тоді як нейронні мережі забезпечують адаптивність і можливості глибокого навчання, але ціною складності та обчислювальних вимог. Найкращий вибір залежить від конкретного контексту та компромісів, які найкраще відповідають наявній проблемі.

1.4 Висновки за розділом

У цьому розділі представлено порівняльний аналіз різних алгоритмів і методів машинного навчання, які використовуються в області виявлення банківського шахрайства. Огляд охоплює широкий спектр методологій, включаючи дерева рішень, випадкові ліси, логістичну регресію, генетичні алгоритми, опорні вектори і нейронні мережі. Кожен із цих підходів має певні переваги та недоліки, а їх придатність залежить від конкретних вимог і складності проблеми. Мета цього дослідження полягає в тому, щоб забезпечити повне розуміння цих алгоритмів, допомагаючи фахівцям-практикам зробити обґрунтований вибір для своїх потреб у виявленні шахрайства.

Виявлення шахрайства є надзвичайно важливою проблемою в банківському секторі, де загроза фінансових втрат і скомпрометована довіра клієнтів є великою. Машинне навчання пропонує різноманітний набір підходів для вирішення цієї проблеми. Цей порівняльний аналіз заглиблюється в

унікальні атрибути кожного алгоритму машинного навчання, допомагаючи зацікавленим сторонам приймати обґрунтовані рішення щодо прийняття найбільш підходящого методу.

У сфері виявлення банківського шахрайства вибір алгоритму машинного навчання залежить від кількох факторів, включаючи характеристики даних, вимоги до інтерпретації, адаптивність до нових тактик шахрайства та доступні обчислювальні ресурси. Дерева рішень і випадкові ліси забезпечують прозорість, але можуть не мати можливості адаптації. Логістична регресія ефективна для бінарних завдань, але обмежена в обробці складних даних. Генетичні алгоритми пропонують потенціал оптимізації, але вимагають налаштування параметрів. SVM забезпечують лінійну класифікацію з хорошим узагальненням, а нейронні мережі пропонують адаптивність і можливості глибокого навчання, хоча й з додатковою складністю та вимогами до обчислень.

2 ДОСЛІДЖЕННЯ МЕТОДІВ ОБРОБКИ ДАНИХ

2.1 Методи передобробки інформації

Після ініціалізації аналізу набору даних дуже важливо оцінити баланс цільової змінної, а потім виконати важливий етап попередньої обробки даних [20]. У реальних сценаріях законні транзакції значно перевищують кількість шахрайських. Для цього аналізу використовувався набір із реальними даними, де численні поля були зашифровані та змінені для забезпечення конфіденційності персональних даних. Незважаючи на велику кількість цих полів, не всі безпосередньо вказують на те, чи є транзакція шахрайською чи законною. Отже, необхідно оцінити, як ці поля корелюють із цільовою змінною.

Коефіцієнт кореляції Пірсона в статистиці — це показник, який використовується для вимірювання лінійного зв'язку між двома наборами даних [21]. Він обчислюється як відношення коваріації двох змінних до добутку їхніх стандартних відхилень. Цей розрахунок стандартизує значення коваріації, обмежуючи діапазон коефіцієнта від -1 до 1 (рис. 2.1).

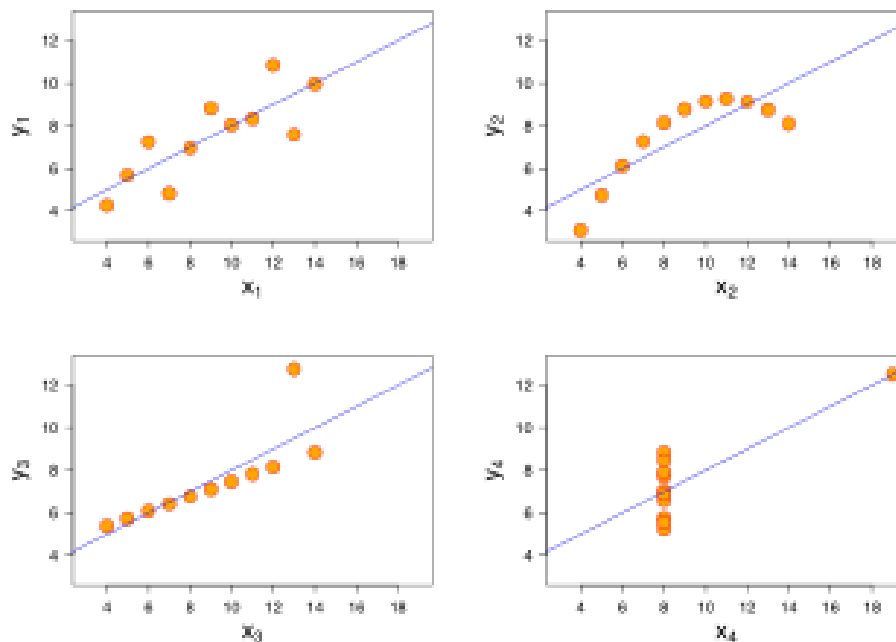


Рисунок 2.1 - Коефіцієнт кореляції Пірсона

Однак важливо зазначити, що цей коефіцієнт вимірює лише лінійні кореляції та не враховує інші типи зв'язків або залежностей, які можуть існувати.

Коефіцієнт кореляції Пірсона розраховується за формулою (2.1):

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}. \quad (2.1)$$

Ця кореляція допомагає відфільтрувати нерелевантні змінні для класифікації та ідентифікувати сильно взаємопов'язані змінні, таким чином оптимізуючи обробку даних під час навчання.

Дисбаланс класів у наборах даних є поширеною проблемою в машинному навчанні та науці про дані, яка часто спостерігається в таких сценаріях, як виявлення шахрайства (рис. 2.2). Цей дисбаланс виникає, коли один клас значно переважає інший, що потенційно може призвести до спотворення продуктивності моделі. Щоб вирішити цю проблему, для досягнення збалансованого співвідношення класів реалізовано такі методи, як недостатня та надмірна вибірка [22].

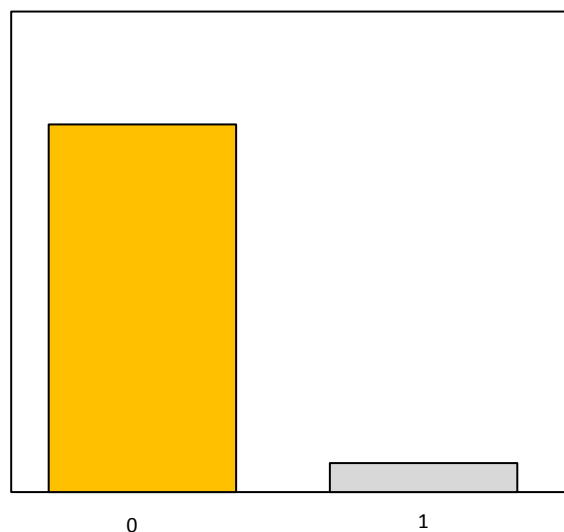


Рисунок 2.2 - Дисбаланс класів у наборах даних

Прості стратегії включають випадкове видалення елементів із основного класу (неповна вибірка) або дублювання даних із меншого класу (надвибірка). Однак ці методи мають обмеження, особливо в задачах виявлення шахрайства. Дублювання даних може призвести до переобладнання моделі без додавання інформаційної цінності, а випадкове видалення може призвести до втрати важливих даних для ідентифікації законних транзакцій [23].

Техніка передискретизації синтетичної меншості (SMOTE) ефективно збільшує розмір другорядного класу, не покладаючись на просте дублювання даних [24]. Цей метод черпає натхнення з успішних методів, що використовуються для розпізнавання рукописного тексту, де розширення навчальних даних передбачало маніпулювання наявними даними шляхом обертання, нахилу та розтягування рукописних символів, що значно покращує результати навчання моделі. На відміну від цих методів, SMOTE працює шляхом створення нових зразків у просторі ознак, а не в просторі даних, пропонуючи більш складний підхід до розширення даних.

SMOTE покращує клас меншості, беручи кожен екземпляр і генеруючи нові синтетичні екземпляри вздовж ліній, які з'єднують кілька або всі K найближчих сусідів у межах одного класу. Кількість сусідів, обраних для синтезу, залежить від необхідного збільшення вибірок класу меншості. Наприклад, якщо метою є подвоїти розмір класу меншості, алгоритм випадковим чином вибирає двох сусідів із K найближчих і генерує одну вибірку для кожного (рис. 2.3).

Процес створення синтетичних прикладів за допомогою SMOTE включає такі кроки [24]:

- обчислити різницю між вектором ознак (конкретним екземпляром) і його найближчим сусідом;
- помножити цю різницю на випадкове число в діапазоні від 0 до 1 і додати її до вихідного вектора ознак. Це випадковим чином вибирає точку уздовж відрізка лінії між двома об'єктами, ефективно розширюючи та

узагальнюючи сферу застосування класу меншості.

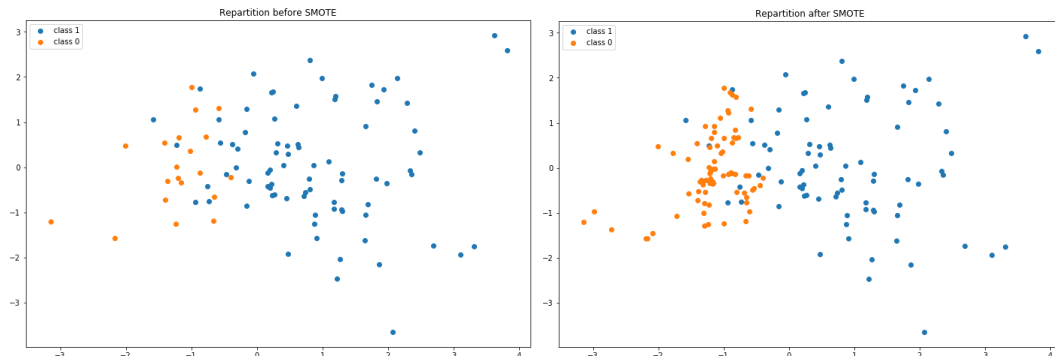


Рисунок 2.3 – SMOTE аналіз

Реалізація алгоритму SMOTE на Python, як показано на рис. 2.4, демонструє застосування цієї техніки в середовищі програмування.

```
def SMOTE(sample: np.array, N: int, k: int) -> np.array:
    T, num_attrs = sample.shape

    # Якщо N менше 100%, рандомізуйте вибірки міноритарного класу, оскільки лиш
    # випадковий процент з них буде підлягати SMOTE
    if N < 100:
        T = round(N / 100 * T)
        N = 100
    # Передбачається, що результат SMOTE буде цілочислено кратний 100
    N = int(N / 100)
    synthetic = np.zeros([T * N, num_attrs])
    new_index = 0
    nbrs = NearestNeighbors(n_neighbors=k+1).fit(sample.values)

    def populate(N, i, nnarray):
        nonlocal new_index
        nonlocal synthetic
        nonlocal sample
        while N != 0:
            nn = randrange(1, k+1)
            for attr in range(num_attrs):
                dif = sample.iloc[nnarray[nn]][attr] - sample.iloc[i][attr]
                gap = uniform(0, 1)
                synthetic[new_index][attr] = sample.iloc[i][attr] + gap * dif
            new_index += 1
            N = N - 1

    for i in range(T):
        nnarray = nbrs.kneighbors(sample.iloc[i].values.reshape(1, -1), return_
distance=False)[0]
        populate(N, i, nnarray)

    return synthetic
```

Рисунок 2.4 – Приклад функції SMOTE

Використовуючи синтетичні приклади, алгоритм SMOTE заохочує

класифікатори формувати менші, більш загальні області прийняття рішень. Це покращує здатність класифікаторів узагальнювати ознаки та запобігає переобладнанню. Крім того, використання SMOTE для збагачення класу меншості показує покращені показники продуктивності на кривій робочих характеристик приймача (ROC).

