

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Комп'ютерних наук і технологій

(повне найменування факультету)

Системний аналіз та обчислювальна математика

(повне найменування кафедри)

Пояснювальна записка

до дипломної роботи

(ступінь вищої освіти)

на тему Виявлення шахрайства в банківських операціях

(назва теми)

Виконав(ла): студент(ка) 2 курсу, групи КНТ-812м

Спеціальності 124 – Системний аналіз

(код і найменування спеціальності)

Освітня програма (спеціалізація)

«Інтелектуальні технології та прийняття

рішень в складних системах»

ЧЕРКАС О.В.

(ПРІЗВИЩЕ та ініціали)

Керівник БАХРУШИН Володимир

(ПРІЗВИЩЕ та ініціали)

Рецензент _____

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
Факультет Комп'ютерних наук і технологій
Кафедра Системний аналіз та обчислювальна математика
Ступінь вищої освіти магістр
Спеціальність Системний аналіз
(код і найменування)
Освітня програма (спеціалізація) «Інтелектуальні технології та прийняття
(назва освітньої програми (спеціалізації)
рішень в складних системах»

ЗАТВЕРДЖУЮ

В. о. завідувача кафедри САОМ
ТЕРЕЩЕНКО Е.В.

« 29 » січня 20 24 року

З А В Д А Н Н Я

Розділ 1 НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ЧЕРКАС Олександр Володимирови

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Виявлення шахрайства в банківських операціях

керівник проєкту (роботи) д.ф.-м.н., проф. БАХРУШИН Володимир Євгенович
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від « 06 » грудня 20 23 № 492

2. Строк подання студентом проєкту (роботи) « 30 » січня 20 23 року

3. Вихідні дані до проєкту (роботи) літературні джерела за темою дослідження:

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Огляд літератури з методів виявлення шахрайства з банківськими операціями. 2. Генерація даних для подальшого використання в навчанні та тестуванні нейронної мережі. 3. Розробка нейронної мережі для виявлення Шахрайства з кредитними картками.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) _____

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1	БАХРУШИН В.Є., д.ф.-м.н., проф.		
2	БАХРУШИН В.Є., д.ф.-м.н., проф.		
Нормоконтроль	ШИРОКОРАД Д.В., к.ф.-м.н., доц.		

7. Дата видачі завдання « 08 » вересня 20 23 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Вибір і затвердження теми дипломної роботи	08.09.2023 – 13.09.2023	
2	Формулювання мети та завдання роботи	14.09.2023 – 22.09.2023	
3	Розробка плану та структури роботи	23.09.2023 – 06.10.2023	
4	Підбір та вивчення літературних джерел	07.10.2023 - 29.10.2023	
5	Розробка моделі виявлення шахрайства	30.10.2023 – 01.12.2023	
6	Оформлення пояснювальної записки	02.12.2023 - 13.12.2023	
7	Попередній захист дипломної роботи та отримання рецензій	14.12.2023 – 29.01.2024	
8	Захист дипломної роботи	30.01.2024	

Студент(ка)

(підпис)

Олександр ЧЕРКАС
(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Володимир БАХРУШИН
(Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

ПЗ: 54 с., 16 рис., 1 табл., 21 джерело.

Об'єкт дослідження: карткові банківські транзакції.

Мета дослідження: розробка методу виявлення шахрайських операцій з банківськими картками.

Предмет дослідження: нейронна мережа для виявлення шахрайств з картковими транзакціями.

Методи дослідження: аналіз джерел, досліджень і публікацій, порівняльний аналіз, порівняльна характеристика.

Збільшення шахрайства з кредитними картками є актуальною проблемою, яка стала більш серйозною в епоху цифрових технологій та електронних фінансів. Шахраї постійно вдосконалюють свої методи та тактики, щоб отримати незаконний доступ до інформації про кредитні картки та використовувати їх для незаконних операцій. Робота присвячена розробці нейромережевої моделі для виявлення шахрайства. Така модель може не тільки ефективно виявляти шахрайські транзакції, але й постійно донавчатися на актуальних даних, що дозволяє адаптуватися до нових патернів поведінки шахраїв.

НЕЙРОННА МЕРЕЖА, ШАХРАЙСЬКІ ТРАНЗАКЦІЇ, ГЕНЕРАЦІЯ
ДАТАСЕТУ, НЕЗБАЛАНСОВАНІ КЛАСИ, НАВЧАННЯ З ВЧИТЕЛЕМ

ЗМІСТ

ЗАВДАННЯ.....	2
РЕФЕРАТ.....	3
ВСТУП.....	7
РОЗДІЛ 1 ТИПИ ТА МЕТОДИ ВИЯВЛЕННЯ ШАХРАЙСТВА В ОПЕРАЦІЯХ З БАНКІВСЬКИМИ КАРТКАМИ.....	8
1.1 Загроза фроду в банківських операціях та сучасні виклики.....	8
1.2 Види шахрайства з банківськими картками	9
1.3 Методи виявлення шахрайства в операціях з банківськими картками	10
1.3.1 Традиційні методи виявлення шахрайства	10
1.3.2 Методи машинного навчання для виявлення шахрайства	13
1.4 Основні виклики при створенні моделі для виявлення шахрайства	20
РОЗДІЛ 2 СИСТЕМА ВИЯВЛЕННЯ ШАХРАЙСТВА З КРЕДИТМИ КАРТКАМИ	22
2.1 Генерація даних транзакцій	22
2.1.1 Постановка задачі.....	22
2.1.2 Генерація необроблених даних.....	22
2.1.3 Генерація профілів клієнтів	23
2.1.4 Генерація профілів терміналів.....	25
2.1.5 Прив'язка профілів клієнтів до терміналів.....	26
2.1.6 Генерація транзакцій.....	27
2.1.7 Масштабування до більшого набору даних.....	28
2.1.8 Генерація сценаріїв шахрайства	29
2.2 Генерація додаткових ознак з початкового набору даних	32
2.2.1 Перетворення дати та часу.....	33
2.2.2 Перетворення ідентифікатора клієнта.....	34
2.2.3 Перетворення ідентифікатора терміналу	35
2.3 Створення моделі для виявлення шахрайства	37
2.3.1 Нейронна мережа прямого розповсюдження.....	37

2.3.2 Стратегія розподілу на тренувальний та тестовий набір	39
2.3.3 Завантаження даних	40
2.3.4 Загальний пайплайн бібліотеки Keras для побудови нейронних мереж	41
2.3.5 Основні налаштування моделі	42
2.3.6 Результати роботи моделі та напрямки її подальшого покращення.....	48
ВИСНОВКИ	50
ПЕРЕЛІК ПОСИЛАНЬ	52

ВСТУП

В сучасному світі, де використання електронних платіжних систем стає все більш популярним, проблема шахрайства в банківських транзакціях набуває найважливішого значення для забезпечення безпеки фінансових операцій. З метою виявлення та запобігання шахрайським діям у фінансовій сфері виникає потреба в розвитку ефективних технологій детектування фроду.

Одним із передових підходів в цьому напрямку є використання нейронних мереж. Нейронні мережі, засновані на принципах штучних нейронних мереж, можуть аналізувати величезний обсяг даних та виявляти в них відмінності та аномалії, які можуть свідчити про можливий фродовий вплив. Ці алгоритми машинного навчання здатні навчатися на основі історичних та реальних даних, що дозволяє їм постійно покращувати свою ефективність та адаптуватися до нових форм шахрайства.

В даному контексті робота з детектування фроду в банківських транзакціях за допомогою нейронних мереж не лише підвищує рівень безпеки електронних фінансових операцій, але й робить цей процес більш точним та ефективним. Застосування передових технологій у цій сфері сприяє збереженню довіри клієнтів до банківської системи, а також допомагає у побудові стійкого фінансового середовища.

РОЗДІЛ 1 ТИПИ ТА МЕТОДИ ВИЯВЛЕННЯ ШАХРАЙСТВА В ОПЕРАЦІЯХ З БАНКІВСЬКИМИ КАРТКАМИ

1.1 Загроза фроду в банківських операціях та сучасні виклики

Банківська система є не лише мотором економіки, але й об'єктом зростаючого інтересу для шахраїв та злочинців, які постійно вдосконалюють свої методи для здійснення фінансового шахрайства. Сучасний банкінг, що віддзеркалюється у безлічі онлайн-транзакцій та інноваційних фінансових послуг, стає сценою для розмаїтих атак та шахрайських схем. В цьому контексті виявлення та запобігання фроду набуває стратегічної важливості для фінансових установ.

Сутність боротьби з фродом в сучасному банкінгу полягає в тому, щоб вчасно розпізнавати та реагувати на неправомірні фінансові операції, забезпечуючи високий рівень безпеки та довіри серед клієнтів. Ідентифікація нових та вдосконалених стратегій шахрайства, а також впровадження передових технологій виявлення є критичним завданням для фінансових інститутів.

Робота присвячена ретельному огляду та аналізу сучасних методів виявлення шахрайства в банківських операціях. Розглянемо традиційні підходи, такі як встановлення правил та порогових значень, а також розкриємо сучасні технології, включаючи методи машинного навчання та використання нейронних мереж. Дослідження також акцентуватиме увагу на викликах, які стоять перед банківськими установами, та надійних стратегіях для забезпечення високого рівня безпеки та захисту від фроду. Практична частина роботи демонструє можливість застосування нейронних мереж для виявлення шахрайства в банківських транзакціях на основі самостійно згенерованого набору даних.

Мета цього дослідження - не лише визначити ключові аспекти виявлення шахрайства, але й виокремити оптимальні технологічні рішення та практики для забезпечення ефективності та надійності фінансових операцій у сучасному банківському середовищі.

1.2 Види шахрайства з банківськими картками

Існує багато вразливостей, якими можуть скористатися зловмисники для здійснення шахрайства з кредитними картками.

1. Втрата або крадіжка картки.

Коли кредитну картку втрачено або викрадено, шахраї можуть використовувати її для здійснення несанкціонованих покупок. Вони можуть використовувати картку в фізичних магазинах, ресторанах або інших закладах до того, як власник картки виявить її зникнення та повідомить про це.

2. Підроблені картки.

Шахраї створюють підроблені кредитні картки, дублюючи інформацію з легальних карток. Це може бути зроблено за допомогою скімінгових пристроїв або шляхом отримання інформації про власника картки з інших джерел. Потім підроблені картки використовуються для особистих транзакцій, часто в роздрібних магазинах або банкоматах.

3. Видавання себе за іншу картку.

При шахрайстві з використанням підробленої картки шахрай видає себе за законного власника картки, пред'являючи фізичну картку з інформацією про власника. Це може статися, коли злодій викрадає фізичну картку, або він може поєднати викрадену інформацію з підробленим документом, що посвідчує особу, з підробленою карткою.

4. Зміна картки.

Шахраї підробляють законні кредитні картки, змінюючи інформацію на картці, наприклад, ім'я власника, термін дії або код безпеки. Це дозволяє їм використовувати змінену картку для шахрайських операцій, не викликаючи підозр.

5. Крадіжка пошти.

Шахраї можуть обкрадати поштові скриньки або перехоплювати пошту, щоб отримати кредитні картки, які надсилаються поштою.

6. Скімінг карток:

Скімінг карток може відбуватися і в режимі офлайн, коли шахраї встановлюють пристрої, що називаються скімерами, на фізичні зчитувачі карток, такі як банкомати або термінали в точках продажу (POS). Ці скімери перехоплюють інформацію про власника картки, коли він проводить по ній пальцем або вставляє картку. Викрадена інформація потім використовується для створення підроблених карток або здійснення шахрайських покупок.

7. Картковий трепінг:

Захоплення картки передбачає використання фізичних пристроїв або механізмів для утримання картки в банкоматі або іншому пристрої, що приймає картки. Коли власник картки виходить, шахрай забирає захоплену картку і використовує її в шахрайських цілях.

