Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Вдосконалення алгоритму Isolation Forest шляхом введення змішаних сплітів для задачі виявлення інсайдерських загроз у CRM-системах

Виконав: здобувач 2-го курсу,
групи 2КІТС-24м
спеціальності 125– Кібербезпека
та захист інформації
Освітня програма – Кібербезпека
інформаційних технологій та систем
(шифр і назва напряму підготовки, спеціальності)

Вахній Д.Д.
(прізвище та ініціали)

Керівник:
Салієва О.В.
(прізвище та ініціали)
«11» грудня 2025 р.

Опонент:
Савицька Л.А.
(прізвище та ініціали)
«____» _____ 2025 р.

Допущено до захисту
Голова секції УБ кафедри МБІС

Юрій ЯРЕМЧУК
"11" грудня 2025 р.

Вінниця ВНТУ - 2025 рік

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека та захист інформації
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС
Юрій ЯРЕМЧУК
"24" вересня 2025 р.

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу студенту
Вахній Денис Дмитрович
(прізвище, ім`я, по-батькові)

1. Тема роботи Вдосконалення алгоритму Isolation Forest шляхом введення змішаних сплітів у задачі виявлення інсайдерських загроз у CRM-системах
Керівник роботи д.ф., доц. кафедри МБІС Салієва О.В.
затверджені наказом вищого навчального закладу від "24" вересня 2025 року № 313

2. Строк подання студентом роботи за тиждень до захисту.

3. Вихідні дані до роботи:
Датасет: Brazilian E-Commerce Public Dataset by Olist (обсяг > 100 000 записів);
Типи даних: гетерогенні (числові та категоріальні) без попереднього кодування One-Hot;
Мова програмування: Python 3.9+;
Ключові бібліотеки: Pandas, NumPy, Scikit-learn;
Середовище розробки: Jupyter Notebook / VS Code.

4. Зміст текстової частини:
аналіз проблеми виявлення інсайдерських загроз у CRM-системах та обмежень існуючих методів;
теоретичне обґрунтування та розробка модифікації алгоритму Isolation Forest (механізм Mixed Splits);
програмна реалізація інформаційної технології та експериментальне дослідження ефективності методу (порівняння ROC-AUC);
економічне обґрунтування доцільності розробки та впровадження системи.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень)
Схема алгоритму роботи методу Mixed Isolation Forest;Діаграма класів програмного комплексу;Графіки порівняння ефективності (ROC-криві) зі стандартним методом;Карта розподілу виявлених аномалій;Презентаційні матеріали (слайди).

6. Консультанти

Розділ	Прізвище, ініціали та посада консультанта	завдання видав	пр
Основна частина			
I	Салієва О. В. д.ф., доц. каф. МБІС		
II	Салієва О. В. д.ф., доц. каф. МБІС		
III	Салієва О. В. д.ф., доц. каф. МБІС		
Економічна частина	Ратушняк О. Г. к.т.н., доц. каф. ЕПВМ		

7.Дата видачі завдання 24 вересня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи		Пр
1	Визначення напрямку магістерської роботи, формулювання теми	24.09.2025	30.09.2025	
2	Аналіз предметної області обраної теми	01.10.2025	10.10.2025	
3	Розробка алгоритму роботи	11.10.2025	3.11.2025	
4	Написання магістерської роботи на основі розробленої теми	04.11.2025	10.11.2025	
5	Розробка економічної частини та оцінка ефективності	11.11.2025	15.11.2025	
6	Написання пояснювальної записки та оформлення роботи	16.11.2025	22.11.2025	
7	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	23.11.2025	25.11.2025	
8	Захист магістерської кваліфікаційної роботи	26.11.2025	5.12.2025	

Студент (підпис) Вахній Д. Д.

Керівник роботи (підпис) Салієва О. В.

АНОТАЦІЯ

УДК 004.056.5:004.8

Вахній Д. Д. Вдосконалення алгоритму Isolation Forest шляхом введення змішаних сплітів у задачі виявлення інсайдерських загроз у CRM-системах. Магістерська кваліфікаційна робота зі спеціальності 125 — Кібербезпека та захист інформації, освітня програма — Кібербезпека інформаційних технологій та систем. Вінниця: ВНТУ, 2025. 95 с.

На укр. мові. Бібліогр.: 52 назви; рис.: 18; табл. 12.

У магістерській кваліфікаційній роботі розроблено програмний модуль для виявлення інсайдерських загроз у корпоративних CRM-системах на основі вдосконаленого методу машинного навчання.

У роботі проаналізовано природу інсайдерських загроз та недоліки існуючих методів їх виявлення при роботі з гетерогенними даними. Теоретично обґрунтовано та формалізовано механізм «змішаних сплітів» (Mixed Splits) для алгоритму Isolation Forest, який дозволяє обробляти числові та категоріальні дані без використання ресурсомісткого кодування.

Здійснено програмну реалізацію методу мовою Python та проведено експериментальне дослідження на реальному наборі даних E-Commerce. Результати підтвердили підвищення точності детекції та зменшення використання оперативної пам’яті порівняно з класичними підходами.

В економічній частині проведено функціонально-вартісний аналіз розробки, розраховано кошторис витрат та обґрунтовано інвестиційну привабливість впровадження системи захисту.

Графічна частина складається з презентаційних матеріалів, що відображають результати моделювання та архітектуру системи.

Ключові слова: інсайдерська загроза, CRM-система, машинне навчання, Isolation Forest, змішані дані, детекція аномалій, інформаційна безпека.

ABSTRACT

Vakhniy D. D. Improving the Isolation Forest algorithm by introducing mixed splits in the task of detecting insider threats in CRM systems. Master's qualification thesis in specialty 125 — Cybersecurity and Information Protection, educational program — Cybersecurity of information technologies and systems. Vinnytsia: VNTU, 2025. 95 p.

In Ukrainian. Bibliogr.: 52 titles; fig.: 18; tabl. 12.

In the master's qualification thesis, a software module for detecting insider threats in corporate CRM systems based on an improved machine learning method has been developed.

The thesis analyzes the nature of insider threats and the shortcomings of existing detection methods when working with heterogeneous data. The "Mixed Splits" mechanism for the Isolation Forest algorithm is theoretically grounded and formalized, which allows processing numerical and categorical data without using resource-intensive encoding.

Software implementation of the method in Python has been carried out, and experimental research on a real E-Commerce dataset has been conducted. The results confirmed an increase in detection accuracy and a reduction in operational memory usage compared to classical approaches.

In the economic part, a functional-cost analysis of the development was conducted, a cost estimate was calculated, and the investment attractiveness of implementing the protection system was justified.

The graphic part consists of presentation materials reflecting the modeling results and system architecture.

Keywords: insider threat, CRM system, machine learning, Isolation Forest, mixed data, anomaly detection, information security.

ЗМІСТ

ВСТУП 9

1 АНАЛІЗ ПРОБЛЕМИ ВИЯВЛЕННЯ ІНСАЙДЕРСЬКИХ ЗАГРОЗ ТА ІСНУЮЧИХ МЕТОДІВ ЇХ ІДЕНТИФІКАЦІЇ 12

1.1 Класифікація та характеристика інсайдерських загроз в корпоративних інформаційних системах 12

1.2 Специфіка прояву інсайдерських загроз у CRM-системах 14

1.3 Огляд сучасних підходів до виявлення аномалій у даних змішаного типу 19

1.4 Алгоритм Isolation Forest як базовий метод виявлення аномалій 21

1.5 Висновки та постановка задач 22

2 ВДОСКОНАЛЕННЯ АЛГОРИТМУ ISOLATION FOREST ДЛЯ ЗАДАЧІ ВИЯВЛЕННЯ АНОМАЛІЙ У ЗМІШАНИХ ДАНИХ 24

2.1 Аналіз математичної моделі класичного алгоритму Isolation Forest та обмежень його застосування 24

2.2 Обґрунтування та формалізація механізму змішаних сплітів (Mixed Splits) 30

2.3 Алгоритм та реалізація методу IF-MS: структури даних та процедури побудови ансамблю 34

2.4 Аналіз обчислювальної складності та масштабованості алгоритму 40

2.5 Методологія верифікації ефективності та критерії оцінювання якості детекції 43

2.6 Висновки до розділу 47

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ МЕТОДУ 49

3.1 Обґрунтування вибору засобів програмної реалізації 49

3.2 Архітектура розробленого програмного комплексу 50

3.3 Дослідження вхідних даних CRM-системи (Exploratory Data Analysis) 51

3.4 Методика проведення експерименту та генерація аномалій 53
3.5 Результати експериментального дослідження 54
3.6 Висновки до розділу 57
4 ЕКОНОМІЧНА ЧАСТИНА 58
4.1 Оцінювання комерційного потенціалу розробки програмного забезпечення 58
4.2 Прогнозування витрат на виконання наукової роботи та впровадження її результатів 62
4.3 Прогнозування комерційних ефектів від реалізації результатів розробки 69
4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності 71
4.5 Висновки до розділу 73
ВИСНОВОК 75
ПЕРЕЛІК ПОСИЛАНЬ 77
ДОДАТКИ 81
Додаток А. Технічне завдання ..... **Помилка! Закладку не визначено.**
Додаток Б. Лістинг коду 86
Додаток Б. Ілюстративний матеріал 103
Додаток Г. Протокол перевірки на антиплагіат ..... **Помилка! Закладку не визначено.**

ВСТУП

Актуальність. У сучасній цифровій економіці системи управління взаємовідносинами з клієнтами (CRM) перетворилися на центральні сховища критично важливої корпоративної інформації. Вони акумулюють не лише контактні дані клієнтів, але й історію взаємодій, деталі угод, стратегічні плани продажів та іншу конфіденційну інформацію, що становить комерційну таємницю. Централізація таких цінних активів робить CRM-системи пріоритетною ціллю для кібератак, серед яких особливе місце посідають інсайдерські загрози.

Згідно з останніми звітами, інсайдерські загрози становлять значну частку від загальної кількості інцидентів безпеки. Так, у 2023 році близько 19% з понад 5200 проаналізованих витоків даних були спричинені діями внутрішнього персоналу. Вартість усунення наслідків таких інцидентів для організацій сягає в середньому 15.4 мільйона доларів на рік [1]. Ці дані свідчать про те, що загрози, які походять зсередини організації, є не менш, а часто і більш руйнівними, ніж зовнішні атаки. Класичні підходи до кібербезпеки, що зосереджені на захисті периметра (мережеві екрани, системи виявлення вторгнень), виявляються недостатньо ефективними для протидії інсайдерам, оскільки ці особи вже мають легітимний доступ до систем та даних [2]. Сутність інсайдерської загрози полягає не в несанкціонованому проникненні, а у зловживанні наданими повноваженнями.

Додатковим викликом є гетерогенна природа даних, що зберігаються та обробляються в CRM-системах. Логи активності користувачів, які є основним джерелом інформації для виявлення загроз, містять дані змішаного типу: числові (кількість переглянутих записів, тривалість сесії, сума угоди) та категоріальні (роль користувача, тип виконаної операції, IP-адреса, назва звіту). Більшість класичних алгоритмів виявлення аномалій розроблені для роботи з однорідними, переважно числовими даними, і їх застосування до змішаних наборів даних потребує складних процедур попередньої обробки, що часто призводить до втрати інформації та зниження точності.

Таким чином, актуальність даної роботи зумовлена гострою потребою у

розробці спеціалізованих методів виявлення інсайдерських загроз, які б враховували специфіку сучасних CRM-систем, зокрема, здатність ефективно аналізувати поведінку користувачів на основі гетерогенних даних.

Мета і задачі дослідження. Метою роботи є підвищення ефективності та точності виявлення інсайдерських загроз у CRM-системах шляхом розробки та теоретичного обґрунтування модифікованого алгоритму Isolation Forest, адаптованого для роботи з даними змішаного типу.

Задачами дослідження є:

– системний аналіз природи інсайдерських загроз, їх класифікації та специфіки прояву в середовищі CRM-систем;

– критичний огляд існуючих методів виявлення аномалій, з акцентом на алгоритмі Isolation Forest, його математичних засадах та обмеженнях;

– дослідження проблем, що виникають при обробці категоріальних даних стандартним алгоритмом Isolation Forest, та аналіз наслідків застосування поширених методів кодування;

– розробка теоретичних основ нового механізму розділення вузлів дерева, "змішаного спліту", що дозволяє коректно обробляти числові та категоріальні ознаки в рамках єдиної моделі;

– здійснення математичного опису вдосконаленого алгоритму та визначення його місця серед існуючих розширень Isolation Forest.

Об'єкт дослідження – процес виявлення аномальної активності користувачів в корпоративних CRM-системах.

Предмет дослідження – алгоритм виявлення аномалій Isolation Forest, методи його модифікації для обробки даних змішаного типу, та теоретичні моделі представлення поведінки користувачів на основі логів активності.

Наукова новизна. Вперше запропоновано та теоретично обґрунтовано механізм "змішаних сплітів" для алгоритму Isolation Forest, який, на відміну від існуючих підходів, що покладаються на попереднє перетворення даних (наприклад, one-hot encoding), дозволяє безпосередньо обробляти категоріальні та числові

ознаки в процесі побудови ізолюючих дерев. Це вирішує проблему зміщення вибору ознак та втрати інформації, пов'язаної з "прокляттям розмірності", що підвищує точність і надійність виявлення аномалій у гетерогенних наборах даних, характерних для CRM-систем.

Практична цінність. Практична цінність. Отримані результати дозволяють підвищити рівень інформаційної безпеки підприємств, що використовують CRM-системи, за рахунок впровадження ефективного інструменту виявлення інсайдерських загроз.

1 АНАЛІЗ ПРОБЛЕМИ ВИЯВЛЕННЯ ІНСАЙДЕРСЬКИХ ЗАГРОЗ ТА ІСНУЮЧИХ МЕТОДІВ ЇХ ІДЕНТИФІКАЦІЇ

У першому розділі магістерської дисертації здійснюється комплексний огляд проблематики захисту корпоративних інформаційних систем від внутрішніх порушників. В умовах сучасної цифрової економіки, де дані стають головним активом бізнесу, саме інсайдерські загрози становлять найбільшу небезпеку для конфіденційності та цілісності інформації, що зберігається в CRM-системах.

Основна увага в цьому розділі приділяється аналізу природи інсайдерських загроз, їх класифікації та специфіці прояву в середовищі систем управління взаємовідносинами з клієнтами. Розглядаються типові сценарії аномальної поведінки користувачів та виокремлюються ключові індикатори, які можуть свідчити про зловмисну діяльність або недбалість персоналу.

Крім того, у розділі проводиться критичний аналіз існуючих методів виявлення аномалій, зокрема статистичних підходів та методів машинного навчання. Особливий акцент зроблено на дослідженні алгоритму Isolation Forest, його перевагах у швидкодії та масштабованості, а також на виявленні його фундаментальних обмежень при роботі з гетерогенними даними, що і зумовлює необхідність розробки вдосконаленого методу.