Крім того, поєднання SMOTE з методами недостатньої вибірки виявилось ефективною стратегією для більш точного балансування пропорцій класів у навчанні класифікатора. Ця комбінація усуває обмеження, притаманні лише надмірній вибірці, пропонуючи більш детальний і ефективний підхід до боротьби з дисбалансом класів у наборах даних. Інтегруючи методи надмірної та недостатньої вибірки, класифікатори можуть досягти кращого узагальнення та точності прогнозування.

Посилання Tomek представляють пари точок даних у наборі даних, які є найближчими сусідами одна одної, але належать до різних класів. Ці пари становлять складність у класифікаційних завданнях, оскільки вони стирають межі між класами. У бінарній класифікації посилання Tomek — це пара прикладів, які є найближчими сусідами один до одного, але з протилежних класів. Візуалізація лінії між ними перетинатиме область, де очікується межа рішення, ускладнюючи точну класифікацію. Ця концепція набула популярності в машинному навчанні для попередньої обробки даних для вдосконалення алгоритмів класифікації, особливо в наборах даних з дисбалансом класів [25].

Математично зв'язок Tomek визначається за допомогою евклідової відстані між двома зразками. Коли такі зв'язки ідентифікуються, вибірки більшості класів, які беруть участь у цих зв'язках, як правило, видаляються, як показано на рис. 2.5.

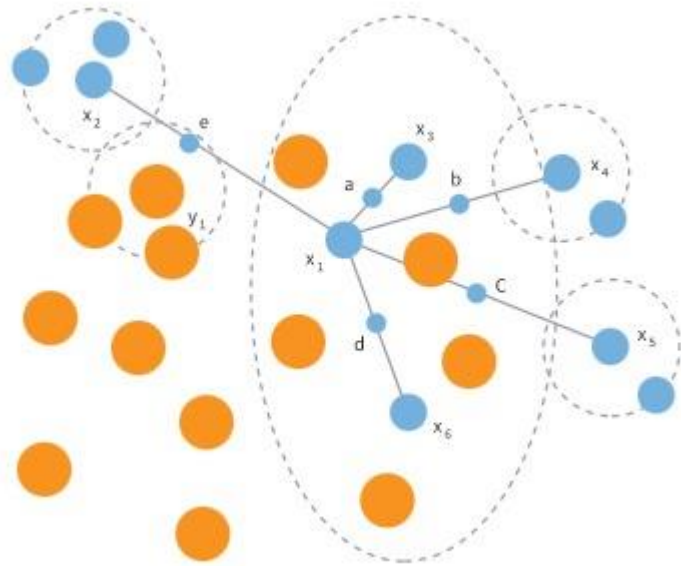


Рисунок 2.5 – Зв’язок Tomek

Правило згорнутого найближчого сусіда (CNN), інший метод, зосереджується на виборі найбільш інформативних екземплярів навчання для класифікації нових даних. Він починається з порожнього набору та поступово додає екземпляри. Примірник додається, лише якщо його неправильно класифіковано поточним набором, гарантуючи збереження лише інформативних прикладів [26]. При незбалансованій класифікації даних процес починається з включення всіх екземплярів класу меншості, а потім продовжується як зазвичай, як показано на рис. 2.6. Цей метод допомагає зменшити надмірність даних, зберігаючи при цьому точність класифікації.

Односторонній вибір (OSS) — це підхід, який поєднує принципи правила CNN і посилянь Tomek. Спочатку застосовується правило CNN, зберігаючи всі приклади класу меншості та лише ті приклади класу більшості, які надають відмінні риси. Після цього екземпляри класу більшості, які утворюють зв’язки Tomek з класом меншості, видаляються. Алгоритм OSS можна підсумувати таким чином: почніть з вихідного набору даних D , створіть новий набір S з усіма екземплярами класу меншості та одним випадковим екземпляром класу більшості.

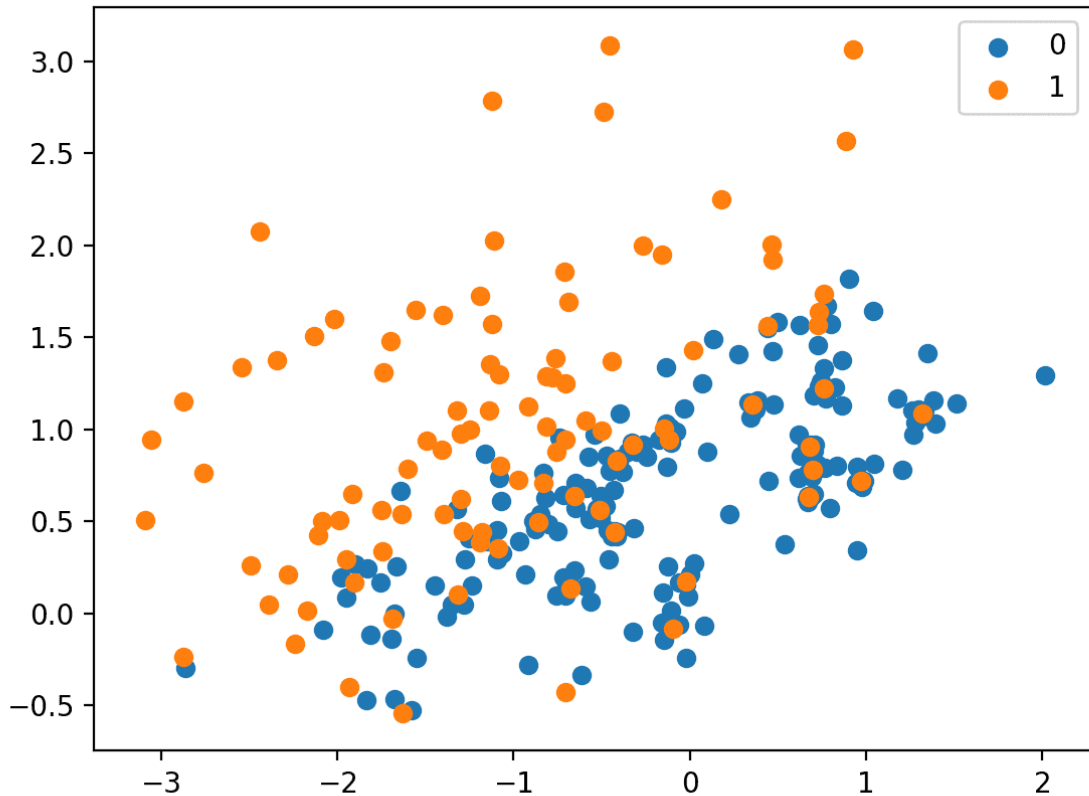


Рисунок 2.6 - Правило згорнутого найближчого сусіда

Використовуючи алгоритм KNN ($k=1$), спробуйте класифікувати екземпляри з D за допомогою S , додаючи будь-які неправильно класифіковані екземпляри до S . Нарешті, видаліть усі екземпляри класу більшості в S , які є частиною посилань Tomek, як показано на рис. 2.7. Цей метод забезпечує збалансоване представлення класів, покращуючи продуктивність класифікатора на незбалансованих наборах даних [28].

Таким чином, у цьому дослідженні використовуються математичні методи для збільшення та фільтрації даних, щоб отримати більш ефективний результат. Цей підхід має на меті підвищити точність моделі, одночасно забезпечуючи її застосовність до даних реального світу та тестових даних, уникаючи підводних каменів основних методів вибірки.

Математичні методи, як-от недостатня та надмірна вибірка, є вдосконаленими методами, які використовуються для збалансування наборів даних і пропонують більш тонкий підхід, ніж випадкове видалення або просте

дублювання даних.

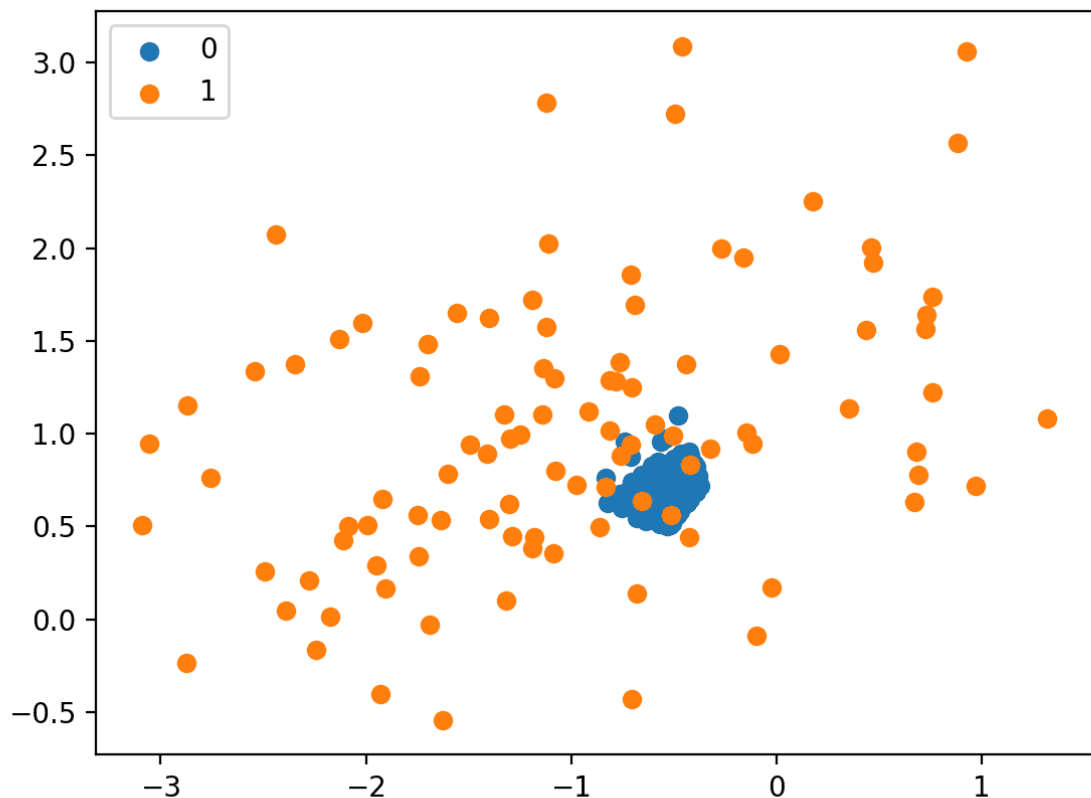


Рисунок 2.7 - Односторонній вибір (OSS)

Хоча ці методи створюють більш збалансований набір даних для навчання моделі, вони вносять певний рівень штучності, змінюючи природний розподіл даних. Тим не менш, їх використання часто необхідне для підвищення продуктивності класифікаторів, особливо при роботі з незбалансованими наборами даних, які становлять значну проблему при навчанні моделі.