8. Нанесення відбитків вручну:

У деяких випадках, особливо в ситуаціях, коли електронні платіжні системи недоступні або ненадійні, торговці можуть використовувати ручні імпринтери для зняття інформації з кредитних карток. Шахраї можуть зловживати цим процесом, знімаючи відбиток картки та використовуючи інформацію для шахрайських операцій [1].

1.3 Методи виявлення шахрайства в операціях з банківськими картками

1.3.1 Традиційні методи виявлення шахрайства

Традиційні підходи, засновані на правилах, вже давно використовуються для виявлення шахрайства з кредитними картками. Ці підходи передбачають визначення набору правил або евристик на основі відомих шаблонів та індикаторів шахрайства для виявлення потенційно шахрайських транзакцій. Традиційні методи, засновані на правилах, використовують заздалегідь

визначені правила та евристики для виявлення шахрайських транзакцій на основі конкретних моделей або умов.

Нижче наведено огляд виявлення шахрайства з кредитними картками з використанням традиційних підходів, заснованих на правилах.

1. Системи виявлення шахрайства на основі правил: Системи на основі правил використовують заздалегідь визначений набір правил, які позначають транзакції на основі певних критеріїв. Ці правила, як правило, розробляються на основі історичних шаблонів шахрайства та експертних знань. Наприклад, правила можуть включати такі умови, як транзакції на великі суми, незвичні місця покупок, кілька транзакцій протягом короткого періоду або транзакції, що виходять за рамки звичних витрат власника картки.

2. Порогові підходи: Порогові підходи визначають порогові значення для певних атрибутів транзакцій, таких як сума транзакції, частота транзакцій або географічне розташування. Транзакції, які перевищують ці заздалегідь визначені порогові значення, позначаються як потенційно шахрайські. Наприклад, якщо транзакція перевищує певну суму в доларах або відбувається в країні, яку власник картки ніколи раніше не відвідував, це може викликати сповіщення.

3. Перевірка швидкості: Перевірка швидкості аналізує швидкість або частоту транзакцій для виявлення підозрілих шаблонів. Наприклад, якщо кілька транзакцій від різних продавців відбуваються протягом короткого проміжку часу, це може свідчити про шахрайську діяльність. Перевірка швидкості має на меті виявити незвичні обсяги транзакцій, які відрізняються від типової поведінки власника картки.

4. Система верифікації адреси (AVS): AVS - це заснована на правилах система, яка звіряє адресу для виставлення рахунку, надану під час транзакції, з адресою у файлі емітента картки. У разі невідповідності або неповної інформації про адресу транзакція може бути відмічена для подальшого розслідування.

5. Перевірка контрольного коду картки (CVC): Перевірка CVC передбачає перевірку тризначного коду безпеки (CVC/CVV), наданого під час

транзакції. Якщо CVC не збігається з інформацією про власника картки, це викликає підозру про можливе шахрайство.

6. Оцінки ризику країни та продавця: Деякі системи, що базуються на правилах, присвоюють бали ризику різним країнам або торговцям на основі історичних моделей шахрайства. Транзакції, що походять з країн з високим рівнем ризику або за участю продавців з високим рівнем ризику, можуть викликати додаткову перевірку або вимагати додаткових кроків перевірки [2].

Переваги традиційних підходів, заснованих на правилах.

1. Прозорість: Підходи, засновані на правилах, є прозорими, оскільки правила та критерії, що використовуються для виявлення шахрайства, чітко визначені та зрозумілі.

2. Швидке впровадження: Системи, засновані на правилах, можуть бути впроваджені відносно швидко, оскільки вони покладаються на заздалегідь визначені правила та критерії.

3. Низький рівень хибних спрацьовувань: Підходи, засновані на правилах, часто мають нижчий рівень помилкових спрацьовувань, оскільки правила розроблені для боротьби з конкретними шахрайськими схемами.

Обмеження традиційних підходів, заснованих на правилах:

1. Обмежена адаптивність: Традиційні системи, що базуються на правилах, можуть мати труднощі з адаптацією до нових або еволюціонуючих схем шахрайства. Вони значною мірою покладаються на заздалегідь визначені правила і не можуть ефективно відстежувати нові схеми шахрайства.

2. Високий рівень хибнонегативних спрацьовувань: Системи, що базуються на правилах, можуть пропускати складні схеми шахрайства, які не відповідають попередньо визначеним правилам, що призводить до хибнонегативних результатів.

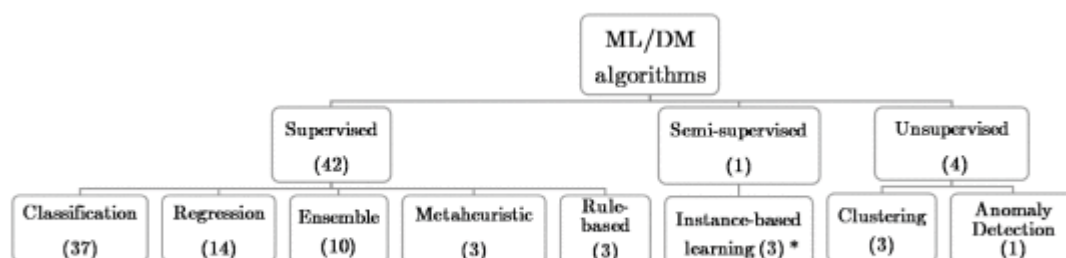
3. Обмежена масштабованість: Оскільки схеми шахрайства стають все більш складними, управління та оновлення великої кількості правил може стати громіздким і непрактичним.

4. Неможливість виявити невідомі схеми шахрайства: Підходи, засновані на правилах, ефективні у виявленні відомих шахрайських схем, але менш здатні виявляти нові або невідомі шахрайські схеми. Загалом, хоча традиційні підходи, засновані на правилах, широко використовуються для виявлення шахрайства з кредитними картками, їхня обмеженість в адаптації до шахрайських схем, що розвиваються, призвела до вивчення більш досконалих методів, таких як машинне навчання та методи обробки концептуального дрейфу.

1.3.2 Методи машинного навчання для виявлення шахрайства

Зі зростанням складності та витонченості шахрайських дій традиційні підходи, засновані на правилах, не встигають за новими шахрайськими схемами. Методи машинного навчання стали потужним інструментом для виявлення шахрайства завдяки своїй здатності навчатися на великих обсягах даних і виявляти складні патерни, що вказують на шахрайську поведінку. Виявлення шахрайства є складним завданням, яке вимагає своєчасної та точної ідентифікації шахрайських дій при мінімізації помилкових спрацьовувань. Методи машинного навчання набули популярності завдяки своїй здатності аналізувати великі масиви даних, виявляти складні закономірності та адаптуватися до шахрайських схем, що постійно змінюються. Логістична регресія, дерева рішень, випадкові ліси, SVM, нейронні мережі, методи ансамблів, виявлення аномалій та методи обробки концептуального дрейфу зазвичай використовуються для виявлення шахрайства. Кожен метод має свої сильні та слабкі сторони, а вибір алгоритму залежить від конкретних вимог та характеристик задачі виявлення шахрайства. Деякі з методів, що використовуються для виявлення шахрайства з кредитними картками, будуть описані нижче.

Найпоширенішим методом машинного навчання в усіх дисциплінах є моделі з вчителем (з керуванням), тобто моделі, навчені на великому наборі належним чином маркованих фінансових звітів. Кожна транзакція позначається як шахрайська або нешахрайська. Для того, щоб навчитися моделям, які найкраще відображають законну поведінку, моделі навчають, подаючи їм велику кількість даних. При розробці моделі з вчителем кількість релевантних навчальних даних прямо корелює з точністю моделі. Загалом, метою розробки моделі без вчителя (без керування) є виявлення аномальної поведінки в ситуаціях, коли кількість тегованих даних обмежена або відсутня. Тому клас самонавчання повинен бути застосований до поверхневих шаблонів у даних, які невидимі для аналітики. Використовуючи комбінацію методів з вчителем та без вчителя, можна виявити раніше невидимі типи підозрілої поведінки у спосіб, який називається методом із слабким керуванням [3].



* Instance based learning algorithms are implemented as either supervised (2) or semi-supervised (1) techniques.

Рисунок 1.1 – Ієрархія методів машинного навчання для виявлення шахрайства [3]

В [3] проведено докладний аналіз статей в розрізів використання різних методів. Алгоритми машинного навчання та інтелектуального аналізу даних використовуються 141 раз у сучасній літературі. На рис. 1.1 представлено ієрархію алгоритмів у проаналізованій літературі та показано, скільки разів застосовується кожен тип методу. Слід зазначити, що в деяких статтях використовувалися декілька методів ML/DM. Алгоритми в першу чергу класифікуються на основі методу навчання на три основні загальні класи: з керуванням, без керування та зі слабким керуванням. Вони, в свою чергу, розподілені на 8 категорій. (рис. 1.1).

1.3.2.1 Класифікація. Методи класифікації є найбільш поширеним підходом, який використовується для виявлення шахрайської фінансової звітності. У літературі для виявлення шахрайства застосовуються різні методи класифікації. Машина опорних векторів (SVM) - це метод керованої класифікації, який шукає гіперплощину з максимальною маржинальністю, що розподіляє вхідні вибірки на два класи. Зокрема, він класифікує нові точки даних на основі маркованого навчального набору даних для кожної категорії. Ядро SVM - це набір математичних функцій, які беруть вхідні дані і відображають їх у простір високої розмірності. Таким чином, SVM може виконувати як лінійні, так і нелінійні (з використанням функцій ядра) задачі класифікації. Послідовна мінімальна оптимізація (SMO), радіально-базисна функція (RBF), лінійне, гаусове, поліноміальне та класово-зважене ядро є прикладами функцій ядра, що використовуються в дослідженнях [4, 5].

Сімейство алгоритмів дерев рішень також привертає значну увагу дослідників. Існують різні способи створення дерева рішень для класифікації. Традиційні алгоритми дерев рішень існують вже десятки років. Так, випадковий ліс є сучасною варіацією дерев рішень і є одним з найпотужніших методів. Дерева класифікації та регресії (CART) відносяться до алгоритмів дерев рішень, які можуть бути застосовані для задач регресійного або класифікаційного прогнозного моделювання. CART - це метод бінарних дерев рішень, який має справу з класифікацією непараметричних або безперервних даних. Умови поділяються на основі атрибутів даних, кількості та індексу Джині. Дані багаторазово розбиваються на дві гілки для визначення наступних умов поділу. Цей поділ даних продовжується для побудови деревовидної структури. Процедура завершується, коли дані більше не піддаються поділу. Алгоритм C4.5 працює як класифікатор дерев рішень серед методів ML/DM. Він приймає рішення на основі конкретних даних (одновимірних або багатовимірних предикторів). Алгоритм дерева рішень C4.5 вимірює інформаційну цінність кожного атрибута і розділяє дані за допомогою атрибута з найбільшою інформаційною цінністю. Іншими поширеними алгоритмами дерев рішень є Chi-

Squared Automatic Interaction Detector (CHAID), Quick Unbiased and Efficient Statistical Tree (QUEST), JRip, Logistic Model Trees та Tree Induction підходи [6, 7].

Ще одне сімейство методів класифікації - це байєсівські методи. Можна виділити наївний Байєс та байєсові мережі. Наївний Байєс - це різновид методів машинного навчання, який базується на теоремі Байєса і використовується для класифікації. Він передбачає ймовірність належності до класу. Зокрема, цей класифікатор прогнозує мітку точки даних на основі ймовірності того, що вона належить до певного класу. Перше припущення наївного байєсівського класифікатора полягає в тому, що взаємні залежності та кореляції між точками даних зручно ігнорувати. Значення певного атрибуту не залежить від значення будь-якого іншого атрибуту. У багатьох практичних застосуваннях на вхід подаються навчальні дані, а на виході - необхідні параметри для класифікатора на етапі навчання. Наївні байєсівські моделі використовують оцінку максимальної правдоподібності (MLE), оптимізацію критерію втрат і байєсівську оцінку (максимальну апостеріорну) для вибору параметрів класифікатора з навчальних даних.