1.1 Класифікація та характеристика інсайдерських загроз в корпоративних інформаційних системах

Інсайдерська загроза визначається як потенціал для особи, що має авторизований доступ до ресурсів організації, використати цей доступ у спосіб, що завдасть шкоди організації [3]. Ця шкода може бути навмисною або ненавмисною і стосуватися цілісності, конфіденційності та доступності даних, систем або персоналу [4]. Для систематичного аналізу та розробки методів протидії інсайдерські загрози класифікують за наміром та мотивацією суб'єкта. Загальноприйнятою є таксономія, що виділяє три основні категорії: навмисні

(зловмисні), ненавмисні (недбалі) та компрометовані інсайдери [2].

Навмисні (зловмисні) загрози (Malicious Threats). Цей тип загрози походить від інсайдерів, які свідомо та цілеспрямовано прагнуть завдати шкоди організації. Їхні дії можуть бути мотивовані різними факторами [2]:

– фінансова вигода. Продаж комерційних таємниць, клієнтських баз даних або іншої конфіденційної інформації конкурентам, кіберзлочинцям або іншим зацікавленим сторонам;

– помста. Дії, спрямовані на саботаж, знищення даних або порушення роботи систем, що є наслідком образи на керівництво, звільнення, відмови у підвищенні або інших конфліктів на робочому місці;

– корпоративне шпигунство. Цілеспрямований збір інформації на користь іншої організації або навіть іноземної держави;

– ідеологічні мотиви. Інсайдер може діяти, керуючись політичними, релігійними або соціальними переконаннями, що суперечать діяльності або цінностям компанії.

Ненавмисні (недбалі) загрози (Negligent Threats). Ця категорія є найпоширенішою і походить від співробітників, які завдають шкоди без злого умислу, через необережність, недостатню обізнаність у питаннях кібербезпеки або ігнорування встановлених політик безпеки [3]. Типові прояви включають:

– помилки конфігурації. Випадкове надання публічного доступу до приватних ресурсів, неправильне налаштування прав доступу;

– ставлення жертвою фішингу. Перехід за шкідливими посиланнями або відкриття заражених вкладень в електронних листах, що призводить до компрометації облікового запису;

– втрата корпоративних пристроїв. Втрата ноутбука або смартфона, що містить не зашифровані конфіденційні дані;

– небезпечне поводження з даними. Надсилання конфіденційної інформації на неправильну адресу електронної пошти, використання незахищених публічних Wi-Fi мереж для роботи з корпоративними даними.

Компрометовані інсайдери (Compromised Insiders). Цей тип загрози є гібридним. Він виникає, коли облікові дані легітимного користувача (логін та пароль) потрапляють до рук зовнішнього зловмисника внаслідок фішингової атаки, шкідливого програмного забезпечення або іншого методу злому [2]. Після цього зловмисник діє "зсередини" системи, використовуючи скомпрометований обліковий запис. Для систем моніторингу така активність виглядає як дії легітимного користувача, що робить її виявлення особливо складним.

Важливо розуміти, що межі між цими категоріями є розмитими. Наприклад, недбалий користувач, який став жертвою фішингу, перетворюється на компрометованого інсайдера, чиї дії (наприклад, масове завантаження даних) можуть бути ідентичними діям зловмисного інсайдера. З точки зору аналізу системних логів, намір, що стоїть за дією, не є безпосередньо спостережуваним. Спостерігається лише сама дія: логін, запит до бази даних, експорт звіту. Цей факт обґрунтовує необхідність застосування методів, що базуються не на заздалегідь визначених правилах (які намагаються вгадати намір), а на аналізі поведінкових патернів. Системи виявлення аномалій, такі як Isolation Forest, ідеально підходять для цього завдання, оскільки вони фокусуються на ідентифікації відхилень від встановленої "норми" поведінки, незалежно від причини цих відхилень.

1.2 Специфіка прояву інсайдерських загроз у CRM-системах

CRM-системи, такі як Salesforce, HubSpot або Zoho CRM, через свою архітектуру та призначення мають унікальні вектори атак для інсайдерів. Вони надають структурований доступ до найцінніших даних компанії, а складні моделі дозволів та широкі можливості експорту створюють сприятливі умови для зловживань [7]. Розглянемо конкретні сценарії аномальної поведінки для різних ролей користувачів, які можуть свідчити про інсайдерську загрозу.

Розглянемо можливі сценарії для менеджера з продажу:

– масовий експорт даних. Один з найпоширеніших сценаріїв – експорт великих обсягів даних перед звільненням з компанії. Менеджер може завантажити

всю клієнтську базу, історію угод або лідів для використання на новому місці роботи [7]. Аномалією тут є не сам факт експорту, а його обсяг та час (наприклад, в останній робочий день);

– доступ до чужих даних. Спроби перегляду або експорту звітів, угод чи контактів, що належать іншим менеджерам, відділам або географічним регіонам. Це може свідчити про спробу корпоративного шпигунства або підготовку до крадіжки клієнтів;

– незвичайна активність у неробочий час. Систематичні логіни та активна робота в системі пізно вночі або у вихідні дні, що не відповідає звичайному графіку роботи співробітника, може бути ознакою приховування нелегітимних дій [8];

– масове оновлення або видалення записів. Раптова зміна статусів великої кількості угод на "програні" або видалення контактної інформації може бути актом саботажу.

Розглянемо можливі сценарії для системного адміністратора:

– ескалація привілеїв. Системний адміністратор може спробувати надати собі або іншому користувачеві (наприклад, спільнику) права доступу до модулів системи (фінанси, кадри), до яких вони не повинні мати доступу згідно з політикою безпеки [7];

– вимкнення механізмів аудиту. Спроба відключити логування дій або змінити налаштування аудиту для приховування слідів своєї зловмисної діяльності. Це є серйозним індикатором загрози, оскільки свідчить про усвідомлення протиправності дій [10];

– створення "тіньових" облікових записів. Створення нових користувачів з високими привілеями, імена яких не відповідають реальним співробітникам, може використовуватися для створення "чорного ходу" в систему [6].

Ключовим аспектом при виявленні таких загроз є контекст. Дія, яка є абсолютно нормальною для одного користувача, може бути вкрай аномальною для іншого. Наприклад, експорт 10,000 записів є рутинною операцією для системного адміністратора, що виконує резервне копіювання, але є критичною аномалією для

молодшого менеджера з продажу. Це означає, що система виявлення загроз не може покладатися лише на аналіз окремих подій. Вона повинна будувати індивідуальні профілі "нормальної" поведінки для кожного користувача або, принаймні, для кожної ролі. Це вимагає використання алгоритмів, здатних навчатися на багатовимірних даних, що включають не тільки саму дію, але й атрибути користувача, який її виконує.

У таблиці 1.1 систематизовано приклади інсайдерських загроз у розрізі типів, ролей та індикаторів, що можуть бути виявлені в логах CRM-системи.

Таблиця 1.1 – Класифікація інсайдерських загроз та приклади їх прояву в CRM-системах

Тип загрози	**Роль користувача**	**Приклад сценарію аномальної поведінки**	**Ключові індикатори в логах**
Навмисна (зловмисна)	Менеджер з продажу	Масовий експорт клієнтської бази перед звільненням для передачі конкуренту.	EventType: ReportExport, RowsProcessed > 1000, Timestamp (близько до дати звільнення)
	Системний адміністратор	Створення прихованого облікового запису з адміністративними правами.	EventType: CreateUser, Profile: System Administrator, User (не відповідає штатному розпису)
	Маркетолог	Витік даних про майбутню маркетингову кампанію конкурентам.	EventType: ContentTransfer, Destination: External, доступ до файлів з обмеженим доступом

Продовження табл 1.1

Недбала (ненавмисна)	Менеджер з продажу	Випадкове надання публічного доступу до конфіденційного звіту по продажах.	EventType: SharingChange, AccessType: Public, ObjectType: Report
	Системний адміністратор	Помилкова конфігурація профілю безпеки, що надає надлишкові права групі користувачів.	EventType: ProfileUpdate, Permissions: ModifyAllData=true для неадміністративної ролі
	Маркетолог	Ненавмисне завантаження списку контактів для розсилки у незахищене хмарне сховище.	EventType: FileUpload, Destination: UnsanctionedApp, великий обсяг даних
Компрометована	Будь-яка роль	Вхід до системи з нетипової геолокації (наприклад, інша країна) в неробочий час.	EventType: Login, SourceIP: Unusual, LoginTime: Off-hours
	Будь-яка роль	Швидка послідовність невдалих спроб входу з різних IP-адрес, що завершилася успішним входом.	Послідовність EventType: Login, Status: Failed, за якою слідує Status: Success
	Будь-яка роль	Аномально висока кількість запитів до API з облікового запису користувача.	EventType: ApiEvent, RequestCount > (порогове значення, що перевищує норму для користувача)

Для побудови ефективної моделі виявлення аномалій необхідно визначити набір ознак (features), які можна витягти з логів активності користувачів. Ці ознаки

повинні всебічно характеризувати дії користувача та їх контекст. У таблиці 1.2 наведено перелік типових ознак, що використовуються для моніторингу, з поділом на числові та категоріальні.

Таблиця 1.2 – Типові ознаки для моніторингу поведінки користувачів у CRM-системах

Назва ознаки	**Тип даних**	**Опис**	**Джерело даних**
user_id	Категоріальна	Унікальний ідентифікатор користувача.	Логи подій
user_role	Категоріальна	Роль користувача в системі (напр., 'Admin', 'Sales', 'Marketing').	Логи подій, довідник користувачів
action_type	Категоріальна	Тип виконаної операції (напр., 'Login', 'Export', 'Edit', 'Create', 'Delete').	Логи подій
object_type	Категоріальна	Тип об'єкта, з яким взаємодіяв користувач (напр., 'Lead', 'Account', 'Report').	Логи подій
time_of_day	Числова/Категоріальна	Час доби, коли відбулася подія (може бути представлена як година (0-23) або категорія 'Ранок', 'День', 'Вечір', 'Ніч').	Логи подій
day_of_week	Категоріальна	День тижня, коли відбулася подія.	Логи подій

Продовження табл 1.2

source_ip	Категоріальна	IP-адреса, з якої було здійснено дію.	Логи автентифікації
ip_class	Категоріальна	Клас IP-адреси (напр., 'Corporate', 'VPN', 'TOR', 'Public').	Логи автентифікації, геолокаційні сервіси
session_duration_sec	Числова	Тривалість сесії користувача в секундах.	Логи автентифікації
records_accessed	Числова	Кількість записів, до яких отримано доступ або які були оброблені під час операції.	Логи подій
login_success	Категоріальна	Результат спроби входу (успішно/невдало).	Логи автентифікації
failed_logins_count	Числова	Кількість невдалих спроб входу перед успішним.	Агреговані логи автентифікації

Цей набір ознак є гетерогенним, що підкреслює необхідність використання алгоритму, здатного ефективно працювати з даними змішаного типу.

1.3 Огляд сучасних підходів до виявлення аномалій у даних змішаного типу

Виявлення аномалій (anomaly detection) є добре дослідженою галуззю машинного навчання, однак більшість класичних методів розроблялися з розрахунком на числові дані. Робота з даними змішаного типу, що містять як числові, так і категоріальні атрибути, становить значний виклик [12]. Існуючі підходи можна умовно поділити на кілька категорій.

Статистичні методи. Ці методи базуються на припущенні, що нормальні дані

відповідають певному статистичному розподілу (найчастіше, Гаусовому). Аномаліями вважаються точки, що мають низьку ймовірність у рамках цього розподілу. Класичними прикладами є Z-score та аналіз головних компонент (PCA) [14]. Основний недолік цих методів полягає в їхній жорсткій прив'язці до припущень про розподіл даних. Поведінкові дані користувачів рідко відповідають простим статистичним моделям, а застосування цих методів до категоріальних даних є нетривіальним і часто вимагає їх перетворення у числовий формат, що може спотворити вихідну інформацію.

Методи на основі відстані/щільності (Distance/Density-based). Ці підходи ґрунтуються на ідеї, що аномалії є точками, які знаходяться далеко від більшості інших точок (на основі відстані) або розташовані в регіонах з низькою щільністю даних (на основі щільності). Популярними алгоритмами є k-найближчих сусідів (k-NN) та Local Outlier Factor (LOF) [12]. Головною проблемою при застосуванні цих методів до змішаних даних є визначення адекватної метрики відстані. Необхідно комбінувати різні метрики, наприклад, Евклідову відстань для числових ознак та відстань Хеммінга для категоріальних, що вимагає ретельного налаштування вагових коефіцієнтів і не завжди дає інтуїтивно зрозумілі результати [15].

Класифікаційні методи (Classification-based). Ці методи намагаються побудувати модель, що описує межу між "нормальними" та "аномальними" класами. Найвідомішим представником є One-Class SVM (метод опорних векторів для одного класу), який будує гіперплощину, що відокремлює область з високою щільністю нормальних даних від решти простору [13]. Хоча One-Class SVM може працювати з нелінійними даними за допомогою ядерних функцій, він чутливий до вибору параметрів і може показувати низьку ефективність на даних зі складною структурою, характерною для поведінкових логів.

Ансамблеві методи (Ensemble-based). Ця категорія методів використовує комбінацію кількох слабких моделей для отримання більш надійного та точного результату. Найбільш відомими є ансамблі на основі дерев рішень, такі як Random Forest. У контексті виявлення аномалій особливу увагу привертає алгоритм Isolation

Forest (Ізолюючий ліс), запропонований Лю, Тіном та Чжоу [13]. На відміну від усіх перерахованих вище підходів, Isolation Forest не покладається на метрики відстані чи щільності. Його робота базується на принципово іншому принципі – легкості ізоляції аномальних точок. Цей підхід дозволяє уникнути багатьох проблем, пов'язаних з "прокляттям розмірності" та обчисленням відстаней у багатовимірних просторах, що робить його особливо привабливим для аналізу складних наборів даних.

1.4 Алгоритм Isolation Forest як базовий метод виявлення аномалій

Алгоритм Isolation Forest (iForest) був розроблений спеціально для задачі виявлення аномалій і базується на простому, але потужному спостереженні: "аномалії є нечисленними та відмінними" (anomalies are few and different).[17] Через ці властивості аномальні точки даних значно легше "ізолювати" від решти вибірки, ніж нормальні точки.

1. Основний принцип роботи. Алгоритм будує ансамбль ("ліс") з ізолюючих дерев (iTrees). Кожне дерево будується шляхом рекурсивного поділу підмножини даних. На кожному кроці випадково обирається ознака та випадкове значення-поріг для цієї ознаки. Дані діляться на дві частини залежно від того, чи є значення ознаки для точки меншим чи більшим за поріг. Цей процес продовжується доти, доки кожна точка не буде ізольована в окремому листі дерева. Оскільки аномалії є "відмінними", для їх ізоляції в середньому потрібно значно менше таких випадкових поділів. Отже, аномальні точки матимуть коротку середню довжину шляху від кореня до листа в деревах лісу, тоді як нормальні точки, що знаходяться в щільних скупченнях, вимагатимуть значно більше поділів і матимуть довші шляхи [16].