Таблиця 2.1 надає порівняльний огляд переваг і недоліків різних методів вибірки.

Ці методи спільно спрямовані на покращення продуктивності класифікаторів на незбалансованих наборах даних. Вони роблять це або збільшуючи клас меншості (як у випадку SMOTE), або скорочуючи набір даних, щоб видалити менш інформативні екземпляри (як у Tomek та CNN).

Таблиця 2.1 – Аналіз методів вибірки даних

Назва методу	Переваги	Недоліки
SMOTE [29]	Зберігає цінну інформацію, не викидаючи дані. Надає моделі більше даних для навчання, зокрема щодо характеристик класу меншини.	Передбачає перенавчання на синтетичних, а не реальних даних. Потенційно збільшує час навчання моделі.
Tomik [30]	Видаляє дані про клас більшості, які дуже нагадують клас меншості, потенційно покращуючи здатність моделі розпізнавати клас меншості. Може скоротити загальний час навчання завдяки більш спрощеному набору даних.	Ризики видалення даних, які, хоч і подібні до класу меншості, можуть бути важливими для процесу навчання моделі.
CNN [31]	Виключає дані, які не сприяють значному розрізненню класів. Подібно до Tomek, він скорочує час навчання за рахунок ущільнення набору даних.	Подібно до Tomek, він може ненавмисно видалити дані, які можуть бути корисними для навчання моделі.

Цей процес допомагає створити ефективніше навчальне середовище для класифікатора, дозволяючи йому навчатися з набору даних, який є кращим представленням різноманітних сценаріїв, з якими він може зіткнутися в реальних програмах. Однак головна проблема полягає в тому, щоб знайти правильний баланс між збереженням цілісності вихідних даних і підвищенням здатності класифікатора навчатися на більш збалансованому наборі даних.

2.2 Нейронні мережі

Численні алгоритми навчання на основі нейронної мережі були розроблені для вирішення широкого кола завдань. Кожен тип архітектури нейронної мережі спеціалізується на конкретних видах завдань. Наприклад, згорткові нейронні мережі (CNN) дуже ефективні для завдань, пов'язаних із зображеннями, тоді як рекурентні нейронні мережі (RNN) розроблені для виконання завдань із послідовними даними, такими як розпізнавання мови та тексту, завдяки їхній здатності запам'ятовувати та обробляти послідовні інформації.

У сфері завдань класифікації, таких як виявлення шахрайських транзакцій, нейронні мережі прямого зв'язку (FNN) є найбільш часто використовуваним методом. FNN, одні з найстаріших і найпростіших типів нейронних мереж, залишаються потужними інструментами для різноманітних застосувань. Багат шаровий перцептрон (MLP), тип FNN, є прикладом цієї архітектури та зображений на рис. 2.8.

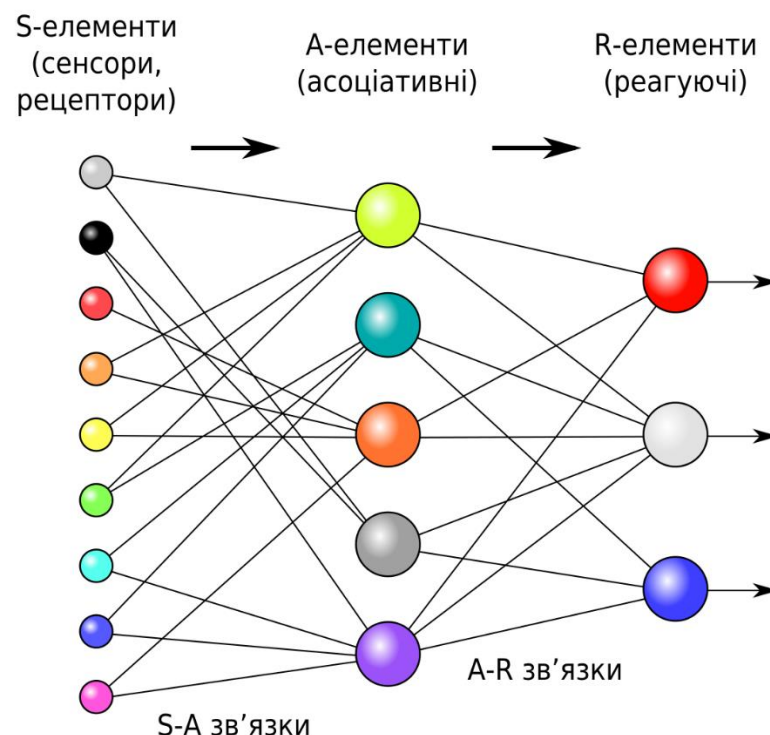


Рисунок 2.8 – Структура MLP

Багатошаровий перцептрон функціонує як апроксиматор, прагнучи точно відтворити оригінальну функцію під час навчання. Складність функції, яку необхідно апроксимувати, визначає розмір мережі, але великі мережі мають компроміси: вони потребують більшої обчислювальної потужності та схильні до переобладнання. Переобладнання відбувається, коли мережа по суті «запам'ятовує» навчальні дані, що призводить до низької продуктивності раніше невидимих даних [29].

Процес навчання для багатошарового перцептрона зазвичай складається з таких кроків:

- ініціалізація випадкових ваг і зміщень у шарах перцептрона;
- навчальні дані передаються через мережу, а вихід генерується шляхом застосування ваг до вхідних функцій;
- функція втрат обчислює помилку (втрату) виходу;
- градієнт обчислюється з використанням часткових похідних похибки щодо ваг. Для мінімізації функції втрат вносяться коригування ваги;
- цей процес повторюється, доки апроксиматор функції не досягне задовільної точності.

Кожен рівень перцептрона використовує функцію активації для інтеграції зважених вхідних даних і отримання виходу. Функція ReLU (випрямлена лінійна одиниця), визначена як $f(x)=\max(0,x)$, зазвичай використовується для прихованих шарів, хоча також використовуються інші функції [30].

Щоб покращити процес навчання та запобігти переобладнанню, реалізовано такі методи, як відсівання. Під час кожної епохи навчання відсівання випадковим чином блокує певні вагові коефіцієнти від оновлення на етапі зворотного поширення. Швидкість навчання, ще один важливий аспект тренувального процесу, визначає ступінь коригування ваги на цьому етапі. Такі стратегії є життєво важливими для вдосконалення процесу навчання, гарантуючи, що нейронні мережі, такі як MLP, можуть ефективно навчатися та узагальнювати навчальні дані без переобладнання.

2.3 Продуктивність моделі

Оцінка точності моделі є критичним кроком у розумінні її продуктивності, особливо в таких завданнях, як класифікація шахрайських транзакцій. Зазвичай точність використовується як метрика, що представляє відсоток правильно класифікованих екземплярів обох класів. Однак ця метрика недостатня при роботі з незбалансованими наборами даних, оскільки вона не враховує перекош розподілу. Щоб вирішити цю проблему, введено матрицю помилок, як показано на рис. 2.9, яка забезпечує більш детальне уявлення про продуктивність моделі.

Прогноз моделі			
	Так	Ні	
Реальні значення	Позитивний клас	Хибно-негативний	Так
	Хибно-позитивний	Негативний клас	Ні

Рисунок 2.9 – Продуктивність моделі

У матриці помилок TP (True Positive) і TN (True Negative) представляють правильно ідентифіковані екземпляри позитивних і негативних класів відповідно. FP (хибно-позитивний) і FN (хибно-негативний) вказують на неправильно класифіковані екземпляри [31]. Використовуючи ці значення, точність розраховується формулою (2.2):

$$A = \frac{TP+TN}{TP+TN+FP+FN} . \quad (2.2)$$

Одна лише точність може не повністю охопити продуктивність моделі, особливо в незбалансованих наборах даних.

Щоб отримати більш детальну оцінку, використовуються додаткові показники у формулі (2.3):

$$P = \frac{TP}{TP+FP} . \quad (2.3)$$

Формула (2.2) вказує на частку правильно ідентифікованих позитивних зразків серед усіх класифікованих як позитивні. Точність (2.3) допомагає оцінити, чи класифікатор надмірно позначає екземпляри як позитивні.

Формула (2.4) відображає відсоток правильно ідентифікованих позитивних зразків серед усіх позитивних зразків:

$$R = \frac{TP}{TP+FN} . \quad (2.4)$$

Цей показник оцінює чутливість класифікатора до позитивних зразків класу.

Оцінка F1 об'єднує точність і запам'ятовування в один показник за допомогою формули (2.5):

$$F1 = \frac{2*P*R}{P+R} . \quad (2.5)$$

Це особливо корисно, оскільки врівноважує обидва показники, на відміну від арифметичних чи середньоквадратичних середніх. Оцінка F1 чутлива як до точності, так і до запам'ятовування та може висвітлити проблеми з моделлю, якщо є забагато помилок типу I або типу II [32].

Іншим важливим показником є площа під кривою робочих характеристик приймача (AUC ROC). Цей показник відображає істинний позитивний коефіцієнт (TPR) проти коефіцієнта помилкових позитивних

результатів (FPR) для різних порогових значень. Крива ROC коливається від $[0,0]$ до $[1,1]$. Значення AUC у діапазоні від 0 до 1 вказує на дискримінаційну здатність моделі, причому 0,5 означає не кращу класифікацію, ніж випадкова, а значення, близькі до 1, вказують на відмінну ефективність класифікації [33].

Ці показники, особливо коли вони використовуються разом, забезпечують повне уявлення про ефективність моделі в класифікації даних, особливо в контексті незбалансованих наборів даних. Вони дозволяють точніше оцінити здатність моделі розрізняти класи, гарантуючи, що модель не тільки точна, але й надійна у своїх прогнозах. Це має вирішальне значення в таких програмах, як виявлення шахрайства, де вартість неправильної класифікації може бути високою, а розуміння обмежень і сильних сторін моделі є ключовим для її практичного застосування.

2.4 Висновки за розділом

У даному розділі розглядаються основні концепції та методології, які є ключовими для створення та розгортання алгоритму машинного навчання, зокрема в контексті виявлення шахрайських транзакцій. У ньому викладаються теоретичні основи, необхідні для використання машинного навчання для вирішення цієї проблеми. Обговорюються ключові стратегії попередньої обробки навчальних наборів даних, такі як кореляція Пірсона, підкреслюючи їх ефективність у виключенні менш значущих даних для прискорення навчання моделі.

Розглядається проблема дисбалансу класів у наборах даних, поширена проблема машинного навчання. Вона досліджує ефективність методів недостатньої та надмірної вибірки для створення більш збалансованого представлення даних. Ці методи допомагають штучно доповнювати набір даних синтетичними даними та видаляти надлишкові дані з основного класу, тим самим підвищуючи загальну продуктивність моделі.

Проаналізовано важливість вибору відповідних показників для оцінки

продуктивності класифікатора, особливо в сценаріях із незбалансованими даними. У ньому обговорюються різні показники, такі як точність, запам'ятовування, F1-оцінка та AUC ROC, з'ясовуючи, як кожен показник сприяє повному розумінню ефективності класифікатора. Ці показники мають вирішальне значення для того, щоб класифікатор не тільки точно ідентифікував шахрайські транзакції, але й підтримував збалансовану чутливість до обох класів у наборі даних.