Байєсівські мережі - це статистичні ймовірнісні графічні моделі, які ілюструють набір змінних та їхні умовні кореляції за допомогою спрямованого ациклічного графа (DAG). Для прогнозування ймовірності того, яка з кількох потенційних розпізнаних причин була основною, байєсівські мережі є чудовим вибором [8].

Штучні нейронні мережі (ШНМ) - це одна категорія алгоритмів класифікації. ШНМ - це моделі обробки інформації, натхненні поведінкою біологічної нервової системи. Нейронні мережі є дуже потужними, коли є великий обсяг даних. Фаза навчання ШНМ може бути керованою, некерованою або зі слабким керуванням. ШНМ можна розділити на різні класи з точки зору стратегії навчання. Багатошарова нейронна мережа прямого поширення (MLFF), імовірнісна нейронна мережа (PNN), самоорганізаційні карти (SOM) є найпопулярнішими ШНМ з точки зору частоти використання в розглянутих

задачах. Крім того, до цієї групи можна віднести і груповий метод обробки даних (GMDH) до нейромережових моделей, оскільки він дещо натхненний дослідженнями персептронів та фільтрів, що навчаються. Це керований підхід, який автоматично самоорганізується (подібно до SOM) і визначає найкращу кількість вузлів та структуру мережі [9, 10].

Алгоритм К-найближчих сусідів (KNN) - це зручний і простий метод машинного навчання з керуванням, який намагається вирішити проблеми регресії та класифікації. Метод KNN визначає мітку класу на основі дуже обмеженої кількості найближчих зразків. Зокрема, він визначає мітку для тестових екземплярів відповідно до міток k сусідніх екземплярів. Отже, цей метод більше підходить для наборів зразків, що перекриваються або перехрещуються, оскільки він значною мірою покладається на навколишні обмежені зразки, а не на метод дискримінантної області для прийняття рішення про клас [11].

Також варто виділити модель дистанційно-зваженої дискримінації (Distance Weighted Discrimination, DWD). Модель DWD має відмінну продуктивність у налаштуваннях високого виміру з малим розміром вибірки і була застосована для виявлення шахрайських фінансових звітів. Дозволяє поєднати текстові та фінансові ознаки для покращення виявлення шахрайства у фінансовій звітності. Також запропоновано метод DWD, заснований на генетичному алгоритмі для оптимізації параметрів і вибору ознак, що дозволяє досягти досить хороших результатів класифікації з використанням обмеженого набору ознак [12].

1.3.2.2 Регресія. Найбільш типовим є використання методу логістичної регресії (ЛР). Логістичні регресійні моделі застосовуються в багатокласових і бінарних класифікаціях. Ще однією причиною популярності моделей логістичної регресії у відповідній літературі з бухгалтерського обліку є те, що вони піддаються статистичній інтерпретації. Базова логістична регресія - це статистичний метод, який застосовує логістичну функцію для моделювання бінарної залежної змінної; однак існує кілька інших складних розширень. Порівняння результатів багаточленної логістики з класифікаторами SVM та BayesNet показує, що логістичний метод часто показує кращі результати [13].

1.3.2.3 Ансамблеві методи. Ансамблеві методи - це метаалгоритми, які об'єднують декілька інтелектуальних моделей в одну прогнозуючу модель. Загальна мета ансамблевих алгоритмів полягає в тому, щоб перетворити окремі слабкі навчальні моделі на сильніші. Кожен ансамблевий метод переслідує різні цілі; наприклад, беггінг намагається зменшити дисперсію, бустінг - зменшити зсув, а стекінг - покращити прогнози. Крім того, Random forest (RF) або ліс випадкових рішень є ще однією технікою ансамблевого навчання для регресії, класифікації та інших задач [14].

1.3.2.4 Метаевристичні алгоритми. Метаевристичні підходи застосовуються як у керованому, так і в некерованому навчанні.

Генетичне програмування (ГП) належить до сімейства еволюційних алгоритмів, яке розширює генетичні алгоритми (ГА) для підтримки дослідження простору комп'ютерних програм. Як і ГА, ГП випадковим чином генерує початкову популяцію розв'язків задачі і маніпулює ними, згодом використовуючи генетичні операції, такі як кросинговер, мутація і розмноження. Ця нова популяція відома як покоління. ГП разом з іншими методами інтелектуального аналізу даних можна використати для виявлення компаній, які вчинили шахрайство з фінансовою звітністю, при цьому ГП перевершило інші класифікатори [15].

1.3.2.5 Підходи, що базуються на правилах. Додаткове використання методів машинного навчання в процесі генерації правил може суттєво розширити можливості традиційного підходу. Так, наприклад, в [16] поєднали генетичний відбір ознак та вилучення правил, щоб запропонувати нечітку модель виявлення шахрайства на основі правил. Усунення нерелевантних атрибутів шляхом відбору ознак та застосування алгоритму нечіткого неупорядкованого виведення правил (FURIA) дозволило цій моделі досягти прийнятної продуктивності.

1.3.2.6 Навчання на основі прикладів. Цей термін відноситься до сімейства алгоритмів навчання, які замінюють виконання явного узагальнення порівнянням небачених зразків зі зразками, які бачили під час навчання і які зберігаються в пам'яті. Методика може бути керованою або зі слабким керуванням. Локально зважене навчання (LWL) - це техніка апроксимації функцій і ще один досліджуваний метод у сімействі навчання на основі прикладів. Метою цих методів є виявлення взаємозв'язку між входом і виходом. Відповідно, для виконання завдання прогнозування використовувалася наближена локальна модель навколо поточної точки [17].

1.3.2.7 Кластеризація. Кластеризація - це метод об'єднання ідентичних екземплярів в однакові групи. Цей метод машинного навчання без вчителя, спрямований на пошук подібностей у даних і групування подібних даних. В [18] було застосовано ієрархічну кластеризацію текстового контенту MD&A для створення моделі виявлення шахрайства. При збереженні максимальної кількості інформації основною схемою зменшення розмірності є створення вектора сингулярної декомпозиції (SVD), що було застосовано в [18] для зменшення розмірності тексту. Потім документи кластеризуються, а результати кластеризації переглядаються та оцінюються. Після цього SVD та алгоритм кластеризації модифікуються, якщо це необхідно.

1.3.2.8 Виявлення аномалій. У літературі існує ще один некерований підхід для виявлення аномалій. Відстань Махаланобіса є одним з найпростіших методів фільтрації аномальних точок даних. В [19] застосували підхід інтелектуального аналізу даних, в якому відстань Махаланобіса використовується для обчислення близькості кожної точки даних до центроїда розподілу для визначення амплітуди аномалії. Запропонована модель змогла ефективно впорядкувати фінансову звітність в сенсі достовірності. Застосування моделі до всіх спостережень показало, що майже чверть з них були дуже аномальними і сумнівними, а більшість фінансових звітів в'єтнамських компаній, що котируються на біржі, є достовірними.

1.4 Основні виклики при створенні моделі для виявлення шахрайства

Процес створення моделі для виявлення шахрайства з кредитними картками з використанням маркованих даних про транзакції має кілька основних обмежень і проблем.

1. Значний дисбаланс у даних: оскільки звичайні операції значно переважають шахрайські транзакції (як правило, близько 99.8%). При цьому, саме клас меншості (шахрайства) становить інтерес, тому їх дефіцит часто перешкоджає здатності класифікатора до навчання, що часто призводить до поганих моделей.

2. Недоступність наборів даних: карткові компанії не можуть надавати їх у відкрите користування дослідникам через конфіденційність, приватність, секретність та юридичні проблеми. Незважаючи на велику кількість алгоритмів машинного навчання і великий інтерес до них, будь-яка дослідницька робота в цьому напрямку потребує реальних даних для прогресу. Саме тому в роботі розглядається використання синтетичних даних як альтернативи.

3. Обмежена кількість опублікованих робіт: дослідники, як правило, не розкривають сутність даних або параметрів, що використовуються в класифікаторах. Це може бути пов'язано з обмеженнями фінансових установ. Розробка нових методів виявлення шахрайства з кредитними картками серйозно гальмується труднощами в обміні ідеями щодо виявлення шахрайства через проблеми безпеки та конфіденційності.

4. Адаптивна та інноваційна шахрайська поведінка: Профілі шахрайства та нормальної поведінки змінюються занадто часто і продовжують розвиватися, вимагаючи постійного повторного вивчення нових шаблонів. Також існує ймовірність того, що минулі шахрайські транзакції будуть помилково сприйняті як дійсні, а алгоритм вивчить ці помилкові факти. Нові шахрайства можуть бути розроблені так, щоб імітувати нормальні транзакції, і один промах може призвести до того, що всі подібні випадки будуть пропущені.

5. Перекриття та обмежена відокремлюваність класів: є прямим наслідком вищезазначеного. За допомогою DBSCAN, змішаної моделі Гауса, біноміальної логістичної регресії та k-середніх було виявлено, що шахрайські транзакції не є викидами (аномаліями) і що вони не є ізольованими або обмеженими певним кластером, а розподілені рівномірно.

6. Відсутність відповідних метрик оцінювання: для сильно незбалансованих даних традиційні показники точності не можуть бути використані. Відзначається відсутність стандартного критерію оцінки або метрик для вимірювання та порівняння ефективності та якості систем виявлення шахрайства. Як наслідок, неможливо проводити бенчмаркінг моделей та алгоритмів виявлення. Дослідження розділяються щодо того, що є ідеальною метрикою для оцінки і які моделі дають кращі результати.

7. Вартість виявлення шахрайства: має бути економічно прийнятною і нижчою, ніж збитки від самого шахрайства. Зважаючи на широкий спектр шахрайств і величезні обсяги, запобігання шахрайству на кілька доларів не має особливого сенсу, і такі фінансові операції повинні бути включені в модель і дизайн системи виявлення [20].

РОЗДІЛ 2 СИСТЕМА ВИЯВЛЕННЯ ШАХРАЙСТВА З КРЕДИТНИМИ КАРТКАМИ

2.1 Генерація даних транзакцій

2.1.1 Постановка задачі

В роботі розв’язується задача виявлення шахрайських операцій серед всіх транзакцій з кредитними картками. Таким чином алгоритм являє собою бінарний класифікатор. Розподіл класів в даних є дуже незбалансованим (низька частка шахрайських транзакцій), що додатково ускладнює розробку стійких алгоритмів.

Формування датасету для навчання та тестування такої моделі є досить нетривіальною задачею. Такі дані, навіть знеособлені, є вкрай чутливою інформацією для будь-якої банківської установи, тому доступ до неї вкрай обмежено. Тому було вирішено використовувати штучний генератор даних транзакцій [21], якій, одночасно з простотою генерації дозволяє отримати велику кількість числових та категоріальних ознак.

2.1.2 Генерація необроблених даних

Транзакція з використанням платіжної картки складається з суми, яку клієнт сплачує продавцю в певний момент. Шість основних параметрів, які характеризують транзакцію:

- Ідентифікатор транзакції: Унікальний ідентифікатор транзакції.
- Дата та час: Дата і час, коли відбулася транзакція.
- Ідентифікатор клієнта. Кожен клієнт має унікальний ідентифікатор.
- Ідентифікатор термінала. Кожен термінал має унікальний ідентифікатор.
- Сума транзакції.

– Мітка шахрайства: Бінарна змінна зі значенням 1 для законної транзакції або значенням 0 для шахрайської транзакції.

Ці параметри будуть позначатися як TRANSACTION_ID, TX_DATETIME, CUSTOMER_ID, TERMINAL_ID, TX_AMOUNT і TX_FRAUD відповідно.

Метою симулятора даних про транзакції буде створення таблиці транзакцій з цими параметрами .

Таблиця 2.1 – Приклад таблиці транзакцій з мітками.