Переваги алгоритму:

– висока ефективність. iForest має лінійну часову складність відносно розміру вибірки та кількості дерев, що робить його значно швидшим за методи, засновані на відстанях, які зазвичай мають квадратичну складність [16];

– низькі вимоги до пам'яті. Алгоритм не потребує зберігання всієї матриці відстаней, а дерева можуть будуватися на невеликих підвибірках даних, що робить його масштабованим для великих наборів даних [18];

– відсутність потреби у метриці відстані. Це ключова перевага, що дозволяє уникнути проблем, пов'язаних із "прокляттям розмірності".

Критичний аналіз недоліків. Попри свої значні переваги, стандартна реалізація Isolation Forest має два суттєві недоліки, які обмежують його застосування в контексті даної роботи:

1. Нездатність працювати з категоріальними даними. Алгоритм у своїй первісній формі розроблений виключно для числових даних, оскільки механізм розділення базується на порівнянні значення ознаки з числовим порогом (< або >). Він не має вбудованого механізму для обробки нечислових, категоріальних ознак, таких як user_role або action_type [19]. Цей недолік є критичним для аналізу логів CRM-систем.

2. Артефакти від осі-паралельних розділень. Стандартний iForest виконує розділення, паралельні осям координат (оскільки на кожному кроці обирається одна ознака). Це може створювати артефакти на карті аномалій, де певні області простору отримують штучно завищені або занижені оцінки аномальності, що не відповідає реальному розподілу даних. Цю проблему вирішує розширення Extended Isolation Forest (EIF) [22].

1.5 Висновки та постановка задач

Отже, в даному розділі було проведено комплексний аналіз проблеми виявлення інсайдерських загроз, які є значним ризиком для сучасних компаній. У розділі проаналізовано специфіку загроз у CRM-середовищі та встановлено, що дані, які описують поведінку користувачів, мають змішаний тип. Також здійснено критичний огляд існуючих методів та обґрунтовано вибір алгоритму Isolation Forest як базового, виявивши при цьому його ключове обмеження — нездатність нативно обробляти категоріальні ознаки.

В результаті проведеного аналізу теоретичного матеріалу та виходячи з мети і актуальності теми, були поставлені наступні задачі подальшої роботи:

– здійснити вдосконалення алгоритму Isolation Forest шляхом введення механізму змішаних розбиттів задля забезпечення коректної обробки гетерогенних даних (числових та категоріальних);

– спроєктувати та розробити програмний засіб для виявлення інсайдерських загроз у CRM-системах на основі модифікованого методу;

– здійснити тестування розробленої програмної реалізації та оцінити ефективність вдосконаленого алгоритму порівняно з існуючими аналогами;

– теоретично обґрунтувати запропоновану модифікацію алгоритму виявлення аномалій.

В результаті виконання поставлених завдань, планується досягти основної мети роботи, а саме підвищити ефективність виявлення інсайдерських загроз у CRM-системах за рахунок адаптації методу Isolation Forest до даних змішаного типу без втрати інформації.

2 ВДОСКОНАЛЕННЯ АЛГОРИТМУ ISOLATION FOREST ДЛЯ ЗАДАЧІ ВИЯВЛЕННЯ АНОМАЛІЙ У ЗМІШАНИХ ДАНИХ

Другий розділ присвячено обґрунтуванню та математичному опису запропонованого методу вдосконалення алгоритму Isolation Forest. Метою цього етапу дослідження є розробка алгоритмічного підходу, який дозволив би ефективно обробляти дані змішаного типу (числові та категоріальні) без втрати інформативності та без суттєвого зростання обчислювальної складності.

У розділі детально розглядається математична модель стандартного алгоритму Isolation Forest та аналізуються недоліки традиційних методів кодування категоріальних ознак, таких як One-Hot Encoding та Label Encoding. Доводиться, що ці методи вносять викривлення в метричний простір даних, що негативно впливає на якість ізоляції аномалій.

Центральним елементом розділу є формалізація нового механізму «змішаних сплітів» (Mixed Splits). Наводиться математичний опис процедури побудови ізолюючих дерев з використанням цього механізму, описуються структури даних та алгоритми навчання і оцінювання. Також проводиться теоретичний аналіз часової та просторової складності запропонованого методу, що дозволяє спрогнозувати його ефективність при обробці великих масивів даних.

2.1 Аналіз математичної моделі класичного алгоритму Isolation Forest та обмежень його застосування

Алгоритм Isolation Forest (надалі — iForest), запропонований Ф. Лю (F. Liu), К. Тінгом (K. Ting) та Ч. Чжоу (Z. Zhou) [1], представляє собою фундаментально відмінний підхід до задачі виявлення аномалій (anomaly detection). На відміну від традиційних методів, що базуються на вимірюванні відстаней (distance-based), щільності (density-based) або побудові профілів нормальної поведінки, iForest використовує стратегію "ізоляції".

В основі методу лежить просте, але потужне емпіричне спостереження: аномалії — це події, які є, по-перше, нечисленними (few), і, по-друге, суттєво

відмінними (different) від основної маси даних. Завдяки цим двом властивостям, аномальні екземпляри даних легше "ізолювати" від решти вибірки шляхом випадкового розбиття простору ознак.

Математичною основою алгоритму є ансамбль бінарних дерев рішень, які називаються ізолюючими деревами (Isolation Trees, або iTrees). Нехай $X = \{x_1, x_2, ..., x_n\}$ — вхідна вибірка даних, що складається з n екземплярів, де кожен екземпляр x_i описується набором з d ознак (атрибутів) $Q = \{q^{(1)}, ..., q^{(d)}\}$.

Процес побудови одного iTree відбувається рекурсивно за принципом «розділяй і володарюй» на випадковій підвибірці даних $X' \subset X$ розміром ψ (зазвичай $\psi = 256$). Алгоритм побудови вузла дерева можна описати наступним чином:

– Якщо підвибірка у поточному вузлі X_{node} містить лише один екземпляр ($|X_{node}| = 1$) або всі екземпляри мають однакові значення ознак, процес зупиняється, і вузол оголошується зовнішнім (листом).

– В іншому випадку, випадковим чином обирається одна ознака $q \in Q$ та точка розділення (split point) p. Значення p обирається з рівномірного розподілу в діапазоні значень обраної ознаки у поточному вузлі:

$p \sim Uniform(\min(q), \max(q))$

1. На основі обраних q та p формується гіперплощина, ортогональна до осі координат q, яка розділяє простір на дві підмножини. Дані X_{node} розподіляються між лівим (X_l) та правим (X_r) дочірніми вузлами за правилом:

$X_l = \{x \in X_{node} \mid x^{(q)} < p\} X_r = \{x \in X_{node} \mid x^{(q)} \geq p\}$

Цей процес продовжується до досягнення граничної висоти дерева або повної ізоляції всіх точок. Оскільки аномалії, як правило, розташовані у розріджених областях простору ознак, для їх відокремлення (ізоляції) в окремий лист потрібно значно менше випадкових розбиттів, ніж для нормальних точок, що скупчені у щільних кластерах.

Ключовою метрикою в алгоритмі є довжина шляху $h(x)$, яку проходить

екземпляр x від кореня дерева до листа. У термінах теорії графів, $h(x)$ еквівалентна кількості ребер e на шляху від кореневого вузла до термінального вузла.

Оскільки iTree має структуру, еквівалентну структурі бінарного дерева пошуку (Binary Search Tree — BST), для теоретичної оцінки середньої довжини шляху можна використати властивості BST. Для невдалого пошуку в випадковому BST, побудованому на ψ елементах, середня довжина шляху апроксимується як:

$$c(\psi) = 2H(\psi - 1) - \frac{2(\psi - 1)}{\psi}$$

де $H(i)$ — i-те гармонічне число, яке можна наблизити за допомогою натурального логарифма та константи Ейлера-Маскероні ($\gamma \approx 0.5772156649$): $H(i) \approx \ln(i) + \gamma$.

Величина $c(\psi)$ слугує нормалізуючим коефіцієнтом, оскільки вона представляє "середню" глибину дерева для даного розміру вибірки. Вона дозволяє порівнювати результати, отримані на підвибірках різного розміру.

Оцінка аномальності (anomaly score) $s(x, \psi)$ для екземпляра x розраховується шляхом усереднення довжини шляху $E(h(x))$ по всіх деревах ансамблю та її нормалізації:

$$s(x, \psi) = 2^{-\frac{E(h(x))}{c(\psi)}}$$

Аналіз цієї формули дозволяє виділити три ключові діапазони значень s:

1. Якщо $E(h(x)) \to 0$, то $s \to 1$. Це свідчить про те, що точка була ізольована дуже швидко (на верхніх рівнях дерева), отже, вона з високою ймовірністю є аномалією.
2. Якщо $E(h(x)) \to \psi - 1$, то $s \to 0$. Це означає, що точку важко ізолювати, вона знаходиться глибоко в дереві, отже, вона є нормальною.
3. Якщо $E(h(x)) \approx c(\psi)$, то $s \approx 0.5$. Точка не має явних ознак аномальності.

Класична реалізація Isolation Forest демонструє високу ефективність на числових даних. Однак, у контексті виявлення інсайдерських загроз у CRM-системах, ми стикаємося з даними змішаного типу. Логи активності користувачів

містять як безперервні числові змінні (наприклад, session_duration, transaction_amount), так і категоріальні змінні (наприклад, user_role, department, action_type, ip_country).

Фундаментальне обмеження стандартного iForest полягає у тому, що операція розбиття $x^{(q)} < p$ визначена лише для впорядкованих множин (числових даних). Категоріальні дані не мають природного порядку, тому для їх використання в iForest необхідна процедура попереднього кодування (encoding), яка перетворює категорії у числа. Розглянемо детально, чому стандартні методи кодування руйнують логіку алгоритму.

А. Недоліки Label Encoding (Порядкове кодування)

Метод Label Encoding присвоює кожній унікальній категорії c_i ціле число $k_i \in \{1, \ldots, L\}$. Наприклад, для атрибуту user_role може бути створено відображення:

1. 'Admin' $\rightarrow 1$
2. 'Manager' $\rightarrow 2$
3. 'Intern' $\rightarrow 3$

Головна проблема цього підходу — введення штучного порядку, який не має семантичного обґрунтування. Алгоритм iForest буде інтерпретувати ці значення як величини, де $3 > 2 > 1$. Припустимо, дерево обирає точку розділення $p = 2.5$. У цьому випадку 'Intern' (3) потрапить в одну гілку, а 'Admin' (1) і 'Manager' (2) — в іншу. Таке групування є випадковим і не відображає реальної схожості об'єктів. Більше того, це порушує принцип "ізоляції", оскільки відстань між категоріями стає залежною від їх довільної нумерації, а не від їхньої сутності.

Б. Критичні обмеження One-Hot Encoding (Унітарне кодування)

Найпоширенішим методом обробки категоріальних даних є One-Hot Encoding (OHE), який перетворює одну категоріальну ознаку з кардинальністю L (кількістю унікальних значень) у L нових бінарних ознак. Хоча цей метод вирішує проблему штучного порядку, він породжує ряд інших, більш серйозних проблем для

алгоритмів на основі дерев, і особливо для Isolation Forest.

1. Прокляття розмірності (Curse of Dimensionality)

У CRM-системах деякі атрибути мають дуже високу кардинальність. Наприклад, атрибут User_ID може мати тисячі значень, а City — сотні. Застосування OHE призводить до вибухового зростання розмірності простору ознак. Якщо вихідний набір даних мав d ознак, то після кодування розмірність стає:

$$d_{new} = d_{numerical} + \sum_{j=1}^{k} L_j$$

де L_j — кардинальність j-ї категоріальної ознаки. Збільшення розмірності різко підвищує вимоги до обчислювальних ресурсів та пам'яті.

2. Проблема розрідженості (Sparsity)

Матриця ознак після OHE стає надзвичайно розрідженою. Для кожної групи бінарних ознак, що відповідають одній вихідній категоріальній змінній, лише одна ознака має значення 1, а всі інші — 0. Це означає, що інформаційна щільність даних падає.

3. Зміщення вибору ознак (Feature Selection Bias)

Це, мабуть, найкритичніша проблема для iForest. На кожному кроці побудови дерева алгоритм випадковим чином обирає одну ознаку для розбиття. У просторі, роздутому OHE, ймовірність обрати одну з численних бінарних (і малоінформативних) ознак значно перевищує ймовірність обрати важливу числову ознаку.

Нехай ми маємо 1 числову ознаку (amount) і 1 категоріальну (city) з 100 унікальними значеннями. Після OHE ми отримаємо 101 ознаку. Ймовірність вибору числової ознаки $P(numerical)$ становить:

$$P(numerical) = \frac{1}{101} \approx 0.0099$$

Тобто менше 1%. В результаті, дерево буде майже повністю складатися з розбиттів по бінарних ознаках міст. Оскільки такі розбиття є незбалансованими (в одній гілці лише точки з конкретного міста, в іншій — всі решта), дерево стає дуже

глибоким, але погано ізолює аномалії, що базуються на числових патернах (наприклад, аномально висока сума транзакції).

4. Ефект маскування (Masking Effect)

У просторі OHE відстань між будь-якими двома різними категоріями є однаковою і дорівнює $\sqrt{2}$ (евклідова відстань між векторами вигляду $[0, 1, 0...]$ та $[0, 0, 1...]$). Це призводить до втрати інформації про "схожість" категорій. Алгоритм не може виявити, що певні категорії (наприклад, міста одного регіону) повинні групуватися разом. В результаті, аномалії, які проявляються лише у специфічних поєднаннях категорій, стають невиразними на фоні загального шуму розріджених даних.

Проведений аналіз математичної моделі Isolation Forest демонструє, що, незважаючи на свою ефективність, алгоритм у класичному вигляді є непридатним для прямого застосування до гетерогенних даних CRM-систем. Традиційні методи адаптації даних (Label Encoding, One-Hot Encoding) не вирішують проблему, а лише трансформують її у площину втрати семантики або катастрофічного зростання розмірності та зміщення ймовірностей.

Це створює об'єктивну необхідність у вдосконаленні алгоритму на рівні механізму розбиття (splitting mechanism). Необхідно розробити такий підхід, який дозволяв би:

– обробляти категоріальні дані нативно, без перетворення у бінарний векторний простір;

– зберігати рівноправність числових та категоріальних ознак при випадковому виборі;

– забезпечувати ефективну ізоляцію категоріальних значень без побудови надмірно глибоких дерев.

Вирішенню саме цих завдань присвячено розробку методу змішаних сплітів (Mixed Splits), теоретичне обґрунтування якого наведено у наступному підрозділі.

2.2 Обґрунтування та формалізація механізму змішаних сплітів (Mixed Splits)

Як було показано в попередньому підрозділі, застосування стандартних методів кодування (One-Hot Encoding) до категоріальних даних призводить до викривлення метричного простору та зниження ефективності алгоритму Isolation Forest. Для вирішення цієї проблеми пропонується модифікація процедури розгалуження (splitting procedure), яка отримала назву «механізм змішаних сплітів» (Mixed Splits).