Крім того, важливо визнати взаємодію між методами попередньої обробки даних і вибором моделей машинного навчання. Ефективність моделі у виявленні шахрайських операцій значною мірою залежить від того, наскільки добре підготовлені та представлені дані. Вибір показників не тільки інформує про поточну продуктивність моделі, але й спрямовує її на подальше вдосконалення та налаштування, що є критично важливим у такій динамічній та ефективній сфері, як виявлення шахрайства. Розуміння та усунення дисбалансу класів, вибір правильної архітектури моделі та застосування відповідних показників оцінки є основними кроками в розробці надійних, ефективних і надійних систем машинного навчання для реальних програм.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДУ ВИЯВЛЕННЯ ШАХРАЙСЬКИХ ДІЙ

3.1 Алгоритм розробки

На основі комплексного аналізу методів можна розробити алгоритм виявлення шахрайських дій за допомогою машинного навчання. Цей алгоритм включатиме обговорювані методології та техніки (рис.3.1).

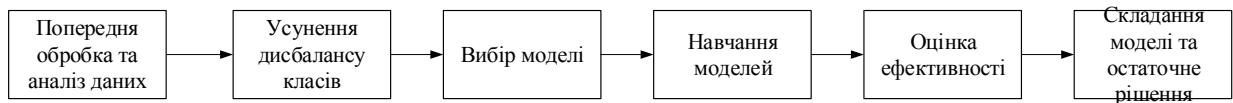


Рисунок 3.1 – Алгоритм виявлення шахрайських дій
за допомогою машинного навчання

Попередня обробка та аналіз даних складається з очищення та попередньої обробки даних. Це включає обробку відсутніх значень, кодування категоріальних змінних і нормалізацію чи стандартизацію числових функцій.

Для усунення дисбалансу класів застосовуються методи недостатньої та надмірної вибірки, щоб збалансувати набір даних. Такі методи, як SMOTE (метод синтетичної надлишкової вибірки меншості), можна використовувати для передискретизації класу меншості, тоді як такі методи, як посилення Tomek, можна використовувати для недобірки класу більшості.

Вибір моделі машинного навчання для завдання класифікації. Пропонується використовувати методи посилення градієнта (XGBoost, LightGBM, CatBoost) і нейронні мережі прямого зв'язку, зокрема багатошаровий перцептрон [34]. Для нейронних мереж слід визначити такі параметри, як кількість шарів, нейронів у кожному шарі, функції активації, показники відсіву та оптимізатор.

Навчання моделей проходить на попередньо обробленому та збалансованому наборі даних. Використовується перехресна перевірка, щоб перевірити продуктивність моделей.

Оцінити моделі, можна використовуючи метрики, які підходять для незбалансованих наборів даних, наприклад точність, прецизійність, запам'ятовування, показник F1 і AUC-ROC.

Проаналізуємо матрицю похибок, щоб зрозуміти продуктивність моделей з точки зору справжніх позитивних результатів, справжніх негативних результатів, помилкових позитивних результатів і помилкових негативних результатів.

Складання моделі та остаточне рішення виходить із поєднання прогнозів з різних моделей, використовуючи комплексні прийоми, наприклад голосування. Цей підхід може використовувати сильні сторони окремих моделей і підвищити загальну точність.

Цей алгоритм забезпечує структурований підхід для виявлення шахрайських транзакцій, використовуючи методи машинного навчання для аналізу шаблонів і аномалій у даних транзакцій. Він адаптований і може бути налаштований відповідно до конкретних вимог або в міру появи нових даних і тенденцій у сфері виявлення шахрайства.

3.2 Програмна реалізація роботи із початковим набором даних

Для аспекту аналізу даних цього проєкту використовувався Python разом із двома його широко використовуваними бібліотеками, DataFrames Library і ChartPlot Library. Бібліотека DataFrames забезпечує ефективну платформу для обробки даних і статистичних обчислень, а бібліотека ChartPlot, зокрема її підмодуль побудови графіків, полегшує візуалізацію даних для кращого розуміння.

Функції, які використовуються для аналізу даних:

- клас `DataFrame`: належить до бібліотеки DataFrames і використовується для зберігання табличних даних;
- метод `csv_reader`: читає файли CSV і вводить дані в змінну `DataFrame`;

- метод `count_values` у класі `DataFrame` підраховує випадки появи кожного унікального значення в стовпці даних, у результаті чого створюється `DataFrame` зі стовпцями «index» і «values»;
- метод `detect_nonnumeric` перетворює нечислові значення на `True` і числові значення на `False` у `DataFrame`;
- метод `unique_counter` підраховує унікальні значення для певного атрибута в `DataFrame`;
- метод `calculate_correlation` обчислює кореляцію між стовпцями даних, за винятком нечислових значень;
- метод `create_figure` створює полотно для візуалізації даних у бібліотеці `ChartPlot`;
- метод `create_bar_chart` створює гістограму для візуалізації частоти вказаних значень атрибутів;
- метод `create_histogram` створює гістограми для відображення розподілу значень;
- метод `color_matrix` показує таблиці даних як кольорові матриці в `seaborn`;
- метод `display_figure` показує створену фігуру з графіками з бібліотеки `ChartPlot`;
- функція `distribute_plot`: створює діаграму, що показує розподіл транзакцій;
- функція `identify_NaNs` підраховує нечислові значення в стовпцях функцій;
- `generate_histograms` створює гістограми для різних розподілів ознак;
- функція `visualize_correlations` створює теплову карту, яка візуалізує кореляційну матрицю між об'єктами з подібним розподілом нечислових значень;
- функція `reduce_features` обчислює кількість унікальних значень у корельованих групах функцій і зберігає одну репрезентативну функцію на

групу.

Набір даних, вибраний для навчання моделі, охоплює 430 760 записів, кожна з яких містить 204 функції.

На рис. 3.1 показано розподіл транзакцій. Існує 756 560 нешахрайських і 19 876 шахрайських транзакцій, що становить 95,3% і 3,8% набору даних відповідно, як показано на рис. 3.2.

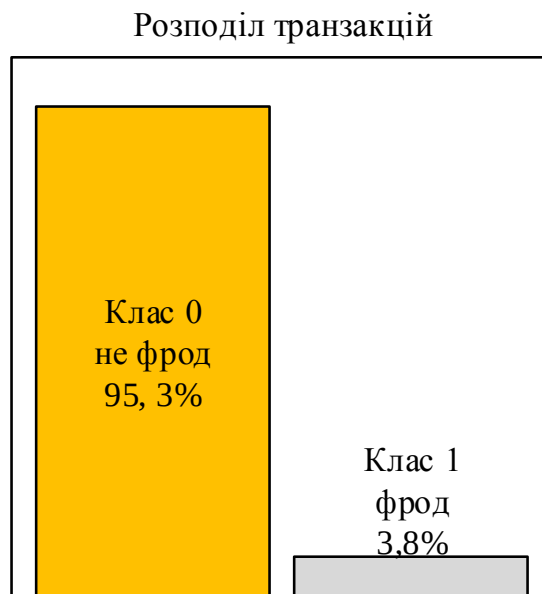


Рисунок 3.2 – Початкові дані

Кожна транзакція в наборі даних включає 204 атрибути, один з яких ('isFraud') вказує на клас транзакції. Деякі атрибути є категоричними, не підлягають прямому порівнянню, тоді як більшість є числовими, але анонімними для збереження конфіденційності. Багато атрибутів мають нечислові (NaN) значення. Тому аналіз цих характеристик для нечислових значень є вирішальним. Виявлення ознак із подібним розподілом нечислових значень може виявити значні кореляції. Згодом можна обчислити кореляції між такими ознаками, і в групах сильно корельованих ознак можна зберегти одну репрезентативну ознаку. Такий підхід оптимізує процес навчання за часом без шкоди для точності моделі.

У контексті аналізу даних транзакцій ці методи та інструменти

дозволяють ретельно досліджувати характеристики даних. Ефективне зменшення простору функцій без втрати важливої інформації може оптимізувати навчання моделі та покращити її здатність розрізняти шахрайські транзакції від нешахрайських. Цей баланс між ефективністю та точністю має вирішальне значення для розробки надійних і практичних моделей машинного навчання.

3.3 Застосування досліджуваних моделей

Для практичного застосування моделей була обрана мова програмування Python разом з онлайн-платформою Google Collab. Python є перевагою завдяки своїй розгалуженій бібліотечній екосистемі, яка підтримує обробку великомасштабних даних і ряд моделей машинного навчання, глибокого навчання та нейронних мереж. Його гнучкість у підтримці різних стилів програмування, включаючи об'єктно-орієнтоване програмування, полегшує розробку чіткого багаторазового коду, необхідного для дослідницьких цілей (табл. 3.1).

Таблиця 3.1 - Застосування досліджуваних моделей

Мова програмування	Python
Платформа розробки	Google Collab
Бібліотеки	GradientBoostingClassifier, LightGBMClassifier, CategoricalBoostingClassifier

Класи GradientBoostingClassifier, LightGBMClassifier і CategoricalBoostingClassifier були використані для точного налаштування моделей за допомогою методів посилення градієнта.

GradientBoostingClassifier, заснований на підході XGBoost, був розроблений за допомогою пакета xgboost. Цей пакет містить комплексні

інструменти для побудови та навчання моделей. Клас приймає кілька гіперпараметрів для конфігурації:

- `boost_rounds` – визначає кількість етапів підвищення для додавання нових дерев рішень;
- `deep_limit` – встановлює максимальну глибину для дерев в ансамблі;
- `leaf_limit` – визначає максимальну кількість листя на дереві;
- `grow_policy` – визначає стратегію розширення дерева, або в глибину, або на основі найбільшої зміни функції втрат;
- `learning_rate` – контролює швидкість навчання моделі, пом'якшуючи вплив кожного дерева для покращеного узагальнення моделі;
- `tree_construction_method` – вибирає метод для побудови дерева, вибираючи між ітерацією всіх вхідних даних при кожному розділенні або використанням попередньо оброблених частотних гістограм, подібних до методів у LightGBM, для підвищення швидкості навчання та збереження точності;
- `min_split_loss` – встановлює мінімальне зменшення втрат, необхідне для подальшого розбиття дерева;
- `data_fraction` – визначає, яка частка навчальних даних використовується для побудови кожного нового дерева, балансуючи між ризиками надмірного та недостатнього оснащення;
- `sample_selection_method` – визначає метод для вибору підвбірок із навчального набору даних, рівномірно або на основі градієнта даних, подібно до методів у CategoricalBoosting;
- `feature_fraction_bytree` – визначає частину функцій, які використовуються для побудови кожного дерева, щоб запобігти переобладнанню;
- `feature_fraction_bylevel` – відсоток ознак, які використовуються для побудови наступних рівнів дерева;
- `feature_fraction_bynode` – частка функцій, що використовуються для розбиття кожного вузла дерева;

- `undefined_data_handling` – вказує, як обробляти відсутні дані в навчальному наборі;

- `training_evaluation_metric` – вибирає метрики для оцінки якості навчання та визначення того, коли подальше навчання перестане давати покращення;

- `stop_training_rounds` – встановлює обмеження на кількість раундів без покращення результатів тренування, після якого тренування припиняється.