TRANSACTION_ID	TX_DATETIME	CUSTOMER_ID	TERMINAL_ID	TX_AMOUNT	TX_FRAUD
0	08-12-2023 14:56:11	4568	4589	150.25	0
1	01-12-2023 19:45:09	568	8574	147.89	0
2	15-12-2023 12:06:21	78	8963	25.63	0
...	...	...	...	...	...

2.1.3 Генерація профілів клієнтів

Кожен клієнт буде визначатися такими властивостями:

- CUSTOMER_ID: унікальний ідентифікатор клієнта;
- (x_customer_id, y_customer_id): пара реальних координат (x_customer_id, y_customer_id) у сітці 100*100, яка визначає географічне розташування клієнта;
- (mean_amount, std_amount): середнє значення та стандартне відхилення сум транзакцій для клієнта, припускаючи, що суми транзакцій відповідають нормальному розподілу. Mean_amount буде взято з рівномірного розподілу (5100), а std_amount буде встановлено як mean_amount, поділене на два;

– `mean_nb_tx_per_day`: Середня кількість транзакцій на день для клієнта, припускаючи, що кількість транзакцій на день відповідає розподілу Пуассона. Це число буде взято з рівномірного розподілу (0,4).

Функція `generate_customer_profiles_table` (рис. 1.1) забезпечує створення таблиці профілів клієнтів. У якості вхідних даних приймається кількість клієнтів, для яких потрібно створити профіль, і «зерно» для відтворюваності. Він повертає `DataFrame`, що містить властивості для кожного клієнта.

```
def generate_customer_profiles_table(n_customers, random_state=0):

    np.random.seed(random_state)

    customer_id_properties=[]

    # Generate customer properties from random distributions
    for customer_id in range(n_customers):

        x_customer_id = np.random.uniform(0,100)
        y_customer_id = np.random.uniform(0,100)

        mean_amount = np.random.uniform(5,100) # Arbitrary (but sensible) value
        std_amount = mean_amount/2 # Arbitrary (but sensible) value

        mean_nb_tx_per_day = np.random.uniform(0,4) # Arbitrary (but sensible) value

        customer_id_properties.append([customer_id,
                                       x_customer_id, y_customer_id,
                                       mean_amount, std_amount,
                                       mean_nb_tx_per_day])

    customer_profiles_table = pd.DataFrame(customer_id_properties, columns=['CUSTOMER_ID',
                                                                           'x_customer_id', 'y_customer_id',
                                                                           'mean_amount', 'std_amount',
                                                                           'mean_nb_tx_per_day'])

    return customer_profiles_table
```

Рисунок 2.1 – Лістинг функції «`generate_customer_profiles_table`»

Симуляція складається з п'яти основних кроків:

– Створення профілів клієнтів: кожен клієнт відрізняється своїми звичками витратити гроші. Це буде змодельовано шляхом визначення деяких властивостей для кожного клієнта. Основними характеристиками будуть їхнє географічне розташування, частота витрат і суми витрат. Властивості клієнта будуть представлені у вигляді таблиці, яку називають таблицею профілю клієнта.

– Генерація профілів терміналу: властивості терміналу складаються лише з географічного розташування. Властивості терміналу будуть представлені у вигляді таблиці, яку будемо називати таблицею профілю терміналу.

- Встановлення відповідностей між профілями клієнтів та терміналами: припустимо, що клієнти здійснюють транзакції лише на терміналах, які знаходяться в радіусі r їхнього географічного положення. Тобто, робиться припущення, що клієнт здійснює транзакції лише на терміналах, географічно близьких до його місцезнаходження.
- Генерація транзакцій: симулятор буде циклічно переглядати набір профілів клієнтів і генерувати транзакції відповідно до їхніх властивостей (частота витрат і суми, а також доступних терміналів). Результатом стане таблиця транзакцій.
- Створення сценаріїв шахрайства: цей останній крок позначає транзакції як законні або справжні. Це буде зроблено за допомогою трьох різних сценаріїв шахрайства.

2.1.4 Генерація профілів терміналів

Кожен термінал буде визначено такими властивостями:

- `TERMINAL_ID`: ідентифікатор терміналу;
- `(x_terminal_id, y_terminal_id)`: пара реальних координат `(x_terminal_id, y_terminal_id)` у сітці 100×100 , яка визначає географічне розташування терміналу;

Функція `generate_terminal_profiles_table` (рис. 1.1) забезпечує реалізацію для створення таблиці профілів терміналу. У якості вхідних даних приймається кількість терміналів, для яких генерується профіль, і «зерно» для відтворюваності. Вона повертає `DataFrame`, що містить властивості для кожного терміналу.

```
def generate_terminal_profiles_table(n_terminals, random_state=0):

    np.random.seed(random_state)

    terminal_id_properties=[]

    # Generate terminal properties from random distributions
    for terminal_id in range(n_terminals):

        x_terminal_id = np.random.uniform(0,100)
        y_terminal_id = np.random.uniform(0,100)

        terminal_id_properties.append([terminal_id,
                                       x_terminal_id, y_terminal_id])

    terminal_profiles_table = pd.DataFrame(terminal_id_properties, columns=['TERMINAL_ID',
                                                                           'x_terminal_id', 'y_terminal_id'])

    return terminal_profiles_table
```

Рисунок 2.2 – Лістинг функції «generate_terminal_profiles_table»

2.1.5 Прив'язка профілів клієнтів до терміналів

Розглянемо етап, на якому пов'яжемо термінали з профілями клієнтів. Згідно з дизайном, клієнти можуть виконувати транзакції лише на терміналах, які знаходяться в радіусі r відносно їхнього географічного розташування.

Функція під назвою `get_list_terminals_within_radius` (рис. 1.3) знаходить ці термінали для профілю клієнта. Функція прийматиме як вхідні дані профіль клієнта (будь-який рядок у таблиці профілів клієнтів), масив, який містить географічне розташування всіх терміналів, і радіус r . Вона повертає список терміналів у радіусі r для цього клієнта.

```
def get_list_terminals_within_radius(customer_profile, x_y_terminals, r):

    # Use numpy arrays in the following to speed up computations

    # Location (x,y) of customer as numpy array
    x_y_customer = customer_profile[['x_customer_id', 'y_customer_id']].values.astype(float)

    # Squared difference in coordinates between customer and terminal locations
    squared_diff_x_y = np.square(x_y_customer - x_y_terminals)

    # Sum along rows and compute squared root to get distance
    dist_x_y = np.sqrt(np.sum(squared_diff_x_y, axis=1))

    # Get the indices of terminals which are at a distance less than r
    available_terminals = list(np.where(dist_x_y < r)[0])

    # Return the list of terminal IDs
    return available_terminals
```

Рисунок 2.3 – Лістинг функції «get_list_terminals_within_radius»

2.1.6 Генерація транзакцій

Профілі клієнтів тепер містять усю інформацію, необхідну для створення транзакцій. Генерація транзакцій здійснюватиметься функцією `generate_transactions_table` (1.4), яка приймає як вхідні дані профіль клієнта, дату початку та кількість днів, для яких потрібно генерувати транзакції. Вона повертає таблицю транзакцій у форматі, наведеному в табл. 1.1 (без мітки транзакції, яка буде додана під час створення сценаріїв шахрайства).

```

def generate_transactions_table(customer_profile, start_date = "2023-01-01", nb_days = 10):

    customer_transactions = []

    random.seed(int(customer_profile.CUSTOMER_ID))
    np.random.seed(int(customer_profile.CUSTOMER_ID))

    # For all days
    for day in range(nb_days):

        nb_tx = np.random.poisson(customer_profile.mean_nb_tx_per_day)

        if nb_tx>0:

            for tx in range(nb_tx):

                time_tx = int(np.random.normal(86400/2, 20000))

                if (time_tx>0) and (time_tx<86400):

                    amount = np.random.normal(customer_profile.mean_amount, customer_profile.std_amount)

                    if amount<0:
                        amount = np.random.uniform(0, customer_profile.mean_amount*2)

                    amount=np.round(amount, decimals=2)

                    if len(customer_profile.available_terminals)>0:

                        terminal_id = random.choice(customer_profile.available_terminals)

                        customer_transactions.append([time_tx+day*86400, day,
                                                    customer_profile.CUSTOMER_ID,
                                                    terminal_id, amount])

    customer_transactions = pd.DataFrame(customer_transactions, columns=['TX_TIME_SECONDS',
                                                                    'TX_TIME_DAYS', 'CUSTOMER_ID',
                                                                    'TERMINAL_ID', 'TX_AMOUNT'])

    if len(customer_transactions)>0:
        customer_transactions['TX_DATETIME'] = pd.to_datetime(customer_transactions['TX_TIME_SECONDS'],
                                                                unit='s', origin=start_date)

        customer_transactions=customer_transactions[['TX_DATETIME', 'CUSTOMER_ID',
                                                    'TERMINAL_ID', 'TX_AMOUNT', 'TX_TIME_SECONDS', 'TX_TIME_DAYS']]

    return customer_transactions

```

Рисунок 2.4 – Лістинг функції «generate_transactions_table»

2.1.7 Масштабування до більшого набору даних

Тепер є всі будівельні блоки для створення більшого набору даних. Напишемо функцію generate_dataset, яка подбає про виконання всіх попередніх кроків. Для цього візьмемо як вхідні дані кількість бажаних клієнтів, терміналів і днів, а також дату початку та радіус r . Функція повертає згенеровану таблицю профілів клієнтів і терміналів, а також DataFrame транзакцій.

```
def generate_dataset(n_customers = 10000, n_terminals = 1000000, nb_days=90, start_date="2023-01-01", r=5):

    start_time=time.time()
    customer_profiles_table = generate_customer_profiles_table(n_customers, random_state = 0)
    print("Time to generate customer profiles table: {0:.2}s".format(time.time()-start_time))

    start_time=time.time()
    terminal_profiles_table = generate_terminal_profiles_table(n_terminals, random_state = 1)
    print("Time to generate terminal profiles table: {0:.2}s".format(time.time()-start_time))

    start_time=time.time()
    x_y_terminals = terminal_profiles_table[['x_terminal_id','y_terminal_id']].values.astype(float)
    customer_profiles_table['available_terminals'] = customer_profiles_table.apply(lambda x :
        get_list_terminals_within_radius(x, x_y_terminals=x_y_terminals, r=r), axis=1)

    customer_profiles_table['nb_terminals']=customer_profiles_table.available_terminals.apply(len)
    print("Time to associate terminals to customers: {0:.2}s".format(time.time()-start_time))

    start_time=time.time()
    transactions_df=customer_profiles_table.groupby('CUSTOMER_ID').apply(lambda x :
        generate_transactions_table(x.iloc[0], nb_days=nb_days)).reset_index(drop=True)

    print("Time to generate transactions: {0:.2}s".format(time.time()-start_time))

    # Sort transactions chronologically
    transactions_df=transactions_df.sort_values('TX_DATETIME')
    # Reset indices, starting from 0
    transactions_df.reset_index(inplace=True,drop=True)
    transactions_df.reset_index(inplace=True)
    # TRANSACTION_ID are the dataframe indices, starting from 0
    transactions_df.rename(columns = {'index':'TRANSACTION_ID'}, inplace = True)

    return (customer_profiles_table, terminal_profiles_table, transactions_df)
```

Рисунок 2.5 – Лістинг функції «generate_dataset»

2.1.8 Генерація сценаріїв шахрайства

Цей останній крок симуляції додає шахрайські транзакції до набору даних, використовуючи такі сценарії шахрайства:

Сценарій 1: будь-яка транзакція, сума якої перевищує 220, є шахрайством. Цей сценарій не натхненний реальним сценарієм. Натомість це забезпечить очевидну модель шахрайства, яку має виявити будь-який базовий детектор шахрайства. Це буде корисно для перевірки реалізації техніки виявлення шахрайства.