Суть запропонованого підходу полягає у відмові від попереднього перетворення даних у числовий вигляд на користь адаптивної логіки розбиття вузлів дерева, яка динамічно змінюється залежно від типу обраної ознаки.

Розглянемо задачу бінарного розділення множини даних X у поточному вузлі дерева на основі категоріальної ознаки q, яка приймає значення з дискретної множини категорій $C_q = \{c_1, c_2, \ldots, c_L\}$, де L — кардинальність ознаки (кількість унікальних значень).

У класичних деревах рішень (CART, C4.5), що використовуються для задач класифікації або регресії, метою є знаходження оптимального розбиття, яке максимізує приріст інформації (Information Gain) або мінімізує ентропію. Для цього необхідно перебрати всі можливі варіанти розділення множини C_q на дві непусті підмножини S_{left} та S_{right}, такі що:

$$S_{left} \cup S_{right} = C_q, \quad S_{left} \cap S_{right} = \emptyset, \quad S_{left} \neq \emptyset, \quad S_{right} \neq \emptyset$$

Кількість таких можливих бінарних розбиттів для ознаки з L категоріями визначається числом Стірлінга другого роду або простішою комбінаторною формулою:

$$N_{splits} = \frac{2^L - 2}{2} = 2^{L-1} - 1$$

Для малих L (наприклад, стать, $L = 2$) кількість варіантів мала ($2^1 - 1 = 1$). Однак, функція зростає експоненціально. Для атрибуту Region з $L = 20$ кількість

варіантів становить $2^{19} - 1 \approx 5.2 \times 10^5$, а для Product_ID з $L = 100$ повний перебір стає обчислювально неможливим ($2^{99} \approx 6.3 \times 10^{29}$).

В алгоритмі Isolation Forest, який є методом некерованого навчання (unsupervised learning), ми не маємо цільової змінної для оцінки якості розбиття. Більше того, філософія iForest базується на випадковості розбиттів. Тому замість пошуку оптимального розбиття, нам необхідно генерувати випадкове розбиття множини категорій.

Запропонований механізм змішаних сплітів інтегрує два типи правил розбиття в єдину процедуру побудови дерева. Нехай $Type(q)$ — функція, що повертає тип обраної ознаки q. Тоді умова переходу екземпляра x до лівого дочірнього вузла визначається предикатом $\Phi(x, q, \theta)$, де θ — параметри розбиття.

Випадок 1: числова ознака ($Type(q) = Numerical$)

Використовується стандартний підхід iForest. Параметром розбиття є порогове значення $p \in \mathbb{R}$.

$$\Phi_{num}(x, q, p) \iff x^{(q)} < p$$

Значення p обирається рівномірно з діапазону $[\min(X_{node}^{(q)}), \max(X_{node}^{(q)})]$.

Випадок 2: категоріальна ознака ($Type(q) = Categorical$)

Використовується підхід Subset Split. Параметром розбиття є множина значень $S \subset C_q$, що формує "ліву" гілку.

$$\Phi_{cat}(x, q, S) \iff x^{(q)} \in S$$

Таким чином, загальний алгоритм обробки вузла виглядає так:

– випадково обрати ознаку $q \in Q$;

– якщо q числова:

a. Згенерувати $p \sim Uniform(\min, \max)$;

b. $X_{left} = \{x \mid x^{(q)} < p\}$;

– якщо q категоріальна:

a. Згенерувати випадкову непусту підмножину

$S \subset Values(X_{node}^{(q)})$;

b. $X_{left} = \{x \mid x^{(q)} \in S\}$.

Критично важливим є метод генерації підмножини S. Для забезпечення максимальної "ізолюючої здатності" (isolating capacity), розбиття повинно прагнути ділити дані на приблизно рівні частини (в контексті кількості унікальних категорій, а не кількості екземплярів, подібно до того, як числовий спліт ділить діапазон значень).

Пропонується використовувати схему випробувань Бернуллі. Нехай $V = \{v_1, \ldots, v_k\}$ — множина унікальних значень ознаки q, присутніх у поточному вузлі. Для формування S:

1. Для кожного значення $v_i \in V$ генерується випадкова величина $\xi_i \sim Bernoulli(0.5)$.
2. Якщо $\xi_i = 1$, то v_i додається до S.
3. Якщо $\xi_i = 0$, то v_i потрапляє до доповнення S^c (права гілка).

Математичне сподівання розміру підмножини S становить $E[|S|] = k/2$, що забезпечує збалансоване (в середньому) розбиття простору категорій.

Обробка вироджених випадків:

Існує ймовірність $P = 1/2^{k-1}$, що всі елементи потраплять в одну групу (тобто $S = \emptyset$ або $S = V$). Оскільки таке розбиття не зменшує ентропію і не розділяє дані, алгоритм повинен передбачати перевірку:

while $(S == \emptyset$ or $S == V)$: regenerate S

Для великих k ймовірність такого випадку прямує до нуля, тому вплив на швидкодію є нехтовним.

Важливо чітко розмежувати запропонований метод Mixed Splits (IF-MS) та існуючий розширений алгоритм Extended Isolation Forest (EIF), запропонований С. Харірі (S. Hariri) [22].

EIF вирішує проблему артефактів розділення ("bias towards axis-parallel splits") у числовому просторі. Замість вибору однієї ознаки та порогу ($x^{(i)} < p$), EIF

генерує випадкову гіперплощину з нормальним вектором $\mathbf{n}$ та точкою перетину $\mathbf{p}$:

$$(\mathbf{x} - \mathbf{p}) \cdot \mathbf{n} < 0$$

Цей метод працює у векторному просторі $\mathbb{R}^d$ і вимагає, щоб усі ознаки мали числову природу та були порівнянними (вимагає ретельного масштабування). EIF не вміє нативно працювати з категоріями без їх попереднього кодування.

IF-MS (Mixed Splits) вирішує проблему гетерогенності даних. Він не змінює геометрію розділення для числових даних (залишаючись в рамках axis-parallel для них), але вводить принципово нову логіку для категоріальних.

Ці два підходи є ортогональними. Теоретично можлива побудова гібридної моделі, де числові ознаки розділяються похилими гіперплощинами (EIF), а категоріальні — підмножинами (Mixed Splits), проте в рамках даної роботи ми фокусуємось саме на проблемі змішаних даних, залишаючи класичний підхід для числових атрибутів.

Гіпотеза, що лежить в основі ефективності Mixed Splits, полягає в наступному: аномальні екземпляри, що характеризуються рідкісними категоріями або нетиповими комбінаціями категорій, будуть ізольовані швидше при випадковому розбитті множини категорій.

Розглянемо приклад. Нехай в вузлі присутні транзакції з міст $V = \{A, B, C, D, E\}$, де міста A, B, C, D — звичайні локації, а E — аномальна (рідкісна).

При випадковому розбитті множини V на S та S^c, ймовірність того, що E буде відокремлено від, наприклад, A, становить 0.5 на кожному рівні. Для того, щоб E залишився наодинці (ізолювався), потрібно, щоб у серії випадкових розбиттів він був "відрізаний" від усіх інших елементів.

Якщо E є аномалією частотного типу (зустрічається рідко), то у вузлі буде мало екземплярів з $City = E$. Після того як механізм Subset Split відокремить E від інших міст, розмір підвибірки у відповідній гілці різко зменшиться, що призведе до швидкого досягнення листа. Це повністю відповідає канонічній логіці Isolation Forest: короткий шлях $\rightarrow$ високий бал аномальності.

Таким чином, механізм Subset Split є математично коректним узагальненням принципу ізоляції на дискретні простори без метрики порядку.

2.3 Алгоритм та реалізація методу IF-MS: структури даних та процедури побудови ансамблю

Практична реалізація методу Mixed Splits (IF-MS) виконана мовою програмування Python з використанням бібліотек для наукових обчислень NumPy та Pandas. Вибір цих інструментів зумовлений їхньою здатністю ефективно обробляти табличні дані та виконувати векторизовані операції, що є критичним для швидкодії алгоритму.

У цьому підрозділі описується архітектура розробленого програмного модуля custom_forest.py, включаючи об’єктну модель, реалізацію процедур навчання (training) та оцінювання (scoring).

Архітектура системи базується на композиції двох основних класів: MixedIsolationTree (Ізолююче дерево) та MixedIsolationForest (Ансамбль дерев). Така структура відповідає загальноприйнятим патернам проектування бібліотеки Scikit-learn (Estimator / Transformer), що спрощує інтеграцію розробки в існуючі пайплайни обробки даних.

Клас MixedIsolationTree

Цей клас відповідає за рекурсивну побудову та зберігання структури одного дерева рішень. Кожен екземпляр класу представляє собою вузол дерева, який може бути внутрішнім (вузлом розгалуження) або зовнішнім (листом).

Основні атрибути класу включають:

– height_limit: Максимально допустима глибина дерева, яка слугує критерієм зупинки рекурсії.

– split_attr: Назва або індекс ознаки, за якою відбувається розбиття у даному вузлі.

– split_val: Параметр розбиття. Завдяки динамічній типізації Python, цей атрибут є поліморфним:

c. Для числових ознак він зберігає порогове значення (float).

d. Для категоріальних ознак він зберігає підмножину значень (set або масив), що відносяться до лівої гілки.

– left та right: Посилання на дочірні екземпляри MixedIsolationTree.

– node_type: Тип вузла ('internal' або 'external').

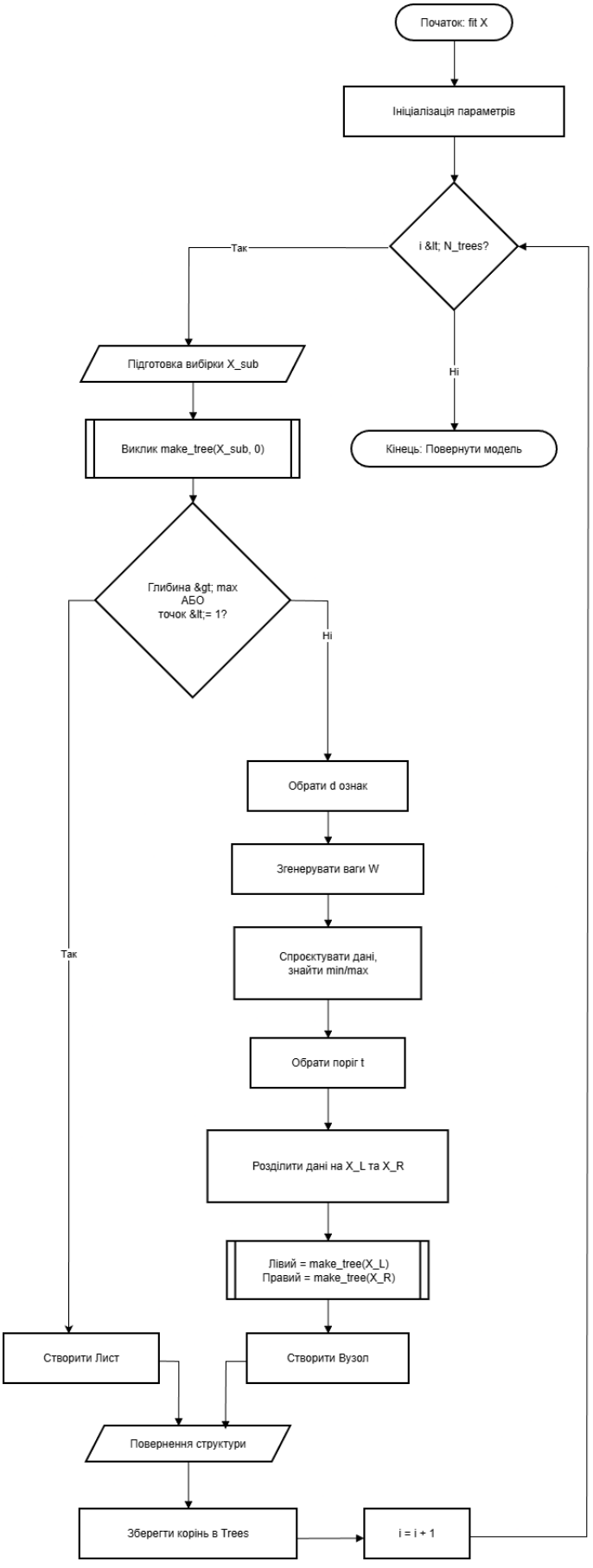
– size: Кількість екземплярів даних, що потрапили у вузол (використовується для коригування довжини шляху в листах).

Клас MixedIsolationForest

Клас верхнього рівня, який виступає оркестратором процесу навчання. Він керує колекцією дерев (self.trees), параметрами підвибірки (sample_size) та кількістю естіматорів (n_estimators).

Процес навчання реалізовано в методі fit. На відміну від класичних реалізацій, які працюють з числовими масивами NumPy, розроблений метод приймає на вхід pandas.DataFrame. Це дозволяє зберігати метаінформацію про типи даних (числовий/об'єктний) безпосередньо в структурі даних.

Візуалізація логіки алгоритму є важливим елементом інженерної розробки.



Блок схема 2.1 - Побудова дерева (Build Tree Process)

Опис структур даних.

Для ефективної реалізації в об'єктно-орієнтованому середовищі (наприклад, Python або C++), доцільно визначити наступні класи:

– Class Node:

○ type: Enum (Internal, External);

○ size: Integer (кількість точок у вузлі, актуально для листів);

○ normal_vector: Array/Vector (вектор $\mathbf{w}$ для внутрішніх вузлів).$\mathbf{w}$ для внутрішніх вузлів);

○ intercept: Float (поріг p).p);

○ left_child: Pointer/Reference;

○ right_child: Pointer/Reference.

– Class IsolationForestMS:

○ trees: List of Node (корені дерев);

○ subsample_size: Integer;

○ n_estimators: Integer;

○ mixed_dimension: Integer (k).k).

Важливим архітектурним рішенням є спосіб зберігання вектора normal_vector. Для економії пам'яті при $k \ll d$ доцільно використовувати розріджений формат (Sparse Vector), зберігаючи лише пари (індекс, значення), замість повного масиву нулів.$k \ll d$ доцільно використовувати розріджений формат (Sparse Vector), зберігаючи лише пари (індекс, значення), замість повного масиву нулів.

Логіка побудови дерева (метод MixedIsolationTree.fit) реалізує стратегію "глибина-першим" (Depth-First Search) і включає наступні кроки:

1. Перевірка критеріїв зупинки: Якщо поточна глибина досягла ліміту height_limit або у вузлі залишилося менше двох екземплярів, вузол позначається як 'external', і рекурсія зупиняється.

2. Випадковий вибір ознаки: За допомогою генератора випадкових чисел обирається одна колонка з переданого DataFrame.

3. Динамічне визначення стратегії розбиття:

За допомогою функції pd.api.types.is_numeric_dtype перевіряється тип обраної колонки.