Для розробки та реалізації нейронної мережі були використані фреймворки TensorFlow та Keras. Ці бібліотеки оснащені широкими функціональними можливостями для побудови різних архітектур нейронних мереж, включаючи мережі прямого зв'язку. Спеціальний клас `NeuralNet` був створений для полегшення ініціалізації та налаштування нейронної мережі зі спеціальною структурою. Цей клас включає рівні та параметри за замовчуванням, а також різні методи керування нейронними мережами

Конструктор класу `NeuralNet` приймає кілька додаткових аргументів зі значеннями за замовчуванням:

- `layer_outputs_1`, `layer_outputs_2`, `layer_outputs_3`: вкажіть кількість нейронів у першому, другому та третьому щільних шарах;

- `hidden_activation` функція активації, що використовується в прихованих шарах;

- `final_activation` функція активації для вихідного рівня;

- `dropout_rate` частка випадково дезактивованих нейронів під час навчання, щоб запобігти переобладнанню;

- `learning_optimizer` визначає оптимізатор для коригування ваг у прихованих шарах;

- `performance_metrics` список показників для оцінки моделі під час і після навчання;

- `lost_function` функція, яка використовується для обчислення розбіжності між прогнозованим і фактичним результатами;

- `batch_dimension` розмір пакетів даних, оброблених у кожній ітерації

навчання;

- `training_epochs` загальна кількість епох для навчання моделі.

Це налаштування забезпечує гнучку конфігурацію та навчання нейронної мережі, адаптацію до різних типів проблем і наборів даних. Клас `NeuralNet` пропонує структурований підхід до визначення та керування моделями нейронних мереж, забезпечуючи легкість експериментів та оптимізацію.

3.4 Висновки за розділом

У третьому було досліджено застосування методологій виявлення транзакцій із запобігання шахрайству на основі стратегій, викладених у попередньому розділі. Для програмування була обрана мова `Python`, відома своїм широким застосуванням у сферах машинного навчання та науки про дані. Універсальність `Python` у різних парадигмах програмування дозволяє використовувати цілісний підхід до вирішення проблем. Він може похвалитися великою кількістю бібліотек, особливо корисних для машинного та глибокого навчання завдяки своїй простоті та ефективності.

Детально описано використання спеціальних бібліотек і методів для перевірки даних, обробки, а також для побудови, навчання та оцінки моделей. Моделі посилення градієнта були розроблені з використанням бібліотек `Xgboost`, `Lightgbm` і `Catboost`. Ці бібліотеки мають подібну структуру та прості функції, що сприяє єдиному процесу розробки моделі. Реалізація прямої нейронної мережі була досягнута за допомогою `Tensorflow` і `Keras`. Ці бібліотеки підтримують навчальні моделі на графічних процесорах (GPU), значно підвищуючи швидкість порівняно з навчанням на основі CPU.

4 ДОСЛІДЖЕННЯ РЕЗУЛЬТАТІВ РОБОТИ МЕТОДУ

4.1 Результати роботи нейронної мережі

Були проведені експерименти, щоб визначити оптимальну конфігурацію архітектури нейронної мережі. Виявлена найефективніша структура включає вхідний рівень із 256 нейронами, за яким йде перший прихований шар, що містить 128 нейронів, і другий прихований шар, що містить 64 нейрони. Вилучення було застосовано після вхідного шару, а пакетна нормалізація слідувала за кожним шаром.

Дизайн нейронної мережі був додатково досліджений за допомогою серії гіперпараметричних тестів. Ці тести були зосереджені на оцінці ключових змінних, таких як швидкість навчання та розмір партії. Результати цих оцінок детально описані в таблиці 4.1, яка пропонує всебічне уявлення про продуктивність моделі при різних налаштуваннях гіперпараметрів.

Таблиця 4.1 – Результати тестування нейронної мережі

Параметри	Швидкість навчання				
Розмір		0.03	0.02	0.01	0.005
	64	0.9178	0.9299	0.9222	0.9212
	128	0.9180	0.9201	0.9213	0.9453
	256	0.9184	0.9201	0.9229	0.9323

На додаток до цих технічних аспектів, значення архітектури моделі та гіперпараметрів неможливо переоцінити. Вибір нейронів у кожному шарі та застосування Dropout і BatchNormalization мають вирішальне значення для управління складністю моделі та запобігання переобладнанню. Цей баланс гарантує, що модель може добре узагальнювати нові, невідомі дані. Функція активації ReLU є особливо корисною для глибоких мереж через її здатність пом'якшувати проблему зникнення градієнта, таким чином допомагаючи в більш ефективному навчанні (рис. 4.1).

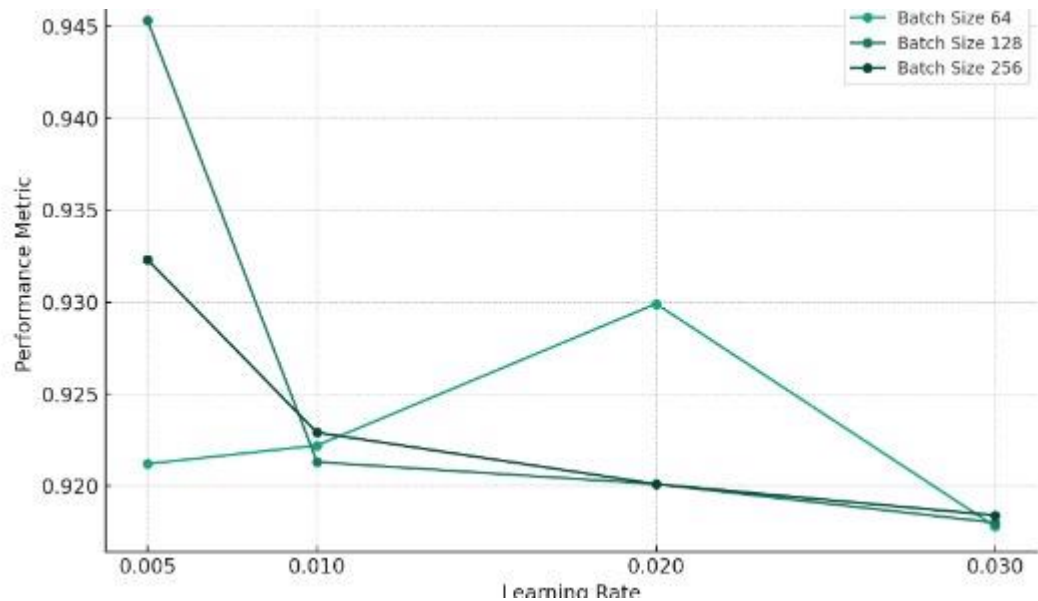


Рисунок 4.1 - Результати тестування нейронної мережі

Крім того, вибір оптимізатора Adam є ключовим. Adam відомий своєю ефективністю в обробці рідкісних градієнтів і адаптивністю в налаштуванні швидкості навчання, що робить його особливо придатним для складних наборів даних, які часто зустрічаються в сценаріях виявлення шахрайства. Нарешті, перевірка швидкості навчання та розміру пакету за допомогою тестування гіперпараметрів є важливою для оптимізації процесу навчання, гарантуючи, що модель не тільки навчається ефективно, але й робить це з економним часом. Ці комплексні тести та конфігурації завершуються створенням надійної та ефективної моделі нейронної мережі, створеної для ефективного виявлення шахрайських транзакцій.

4.2 Результати оптимізації моделі

Для підвищення точності прогнозів була реалізована стратегія, яка передбачає об'єднання різних моделей з оптимальними параметрами. Обґрунтування цього методу полягає в тому, що кожна модель має свої сильні та слабкі сторони у виявленні певних типів транзакцій. Однак цей метод не є надійним; якщо декілька моделей одночасно дають неточні висновки,

сукупний результат також буде помилковим. Для цієї інтеграції були обрані такі моделі, як XGBoos і LightGBM, які засновані на градієнтному спуску. Нейронна мережа також була оцінена, але через її відносно нижчу продуктивність вона була виключена з комплексу. У цьому випадку показники продуктивності об'єднаних моделей продемонстрували незначне покращення (рис. 4.2) порівняно з однією моделлю з найкращими показниками, як зазначено в таблиці 4.2.

Таблиця 4.2 – Моделі градієнтного бустінгу

Характеристика	XGBoos	LightGBM	Комбінація моделей
Влучність	0,9103	0,9154	0,9216
Повнота	0,8597	0,8623	0,8645
F1	0,8987	0,8991	0,9001

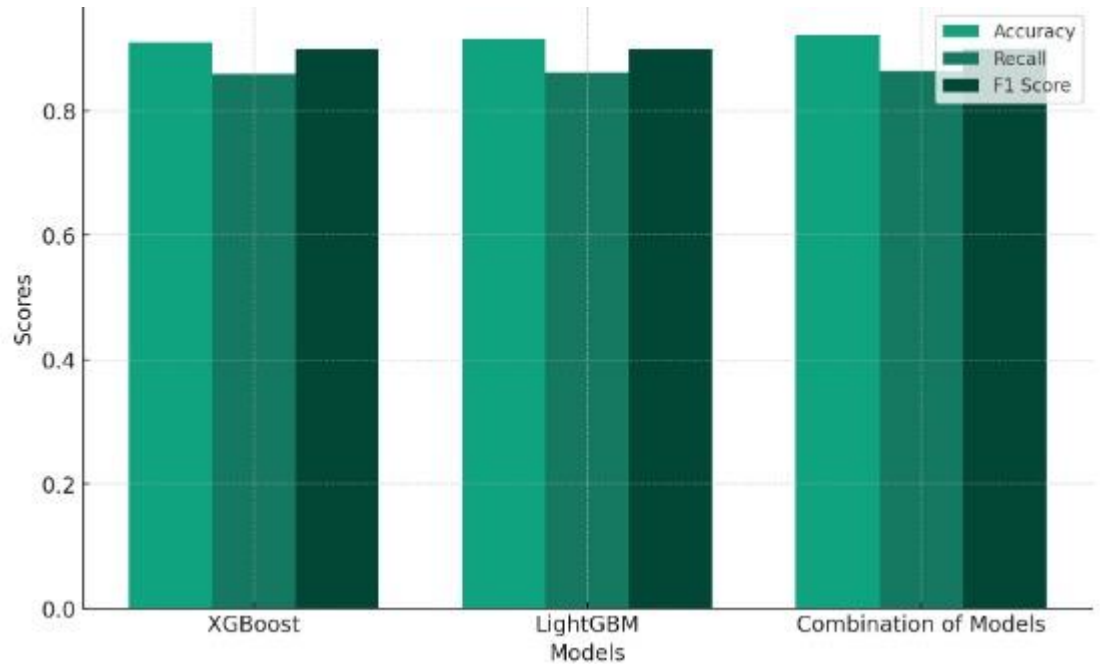


Рисунок 4.2 - Моделі градієнтного бустінгу

Стратегія, використана для досягнення покращеної ефективності прогнозування, передбачала синергетичну інтеграцію кількох моделей, кожену

з яких було обрано на основі їхньої кращої ефективності в певних областях. Цей метод використовує ідею про те, що хоча одна модель не може ефективно ідентифікувати конкретні транзакції, інша може. Однак важливо зазначити, що ця техніка не є надійною. Якщо більш ніж одна модель приходить до неправильного аналізу, сукупний результат відображає цю помилку. Комбінація включає такі моделі, як XGBoost і LightGBM, усі з яких використовують техніку градієнтного спуску. Нейронна мережа спочатку розглядалася, але зрештою не була включена через її порівняно низьку продуктивність. Сукупна ефективність цих моделей, як задокументовано в таблиці 4.2, була дещо вищою, ніж найкраща окрема модель.