Сценарій 2: кожного дня випадковим чином складається список із двох терміналів. Усі транзакції на цих терміналах протягом наступних 28 днів будуть позначені як шахрайські. Цей сценарій імітує злочинне використання терміналу,

наприклад, через фішинг. Виявлення цього сценарію стане можливим, додавши функції, які відстежують кількість шахрайських транзакцій на терміналі. Оскільки термінал скомпрометований лише протягом 28 днів, необхідно розробити додаткові стратегії, які передбачають зміну концепції, щоб ефективно впоратися з цим сценарієм.

Сценарій 3: кожного дня випадковим чином складається список із 3 клієнтів. Протягом наступних 14 днів суми $1/3$ їхніх транзакцій помножуються на 5 і позначаються як шахрайські. Цей сценарій імітує шахрайство з відсутністю картки, коли облікові дані клієнта витікають. Клієнт продовжує здійснювати транзакції, а транзакції більшої вартості здійснюються шахраями, які намагаються максимізувати свої прибутки. Виявлення цього сценарію вимагатиме додавання функцій, які відстежуватимуть звички витратити клієнта. Що стосується сценарію 2, оскільки карта лише тимчасово скомпрометована, також слід розробити додаткові стратегії, які включають зміни концепції.

Додамо шахрайські транзакції за такими сценаріями:

```
transactions_df = add_frauds(
customer_profiles_table, terminal_profiles_table,
transactions_df)
```

Кількість транзакцій за кожним зі сценаріїв:

Number of frauds from scenario 1: 978

Number of frauds from scenario 2: 9099

Number of frauds from scenario 3: 4604

Кількість шахрайських операцій:

```
transactions_df.TX_FRAUD.sum()
14681
```

Всього 14681 транзакція була позначена як шахрайська. Це становить 0,8% угод. Зверніть увагу, що сума шахрайства для кожного сценарію не дорівнює загальній кількості шахрайських транзакцій. Це пояснюється тим, що ті самі транзакції могли бути позначені як шахрайські за двома або більше сценаріями шахрайства.

```

def add_frauds(customer_profiles_table, terminal_profiles_table, transactions_df):

    # By default, all transactions are genuine
    transactions_df['TX_FRAUD']=0
    transactions_df['TX_FRAUD_SCENARIO']=0

    # Scenario 1
    transactions_df.loc[transactions_df.TX_AMOUNT>220, 'TX_FRAUD']=1
    transactions_df.loc[transactions_df.TX_AMOUNT>220, 'TX_FRAUD_SCENARIO']=1
    nb_frauds_scenario_1=transactions_df.TX_FRAUD.sum()
    print("Number of frauds from scenario 1: "+str(nb_frauds_scenario_1))

    # Scenario 2
    for day in range(transactions_df.TX_TIME_DAYS.max()):

        compromised_terminals = terminal_profiles_table.TERMINAL_ID.sample(n=2, random_state=day)

        compromised_transactions=transactions_df[(transactions_df.TX_TIME_DAYS>=day) &
            (transactions_df.TX_TIME_DAYS<day+28) &
            (transactions_df.TERMINAL_ID.isin(compromised_terminals))]

        transactions_df.loc[compromised_transactions.index, 'TX_FRAUD']=1
        transactions_df.loc[compromised_transactions.index, 'TX_FRAUD_SCENARIO']=2

    nb_frauds_scenario_2=transactions_df.TX_FRAUD.sum()-nb_frauds_scenario_1
    print("Number of frauds from scenario 2: "+str(nb_frauds_scenario_2))

    # Scenario 3
    for day in range(transactions_df.TX_TIME_DAYS.max()):

        compromised_customers = customer_profiles_table.CUSTOMER_ID.sample(n=3, random_state=day).values

        compromised_transactions=transactions_df[(transactions_df.TX_TIME_DAYS>=day) &
            (transactions_df.TX_TIME_DAYS<day+14) &
            (transactions_df.CUSTOMER_ID.isin(compromised_customers))]

        nb_compromised_transactions=len(compromised_transactions)

        random.seed(day)
        index_frauds = random.sample(list(compromised_transactions.index.values),k=int(nb_compromised_transactions/3))

        transactions_df.loc[index_frauds, 'TX_AMOUNT']=transactions_df.loc[index_frauds, 'TX_AMOUNT']*5
        transactions_df.loc[index_frauds, 'TX_FRAUD']=1
        transactions_df.loc[index_frauds, 'TX_FRAUD_SCENARIO']=3

    nb_frauds_scenario_3=transactions_df.TX_FRAUD.sum()-nb_frauds_scenario_2-nb_frauds_scenario_1
    print("Number of frauds from scenario 3: "+str(nb_frauds_scenario_3))

    return transactions_df

```

Рисунок 2.6 – Лістинг функції «add_frauds»

Набір даних змодельованих транзакцій завершено, до всіх транзакцій додано шахрайську мітку.

Замість того, щоб зберігати весь набір даних транзакцій, розіб'ємо набір даних на щоденні пакети. Це дозволить пізніше завантажувати конкретні періоди замість усього набору даних. Для прискорення завантаження використовується формат pickle, а не CSV. Усі файли зберігаються в папці DIR_OUTPUT. Назви файлів — це дати з розширенням .pkl.

```

DIR_OUTPUT = "./simulated-data-raw/"

if not os.path.exists(DIR_OUTPUT):
    os.makedirs(DIR_OUTPUT)

start_date = datetime.datetime.strptime("2023-01-01", "%Y-%m-%d")

for day in range(transactions_df.TX_TIME_DAYS.max()+1):

    transactions_day = transactions_df[transactions_df.TX_TIME_DAYS==day].sort_values('TX_TIME_SECONDS')

    date = start_date + datetime.timedelta(days=day)
    filename_output = date.strftime("%Y-%m-%d")+'.pkl'

    transactions_day.to_pickle(DIR_OUTPUT+filename_output, protocol=4)

```

Рисунок 2.7 – Лістинг скрипту, що записує датасет на диск

2.2 Генерація додаткових ознак з початкового набору даних

Змодельований набір даних містить лише основні характеристики, які характеризують операцію з платіжною картою.

Хоча він концептуально простий для людини, такий набір даних не підходить для прогнозної моделі машинного навчання. Алгоритми машинного навчання зазвичай потребують числових і впорядкованих ознак. Тобто, тип змінної має бути цілим або дійсним числом, та порядок значень змінної є значущим.

У цьому наборі даних єдиними числовими та впорядкованими функціями є сума транзакції та позначка шахрайства. Дата є міткою часу Panda, а отже, не числом. Ідентифікатори транзакцій, клієнтів і терміналів є числовими, але не впорядкованими: не має сенсу припускати, наприклад, що термінал з ідентифікатором 3548 «більший», ніж термінал з ідентифікатором 1983. Ці ідентифікатори представляють окремі «сутності», які називаються категоріальними ознаками.

На жаль, не існує стандартної процедури для роботи з нечисловими або категоріальними ознаками. Ця тема відома в літературі з машинного навчання як інженерія ознак або перетворення функцій. По суті, метою розробки функцій є

розробка нових ознак, які вважаються релевантними для прогнозної проблеми. Розробка цих ознак зазвичай залежить від проблеми та передбачає знання предметної області.

2.2.1 Перетворення дати та часу

Створимо дві нові бінарні ознаки з дат і часу транзакції:

Перша характеризуватиме, чи відбувається транзакція в будній день (значення 0) чи у вихідні (1), і називатиметься `TX_DURING_WEEKEND`.

Друга характеризуватиме, чи відбувається транзакція вдень (0) чи вночі (1). Ніч визначається як години між 0 вечора та 6 ранку. Вона називатиметься `TX_DURING_NIGHT`.

Для функції `TX_DURING_WEEKEND` ми визначаємо функцію `is_weekend`, яка приймає як вхідні дані мітку часу Panda та повертає 1, якщо дата припадає на вихідні, або 0 в іншому випадку. Об'єкт `timestamp` надає функцію `weekday` для визначення дня тижня.

Дотримуємося тієї ж логіки, щоб реалізувати функцію `TX_DURING_NIGHT`. Реалізуємо функцію `is_night`, яка приймає як вхідні дані позначку часу Panda та повертає 1, якщо час вночі, або 0 в іншому випадку. Об'єкт `timestamp` зручно надає властивість `hour`, яка використовується в даній функції.

2.2.2 Перетворення ідентифікатора клієнта

Тепер приступимо до трансформації ідентифікатора клієнта. Ми візьмемо за основу підхід RFM (Recency, Frequency, Monetary value), і обчислимо дві з цих функцій у трьох часових вікнах. Першою ознакою буде кількість транзакцій, які відбуваються в межах часового вікна (Frequency). Другою буде середня сума, витрачена на ці операції (Monetary value). Часові вікна будуть встановлені на один, сім і тридцять днів. Це створить шість нових ознак. Ці часові вікна пізніше можуть бути оптимізовані разом із іншими параметрами моделі.

Реалізуємо ці перетворення, написавши функцію `get_customer_spending_behaviour_features`. Функція приймає як вхідні дані набір транзакцій для клієнта та набір розмірів вікон. Вона повертає DataFrame із шістьма новими ознаками. Реалізація покладається на функцію Panda `rolling`, яка полегшує обчислення агрегатів у часовому вікні.

```
def get_customer_spending_behaviour_features(customer_transactions, windows_size_in_days=[1,7,30]):

    # Let us first order transactions chronologically
    customer_transactions=customer_transactions.sort_values('TX_DATETIME')

    customer_transactions.index=customer_transactions.TX_DATETIME

    for window_size in windows_size_in_days:

        SUM_AMOUNT_TX_WINDOW=customer_transactions['TX_AMOUNT'].rolling(str(window_size)+'d').sum()
        NB_TX_WINDOW=customer_transactions['TX_AMOUNT'].rolling(str(window_size)+'d').count()

        AVG_AMOUNT_TX_WINDOW=SUM_AMOUNT_TX_WINDOW/NB_TX_WINDOW

        customer_transactions['CUSTOMER_ID_NB_TX_'+str(window_size)+'DAY_WINDOW']=list(NB_TX_WINDOW)
        customer_transactions['CUSTOMER_ID_AVG_AMOUNT_'+str(window_size)+'DAY_WINDOW']=list(AVG_AMOUNT_TX_WINDOW)

    customer_transactions.index=customer_transactions.TRANSACTION_ID

    return customer_transactions
```

Рисунок 2.8 – Лістинг функції «`get_customer_spending_behaviour_features`»

Тепер створимо ці функції для всіх клієнтів. Це просто за допомогою методів Panda `groupby` і `apply`.

```
transactions_df=transactions_df.groupby('CUSTOMER_ID')
transactions_df.apply(lambda x:
    get_customer_spending_behaviour_features(x,
    windows_size_in_days=[1,7,30]))
```

2.2.3 Перетворення ідентифікатора терміналу

Перейдемо до трансформації ідентифікатора терміналу. Основна мета полягатиме в отриманні оцінки ризику, яка показує вразливість певного ідентифікатора терміналу для шахрайських транзакцій. Оцінка ризику визначатиметься як середня кількість шахрайських транзакцій, які відбулися на ідентифікаторі терміналу за часовий проміжок. Тут також будемо використовувати три розміри вікна: 1, 7 і 30 днів.

На відміну від перетворень ідентифікатора клієнта, часові вікна не будуть безпосередньо передувати даній транзакції. Натомість їх буде зміщено назад на період затримки. Період затримки пояснюється тим фактом, що на практиці шахрайські транзакції виявляються лише після розслідування шахрайства або скарги клієнта. Отже, шахрайські мітки, необхідні для обчислення оцінки ризику, доступні лише після цього періоду затримки. У першому наближенні цей період затримки буде встановлено на один тиждень.