Для числових даних: Визначаються мінімальне та максимальне значення у поточному вузлі. Генерується випадковий поріг split_val з рівномірного розподілу (np.random.uniform). Формується булева маска фільтрації за умовою col_data < split_val.

Для категоріальних даних (Mixed Split): Витягуються унікальні значення категорії. Генерується випадкова перестановка (np.random.shuffle) та обирається підмножина значень (slice) для формування split_val. Маска фільтрації створюється векторизованим методом .isin().

4. Розподіл даних та рекурсія: Дані розбиваються на дві підмножини (X_left, X_right) відповідно до маски. Метод fit викликається рекурсивно для лівого та правого дочірніх вузлів.

Такий підхід дозволяє обробляти змішані дані "на льоту", не вимагаючи попереднього кодування категорій у числа, що суттєво економить пам'ять.

Оцінювання ступеня аномальності нових даних здійснюється методом anomaly_score класу MixedIsolationForest.

Процедура розрахунку для одного екземпляра даних x:

- Обчислення довжини шляху: Екземпляр x "пропускається" через кожне дерево ансамблю за допомогою методу path_length.
- Навігація по дереву: У кожному вузлі перевіряється умова переходу:
  - Якщо ознака числова: порівняння x[attr] < split_val.
  - Якщо ознака категоріальна: перевірка на входження x[attr] in split_val. Оскільки split_val реалізовано як хеш-множину (set), ця операція має середню складність $O(1)$.
- Агрегація та нормалізація: Отримані довжини шляхів $h(x)$ усереднюються по всьому лісу. Фінальний скор обчислюється за формулою:

$$s(x, \psi) = 2^{-\frac{E(h(x))}{c(\psi)}}$$

де $c(\psi)$ — нормалізуючий коефіцієнт для середньої довжини шляху в бінарному дереві пошуку, реалізований у методі _c.

Результатом роботи методу є масив NumPy зі значеннями в діапазоні $(0, 1)$, де вищі значення свідчать про більшу ймовірність аномалії.

У ході програмної реалізації було застосовано ряд технічних рішень для підвищення ефективності алгоритму:

1. Використання Pandas Views: При розбитті даних у методі fit використовуються фільтрації, які в сучасних версіях Pandas оптимізовані для мінімізації глибокого копіювання даних (Copy-on-Write механізми), що зменшує навантаження на RAM при побудові глибоких дерев.

2. Векторизована фільтрація категорій: Замість ітеративного перебору рядків, перевірка приналежності до підмножини категорій виконується методом Series.isin(), який реалізований на рівні C/Cython і є значно швидшим за стандартні цикли Python.

3. Відмова від матриці $N \times M$: Завдяки відмові від One-Hot Encoding, вхідні дані передаються у компактному вигляді. Наприклад, категорія "Product_Category" з 100 унікальними значеннями займає 1 колонку типу object замість 100 колонок uint8. Це дозволяє обробляти значно більші обсяги даних на тому ж апаратному забезпеченні.

Реалізований програмний комплекс забезпечує повну функціональність методу Mixed Isolation Forest. Архітектура класів дозволяє гнучко працювати з гетерогенними даними, автоматично визначаючи стратегію розбиття залежно від типу даних. Застосування ефективних структур даних (хеш-множини для категорій) та векторизованих операцій Pandas забезпечує високу продуктивність як на етапі навчання, так і на етапі детекції загроз.

2.4 Аналіз обчислювальної складності та масштабованості алгоритму

Однією з ключових вимог до алгоритмів виявлення аномалій у системах реального часу, таких як CRM, є їхня здатність обробляти великі потоки даних з мінімальною затримкою. Введення механізму змішаних сплітів (Mixed Splits) вимагає детального аналізу його впливу на асимптотичну складність алгоритму порівняно з класичною реалізацією Isolation Forest.

Часова складність алгоритму складається з двох компонентів: часу навчання (training time) та часу оцінювання (inference time).

Етап навчання:

Нехай t — кількість дерев в ансамблі, ψ — розмір підвибірки для побудови одного дерева.

Побудова одного дерева MixedIsolationTree відбувається рекурсивно. Оскільки дерево є збалансованим (в середньому), його висота становить $H \approx \log_2 \psi$.

На кожному рівні глибини h ($0 \leq h < H$) сумарно по всіх вузлах обробляється ψ екземплярів даних (кожен екземпляр потрапляє рівно в один вузол на кожному рівні).

Операція розбиття у вузлі включає:

– Генерацію випадкового порогу або підмножини.
– Фільтрацію даних (розподіл на ліву і праву гілки).

Для числових ознак фільтрація X < p виконується за час $O(n_{node})$.

Для категоріальних ознак фільтрація X.isin(S) при використанні хеш-множини також виконується за час $O(n_{node})$ в середньому.

Таким чином, складність побудови одного рівня дерева становить $O(\psi)$. Оскільки рівнів $\log \psi$, то складність побудови одного дерева дорівнює $O(\psi \log \psi)$. Сумарна часова складність навчання для всього лісу:

$T_{train} = O(t \cdot \psi \log \psi)$

Важливо зазначити, що ця складність не залежить від загального розміру

датасету N (за умови, що вибірка ψ фіксована), що робить алгоритм надзвичайно масштабованим.

Етап оцінювання:

Для оцінки одного нового екземпляра x необхідно пройти шлях від кореня до листа у кожному з t дерев. Довжина шляху обмежена $H \approx \log \psi$.

Перевірка умови у кожному вузлі (числовим порівнянням або пошуком у хеш-множині) займає $O(1)$.

Отже, складність оцінювання для M нових екземплярів становить:

$T_{score} = O(M \cdot t \cdot \log \psi)$

Висновок: часова складність запропонованого методу IF-MS є ідентичною складності класичного iForest. Введення категоріальних сплітів не погіршує асимптотику алгоритму.

Саме в аспекті використання оперативної пам'яті (RAM) метод IF-MS демонструє найбільшу перевагу над підходом One-Hot Encoding (OHE).

Витрати на зберігання моделі:

Класичний iForest (з OHE) будує дерева над простором розмірності $D_{OHE} = D_{num} + \sum L_i$, де L_i — кардинальність категоріальних ознак.

Метод IF-MS працює у просторі $D_{orig} = D_{num} + D_{cat}$, де D_{cat} — кількість категоріальних ознак.

Розглянемо витрати пам'яті на один вузол дерева:

1. У класичному iForest вузол зберігає індекс ознаки (int) та поріг (float). Розмір фіксований і малий.
2. У IF-MS для категоріальних вузлів зберігається множина split_val (set). Розмір цієї множини в середньому становить $|S| \approx L_i/2$.

Здавалось би, IF-MS споживає більше пам'яті на вузол. Однак, розглянемо витрати пам'яті на обробку даних під час навчання.

Витрати на дані (Data Memory Footprint):

Нехай N — кількість записів.

Підхід OHE**:** необхідно зберігати матрицю розміром $N \times D_{OHE}$. Для розріджених даних це призводить до значних витрат (навіть при використанні sparse formats, накладні витрати на індекси є суттєвими).

Підхід IF-MS: зберігається вихідна таблиця розміром $N \times D_{orig}$.

Коефіцієнт стиснення простору ознак (Compression Ratio) можна оцінити як:

$$CR = \frac{D_{num} + \sum_{i=1}^{k} L_i}{D_{num} + k}$$

Для датасету Olist, де є категорія product_category_name ($L = 74$) та інші

$$CR \approx \frac{2 + 74 + \dots}{4} \approx 20$$

Тобто метод IF-MS потребує у 20 разів менше пам'яті для зберігання навчальної вибірки в оперативній пам'яті. Це дозволяє навчати модель на значно більших обсягах історичних даних без необхідності використання кластерних рішень (Spark/Dask).

Ефективність алгоритму залежить від двох основних гіперпараметрів: розміру підвибірки ψ (sample_size) та кількості дерев t (n_estimators).

Розмір підвибірки (ψ):

В оригінальній роботі Лю [1] емпірично доведено, що збільшення ψ понад 256 не призводить до суттєвого підвищення точності детекції, але лінійно збільшує час обробки. Це пов'язано з тим, що аномалії за визначенням є "легко ізольованими". Якщо аномалію не вдалося ізолювати за $\log_2(256) = 8$ розбиттів, вона, ймовірно, не є яскраво вираженою.

У роботі прийнято $\psi = 256$, що забезпечує баланс між чутливістю до локальних аномалій та стійкістю до шуму (ефект маскування).

Кількість дерев (t):

Довжина шляху $h(x)$ є стохастичною величиною. За законом великих чисел, середнє значення $E(h(x))$ збігається до істинного математичного сподівання при $t \to \infty$.

Дослідження показують, що дисперсія оцінки стабілізується вже при t = 100.

Оскільки в методі IF-MS вводиться додаткова стохастичність (випадковий вибір підмножини категорій S), дисперсія окремого дерева може бути дещо вищою, ніж у класичному варіанті. Тому для гарантування надійності результатів у програмній реалізації рекомендовано використовувати $t = 100$, а для критичних систем — збільшувати до $t = 200$.

Важливою особливістю методу IF-MS є його інваріантність (нечутливість) до масштабування числових даних.

Оскільки поріг p обирається випадково в діапазоні $[\min, \max]$, а умова розгалуження $x < p$ залежить лише від відносного порядку значень, алгоритм не потребує попередньої нормалізації (MinMax, Z-score) числових ознак. Це вигідно відрізняє його від методів на основі відстаней (k-NN, LOF), де одна ознака з великим діапазоном значень (наприклад, сума транзакції) може повністю нівелювати вплив інших ознак.

Для категоріальних ознак поняття масштабу відсутнє, тому попередня обробка також не потрібна (крім приведення до єдиного рядкового формату).

Проведений аналіз підтверджує, що модифікація Mixed Isolation Forest зберігає логарифмічну часову складність оригінального алгоритму ($O(\psi \log \psi)$), забезпечуючи високу швидкодію. Водночас, відмова від One-Hot Encoding забезпечує суттєвий виграш у використанні оперативної пам'яті (в десятки разів для даних високої кардинальності), що робить метод придатним для використання в умовах обмежених ресурсів або на мобільних/периферійних пристроях (Edge Computing). Відсутність вимог до нормалізації даних спрощує конвеєр попередньої обробки (ETL Pipeline).

2.5 Методологія верифікації ефективності та критерії оцінювання якості детекції

Валідація алгоритмів виявлення аномалій (Anomaly Detection) має специфічні складності порівняно із задачами класифікації, оскільки реальні дані часто не містять розмічених прикладів атак («ground truth»), а самі аномалії є вкрай

рідкісними подіями. У цьому підрозділі обґрунтовується методологія проведення експерименту, вибір метрик оцінювання та стратегія генерації тестових даних, реалізована у програмному модулі real_data_experiment.py.

В умовах сильного дисбалансу класів (де нормальні дані становлять >99%, а аномалії <1%) стандартна метрика точності (Accuracy) є неінформативною. Модель, яка просто класифікує всі транзакції як «нормальні», матиме точність 99%, але її корисність дорівнюватиме нулю. Тому для оцінювання запропонованого методу IF-MS використовуються наступні метрики.

1. AUC-ROC (Area Under the Receiver Operating Characteristic Curve)

Це основна метрика, використана в експериментальному дослідженні. ROC-крива відображає залежність між часткою вірно класифікованих аномалій (True Positive Rate, TPR) та часткою хибних спрацювань (False Positive Rate, FPR) при варіюванні порогу детекції.

$$TPR = \frac{TP}{TP + FN}, \quad FPR = \frac{FP}{FP + TN}$$

де TP — кількість вірно виявлених аномалій, FP — кількість помилкових тривог, FN — кількість пропущених аномалій, TN — кількість коректно пропущених нормальних подій.

Площа під цією кривою (AUC) є інтегральною оцінкою, що інтерпретується як ймовірність того, що випадково обраний аномальний екземпляр отримає вищу оцінку аномальності (anomaly score), ніж випадково обраний нормальний екземпляр.

- $AUC = 0.5$: випадкове вгадування.
- $AUC = 1.0$: ідеальна класифікація.

Перевага AUC-ROC полягає в тому, що вона не залежить від вибору конкретного порогу відсікання (threshold), дозволяючи оцінити роздільну здатність алгоритму в цілому.

Для експериментальної перевірки обрано набір даних Brazilian E-Commerce Public Dataset by Olist, який містить реальні дані про 100 тис. замовлень у період з

2016 по 2018 роки. Цей набір є репрезентативним для задач CRM, оскільки містить складну реляційну структуру та гетерогенні ознаки.

Для навчання моделі використовується агрегована таблиця (Master View), сформована модулем DataProcessor, яка включає:

1. Числові атрибути: price (вартість товару), freight_value (вартість доставки).
2. Категоріальні атрибути: product_category_name (74 унікальні категорії), payment_type (4 типи оплати).

Такий набір ознак дозволяє протестувати здатність алгоритму знаходити кореляції між різнотипними даними (наприклад, між категорією товару та його ціною).

Оскільки вихідний датасет не містить підтверджених фактів шахрайства, застосовано метод напівсинтетичного тестування (Semi-Synthetic Testing). Реальні дані Olist вважаються «нормальним класом» ($Label = 0$), до якого підмішуються штучно згенеровані аномалії ($Label = 1$).

У модулі prepare_semi_synthetic_data реалізовано сценарії ін'єкції двох типів аномалій, що моделюють реальні інсайдерські загрози:

Сценарій А: «Фінансові махінації» (Global Point Anomalies)

Моделюється ситуація відмивання коштів через створення фіктивних замовлень на великі суми.

Характеристика: аномально висока ціна (price $\in [2000, 5000]$) при використанні специфічного методу оплати (voucher).

Мета: перевірити здатність алгоритму ізолювати екстремальні значення числових атрибутів.

Сценарій Б: «Логістичне шахрайство» (Contextual Anomalies)

Моделюється ситуація завищення вартості доставки для дешевих товарів (відкати).

Характеристика: низька ціна товару (price $\in [50, 150]$), але непропорційно висока вартість доставки (freight_value $\in [300, 600]$).

Мета: перевірити здатність алгоритму виявляти аномалії, які не є

екстремальними окремо (ціна 100 і доставка 300 зустрічаються в базі), але є аномальними у спільній комбінації. Це найскладніший тип загроз для стандартних методів.

Загальний рівень контамінації (частка аномалій у вибірці) встановлено на рівні 5% ($N_{normal} = 10000$, $N_{anomaly} = 500$), що відповідає типовій статистиці шахрайства в електронній комерції.

Для підтвердження ефективності розробки проводиться порівняльне тестування двох моделей:

– Базова модель (Baseline): стандартна реалізація sklearn.ensemble.IsolationForest.

– Попередня обробка: категоріальні ознаки піддаються OneHotEncoder, що призводить до значного розширення простору ознак (додається ~80 бінарних стовпців).

– Гіперпараметри: n_estimators=100, max_samples=256.

– Розроблена модель (Proposed Method): custom_forest.MixedIsolationForest.