4.3 Висновки за розділом

У даному розділі були проведені і оцінки моделей зосереджуючись на визначенні потенційних покращень. Основні оцінки для моделі посилення градієнта передбачали зміну таких параметрів, як швидкість навчання, кількість дерев рішень і розділів у цих деревах. Результати комбінації моделей – 0,9216.

Подальше тестування було проведено на моделі нейронної мережі, щоб визначити найбільш ефективну архітектуру для поставленого завдання. Це включало експериментування зі змінними, такими як швидкість навчання та розмір підвибірки, що зрештою призвело до найкращої моделі з оцінкою 0,9323. Ці тести мали вирішальне значення для тонкого налаштування нейронної мережі для оптимізації її продуктивності для конкретних вимог цього додатка.

У цьому розділі не лише висвітлюється ефективність індивідуального налаштування моделі, але й демонструється порівняння різних моделей для досягнення чудової продуктивності у таких складних завданнях, як виявлення шахрайства.

ВИСНОВКИ

У цьому дослідженні було розроблено підхід машинного навчання для виявлення шахрайських транзакцій на основі різних технологій і методологій у цій галузі. Він демонструє еволюцію методів машинного навчання та їх застосування у вирішенні різноманітних проблем. Ретельний аналіз проблем із виявленням шахрайських транзакцій зосереджений на таких питаннях, як дисбаланс даних, коли кількість нешахрайських транзакцій значно перевищує кількість шахрайських.

Щоб вирішити проблему дисбалансу даних, було досліджено методи вирівнювання представлення позитивних (шахрайських) і негативних (не шахрайських) екземплярів класу. Такі методи, як надмірна та недостатня вибірка, були проаналізовані на предмет їх ефективності у створенні збалансованого набору даних, що є вирішальним для точного навчання моделі. Було проведено комплексний аналіз набору навчальних даних. Це включало виявлення надлишкових даних і функцій, які були сильно корельовані, що могло перешкоджати швидкості навчання моделі.

Були досліджені методи посилення градієнта та нейронних мереж прямого поширення.

Комплексний підхід дослідження, від аналізу проблем із даними до впровадження та вдосконалення моделей машинного навчання, підкреслює ретельний процес розробки ефективних рішень для таких складних проблем, як виявлення шахрайства. Інтеграція різних методів машинного навчання не тільки демонструє універсальність цих методів, але й підкреслює важливість постійного розвитку та адаптації в галузі для ефективного вирішення проблем, що розвиваються.

Основні оцінки для моделі посилення градієнта передбачали зміну таких параметрів, як швидкість навчання, кількість дерев рішень і розділів у цих деревах. Результати комбінації моделей – 0,9216.

Подальше тестування було проведено на моделі нейронної мережі, щоб

визначити найбільш ефективну архітектуру для поставленого завдання. Це включало експериментування зі змінними, такими як швидкість навчання та розмір підвибірki, що зрештою призвело до найкращої моделі з оцінкою 0,9323. Ці тести мали вирішальне значення для тонкого налаштування нейронної мережі для оптимізації її продуктивності для конкретних вимог цього додатка.

В даній роботі не лише висвітлюється ефективність індивідуального налаштування моделі, але й демонструється порівняння різних моделей для досягнення чудової продуктивності у таких складних завданнях, як виявлення шахрайства.

Матеріали, методи та технічні засоби які були використані - математичні методи роботи із незбалансованими наборами даних, структурне та об'єктно-орієнтоване програмування, мови програмування Python, бібліотеки TensorFlow, Keras, програмні середовища Collab, персональний комп'ютер з процесором Intel Core 4 Duo під управлінням операційної системи Microsoft Windows 10.

Практичні результати роботи, це створена програмна реалізація для визначення шахрайських дій в транзакціях для малих об'ємів даних.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Geetha T. V. Classification Algorithms [Electronic resource]. / T. V. Geetha, S. Sendhilkumar // Machine Learning. – Boca Raton, 2023. – P. 127–151. – Access mode: <https://doi.org/10.1201/9781003290100-6>.
2. Li F. On the Adversarial Robustness of LASSO Based Feature Selection [Electronic resource]. / Fuwei Li, Lifeng Lai, Shuguang Cui // Machine Learning Algorithms. – Cham, 2022. – P. 39–61. – Access mode: https://doi.org/10.1007/978-3-031-16375-3_3.
3. Li F. Summary and Extensions [Electronic resource]. / Fuwei Li, Lifeng Lai, Shuguang Cui // Machine Learning Algorithms. – Cham, 2022. – P. 83–85. – Access mode: https://doi.org/10.1007/978-3-031-16375-3_5.
4. Mitra M. Algorithms and Machine Learning [Electronic resource]. / Manu Mitra // American Research Journal of Electronics and Communication Engineering. – 2019. – Vol. 1, no. 1. – Access mode: <https://doi.org/10.21694/2643-3486.19008>.
5. Ling Q. Machine learning algorithms review [Electronic resource]. / Qingyang Ling // Applied and Computational Engineering. – 2023. – Vol. 4, no. 1. – P. 91–98. – Access mode: <https://doi.org/10.54254/2755-2721/4/20230355>.
6. Ayyadevara V. K. Basics of machine learning [Electronic resource]. / V. Kishore Ayyadevara // Pro machine learning algorithms. – Berkeley, CA, 2018. – P. 1–15. – Access mode: https://doi.org/10.1007/978-1-4842-3564-5_1.
7. Ayyadevara V. K. Recurrent Neural Network [Electronic resource]. / V. Kishore Ayyadevara // Pro Machine Learning Algorithms. – Berkeley, CA, 2018. – P. 217–257. – Access mode: https://doi.org/10.1007/978-1-4842-3564-5_10.
8. Ayyadevara V. K. Clustering [Electronic resource]. / V. Kishore Ayyadevara // Pro machine learning algorithms. – Berkeley, CA, 2018. – P. 259–281. – Access mode: https://doi.org/10.1007/978-1-4842-3564-5_11.
9. Ayyadevara V. K. Principal Component Analysis [Electronic resource]. / V. Kishore Ayyadevara // Pro Machine Learning Algorithms. – Berkeley, CA, 2018.

– P. 283–297. – Access mode: https://doi.org/10.1007/978-1-4842-3564-5_12.

10. Ayyadevara V. K. Decision tree [Electronic resource]. / V. Kishore Ayyadevara // Pro machine learning algorithms. – Berkeley, CA, 2018. – P. 71–103. – Access mode: https://doi.org/10.1007/978-1-4842-3564-5_4.

11. Ayyadevara V. K. Random Forest [Electronic resource]. / V. Kishore Ayyadevara // Pro Machine Learning Algorithms. – Berkeley, CA, 2018. – P. 105–116. – Access mode: https://doi.org/10.1007/978-1-4842-3564-5_5.

12. Ayyadevara V. K. Gradient boosting machine [Electronic resource]. / V. Kishore Ayyadevara // Pro machine learning algorithms. – Berkeley, CA, 2018. – P. 117–134. – Access mode: https://doi.org/10.1007/978-1-4842-3564-5_6.

13. Ayyadevara V. K. Artificial neural network [Electronic resource]. / V. Kishore Ayyadevara // Pro machine learning algorithms. – Berkeley, CA, 2018. – P. 135–165. – Access mode: https://doi.org/10.1007/978-1-4842-3564-5_7.

14. Ayyadevara V. K. Word2vec [Electronic resource]. / V. Kishore Ayyadevara // Pro Machine Learning Algorithms. – Berkeley, CA, 2018. – P. 167–178. – Access mode: https://doi.org/10.1007/978-1-4842-3564-5_8.

15. Ayyadevara V. K. Pro Machine Learning Algorithms [Electronic resource]. / V. Kishore Ayyadevara. – Berkeley, CA : Apress, 2018. – Access mode: <https://doi.org/10.1007/978-1-4842-3564-5>.

16. Jo T. Simple Clustering Algorithms [Electronic resource]. / Taeho Jo // Machine Learning Foundations. – Cham, 2020. – P. 191–215. – Access mode: https://doi.org/10.1007/978-3-030-65900-4_9.

17. Theodoridis S. Sparsity-Aware Learning: Algorithms and Applications [Electronic resource]. / Sergios Theodoridis // Machine Learning. – [S. l.], 2020. – P. 473–529. – Access mode: <https://doi.org/10.1016/b978-0-12-818803-3.00020-9>.

18. Gavrilova M. L. Adaptive Algorithms for Intelligent Geometric Computing [Electronic resource]. / M. L. Gavrilova // Machine Learning. – [S. l.], 2012. – P. 97–104. – Access mode: <https://doi.org/10.4018/978-1-60960-818-7.ch109>.

19. Learning Algorithms [Electronic resource]. // Machine Learning for

Speaker Recognition. – [S. 1.], 2020. – P. 13–35. – Access mode: <https://doi.org/10.1017/9781108552332.003>.

20. Japkowicz N. Machine Learning and Statistics Overview [Electronic resource]. / Nathalie Japkowicz, Mohak Shah // Evaluating Learning Algorithms. – Cambridge. – P. 23–73. – Access mode: <https://doi.org/10.1017/cbo9780511921803.003>.

21. Alinnor U. Introduction to machine learning algorithms: 10 commonly used algorithms machine learning experts / Ugonna Alinnor. – [S. 1.] : Independently Published, 2022.

22. Drowsiness Alert Using Machine Learning Algorithms and Deep Learning Algorithms [Electronic resource]. / Prof. Malan Sale [et al.] // International Journal of Advanced Research in Science, Communication and Technology. – 2022. – P. 182–185. – Access mode: <https://doi.org/10.48175/ijarsct-3904>.

23. Turan S. C. Ensemble learning [Electronic resource]. / Selin Ceren turan // Journal of Science and Arts. – 2022. – Vol. 22, no. 2. – P. 459–470. – Access mode: <https://doi.org/10.46939/j.sci.arts-22.2-a18>.

24. Ling Q. Machine learning algorithms review [Electronic resource]. / Qingyang Ling // Applied and Computational Engineering. – 2023. – Vol. 4, no. 1. – P. 91–98. – Access mode: <https://doi.org/10.54254/2755-2721/4/20230355>.

25. K.M. U. Road Accident Perusal Using Machine Learning Algorithms [Electronic resource]. / Umamaheswari K.M. // International Journal of Psychosocial Rehabilitation. – 2020. – Vol. 24, no. 5. – P. 1676–1682. – Access mode: <https://doi.org/10.37200/ijpr/v24i5/pr201839>.

26. Nair D. P. S. Analyzing Titanic Disaster using Machine Learning Algorithms [Electronic resource]. / Dr Prabha Shreeraj Nair // International Journal of Trend in Scientific Research and Development. – 2017. – Volume-2, Issue-1. – P. 410–416. – Access mode: <https://doi.org/10.31142/ijtsrd7003>.

27. Meena M. Breakdown of Machine Learning Algorithms [Electronic resource]. / Munesh Meena, Ruchi Sehrawat // Recent Trends in Artificial Intelligence & it's Applications. – 2022. – Vol. 1, no. 3. – P. 25–29. – Access mode:

<https://doi.org/10.46610/rtaia.2022.v01i03.005>.