Виконаємо обчислення показників ризику, визначивши функцію `get_count_risk_rolling_window`. Функція приймає як вхідні дані `DataFrame` транзакцій для даного ідентифікатора терміналу, період затримки та список розмірів вікон. На першому етапі обчислюється кількість транзакцій і шахрайських транзакцій для періоду затримки (`NB_TX_DELAY` і `NB_FRAUD_DELAY`). На другому етапі кількість транзакцій і шахрайських транзакцій обчислюється для кожного розміру вікна плюс період затримки (`NB_TX_DELAY_WINDOW` і `NB_FRAUD_DELAY_WINDOW`). Кількість транзакцій і шахрайських транзакцій, які відбулися для заданого розміру вікна, зміщена назад на період затримки, потім отримується шляхом простого обчислення різниці кількостей, отриманих за період затримки, і розмір вікна плюс період затримки.

Оцінка ризику визначається шляхом обчислення частки шахрайських транзакцій для кожного розміру вікна (або 0, якщо в даному вікні не відбулося жодної транзакції). На додаток до оцінки ризику, функція також повертає кількість транзакцій для кожного розміру вікна. Це призводить до додавання шести нових ознак: ризик і кількість транзакцій для трьох розмірів вікна.

```
def get_count_risk_rolling_window(terminal_transactions, delay_period=7, windows_size_in_days=[1,7,30], feature="TERMINAL_ID"):

    terminal_transactions=terminal_transactions.sort_values('TX_DATETIME')

    terminal_transactions.index=terminal_transactions.TX_DATETIME

    NB_FRAUD_DELAY=terminal_transactions['TX_FRAUD'].rolling(str(delay_period)+'d').sum()
    NB_TX_DELAY=terminal_transactions['TX_FRAUD'].rolling(str(delay_period)+'d').count()

    for window_size in windows_size_in_days:

        NB_FRAUD_DELAY_WINDOW=terminal_transactions['TX_FRAUD'].rolling(str(delay_period+window_size)+'d').sum()
        NB_TX_DELAY_WINDOW=terminal_transactions['TX_FRAUD'].rolling(str(delay_period+window_size)+'d').count()

        NB_FRAUD_WINDOW=NB_FRAUD_DELAY_WINDOW-NB_FRAUD_DELAY
        NB_TX_WINDOW=NB_TX_DELAY_WINDOW-NB_TX_DELAY

        RISK_WINDOW=NB_FRAUD_WINDOW/NB_TX_WINDOW

        terminal_transactions[feature+'_NB_TX_'+str(window_size)+'DAY_WINDOW']=list(NB_TX_WINDOW)
        terminal_transactions[feature+'_RISK_'+str(window_size)+'DAY_WINDOW']=list(RISK_WINDOW)

    terminal_transactions.index=terminal_transactions.TRANSACTION_ID

    # Replace NA values with 0 (all undefined risk scores where NB_TX_WINDOW is 0)
    terminal_transactions.fillna(0,inplace=True)

    return terminal_transactions
```

Рисунок 2.9 – Лістинг функції «get_count_risk_rolling_window»

Збережемо набір даних, розбитий на щоденні пакети, використовуючи формат pickle.

```
DIR_OUTPUT = "./simulated-data-transformed/"

if not os.path.exists(DIR_OUTPUT):
    os.makedirs(DIR_OUTPUT)

start_date = datetime.datetime.strptime("2023-01-01", "%Y-%m-%d")

for day in range(transactions_df.TX_TIME_DAYS.max()+1):

    transactions_day = transactions_df[transactions_df.TX_TIME_DAYS==day].sort_values('TX_TIME_SECONDS')

    date = start_date + datetime.timedelta(days=day)
    filename_output = date.strftime("%Y-%m-%d")+'.pkl'

    # Protocol=4 required for Google Colab
    transactions_day.to_pickle(DIR_OUTPUT+filename_output, protocol=4)
```

Таблиця 2.2 – Лістинг скрипта для збереження датасету

2.3 Створення моделі для виявлення шахрайства

2.3.1 Нейронна мережа прямого розповсюдження

Оскільки нейронні мережі є основою як ранніх, так і останніх досягнень штучного інтелекту, їх використання для виявлення шахрайства з кредитними картками не викликає подиву. Перші приклади застосування простих нейронних мереж прямого поширення для виявлення шахрайства відносяться до початку 90-х років. Природно, що в останніх дослідженнях FDS нейронні мережі часто зустрічаються в експериментальних бенчмарках, поряд з випадковими лісами, XGBoost або логістичною регресією.

В основі нейронної мережі прямого поширення лежить штучний нейрон - проста модель машинного навчання, яка складається з лінійної комбінації вхідних змінних з подальшим застосуванням функції активації (сигмоїд, ReLU, tanh тощо). Точніше, маючи список з n вхідних змінних x_i , вихід штучного нейрона h_i обчислюється наступним чином:

$$h = \sigma\left(\sum_{i=1}^n w_i * x_i\right),$$

де w_i – вагові коефіцієнти моделі.

Вся мережа складається з послідовності шарів, що містять нейрони, які отримують на вході вихідні значення попереднього шару (рис. 2.10).

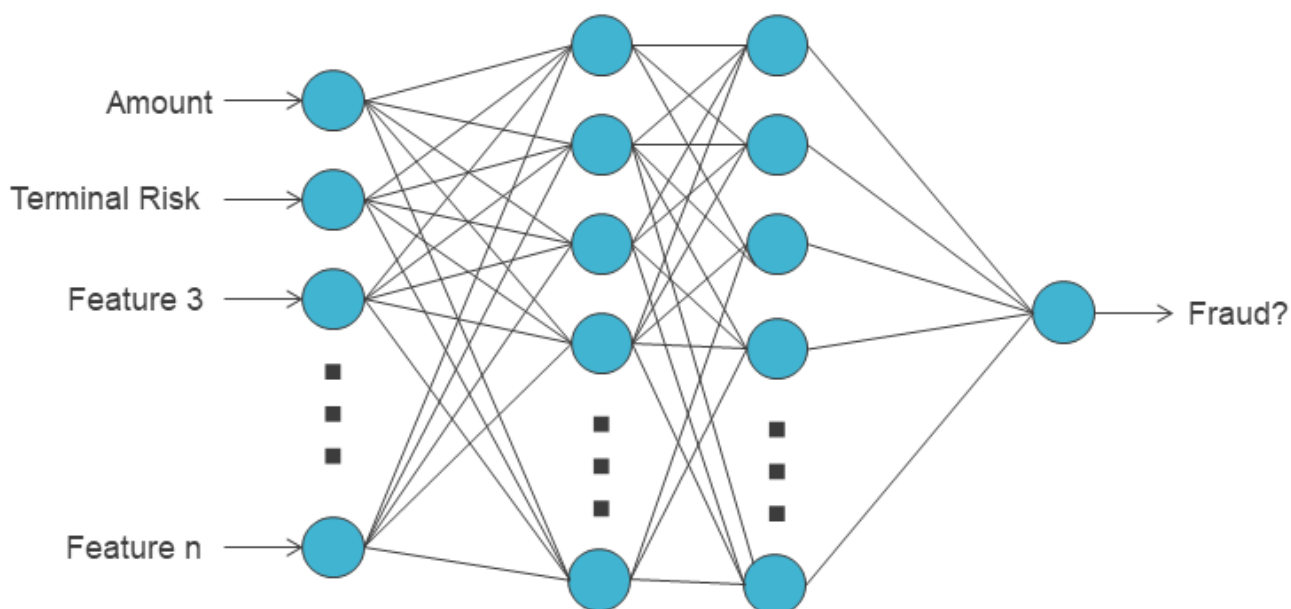


Рисунок 2.10 – Загальна схема нейронної мережі для виявлення шахрайства

У застосуванні до задачі виявлення шахрайства архітектура мережі виглядає наступним чином. На початку мережі нейрони приймають на вхід характеристики транзакції з кредитною картою, тобто ознаки, які були визначені раніше. В кінці мережа видає один нейрон, який має на меті представити ймовірність того, що вхідна транзакція є шахрайською.

Решта архітектури (інші шари), специфіка нейронів (функції активації) та інші гіперпараметри (оптимізація, обробка даних тощо) залишаються на вибір практиків.

Найпопулярнішим алгоритмом навчання для прямих архітектур є зворотне поширення. Ідея полягає в ітерації над усіма вибірками набору даних і виконанні двох ключових операцій:

- прямий прохід: встановлення значень ознак вибірки у вхідних нейронах і обчислення всіх шарів для отримання прогнозованого результату;
- зворотний прохід: обчислення функції вартості, тобто розбіжності між прогнозом і очікуваним вихідним результатом, і спроба мінімізувати її за допомогою оптимізатора (наприклад, градієнтного спуску) шляхом оновлення ваг шар за шаром, від виходу до входу.

2.3.2 Стратегія розподілу на тренувальний та тестовий набір

Найбільш простий підхід до навчання та оцінки моделі прогнозування для системи виявлення шахрайства показано на рис. 1. Для наочності ілюстрації потік даних про транзакції дискретизовано на блоки. Блок відповідає часовому періоду (наприклад, одному тижню або одному дню). Ми припускаємо, що модель прогнозування повинна оновлюватися для кожного нового блоку.

Блоки наразі відповідають одному тижню даних про транзакції. Один блок даних використовувався для навчання, а інший - для тестування. Тижнева затримка також використовувалася для розділення навчальних і тестових блоків.

У більш загальному випадку, маючи набір блоків минулих транзакцій, зібраних до поточного моменту часу метою моделі прогнозування є надання прогнозів щодо транзакцій, які відбудуться в наступному блоці (майбутні транзакції, які також називаються тестовими транзакціями, виділені жовтим кольором). Набір блоків, які використовуються для навчання, є навчальними блоками (темно-синім кольором). Через затримку в отриманні міток транзакцій, останні транзакції зазвичай не можуть бути використані для навчання (період затримки, світло-блакитний колір) і видаляються. Старіші блоки також зазвичай відкидаються через дрейф концепції (виділено блакитним кольором, також видаляються). Позначимо довжини періодів навчання, затримки та тестування через δ_{train} , δ_{delay} та δ_{test} відповідно.

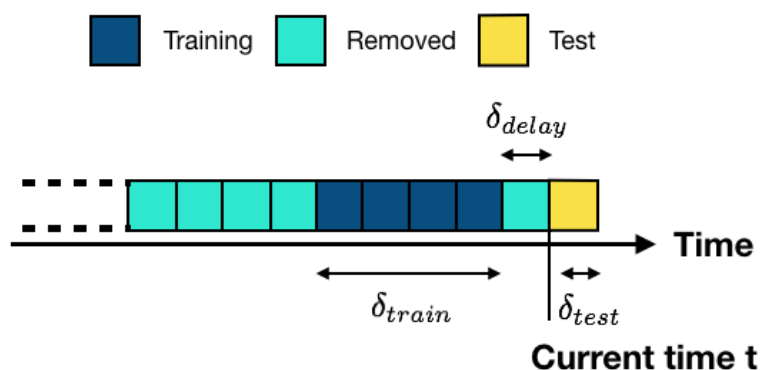


Рисунок 2.11 – Схема розподілу на навчальний на тестувальний набори

Набір для валідації вибирається випадково з тренувального датасета у розмірі 20% від загальної кількості транзакцій.

2.3.3 Завантаження даних

Для навчання нейронної мережі використовуватиметься 150 днів, такий же період використовувався і для перевірки прогнозів.