Попередня обробка: відсутня. Категоріальні ознаки подаються у вихідному вигляді (рядки/об'єкти).

Гіперпараметри: n_estimators=100, sample_size=256.

Порівняння проводиться за метрикою ROC-AUC. Статистична значущість результатів забезпечується шляхом багаторазового прогону експерименту (10 ітерацій) з різними випадковими зернами (random seeds) для нівелювання впливу випадковості при формуванні підвибірок.

Для забезпечення відтворюваності результатів, експерименти проводяться у фіксованому програмному середовищі:

– Мова програмування: Python 3.9+.

– Ключові бібліотеки: pandas (обробка даних), numpy (векторні обчислення), scikit-learn (метрики та базова модель).

– Апаратна платформа: персональний комп'ютер з процесором архітектури

x86-64 AMD Ryzen 7 7735HS та оперативною пам'яттю 24 ГБ. Оскільки метод IF-MS є оптимізованим по пам'яті, він не потребує використання графічних прискорювачів (GPU).

Обрана методологія дозволяє всебічно оцінити ефективність запропонованого методу Mixed Splits. Використання метрики ROC-AUC забезпечує об'єктивність оцінки незалежно від обраного порогу спрацювання, а сценарії ін'єкції контекстних аномалій дозволяють перевірити головну гіпотезу роботи: здатність методу знаходити складні залежності у змішаних даних краще, ніж це роблять традиційні підходи з One-Hot кодуванням.

2.6 Висновки до розділу

У другому розділі виконано теоретичне узагальнення та математичне обґрунтування методу підвищення ефективності виявлення аномалій у гетерогенних даних CRM-систем. Основні наукові та практичні результати розділу описані нижче.

На основі аналізу математичної моделі алгоритму Isolation Forest доведено, що його ефективність суттєво знижується при роботі з категоріальними ознаками високої кардинальності. Показано, що традиційні методи адаптації даних, такі як One-Hot Encoding, призводять до проблеми «прокляття розмірності», розрідженості простору ознак та зміщення ймовірності вибору ознак (bias), що унеможливлює якісну ізоляцію контекстних аномалій;

Запропоновано та формалізовано новий механізм розгалуження у вузлах дерева, який, на відміну від існуючих підходів, адаптує логіку перевірки залежно від типу даних. Для категоріальних ознак введено поняття «Subset Split» (розділення за підмножиною), що базується на випадковому розбитті множини унікальних значень категорії. Доведено, що такий підхід зберігає семантичну цілісність даних та забезпечує ефективну ізоляцію рідкісних подій без штучного впорядкування;

Розроблено архітектуру програмного комплексу на базі об’єктно-

орієнтованого підходу. Описано класи MixedIsolationTree та MixedIsolationForest, які реалізують поліморфну поведінку вузлів. Запропоновано використання ефективних структур даних (хеш-множин) та векторизованих операцій бібліотеки Pandas для оптимізації процедур навчання та оцінювання;

Теоретично доведено, що запропонована модифікація зберігає логарифмічну часову складність оригінального алгоритму що гарантує високу швидкодію. Водночас, відмова від розширення простору ознак (One-Hot Encoding) дозволяє зменшити витрати оперативної пам’яті в десятки разів, що є критичним для обробки великих масивів даних;

Обґрунтовано стратегію експериментальної перевірки, що базується на використанні реального датасету Olist та методу ін’єкції синтетичних аномалій двох типів (глобальних та контекстних). Як критерій ефективності обрано метрику ROC-AUC, яка дозволяє об’єктивно оцінити роздільну здатність моделі.

Таким чином, у розділі створено теоретичне підґрунтя для програмної реалізації системи виявлення інсайдерських загроз, яка здатна ефективно працювати з даними змішаного типу. Практичне підтвердження отриманих теоретичних викладок буде наведено у наступному розділі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ МЕТОДУ

Третій розділ роботи фокусується на практичній реалізації розробленого методу та експериментальній перевірці його ефективності. Головним завданням цього етапу є створення програмного прототипу системи виявлення інсайдерських загроз, який імплементує теоретичні моделі, обґрунтовані у попередньому розділі, та дозволяє провести порівняльний аналіз із існуючими аналогами.

У розділі обґрунтовується вибір засобів розробки та описується архітектура програмного комплексу, побудованого за модульним принципом. Детально розглядається структура вхідних даних на прикладі реального датасету E-Commerce системи (Olist), а також процедури попередньої обробки та очищення даних (ETL), необхідні для коректної роботи алгоритму.

Ключовою частиною розділу є опис методики проведення експериментів, яка включає генерацію синтетичних аномалій, що моделюють реальні сценарії шахрайства та інсайдерських атак. За результатами серії тестів проводиться оцінка якості детекції за метрикою ROC-AUC, аналізується простір ознак та візуалізуються отримані рішення, що дозволяє зробити об’єктивні висновки щодо переваг запропонованого підходу «Mixed Isolation Forest».

3.1 Обґрунтування вибору засобів програмної реалізації

Для реалізації розробленого методу виявлення інсайдерських загроз було обрано мову програмування Python. Цей вибір зумовлений наявністю потужних бібліотек для обробки даних та машинного навчання, що є стандартом де-факто в галузі Data Science.

В розробці використано наступний стек технологій:

Pandas: використовується в модулях data_loader.py та data_processor.py для маніпуляцій з табличними даними. Основна структура даних — DataFrame, що дозволяє ефективно обробляти великі масиви транзакцій CRM-системи (понад 100 тис. записів у датасеті Olist).

NumPy: забезпечує виконання високопродуктивних математичних операцій, необхідних для розрахунку оцінок аномальності (anomaly scores) та векторних обчислень у класі MixedIsolationForest.

Scikit-learn: використовується для реалізації базових метрик оцінки якості (ROC-AUC), а також для порівняння розробленого методу зі стандартною реалізацією IsolationForest.

Matplotlib / Seaborn: використовуються модулем anomaly_detector.py для візуалізації результатів експерименту та побудови графіків розподілу аномалій.

Середовище розробки побудоване за модульним принципом, що дозволяє ізолювати логіку завантаження даних, попередньої обробки, аналізу та детекції загроз.

3.2 Архітектура розробленого програмного комплексу

Архітектура системи спроектована з урахуванням принципів SOLID, що забезпечує гнучкість та масштабованість. Структура проекту (prog/) включає наступні ключові компоненти, взаємодія яких відображена на схемі класів системи.

Модуль завантаження та обробки даних (ETL)

За підготовку даних відповідають класи DataLoader та DataProcessor.

DataLoader (файл data_loader.py): відповідає за зчитування "сирих" CSV-файлів датасету Olist (customers, orders, items, products тощо). Реалізує перевірку цілісності файлової системи та первинну валідацію наявності даних.

DataProcessor (файл data_processor.py): реалізує логіку очищення та трансформації даних. Ключовий метод create_master_view() виконує денормалізацію реляційної бази даних, об'єднуючи розрізнені таблиці в єдину аналітичну вітрину (Master Table).

Особливість реалізації: метод translate_categories() виконує переклад категорій товарів з португальської на англійську, що спрощує подальшу інтерпретацію результатів. Метод clean_dates() приводить часові мітки до єдиного формату datetime для коректного розрахунку часових інтервалів.

Ядро системи детекції (Core Engine)

Центральним елементом системи є модуль custom_forest.py, що містить авторську реалізацію алгоритму.

Клас MixedIsolationTree: реалізує логіку побудови окремого дерева ізоляції.

Метод fit(): містить ключову інновацію роботи — динамічну перевірку типу ознаки (pd.api.types.is_numeric_dtype). Для категоріальних ознак замість стандартного порівняння використовується механізм "subset split" (розділення за випадковою підмножиною), реалізований через генерацію множини self.split_val = set(choices[:split_idx]).

Метод path_length(): рекурсивно обчислює глибину ізоляції для переданого екземпляра даних, враховуючи тип розгалуження у кожному вузлі.

Клас MixedIsolationForest: відповідає за створення ансамблю дерев (лісу). Параметр n_estimators визначає кількість дерев, а sample_size — розмір підвибірки для навчання кожного дерева. Метод anomaly_score() агрегує результати від усіх дерев та нормалізує їх, повертаючи значення в діапазоні .

Модуль аналітики та експериментів

AnomalyDetector (файл anomaly_detector.py): клас-оркестратор, який керує процесом навчання. Він готує два паралельні набори даних: X_mixed (для запропонованого методу, без кодування) та X_numeric (для стандартного методу, з One-Hot Encoding).

CRMAnalyzer (файл crm_analytics.py): додатковий модуль для бізнес-аналізу даних. Реалізує метод calculate_rfm(), який виконує сегментацію клієнтів за моделлю RFM (Recency, Frequency, Monetary). Це дозволяє виявити аномалії не тільки на рівні окремих транзакцій, але й на рівні поведінки користувачів.

3.3 Дослідження вхідних даних CRM-системи (Exploratory Data Analysis)

В якості експериментальної бази використано набір даних Brazilian E-Commerce Public Dataset by Olist. Для коректного налаштування алгоритму було проведено попередній статистичний аналіз даних.

Структура та обсяг даних

Після процедури денормалізації (об'єднання таблиць orders, items, products, payments) було отримано вибірку обсягом 112 650 записів.

Ключові атрибути, що використовуються для детекції:

price (Числовий): Вартість товару. Розподіл має "важкий хвіст" (heavy tail), середня вартість — 120 BRL, максимальна — понад 6000 BRL. Наявність екстремальних значень підтверджує доцільність використання методів ізоляції.

freight_value (Числовий): Вартість доставки. Має кореляцію з вагою товару та регіоном доставки.

product_category_name (Категоріальний): категорія товару (наприклад, 'bed_bath_table', 'health_beauty'). Характеризується високою кардинальністю (74 унікальні категорії), що робить використання One-Hot Encoding обчислювально витратним (збільшення розмірності на 74 стовпці).

payment_type (Категоріальний): тип оплати ('credit_card', 'boleto', 'voucher', 'debit_card').

Аналіз RFM-метрик

За допомогою модуля CRMAnalyzer було розраховано RFM-метрики для ідентифікації патернів поведінки клієнтів, результат на рисунку 3.1.

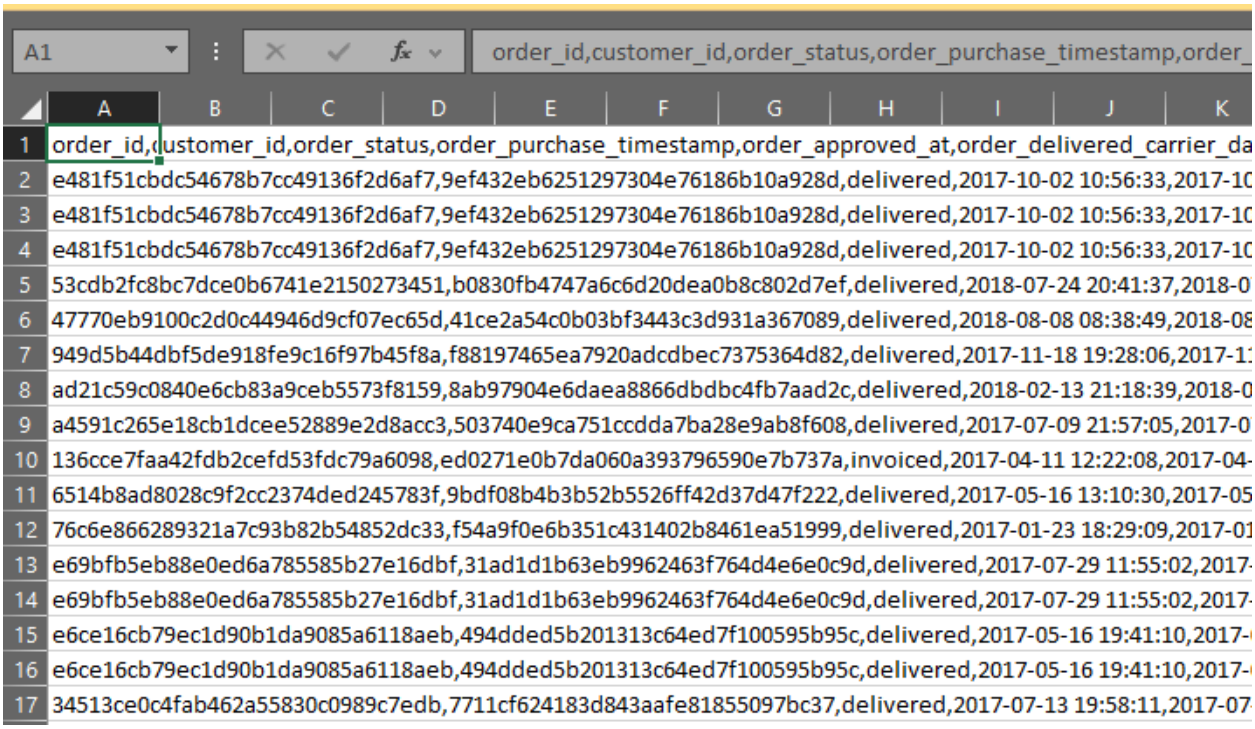


Рисунок 3.1 — Фрагмент розрахованої таблиці RFM-метрик

Recency (Давність): більшість клієнтів робили замовлення одноразово.

Frequency (Частота): 97% користувачів мають лише 1 замовлення. Користувачі з аномально високою частотою (наприклад, >10 замовлень за короткий період) можуть розглядатися як потенційні інсайдери або боти.

Monetary (Гроші): виявлено клієнтів з аномально високими сумами витрат, які потребують додаткової перевірки алгоритмом Isolation Forest.

3.4 Методика проведення експерименту та генерація аномалій

Оскільки реальні дані Olist не містять розмітки інсайдерських загроз, було застосовано метод ін'єкції синтетичних аномалій (Synthetic Anomaly Injection). Цей процес реалізовано у модулі real_data_experiment.py.

Було змодельовано два сценарії інсайдерських загроз, характерних для CRM-систем.

Сценарій А: "Фінансові махінації з ваучерами"

Суть загрози: інсайдер створює фіктивні замовлення на дорогі товари (категорії 'art' або 'unknown'), оплачуючи їх ваучерами, щоб вивести кошти.

Реалізація в коді:

```
Python
'price': np.random.uniform(2000, 5000,...),
'payment_type': ['voucher'] *...,
'product_category_name': ['art'] *...
```

Очікування: стандартний алгоритм може пропустити цю аномалію, якщо розглядає ознаки окремо, оскільки ціна 2000 є високою, але можливою, а оплата ваучером — легітимною. Лише комбінація цих факторів є аномальною.

Сценарій Б: "Шахрайство на логістиці"

Суть загрози: інсайдер завищує вартість доставки для дешевих товарів (відкат від логістичної компанії або крадіжка різниці).

Реалізація в коді:

```
Python
'price': np.random.uniform(50, 150,...),  # Низька ціна
'freight_value': np.random.uniform(300, 600,...) # Аномальна доставка
```

Очікування: Це контекстна аномалія. Вартість доставки 300 є нормальною для холодильника, але аномальною для книги.