28. Pandey M. A. Machine Learning [Electronic resource]. / Mrs Arjoo Pandey // International Journal for Research in Applied Science and Engineering Technology. – 2023. – Vol. 11, no. 8. – P. 864–869. – Access mode: <https://doi.org/10.22214/ijraset.2023.55224>.

29. A M. V. Brain Tumor Classification Using Various Machine Learning Algorithms [Electronic resource]. / Ms Vidhya A, Dr Parameswari R, Ms Sathya S // International Journal of Research in Arts and Science. – 2019. – Vol. 5, Special Issue. – P. 258–270. – Access mode: <https://doi.org/10.9756/bp2019.1002/25>.

30. Mishra A. Supervised Machine Learning Classification Algorithms for Detection of Fracture Location in Dissimilar Friction Stir Welded Joints [Electronic resource]. / Akshansh Mishra, Apoorv Vats // Frattura ed Integrità Strutturale. – 2021. – Vol. 15, no. 58. – P. 242–253. – Access mode: <https://doi.org/10.3221/igf-esis.58.18>.

31. Coe J. Evaluating Impact of Race in Facial Recognition across Machine Learning and Deep Learning Algorithms [Electronic resource]. / James Coe, Mustafa Atay // Computers. – 2021. – Vol. 10, no. 9. – P. 113. – Access mode: <https://doi.org/10.3390/computers10090113>.

32. Sameer S. K. L. Improving the Efficiency by Novel Feature Extraction Technique Using Decision Tree Algorithm Comparing with SVM Classifier Algorithm for Predicting Heart Disease [Electronic resource]. / S. K. L. Sameer, P. Sriramya // Alinteri Journal of Agriculture Sciences. – 2021. – Vol. 36, no. 1. – P. 713–720. – Access mode: <https://doi.org/10.47059/alinteri/v36i1/ajas21100>.

33. Bisong E. Google Colaboratory [Electronic resource]. / Ekaba Bisong // Building Machine Learning and Deep Learning Models on Google Cloud Platform. – Berkeley, CA, 2019. – P. 59–64. – Access mode: https://doi.org/10.1007/978-1-4842-4470-8_7.

34. Scarpino M. TensorFlow / Matthew Scarpino. – Hoboken, New Jersey : John Wiley & Sons, Inc., 2018. – 336 p.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для виявлення шахрайських дій у банківській системі.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему "Дослідження та реалізація методу виявлення шахрайських дій у банківській системі", затверджене наказом Національного університету "Запорізька політехніка" № 437 від 14 листопада 2023 р.

A.2 Призначення розробки

Програмний продукт призначений для автоматизованого виявлення шахрайських транзакцій в банківській системі з використанням методів машинного навчання.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

При розробці програмного забезпечення необхідно забезпечити:

- завантаження та попередню обробку даних про транзакції;
- побудову та навчання моделей машинного навчання (градієнтний бустинг, нейронні мережі);
- оцінку та порівняння моделей за метриками якості;
- виявлення шахрайських транзакцій в тестових даних.

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс користувача передбачається як командний рядок.

A.3.3 Вимоги до надійності

Надійне виявлення шахрайських транзакцій з врахуванням особливостей даних.

A.3.4 Умови експлуатації

Для експлуатації потрібен персональний комп'ютер під управлінням ОС Windows, Linux або MacOS, Python 3.6+, бібліотеки sklearn, TensorFlow, Keras.

A.3.5 Вимоги до складу та параметрів технічний засобів

Персональний комп'ютер з процесором Intel Core i5 або аналогічним, оперативна пам'ять не менше 8Гб.

A.3.6 Вимоги до маркування і пакування

Програмне забезпечення може поширюватись у вигляді вихідного коду або виконуваних файлів. На пакуванні має бути назва "Метод виявлення шахрайських дій у банківській системі".

ДОДАТОК Б**Лістинги програмної реалізації методів**

```

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Reading data from a CSV file
data = pd.read_csv('test.csv') # csv_reader method

# Display the first few records of the DataFrame
print(data.head())

# Counting the occurrences of each unique value in a data column
value_count = data['your_column_name'].value_counts() #
count_values method
print(value_count)

# Detecting non-numeric values in the DataFrame
non_numeric = data.applymap(lambda x: not str(x).isnumeric()) #
detect_nonnumeric method
print(non_numeric)

# Counting unique values for a specific attribute
unique_count = data['your_column_name'].nunique() #
unique_counter method
print(f"Number of unique values in 'your_column_name':
{unique_count}")

# Calculating the correlation between data columns
correlation_matrix = data.corr() # calculate_correlation method
print(correlation_matrix)

# Creating a figure canvas for data visualization
plt.figure() # create_figure method

# Creating a bar chart for visualizing the frequency of attribute
values
data['your_column_name'].value_counts().plot(kind='bar') #

```

```

create_bar_chart method
plt.title("Bar Chart of your_column_name")
plt.show()

# Creating a histogram for displaying the distribution of values
data['your_numeric_column'].plot(kind='hist', bins=30) #
create_histogram method
plt.title("Histogram of your_numeric_column")
plt.show()

# Showing data tables as color matrices
sns.heatmap(correlation_matrix, annot=True) # color_matrix method
plt.title("Correlation Matrix")
plt.show()

# Function to create a distribution plot
def distribute_plot(data, column):
    data[column].plot(kind='hist', bins=30)
    plt.title(f"Distribution of {column}")
    plt.show()

# Function to count non-numeric values in columns
def identify_NaNs(data):
    return data.applymap(lambda x: not str(x).isnumeric()).sum()

# Function to generate histograms for feature distributions
def generate_histograms(data, columns):
    for column in columns:
        data[column].plot(kind='hist', bins=30)
        plt.title(f"Histogram of {column}")
        plt.show()

# Function to visualize correlations
def visualize_correlations(data):
    corr_matrix = data.corr()
    sns.heatmap(corr_matrix, annot=True)

```

```

plt.title("Correlation Matrix Heatmap")
plt.show()

# Function to reduce features based on unique values in correlated
groups
def reduce_features(data):
    # Implementation depends on specific criteria for reduction
    pass

# Example usage of the custom functions
distribute_plot(data, 'your_column_name')
nan_counts = identify_NaNs(data)
print(nan_counts)
generate_histograms(data, ['your_numeric_column1',
'your_numeric_column2'])
visualize_correlations(data)

# Initialize the GradientBoostingClassifier (XGBoost)
gb_classifier = xgb.XGBClassifier(
    n_estimators=100,      # boost_rounds
    max_depth=6,          # depth_limit
    max_leaves=None,      # leaf_limit
    grow_policy='depthwise', # grow_policy
    learning_rate=0.3,    # learning_rate
    tree_method='auto',   # tree_construction_method
    gamma=0,              # min_split_loss
    subsample=1,          # data_fraction
    sampling_method='uniform', # sample_selection_method
    colsample_bytree=1,    # feature_fraction_bytree
    colsample_bylevel=1,   # feature_fraction_bylevel
    colsample_bynode=1,    # feature_fraction_bynode
    missing=None,         # undefined_data_handling
    eval_metric='logloss', # training_evaluation_metric
    early_stopping_rounds=10 # stop_training_rounds
)

```

```

# Training the model
gb_classifier.fit(X_train,          y_train,          eval_set=[(X_train,
y_train)])
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam

class NeuralNet:
    def __init__(self, layer_outputs_1=64, layer_outputs_2=64,
layer_outputs_3=64,
                    hidden_activation='relu',
final_activation='sigmoid',
                    dropout_rate=0.5, learning_optimizer='adam',
                    performance_metrics=['accuracy'],
loss_function='binary_crossentropy',
                    batch_dimension=32, training_epochs=10):
        self.model = Sequential()
        self.model.add(Dense(layer_outputs_1,
activation=hidden_activation))
        self.model.add(Dropout(dropout_rate))
        self.model.add(Dense(layer_outputs_2,
activation=hidden_activation))
        self.model.add(Dropout(dropout_rate))
        self.model.add(Dense(layer_outputs_3,
activation=hidden_activation))
        self.model.add(Dense(1, activation=final_activation))

        optimizer = Adam(learning_rate=0.001) if
learning_optimizer == 'adam' else learning_optimizer
        self.model.compile(optimizer=optimizer,
loss=loss_function, metrics=performance_metrics)
        self.batch_dimension = batch_dimension
        self.training_epochs = training_epochs

```

```
def train(self, X_train, y_train):
    self.model.fit(X_train, y_train,
batch_size=self.batch_dimension, epochs=self.training_epochs)

def evaluate(self, X_test, y_test):
    return self.model.evaluate(X_test, y_test)

# Example usage
neural_net = NeuralNet()
X_train, y_train = ... # Load or generate your training data
X_test, y_test = ...   # Load or generate your test data

# Train the model
neural_net.train(X_train, y_train)

# Evaluate the model
evaluation_results = neural_net.evaluate(X_test, y_test)
print("Evaluation Results:", evaluation_results)
```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ МЕТОДІВ ВИЯВЛЕННЯ ШАХРАЙСЬКИХ ДІЙ В БАНКІВСЬКІЙ СИСТЕМІ

Виконав: студент гр. КНТ-122м Віталій ЄВТУШЕНКО
Керівник: доцент Тетяна ГОЛУБ

Рисунок В.1 – Слайд 1

Мета і задачі проєкту

Метою проєкту є дослідження та розробка методу виявлення шахрайських дій у банківській системі за допомогою машинного навчання.

Завдання, які треба вирішити:

- ▶ проаналізувати стратегії виявлення шахрайства та порівняти підходи машинного навчання;
- ▶ дослідити методи обробки даних, зокрема боротьбу з дисбалансом класів;
- ▶ розробити програмну реалізацію методу на основі градієнтного бустингу та нейронних мереж;
- ▶ протестувати та оптимізувати моделі, порівняно їх ефективності.

Проблема шахрайства залишається серйозною загрозою для банківської системи і є досить актуальною для дослідження. Машинне навчання пропонує ефективний підхід для автоматизації виявлення шахрайства. Розроблений метод може бути застосований банками для захисту від збитків та підтримки довіри клієнтів

Рисунок В.2 – Слайд 2

Аналіз машинного навчання для виявлення шахрайства

Параметри	Опис
Переваги	
Адаптивність	Моделі машинного навчання здатні адаптуватися до мінливих моделей шахрайства. Вони можуть вивчати нові дані та виявляти складні схеми шахрайства, що розвиваються, які системи на основі правил можуть пропустити.
Масштабованість	Системи машинного навчання можуть ефективно обробляти великі обсяги даних. Вони можуть обробляти величезні набори даних для виявлення тонких закономірностей і аномалій, що має вирішальне значення в банківському середовищі з мільйонами транзакцій.
Автоматизація	Після навчання моделі машинного навчання можуть автоматизувати процес виявлення шахрайства, зменшуючи потребу в постійному ручному втручанні.
Недоліки	
Складність	Моделі машинного навчання часто розглядаються як «чорні ящики». Їхні процеси прийняття рішень можуть бути складними для тлумачення, що може спричинити труднощі з поясненням обґрунтування рішень щодо виявлення шахрайства, особливо щодо дотримання нормативних вимог
Залежність від даних	Ефективність моделей машинного навчання значною мірою залежить від якості та репрезентативності навчальних даних. Упереджені або недостатні дані можуть призвести до упереджених або неефективних моделей.
Ресурсовитратність	Розробка, навчання та підтримка моделей машинного навчання вимагає значних обчислювальних ресурсів і досвіду. Інституції повинні інвестувати в науковців з обробки даних, обчислювальну інфраструктуру та постійний моніторинг і оновлення моделей.