Навчання мережі буде базуватися на трансформованих модельних даних (simulated-data-transformed/data/).

```
DIR_INPUT="./Documents/simulated-data-transformed/"

BEGIN_DATE = "2023-01-01"
END_DATE = "2023-12-31"

print("Load files")
time transactions_df=read_from_files(DIR_INPUT, BEGIN_DATE, END_DATE)
print("{0} transactions loaded, containing {1} fraudulent transactions".format(len(transactions_df),transactions_df.TX_FRAUD.sum()))

output_feature="TX_FRAUD"

input_features=['TX_AMOUNT', 'TX_DURING_WEEKEND', 'TX_DURING_NIGHT', 'CUSTOMER_ID_NB_TX_1DAY_WINDOW',
'CUSTOMER_ID_AVG_AMOUNT_1DAY_WINDOW', 'CUSTOMER_ID_NB_TX_7DAY_WINDOW',
'CUSTOMER_ID_AVG_AMOUNT_7DAY_WINDOW', 'CUSTOMER_ID_NB_TX_30DAY_WINDOW',
'CUSTOMER_ID_AVG_AMOUNT_30DAY_WINDOW', 'TERMINAL_ID_NB_TX_1DAY_WINDOW',
'TERMINAL_ID_RISK_1DAY_WINDOW', 'TERMINAL_ID_NB_TX_7DAY_WINDOW',
'TERMINAL_ID_RISK_7DAY_WINDOW', 'TERMINAL_ID_NB_TX_30DAY_WINDOW',
'TERMINAL_ID_RISK_30DAY_WINDOW']
```

Рисунок 2.12 – Лістинг скрипту для завантаження даних

```

start_date_training = datetime.datetime.strptime("2023-01-01", "%Y-%m-%d")
delta_train=150
delta_delay=7
delta_test=150

(train_df, test_df)=get_train_test_set(transactions_df,start_date_training,
                                     delta_train=delta_train,delta_delay=delta_delay,delta_test=delta_test)

(train_df, test_df)=scaleData(train_df,test_df,input_features)

```

Рисунок 2.13 – Встановлення часових проміжків та масштабування даних

2.3.4 Загальний пайплайн бібліотеки Keras для побудови нейронних мереж

У Keras, який є високорівневим інтерфейсом для роботи з нейронними мережами на базі TensorFlow, створення моделей зазвичай включає в себе визначення та компіляцію моделі, а також тренування та оцінку. Основні етапи створення та використання моделі нейронної мережі в Keras виглядають наступним чином:

- визначення моделі. Можливо використовувати клас Sequential для лінійної послідовності шарів або створюйте модель із використанням функціонального API для більш складних архітектур;
- компіляція моделі. Визначення оптимізатору, функції втрат та метрики для оцінки моделі під час тренування;
- підготовка даних. Забезпечення відповідних даних для тренування та валідації;
- тренування моделі. Використання методу fit для тренування моделі на навчальних даних:
- оцінка та передбачення. Використання методу evaluate для оцінки моделі на тестових даних та predict для здійснення передбачень.

2.3.5 Основні налаштування моделі

Після визначення пайплайна наступним кроком є визначення гіперпараметрів, що є ітераційним процесом.

Як зазначалось, ідея полягає у визначенні декількох повністю з'єднаних шарів. Останнім йде вихідний шар, який має один вихідний нейрон, що відповідає мітці (шахрайство або справжність, представлений 1 і 0). В результаті серії експериментів була обрана архітектура, показана на рис. 2.14

У минулому сигмоїдна функція активації була основним вибором для всіх функцій активації у всіх шарах нейронної мережі. Сьогодні кращим вибором є ReLU (або такі варіанти, як eLU, leaky ReLU), принаймні для проміжних нейронів. Емпірично доведено, що це кращий вибір для оптимізації та швидкості. Для вихідних нейронів вибір залежить від діапазону або очікуваного розподілу вихідного значення, яке потрібно передбачити.

Для даної нейронної мережі для виявлення шахрайства було використано активацію ReLU для прихованого шару та активацію Sigmoid для вихідного шару. Перша є основним вибором для проміжних шарів у глибокому навчанні. Другий тип є основним вибором для вихідних нейронів у задачах бінарної класифікації, оскільки він виводить значення між 0 і 1, які можна інтерпретувати як ймовірності.

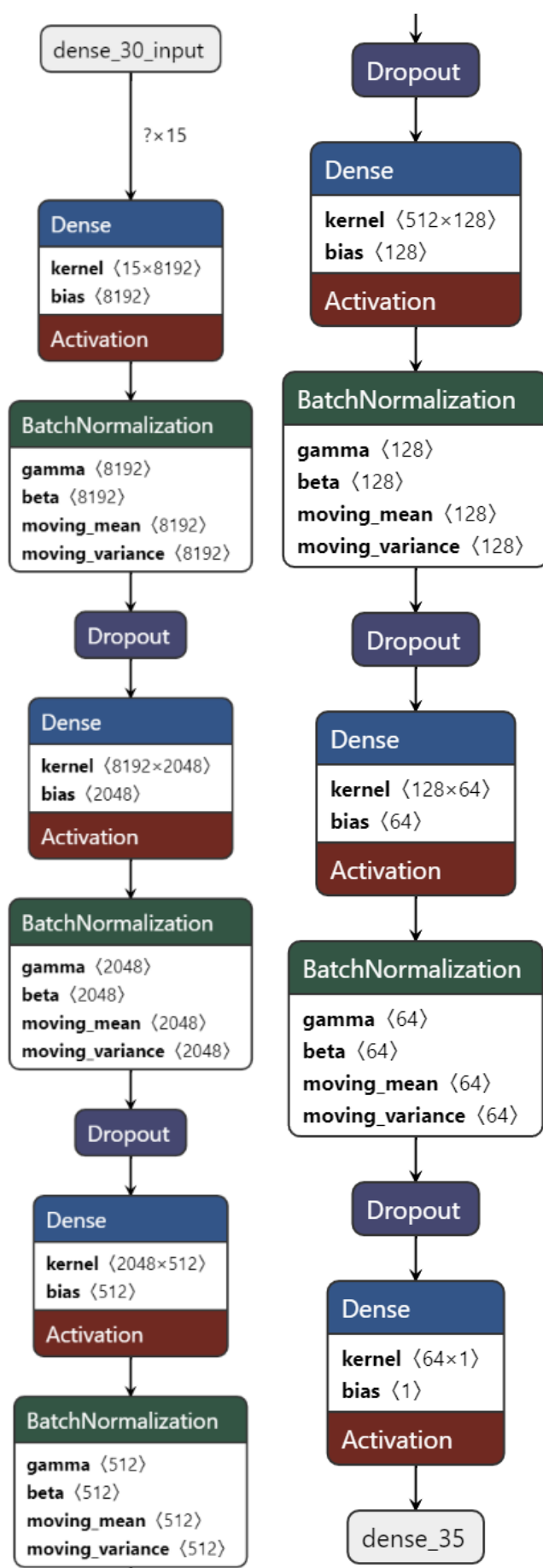


Рисунок 2.14 – Архітектура нейронної мережі

Оптимізація лежить в основі навчання нейронних мереж. Наведена вище нейронна мережа призначена для виведення одного значення між 0 і 1. Мета полягає в тому, щоб це значення було якомога ближче до 1 (або 0) для вхідних даних, що описують шахрайську (або справжню) транзакцію.

На практиці ця мета формулюється у вигляді оптимізаційної задачі, яка спрямована на мінімізацію або максимізацію певної функції витрат/втрат. Роль функції втрат полягає саме в тому, щоб виміряти розбіжність між прогнозованим значенням і очікуваним значенням (0 або 1). У Keras є багато функцій втрат (середня квадратична похибка, перехресна ентропія, KL-дивергенція, петльові втрати, середня абсолютна похибка), і кожна з них слугує певній меті. Тут ми зосередимося лише на бінарній перехресній ентропії, оскільки вона є найбільш релевантною функцією втрат для задач бінарної класифікації, таких як виявлення шахрайства. Вона визначається наступним чином:

$$BCE(y, p) = -(y * \log(p) + (1 - y) * \log(1 - p)),$$

де y – відомою істиною (в $\{0,1\}$),

p – прогнозований результат (в $[0,1]$).

Таким чином задача оптимізації визначається наступним чином: мінімізувати загальну/середню бінарну перехресну ентропію моделі на всіх вибірках з навчального набору даних. Отже, навчання моделі полягає в застосуванні алгоритму оптимізації (зворотного поширення) для чисельного розв'язання оптимізаційної задачі.

Алгоритм оптимізації або оптимізатор може бути стандартним стохастичним градієнтним спуском з постійною швидкістю навчання (SGD) або з адаптивною швидкістю навчання (Adagrad, Adam тощо...). Декілька гіперпараметрів оптимізації (швидкість навчання, імпульс, розмір пакету та ін.) можна налаштовувати. Вибір правильного оптимізатора та гіперпараметрів впливає на швидкість збіжності та якість досягнутого оптимуму. Для нейронної мережі було обрано оптимізатор Adam, який описується наступним чином:

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} * \hat{m}_t,$$

де

$$\begin{aligned} m_t &= \beta_1 * m_{t-1} + (1 - \beta_1) * \nabla w_t, \\ v_t &= \beta_2 * v_{t-1} + (1 - \beta_2) * (\nabla w_t)^2, \\ \hat{m}_t &= \frac{m_t}{1 - \beta_1^t}, \\ \hat{v}_t &= \frac{v_t}{1 - \beta_2^t}. \end{aligned}$$

Розглянемо цикл навчання для нейронної мережі. Спочатку виконаємо кілька ітерацій над навчальним генератором (кожна ітерація називається епохою). Під час кожної ітерації генератор видає послідовність навчальних пакетів і виконує прохід вперед для отримання прогнозів моделі. Потім обчислюється критерій між прогнозами та істиною, і, нарешті, виконується прохід назад для оновлення моделі за допомогою оптимізатора. Після ряду експериментів було обрано 25 епох.

Нейронні мережі, як відомо, є дуже експерсивними моделями. Насправді, універсальна теорема про апроксимацію показує, що при достатній кількості нейронів/шарів можна змодельовати будь-яку функцію за допомогою нейронної мережі. Тому очікується, що складна нейронна мережа буде здатна майже ідеально відповідати навчальним даним. Але кінцевою метою є отримання моделі, яка добре узагальнює невідомі дані (наприклад, валідаційний набір). Дивлячись на валідаційні помилки, часто можна побачити, що вони спочатку зменшуються (з коливаннями) і досягають оптимуму близько 0,019, а потім перестають зменшуватися (або навіть починають збільшуватися). Це явище називається перенавчанням.

Багато аспектів можна покращити під час тренувань. Хоча під час навчання неможливо виміряти продуктивність на остаточному тестовому наборі, можна покластися на перевірочний набір і спробувати зупинити навчання до того, як відбудеться перенавчання. Також можна змінити алгоритм та параметри оптимізації, щоб пришвидшити навчання та досягти кращого оптимуму.

В моделі Реалізовано стратегію ранньої зупинки. Ідея полягає в тому, щоб виявити перенавчання, тобто коли помилка валідації починає зростати, і зупинити процес навчання. Іноді помилка валідації може збільшуватися в певний момент часу, але потім знову зменшуватися. З цієї причини важливо також враховувати параметр «терпіння», тобто кількість ітерацій, на яку процес навчання чекає, щоб переконатися, що помилка точно зростає.

Класичний спосіб покращити узагальнення та досягти кращої валідації - це регуляризація моделі. Грубо кажучи, регуляризація полягає в обмеженні виразності моделі з метою зменшення перенавчання.

Ще один поширений прийом – регуляризація моделі машинного навчання, що є обмеженням простору її параметрів, наприклад, норми, шляхом додавання члена в оптимізаційну задачу. Крім того, щоб мінімізувати розбіжність між істинним станом і прогнозом, інтегрування члена з нормою $L1$ (або $L2$) у втратах призведе до розрідженості параметрів (тобто обмежить амплітуду параметрів). Простір початкових розв'язків, як правило, повний еквівалентних розв'язків (наприклад, при лінійній активації, ділення всіх вхідних ваг нейрона на 2 і множення всіх його вихідних ваг на 2 призводить до еквівалентної моделі), тому обмеження, що накладаються регуляризацією, не тільки обмежують перенавчання, але й скорочують час пошуку і можуть допомогти в оптимізації. Нарешті, вибір рішення з мінімальною нормою відповідає принципу "За інших рівних умов найпростіше рішення, як правило, є найкращим", науковому принципу, який часто називають "бритвою Оккама".