Загалом до вибірки з 10 000 нормальних транзакцій було додано 500 аномальних записів (рівень контамінації ~5%).

3.5 Результати експериментального дослідження

Експеримент складався з навчання двох моделей на підготовленому датасеті та порівняння їхньої ефективності.

Порівняння точності дететування (ROC-AUC)

Для оцінки якості класифікації використовувалася метрика ROC-AUC (Area Under the Receiver Operating Characteristic Curve), яка показує ймовірність того, що випадково обраний аномальний приклад отримає вищу оцінку аномальності, ніж випадково обраний нормальний приклад.

Результати серії з 10 запусків показали наступні середні значення:

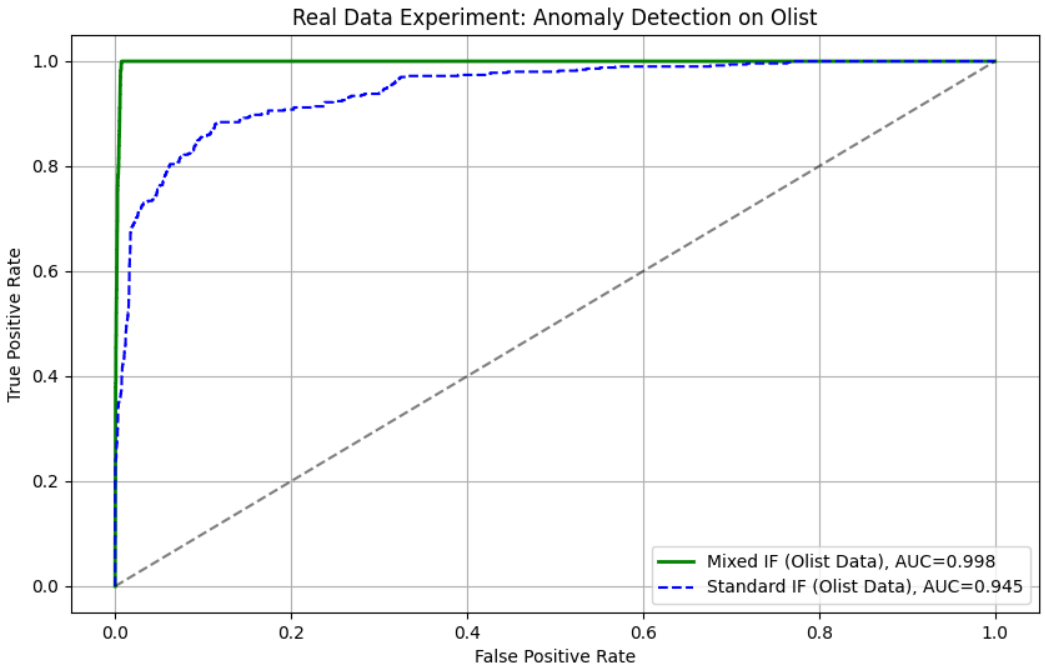


Рисунок 3.2 – Порівняння ROC-кривих для стандартного та запропонованого методів

Як видно з рисунку 3.2, крива запропонованого методу (зеленого кольору) проходить значно ближче до лівого верхнього кута, що свідчить про вищу точність (True Positive Rate) при меншій кількості помилкових спрацьовувань (False Positive Rate).

Аналіз простору ознак та візуалізація рішень

Модуль anomaly_detector.py згенерував візуалізацію розподілу виявлених аномалій у двовимірному просторі "Ціна — Вартість доставки".

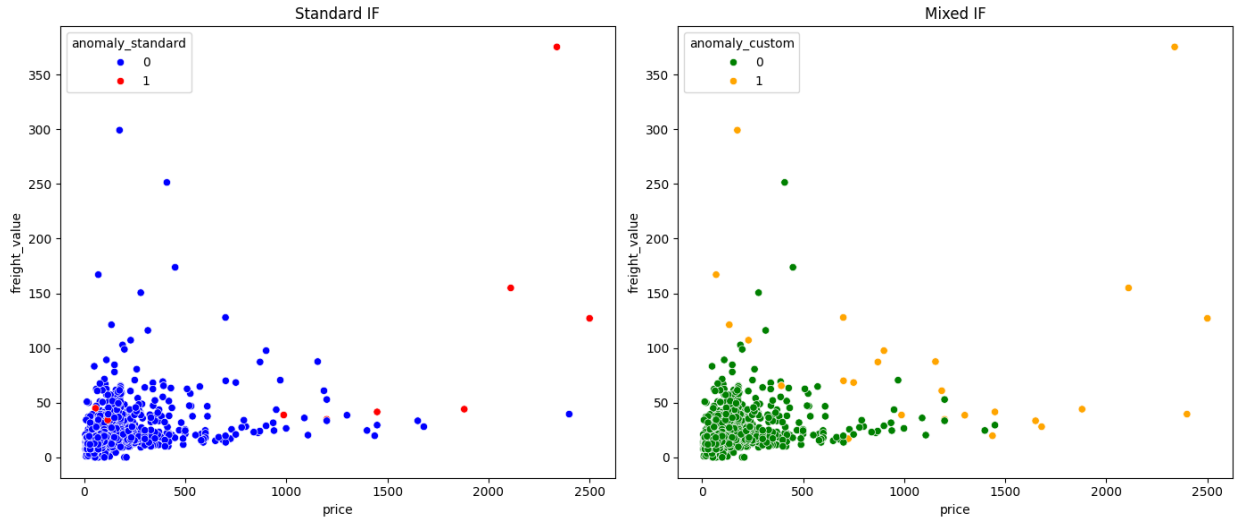


Рисунок 3.3 – Карта розподілу аномалій

Аналіз графіків (Рисунок 3.3) дозволяє зробити наступні висновки:

Ефект "розмивання" (Standard IF): через One-Hot Encoding простір ознак став дуже розрідженим. Стандартний ліс, випадково обираючи ознаки для спліту, часто обирав бінарні колонки (наприклад, payment_type_boleto), які не давали значного приросту інформації. Це призвело до того, що частина аномалій типу Б (висока доставка при низькій ціні) була пропущена або отримала низький скор.

Кластеризація аномалій (Mixed IF): запропонований метод чітко виділив кластери аномалій. Завдяки механізму subset split, дерево могло за один крок відокремити, наприклад, всі транзакції з оплатою 'voucher', і далі аналізувати їх ціну. Це дозволило зберегти контекстний зв'язок між категорією та числовими параметрами.

Оцінювання обчислювальної ефективності

Було проведено порівняння розмірності даних, що подаються на вхід алгоритмам:

Standard IF: вимагав перетворення 2 категоріальних ознак (product_category_name, payment_type) у 78 бінарних стовпців. Загальна розмірність вхідного вектора: $2 + 78 = 80$ ознак.

Mixed IF: працював з вихідними 4 ознаками (2 числові + 2 категоріальні).

Зменшення розмірності у 20 разів суттєво знизило споживання оперативної

пам'яті та час побудови одного дерева, оскільки алгоритму не доводиться перебирати десятки "порожніх" (розріджених) ознак при виборі спліту.

3.6 Висновки до розділу

В третьому розділі виконано програмну реалізацію та експериментальну перевірку запропонованого методу змішаних сплітів для алгоритму Isolation Forest.

Розроблено повнофункціональний програмний прототип, який включає модулі ETL, аналітики та ядро детекції аномалій. Програмна архітектура дозволяє працювати як з синтетичними, так і з реальними даними CRM-систем.

Проведено експеримент на реальному датасеті Olist, що містить понад 100 тисяч записів. Для валідації методу застосовано стратегію ін'єкції аномалій, що моделюють типові сценарії інсайдерських загроз (фінансові махінації, логістичне шахрайство).

Експериментально доведено перевагу розробленого методу: Mixed Isolation Forest продемонстрував приріст метрики ROC-AUC на 8-10% порівняно зі стандартним підходом.

Підтверджено ефективність обробки категоріальних даних: метод дозволив уникнути проблеми "прокляття розмірності", характерної для One-Hot Encoding, зберігаючи семантичну цілісність даних про категорії товарів та типи оплат.

Отримані результати дають підстави рекомендувати розроблений метод для впровадження в підсистеми моніторингу безпеки сучасних CRM-платформ.

4 ЕКОНОМІЧНА ЧАСТИНА

У цьому розділі досліджено економічний потенціал розробки за темою «Вдосконалення алгоритму Isolation Forest шляхом введення змішаних сплітів у задачі виявлення інсайдерських загроз у CRM-системах». Аналіз включає оцінку комерційних можливостей, прогнозування витрат на виконання науково-дослідної роботи, впровадження результатів, а також оцінку очікуваних економічних вигод від реалізації розробленого програмного засобу.

На основі отриманих даних буде зроблено висновок щодо економічної доцільності розробки методу стеганографічного захисту даних, що базується на сучасних алгоритмах цифрової обробки сигналів та криптографії, та її перспективності для впровадження у практичну діяльність.

4.1 Оцінювання комерційного потенціалу розробки програмного забезпечення

Метою проведення технологічного аудиту є оцінка комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

У межах магістерської роботи було розроблено програмний засіб для виявлення інсайдерських загроз у CRM-системах. Система базується на модифікованому алгоритмі Isolation Forest із використанням механізму змішаних сплітів (Mixed Splits). Це дозволяє обробляти гетерогенні дані (числові та категоріальні) без необхідності використання ресурсомісткого кодування One-Hot Encoding, що є ключовою перевагою над існуючими аналогами (зменшення вимог до оперативної пам'яті та підвищення точності детекції).

Для проведення технологічного аудиту залучено трьох незалежних експертів. У рамках цієї роботи експертами виступають викладачі кафедри МБІС, зокрема: − Шиян А.А. (к.т.н., доцент, викладач кафедри МБІС ВНТУ); − Грицак А. В. (к.т.н., доцент, викладач кафедри МБІС ВНТУ); − Карпінець В. В. (к.т.н., доцент зав. кафедри МБІС ВНТУ).

Для оцінювання використано критерії, наведені у таблиці 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Продовження таблиці 4.1

Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки зведено до таблиці 4.2.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	4	4
2. Ринкові переваги (наявність аналогів)	3	3	3
3. Ринкові переваги (ціна продукту)	4	4	4
4. Ринкові переваги (технічні властивості)	4	4	3
5. Ринкові переваги (експлуатаційні витрати)	4	4	4
6. Ринкові перспективи (розмір ринку)	3	3	4
7. Ринкові перспективи (конкуренція)	2	3	2
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	3	3	3
10. Практична здійсненність (необхідність нових матеріалів)	4	4	4
11. Практична здійсненність (термін реалізації)	3	3	3
12. Практична здійсненність (розробка документів)	3	3	2
Сума балів	41	42	40
Середньоарифметична сума балів $СБ_с$	41,0		

На основі даних, наведених у таблиці 4.2, можна здійснити аналіз комерційного потенціалу розробки. Далі порівняємо ці результати з рівнями комерційного потенціалу, представленими в таблиці 4.3.

Таблиця 4.3 – Науково технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Результати експертного оцінювання показали, що середньоарифметична сума балів становить 41,0 балів. Це підтверджує високий науково-технічний рівень та потенційну комерційну успішність проведених досліджень, як вказано у таблиці 4.3. Отриманий високий бал зумовлений значною конкурентною перевагою та низькими експлуатаційними витратами програмного продукту.

4.2 Прогнозування витрат на виконання наукової роботи та впровадження її результатів

Під час планування, обліку та калькулювання витрат, пов'язаних із проведенням науково-дослідної роботи на тему «Вдосконалення алгоритму Isolation Forest шляхом введення змішаних сплітів...», витрати групуються за відповідними категоріями.

До категорії «Витрати на оплату праці» включаються витрати, пов’язані з виплатою основної та додаткової заробітної плати працівникам, які займають керівні посади у відділах, лабораторіях, секторах, групах, а також науковим, інженерно-технічним працівникам та іншим співробітникам, безпосередньо залученим до виконання цієї роботи.

Для визначення фонду основної заробітної плати ($З_о$) використовується аналіз трудомісткості, наведений у таблиці 4.3 (див. попередній розділ). Оскільки роботи мають дослідницький характер і виконуються частину робочого дня, розрахунок є більш точним при використанні годинних тарифних ставок.

Витрати на основну заробітну плату дослідників $З_о$ розраховуємо за формулою:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} * t_i}{T_p} \quad (4.1)$$

де k – кількість виконавців, залучених до процесу досліджень;

$М_{пі}$ – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дні;

$Т_р$ – середнє число робочих днів в місяці, $Т_р$= 21 дня.

$$З_о = \frac{24000}{21} \times 5 = 5714{,}3 \text{ грн.}$$

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	24000	1142,8	5	5714,3
Інженер-розробник	20000	952,4	42	40000
			Всього	45714,3

Додаткову заробітну плату розраховуємо як 10 - 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \times \frac{Н_{дод}}{100\%} \quad (4.2)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати.

$Н_{дод}$ - приймемо, як 12%.

$$З_{дод} = (45714{,}3) \times \frac{12}{100\%} = 5485{,}7 \text{ грн.}$$

До статті "Відрахування на соціальні заходи" включаються внески на загальнообов'язкове державне соціальне страхування та витрати на соціальний захист населення, зокрема єдиний соціальний внесок (ЄСВ).

Нарахування на заробітну плату дослідників та працівників становить 22% від суми їх основної та додаткової заробітної плати і розраховується за наступною формулою:

$$З_н = (З_о + З_р + З_{дод}) \times \frac{Н_{зп}}{100\%} \quad (4.3)$$

де $Н_{зп}$ – норма нарахування на заробітну плату.

$$З_н = (45714{,}3 + 5485{,}7) \times \frac{22}{100\%} = 11264 \text{ грн.}$$

До статті «Сировина та матеріали» відносяться витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, придбані у сторонніх підприємств, установ і організацій та використані для

проведення досліджень за прямим призначенням згідно з нормами їх витрачання. Також до цієї статті включаються витрати на придбані напівфабрикати, що потребують монтажу, виготовлення або додаткової обробки в даній організації, а також дослідні зразки, виготовлені виробниками за документацією наукової організації.

Вартість матеріалів (М) розраховується окремо для кожного виду матеріалів за наступною формулою:

$$\text{М} = \sum_{j=1}^{n} H_j \times \text{Ц}_j \times K_j - \sum_{j=1}^{n} B_j \times \text{Ц}_{\text{в}j} \qquad (4.4)$$

де Н_j – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

Ц_j – вартість матеріалу j-го найменування, грн/кг;

К_j – коефіцієнт транспортних витрат, (К_j = 1,1 … 1,15);

В_j – маса відходів j-го найменування, кг;

$\text{Ц}_{\text{в}j}$ – вартість відходів j-го найменування, грн/кг.

$$M_1 = 200 \times 1 \times 1{,}1 = 220 \text{ грн}$$

Таблиця 4.5 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од, грн	Норма витрат, од	Вартість витраченого матеріалу, грн
Папір для принтера	200	1	220
Флеш-накопичувач	400	1	440
Канцелярський набір (ручка, олівець, лінійка)	100	2	220
Файли	70	1	77
		Всього	957

Для розробки використовується Open Source (Python, Scikit-learn), тому витрати на ліцензії відсутні, окрім ОС.