Рисунок В.3 – Слайд 3

Аналіз експертних систем для виявлення шахрайства

Параметри	Опис
Переваги	
Прозорість і можливість інтерпретації:	Експертні системи засновані на правилах і чітко визначені, що робить їхні процеси прийняття рішень прозорими та такими, що їх можна інтерпретувати. Це має вирішальне значення для відповідності, оскільки регулятори часто вимагають від установ пояснення їхніх механізмів виявлення шахрайства.
Налаштування	Експерти з безпеки можуть адаптувати експертні системи відповідно до конкретних потреб і вимог своєї банківської установи. Вони можуть використовувати специфічні для домену знання та точно налаштувати правила для ефективного націлювання на відомі схеми шахрайства.
Швидке впровадження	Експертні системи можна розгорнути відносно швидко, оскільки вони покладаються на заздалегідь визначені правила. Це робить їх привабливими варіантами для установ, яким потрібні рішення для негайного запобігання шахрайству.
Недоліки	
Відсутність адаптивності	Експертним системам може бути важко адаптуватися до нових тактик шахрайства. Вони покладаються на заздалегідь визначені правила, які можуть застаріти, оскільки шахраї постійно розробляють нові стратегії. Системи можуть не виявити нові загрози.
Помилкові спрацювання	Надто суворі набори правил можуть призвести до збільшення кількості хибних спрацювань, що створює незручності для законних клієнтів і може призвести до операційної неефективності.

Рисунок В.4 – Слайд 4

Методи виявлення аномалій

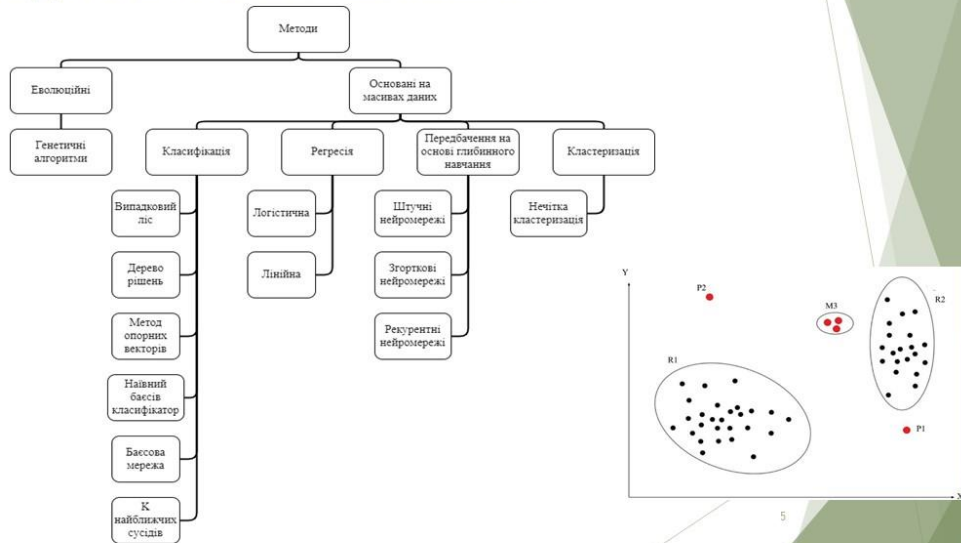


Рисунок В.5 – Слайд 5

Дослідження методів машинного навчання для знаходження шахрайських дій

Назва методу	Переваги	Недоліки
Дерева рішень	Прозорість і можливість інтерпретації, зручний для користувача, гнучкість з різними типами даних	Схильність до переобладнання та може погано адаптуватися до нових тактик шахрайства
Випадковий ліс	Покращена продуктивність у порівнянні з окремими деревами рішень, адаптованість до змінних даних, складна модель з обмеженою інтерпретацією	Недоліки: у деяких випадках все ще може перевищувати, відмінна масштабованість і надійність
Логістична регресія	Підходить для задач бінарної класифікації, інтерпретується як ймовірності, ефективний в обробці вхідних даних.	Обмежені можливості обробки складних нелінійних зв'язків у даних.
Генетичні алгоритми	Може оптимізувати складні моделі, підходить для вибору складних функцій	Сильно залежить від параметрів, складний у налаштуванні та реалізації.
Підтримка векторних машин (SVM)	Лінійний класифікатор здатний обробляти дані великої розмірності, хороше узагальнення, ефективний для великих наборів даних, потребує налаштування максимального розділення класів.	Може потребувати інтенсивних обчислень, особливо для розподілу параметрів, менш піддається інтерпретації.
Нейронні мережі	Надзвичайно гнучкий і потужний, здатний обробляти різні типи даних, варіанти глибокого навчання можуть виявляти складні шаблони, добре вилучати ознаки, висока швидкість обробки	Схильність до переобладнання (особливо для глибоких архітектур), складна в налаштуванні, може вимагати значних обчислювальних ресурсів, їх можна розглядати як «чорні ящики» з обмеженою інтерпретацією

Рисунок В.6 – Слайд 6

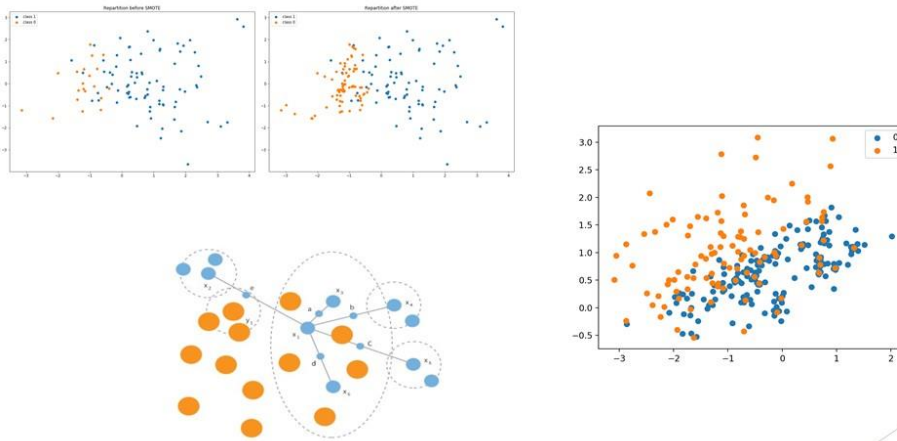
Аналіз методів вибірки даних

Назва методу	Переваги	Недоліки
SMOTE	Зберігає цінну інформацію, не викидаючи дані. Надає моделі більше даних для навчання, зокрема щодо характеристик класу меншини.	Передбачає перенавчання на синтетичних, а не реальних даних. Потенційно збільшує час навчання моделі.
Tomik	Видаляє дані про клас більшості, які дуже нагадують клас меншості, потенційно покращуючи здатність моделі розпізнавати клас меншості. Може скоротити загальний час навчання завдяки більш спрощеному набору даних.	Ризики видалення даних, які, хоч і подібні до класу меншості, можуть бути важливими для процесу навчання моделі.
CNN	Виключає дані, які не сприяють значному розрізненню класів. Подібно до Tomek, він скорочує час навчання за рахунок ущільнення набору даних.	Подібно до Tomek, він може ненавмисно видалити дані, які можуть бути корисними для навчання моделі.

7

Рисунок В.7 – Слайд 7

Аналіз методів вибірки даних



8

Рисунок В.8 – Слайд 8

Продуктивність моделі

Прогноз моделі		Реальні значення
Так	Ні	
Позитивний клас	Хибно-негативний	Так
Хибно-позитивний	Негативний клас	Ні

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

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 * precision * recall}{precision + recall}$$

Рисунок В.9 – Слайд 9

Алгоритм розробки

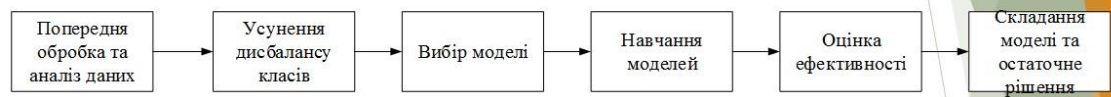


Рисунок В.10 – Слайд 10

Початкові дані



Рисунок В.11 – Слайд 11

Застосування досліджуваних моделей

Мова програмування	Python
Платформа розробки	Google Collab
Бібліотеки	GradientBoostingClassifier, LightGBMClassifier, CategoricalBoostingClassifier

Рисунок В.12 – Слайд 12

Результати тестування нейронної мережі

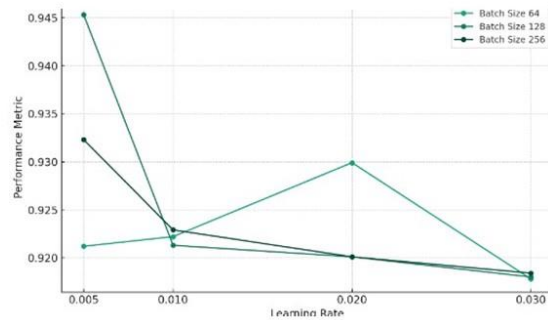


Рисунок В.13 – Слайд 13

Результати оптимізації моделі

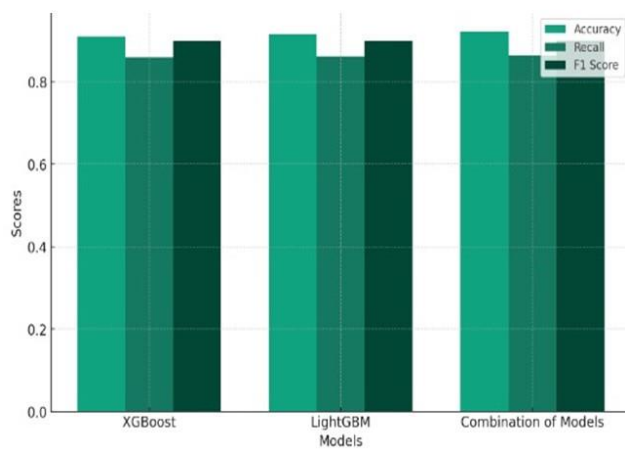


Рисунок В.14 – Слайд 14

Висновки

- Комплексний підхід дослідження, від аналізу проблем із даними до впровадження та вдосконалення моделей машинного навчання, підкреслює ретельний процес розробки ефективних рішень для таких складних проблем, як виявлення шахрайства. Інтеграція різних методів машинного навчання не тільки демонструє універсальність цих методів, але й підкреслює важливість постійного розвитку та адаптації в галузі для ефективного вирішення проблем.
- Основні оцінки для моделі посилення градієнта передбачали зміну таких параметрів, як швидкість навчання, кількість дерев рішень і розділів у цих деревах. Результати комбінації моделей - 0,9216. Подальше тестування було проведено на моделі нейронної мережі, щоб визначити найбільш ефективну архітектуру для поставленого завдання. Це включало експериментування зі змінними, такими як швидкість навчання та розмір підвибірки, що зрештою призвело до найкращої моделі з оцінкою 0,9184.

Рисунок В.15 – Слайд 15