На противагу додаванню доданків втрат, існує техніка регуляризації, спеціально розроблена для нейронних мереж, яка називається виключення (dropout). Виключення полягає у випадковому вилученні деяких нейронів з мережі на кожному кроці навчання. Точніше, фіксується параметр виключення $p \in [0,1]$, і для кожного міні-пакету для кожного нейрона виконується підкидання монети з ймовірністю p . Якщо результат позитивний, ваги нейрона тимчасово встановлюються на нуль (так, що відсіяний нейрон не враховується під час прямого та зворотного проходів). Це еквівалентно навчанню випадкової

підмережі в кожному міні-пакеті, і можна довести, що це має ефект L2-регуляризації на конкретних архітектурах.

В розробленій мережі параметр відсікання становив 0.2

XGBoost і випадкові ліси вивчають розбиття на окремі ознаки і тому є стійкими до масштабу і розподілу значень. Навпаки, в нейронній мережі кожен нейрон першого шару вивчає лінійну комбінацію всіх ознак. Тому нейрони легше навчати, коли всі ознаки мають однаковий діапазон і нормально розподілені. Першу властивість можна легко реалізувати, застосувавши до ознак `min-max` або стандартне масштабування. Що стосується другої властивості, то вона залежить від початкового розподілу ознак. Деякі з них не є нормально розподіленими і мають нелінійні шкали (наприклад, сума): збільшення суми на 5 доларів не повинно мати однаковий ефект, якщо початкова точка дорівнює 5 доларам або 100 доларам. Виявляється, що застосування лог-функції до таких ознак може зробити їх розподіл трохи більш нормальним, що полегшує навчання нейронних мереж прямого поширення.

Також додатково використовується параметр `class_weight` у методі `fit`, який призначений для врахування дисбалансу класів під час навчання моделі. Це особливо актуально в задачах класифікації, де один або декілька класів можуть бути представлені значно меншою кількістю прикладів, ніж інші класи.

При використанні `class_weight` визначаються ваги для кожного класу. Ці ваги впливають на те, як модель "сприймає" важливість різних класів під час навчання. Збільшення ваги класу призводить до того, що помилки на прикладах цього класу матимуть більший вплив на загальну функцію втрат, яку мінімізує модель. Це може допомогти збалансувати вплив часто зустрічених класів та тих, що зустрічаються рідко.

```

random.seed(20)
from sklearn.preprocessing import StandardScaler

print(np.unique(y_train))
from sklearn.utils.class_weight import compute_class_weight
class_weights = compute_class_weight('balanced', classes=np.unique(y_train), y=np.ravel(y_train))
class_weight_dict = dict(zip(np.unique(y_train), class_weights))

model = Sequential()
model.add(Dense(8192, activation='relu', input_dim=len(input_features)))
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(Dense(2048, activation='relu',))
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu', ))
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(Dense(128, activation='relu', ))
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(Dense(64, activation='relu', ))
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
early_stopping = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)

# Train the model
model.fit(X_train, y_train, epochs=25, batch_size=64, validation_split=0.2, class_weight=class_weight_dict)

# Evaluate the model on the test set
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Test Loss: {loss}, Test Accuracy: {accuracy}')

```

Рисунок 2.15 – Лістинг скрипту для навчання та оцінки моделі.

2.3.6 Результати роботи моделі та напрямки її подальшого покращення

Навчання виконувалось протягом 11 годин. В результаті були отримані такі результати:

Test Loss: 0.24832622706890106, Test Accuracy: 0.9839848279953003

Враховуючи нерівномірний розподіл класів, більш наочною метрикою стають влучність та повнота:

Precision: 0.2018436719800269, Recall: 0.717896174863388

При цьому матриця невідповідностей виглядає наступним чином:

[[205680 4156]

[413 1051]]

Площа під кривою ROC-AUC (Receiver Operating Characteristic - Area Under Curve) є метрикою, яка використовується для оцінки якості бінарної класифікації моделі машинного навчання. ROC-AUC вимірює здатність моделі розрізняти між позитивними і негативними класами при різних порогах класифікації. Ця метрика надає інформацію про те, наскільки добре модель відрізняє справжні позитивні випадки від справжніх негативних випадків.

Якщо ROC-AUC дорівнює 0.5, це вказує на випадковий класифікатор, який не відрізняється від випадкового вибору класів. Якщо ROC-AUC менше 0.5, це вказує на погану модель, яка робить гірше, ніж випадковий вибір. Якщо ROC-AUC більше 0.5, то модель вважається кращою, ніж випадковий вибір, і значення більше 0.5 вказує на її здатність до класифікації.

Значення ROC-AUC для нашої моделі дорівнює 0,85.

Таким чином, модель показує результати на прийнятному рівні. Тим не менш, є можливості для підвищення якості класифікатору. Подальшими напрямками розвитку моделі можуть стати:

- ансамблеве навчання;
- пошук по сітці для визначення оптимальних гіперпараметрів;
- введення додаткових ознак.

ВИСНОВКИ

У цій роботі було розглянуто і розроблено нейромережеву модель для виявлення шахрайства. Даний проект є важливим у сучасному світі, де зростає кількість шахраїв та шахрайських схем у фінансовій сфері та інших галузях.

У процесі розробки було проведено наступні кроки.

1. Збір та підготовка даних. Були згенеровані та підготовлені дані для тренування та тестування моделі. Це включало генерацію початкових даних та подальше виокремлення з них додаткових ознак. Також проведено розподіл датасету на навчальні та тестові вибірки.

2. Вибір архітектури моделі. Була вибрана архітектура нейромережі, яка включала в себе вхідний шар, 6 прихованих шарів з функціями активації ReLU та вихідний шар з сигмоїдною функцією активації.

3. Навчання моделі. Модель була навчена на навчальних даних з використанням алгоритму оптимізації adam та бінарної перехресної ентропії. Процес навчання включав в себе звіряння прогнозів моделі з фактичними значеннями та підгонку параметрів для досягнення кращої точності.

4. Оцінка та тестування моделі: Після навчання модель була протестована на тестових даних для оцінки її ефективності. Були виміряні метрики, такі як точність, влучність та повнота, щоб визначити, наскільки добре модель виявляє шахрайство.

5. Покращення та налаштування: Це включало в себе зміну гіперпараметрів та додавання додаткових функцій для покращення виявлення шахрайства.

У результаті роботи була створена нейромережева модель, яка може бути використана для виявлення шахрайства з кредитними картами. Модель показує достатньо добрі результати, але існують можливості для її подальшого покращення.

Важливо пам'ятати, що робота з виявленням шахрайства є постійною задачею, оскільки шахраї постійно шукають нові способи обходу захисту. Тому модель слід постійно вдосконалювати та оновлювати для забезпечення ефективного виявлення шахрайства в майбутньому та донавчати її в тому числі і на реальних даних.

ПЕРЕЛІК ПОСИЛАНЬ

1. Pundkar S. N. Credit card fraud detection methods: a review / S. N. Pundkar, M. Zubei // E3S Web of Conferences. — 2023. — Vol. 453. — P. 01015.
2. Adebayo O. S. Comparative review of credit card fraud detection using machine learning and concept drift techniques / O. S. Adebayo, T. A. Favour-Bethy, O. Otasowie, O. A. Okunola // International Journal of Computer Science and Mobile Computing. — 2023. — Vol. 12, No. 7. — P. 24–48.
3. Ashtiani M. N. Intelligent fraud detection in financial statements using machine learning and data mining: a systematic literature review / M. N. Ashtiani, B. Raahemi // IEEE Access. — 2022. — Vol. 10. — P. 72504–72525.
4. Throckmorton C. S. Financial fraud detection using vocal, linguistic and financial cues / C. S. Throckmorton, W. J. Mayew, M. Venkatachalam, L. M. Collins // Decision Support Systems. — 2015. — Vol. 74. — P. 78–87.
5. Fanning K. Detection of management fraud: a neural network approach / K. Fanning, K. O. Cogger, R. Srivastava // Intelligent Systems in Accounting, Finance and Management. — 1995. — Vol. 4, No. 2. — P. 113–126.
6. Gupta R. Prevention and detection of financial statement fraud – an implementation of data mining framework / R. Gupta, N. Singh // International Journal of Advanced Computer Science and Applications. — 2012. — Vol. 3, No. 8.
7. Humpherys S. L. Identification of fraudulent financial statements using linguistic credibility analysis / S. L. Humpherys, K. C. Moffitt, M. B. Burns, [et al.] // Decision Support Systems. — 2011. — Vol. 50, No. 3. — P. 585–594.
8. Kirkos E. Data mining techniques for the detection of fraudulent financial statements / E. Kirkos, C. Spathis, Y. Manolopoulos // Expert Systems with Applications. — 2007. — Vol. 32, No. 4. — P. 995–1003.
9. Chen J. I.-Z. Deep convolution neural network model for credit-card fraud detection and alert / J. I.-Z. Chen, K.-L. Lai // Journal of Artificial Intelligence and Capsule Networks. — 2021. — Vol. 3, No. 2. — P. 101–112.

10. Cheng D. Graph neural network for fraud detection via spatial-temporal attention / D. Cheng, X. Wang, Y. Zhang, L. Zhang // *IEEE Transactions on Knowledge and Data Engineering*. — 2022. — Vol. 34, No. 8. — P. 3800–3813.
11. Chung J. Credit card fraud detection: an improved strategy for high recall using knn, lda, and linear regression / J. Chung, K. Lee // *Sensors*. — 2023. — Vol. 23, No. 18. — P. 7788.
12. Li X. How to protect investors? a ga-based dwd approach for financial statement fraud detection / X. Li, W. Xu, X. Tian. — San Diego, CA, USA : IEEE, 2014.
13. Itoo F. Comparison and analysis of logistic regression, naïve bayes and knn machine learning algorithms for credit card fraud detection / F. Itoo, Meenakshi, S. Singh // *International Journal of Information Technology*. — 2021. — Vol. 13, No. 4. — P. 1503–1511.
14. Forough J. Ensemble of deep sequential models for credit card fraud detection / J. Forough, S. Momtazi // *Applied Soft Computing*. — 2021. — Vol. 99. — P. 106883.
15. Assis C. A. S. A genetic programming approach for fraud detection in electronic transactions / C. A. S. Assis, A. C. M. Pereira, M. A. Pereira, E. G. Carrano. — Orlando, FL, USA : IEEE, 2014.
16. Hajek P. Interpretable fuzzy rule-based systems for detecting financial statement fraud / P. Hajek // *Artificial Intelligence Applications and Innovations* / J. MacIntyre, I. Maglogiannis, L. Iliadis, E. Pimenidis. — Cham : Springer International Publishing, 2019. — P. 425–436.
17. Malekolkalami T. Application of data mining to detect accounting fraud in information systems / T. Malekolkalami, K. Khodabakhshi Parijani, M. Alifari // *International Journal of Knowledge Processing Studies*. — 2023. — Vol. 3, No. 4.
18. Glancy F. H. A computational model for financial reporting fraud detection / F. H. Glancy, S. B. Yadav // *Decision Support Systems*. — 2011. — Vol. 50, No. 3. — P. 595–601.

19. Lokanan M. Detecting anomalies in financial statements using machine learning algorithm: the case of vietnamese listed firms / M. Lokanan, V. Tran, N. H. Vuong // Asian Journal of Accounting Research. — 2019. — Vol. 4, No. 2. — P. 181–201.
20. Kulatilleke G. K. Challenges and complexities in machine learning based credit card fraud detection / G. K. Kulatilleke // 2022.
21. Le Borgne Y.-A. Reproducible machine learning for credit card fraud detection - practical handbook / Y.-A. Le Borgne, W. Siblini, B. Lebichot, G. Bontempi. — Universite Libre de Bruxelles, 2022.