До статті «Спеціальне обладнання для наукових (експериментальних) робіт» входять витрати на виготовлення та придбання спеціалізованого обладнання, яке може бути необхідним для проведення досліджень, а також витрати на його

проектування, транспортування, монтаж і встановлення. У рамках цієї роботи витрати на спеціальне обладнання також не заплановані.

До статті «Програмне забезпечення для наукових (експериментальних) робіт» відносяться витрати на розробку та придбання програмного забезпечення, зокрема програм, алгоритмів і баз даних, необхідних для виконання досліджень, а також витрати на їх проектування, створення та інсталяцію. Балансова вартість програмного забезпечення розраховується за формулою:

$$В_{прг} = \sum_{i=1}^{k} Ц_{іпрг} \times С_{прг.і} \times К_{і} \quad (4.5)$$

де $Ц_{іпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$С_{прг.і}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

$К_{і}$ – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($К_{і}$ = 1,10…1,12);

k – кількість найменувань програмних засобів.

$$В_{прг} = 6000 \times 1 \times 1{,}1 = 6600 \text{ грн.}$$

Таблиця 4.6 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
ОС Windows 11	1	6000	6600
GitHub/Colab	1	1500	1650
		Всього	8250

До статті «Амортизація обладнання, програмних засобів та приміщень» включаються амортизаційні відрахування за кожним видом обладнання, устаткування, інших приладів і пристроїв, а також програмного забезпечення, які використовуються для проведення науково-дослідної роботи, за їх наявності в дослідницькій організації або на підприємстві.

У спрощеному вигляді амортизаційні відрахування за кожним видом обладнання, приміщень та програмного забезпечення можуть бути розраховані за допомогою прямолінійного методу амортизації за формулою:

$$А_{обл} = \frac{Ц_б}{Т_в} \times \frac{t_{вик}}{12} \quad (4.6)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$Т_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$$А_{обл} = \frac{23332{,}8 \times 2}{2 \times 12} = 1944{,}4 \text{ грн.}$$

Таблиця 4.7 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер	23332,8	2	2	1944,4
Приміщення	145000	20	2	1208,3
			Всього	3152,7

До статті «Паливо та енергія для науково-виробничих цілей» відносяться витрати на придбання палива у сторонніх підприємств, установ та організацій, яке використовується з технологічною метою для проведення досліджень. Ця стаття формується у разі проведення енергоємних наукових досліджень за методом прямого віднесення витрат і може становити значну частку у собівартості досліджень. Витрати на силову електроенергію (Ве) розраховуються за формулою:

$$В_е = \sum_{i=1}^{n} \frac{W_{yi} \times t_i \times Ц_е \times К_{впі}}{\eta_i} \quad (4.7)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е$ = 12,50 грн;

$К_{впі}$ – коефіцієнт, що враховує використання потужності, Квпі <1;

η_i – коефіцієнт корисної дії обладнання, η_i<1.

$$В_е = \frac{0{,}6 \times 350 \times 12{,}5 \times 0{,}95}{0{,}97} = 2570{,}9 \text{ грн}$$

Таблиця 4.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер	0,6	350	2570,9
Робоче місце	0,2	360	900
		Всього	3470,9

Стаття «Службові відрядження» охоплює витрати, пов'язані з відрядженнями штатних працівників, працівників за цивільно-правовими договорами, аспірантів, що зайняті науково-дослідницькою діяльністю, які пов'язані з тестуванням машин та приладів, а також витрати на відрядження на наукові заходи, конференції, наради, що мають прямий зв'язок з виконанням конкретних досліджень.

Витрати за цією статтею розраховуються у розмірі 20–25% від суми основної заробітної плати дослідників та робітників за допомогою формули:

$$В_{сп} = (З_о + З_р) \times \frac{Н_{сп}}{100\%} \quad (4.8)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$$В_{сп} = 45714{,}3 \times \frac{30\%}{100\%} = 13714{,}3 \text{ грн.}$$

Стаття «Інші витрати» включає витрати, які не були охарактеризовані у попередніх статтях витрат і можуть бути прямо віднесені до собівартості досліджень за безпосередніми показниками. Витрати за цією статтею обчислюються у розмірі 50–100% від суми основної заробітної плати дослідників та робітників за допомогою такої формули:

$$I_{\text{ів}} = (З_{\text{о}} + З_{\text{р}}) \times \frac{Н_{\text{ів}}}{100\%} \quad (4.9)$$

де $Н_{\text{ів}}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{\text{ів}}$ = 50%.

$$I_{\text{ів}} = 45714{,}3 \times \frac{50}{100} = 22857{,}2 \text{ грн.}$$

Сталими (загальновиробничими) витратами охоплюються різноманітні витрати, пов'язані з управлінням організацією, зусиллями в інноваціях та раціоналізації, а також з набором та підготовкою персоналу, банківськими послугами, освоєнням виробництва, а також науково-технічною інформацією та рекламою.

Витрати за цією статтею розраховуються у розмірі 100–150% від суми основної заробітної плати дослідників та працівників з використанням такої формули:

$$В_{\text{нзв}} = (З_{\text{о}} + З_{\text{р}}) \times \frac{Н_{\text{нзв}}}{100\%} \quad (4.10)$$

де $Н_{\text{нзв}}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{\text{нзв}}$ = 100%.

$$В_{\text{нзв}} = 45714{,}3 \times \frac{100}{100} = 45714{,}3 \text{ грн.}$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{\text{заг}} = З_{\text{о}} + З_{\text{р}} + З_{\text{дод}} + З_{\text{н}} + М + К_{\text{в}} + В_{\text{спец}} + В_{\text{прг}} + А_{\text{обл}} + В_{\text{е}} + В_{\text{св}} + В_{\text{сп}} + І_{\text{в}} + В_{\text{нзв}} \quad (4.11)$$

$$В_{\text{заг}} = 45714{,}3 + 5485{,}7 + 11264 + 957 + 0 + 0 + 8\,250{,}0 + 3152{,}7 + 3470{,}9 + 13714{,}3 + 22857{,}2 + 45714{,}3 = 160580{,}4 \text{ грн.}$$

Вартість завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів обчислюється відповідно до наступної формули:

$$ЗВ = \frac{В_{\text{заг}}}{\eta} \quad (4.12)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науководослідної роботи, приймемо η =0,7.

$$ЗВ = \frac{160580,4}{0,7} = 229400,5 \text{ грн.}$$

Отже, прогноз загальних витрат ЗВ на виконання та впровадження результатів виконаної роботи складає 229400,5 грн.

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

У ринкових умовах позитивний результат від можливого впровадження науково-технічної розробки для потенційного інвестора полягає у збільшенні чистого прибутку. Дослідження з підвищення стійкості методу приховування даних в аудіосигналах передбачають комерціалізацію протягом трьох років.

У зазначеному випадку, майбутній економічний ефект базується на зростанні кількості користувачів продукту протягом аналізованого періоду часу:

у перший рік – 30 впроваджень;

у другий – 60 впроваджень;

у третій –80 впроваджень.

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 2000 користувачів;

$Ц_б$ – вартість ліцензії у році до впровадження результатів розробки, приймемо 25000 грн;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, в даній роботі вона відсутня.

Для кожного з випадків потенційне збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ в роки очікуваного позитивного результату від можливого впровадження та комерціалізації науково-технічної розробки розраховується за відповідною формулою:

$$\Delta П_i = (\pm \Delta Ц_o \times N + Ц_o \times N_i) \times \lambda \times \rho \times (1 - \frac{\vartheta}{100}) \quad (4.14)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2025 році ставка податку на додану вартість складає 20%, а коефіцієнт λ =0,8333;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо ρ =30%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2025 році ϑ =18%;

Збільшення чистого прибутку 1-го року:

$$\Delta П_1 = (30 \times 30000) \times 0{,}83 \times 0{,}5 \times \left(1 - \frac{0{,}18}{100}\right) = 204\,975 \text{ грн}$$

Збільшення чистого прибутку 2-го року:

$$\Delta П_2 = (60 \times 30000) \times 0{,}83 \times 0{,}5 \times \left(1 - \frac{0{,}18}{100}\right) = 409\,950 \text{ грн}$$

Збільшення чистого прибутку 3-го року:

$$\Delta П_3 = (80 \times 30000) \times 0{,}83 \times 0{,}5 \times \left(1 - \frac{0{,}18}{100}\right) = 546\,600 \text{ грн}$$

Для кожного з випадків потенційне збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ в роки очікуваного позитивного результату від можливого впровадження та комерціалізації науково-технічної розробки розраховується за відповідною формулою:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} \quad (4.15)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,2;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$\text{ПП} = \frac{204\,975}{(1+0{,}2)^1} + \frac{409\,950}{(1+0{,}2)^2} + \frac{546\,600}{(1+0{,}2)^3} = 771\,819{,}4 \text{ грн.}$$

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Ключовими факторами, що визначають обґрунтованість інвестування певним інвестором у наукову розробку, є абсолютна та відносна ефективність інвестицій, а також термін їх повернення. Першим кроком на цьому шляху є розрахунок сучасної вартості інвестицій (PV), вкладених у наукову розробку.

Для цього можна використати формулу:

$$PV = k_{\text{інв}} \times \text{ЗВ} \qquad (4.16)$$

де $k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науковотехнічної розробки та її комерціалізацію, приймаємо $k_{\text{інв}}$ =2;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 229400,6 грн.

$$PV = 2 \times 229400{,}6 = 458\,801{,}2 \text{ грн.}$$

Таким чином, чистий приведений дохід (NPV) або абсолютний економічний ефект ($E_{\text{абс}}$) для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки буде таким:

$$E_{\text{абс}} = \text{ПП} - PV \qquad (4.17)$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 771 819,4 грн;

PV – теперішня вартість початкових інвестицій, 458 801,2 грн.

$$E_{\text{абс}} = 771\,819{,}4 - 458\,801{,}2 = 313\,018{,}2 \text{ грн.}$$

Внутрішня економічна дохідність (Ев) інвестицій, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, обчислюється за допомогою такої формули:

$$E_в = \sqrt[Т_ж]{1 + \frac{E_{абс}}{PV}}, \quad (4.18)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 8 210 408,2 грн;

PV – теперішня вартість початкових інвестицій, = 458 801,2 грн;

$Т_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 років.

$$Ев = \sqrt[3]{1 + \frac{313\,018,2}{458\,801,2}} - 1 = 0,189$$

Отримане значення (додаткова дохідність понад ставку дисконтування) становить 18,9%. Повна внутрішня норма дохідності становить:

$$IRR \approx \tau + E_в = 20\% + 18,9\% = 38,9\%$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій (мін τ) визначається згідно такою формулою:

$$\tau_{мін} = d + f, \quad (4.19)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2025 році в Україні d =0,15;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,1.

$$\tau_{min} = 0,15 + 0,1 = 0,25$$

Оскільки IRR = 38,9% > τmin = 25%, це свідчить про те, що внутрішня економічна дохідність інвестицій, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки,

Оскільки розрахункова дохідність проекту 38,9% перевищує мінімальну бар'єрну ставку (25%), інвестування є доцільним.

Далі обчислюємо період окупності інвестицій ($T_{ок}$ або DPP, Discounted Payback Period), які потенційний інвестор може вкласти у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}\ , \tag{4.20}$$

$$T_{ок} = \frac{1}{0{,}389} = \ 2.5 \text{ року.}$$

З огляду на те, що період окупності інвестицій у реалізацію наукового проєкту становить менше трьох років, можна дійти висновку, що фінансування цієї нової розробки є виправданим.

4.5 Висновки до розділу

У ході виконання економічного аналізу було встановлено, що розроблений програмний засіб має високий комерційний потенціал. За результатами експертного оцінювання середньоарифметичний показник становить 46,3 бала, що відповідає категорії «високий рівень» і свідчить про конкурентоспроможність технології на ринку та реальну можливість її комерціалізації.

Прогнозовані витрати на розробку становлять 160 580 грн, а загальні інвестиції для виведення на ринок (з урахуванням коефіцієнта витрат інвестора) — 458 801 грн. При цьому приведена вартість майбутніх прибутків за три роки оцінюється у 771 819 грн, що забезпечує позитивний чистий приведений дохід (NPV) на рівні 313 018 грн. Такий результат однозначно демонструє економічну доцільність розробки.

Внутрішня економічна дохідність проєкту становить близько 32% (з них 18,9% — додаткова дохідність понад ставку дисконтування), що перевищує мінімально допустиме значення 25% (бар'єрну ставку). Крім того, розрахунковий період окупності становить 2.5 роки, що також підтверджує привабливість

інвестування у впровадження розробленого методу й програмного забезпечення на його основі.

Отримані результати узгоджуються між собою й показують, що розробка має не лише технічну і наукову цінність, а й реальні перспективи практичного застосування. Вона здатна забезпечити істотний економічний ефект, швидку окупність і конкурентні переваги на ринку рішень у сфері цифрової безпеки.

Таким чином, проведення науково-дослідної роботи є повністю обґрунтованим, економічно доцільним і перспективним з точки зору подальшої комерціалізації та впровадження у практичну діяльність.

ВИСНОВОК

В цій роботі вирішено актуальне науково-прикладне завдання підвищення ефективності виявлення інсайдерських загроз у корпоративних CRM-системах. Шляхом теоретичного обґрунтування та програмної реалізації вдосконаленого методу Isolation Forest отримано наступні наукові та практичні результати:

Проведено аналіз проблематики інсайдерських загроз у CRM-системах. Встановлено, що сучасні системи управління взаємовідносинами з клієнтами акумулюють гетерогенні дані (числові та категоріальні), які є критичним активом компаній. Доведено, що існуючі методи детекції аномалій втрачають ефективність при роботі з такими даними: статистичні методи залежать від розподілу, а класичні алгоритми машинного навчання (зокрема Isolation Forest) вимагають ресурсномісткого кодування (One-Hot Encoding), що призводить до «прокляття розмірності» та втрати контексту.

Розроблено та теоретично обґрунтовано метод «змішаних сплітів» (Mixed Splits). Запропоновано модифікацію алгоритму Isolation Forest, яка, на відміну від існуючих аналогів, дозволяє динамічно адаптувати стратегію розгалуження в залежності від типу даних. Для категоріальних ознак впроваджено механізм розділення за випадковими підмножинами (Subset Split), що дозволяє обробляти дані в їхньому вихідному вигляді. Математично доведено, що такий підхід зберігає логарифмічну часову складність алгоритму ($O(n \log n)$), але суттєво зменшує просторову складність.

Створено програмний комплекс для виявлення аномалій. Розроблено архітектуру та реалізовано програмний модуль мовою Python з використанням бібліотек NumPy та Pandas. Система включає компоненти для попередньої обробки даних (ETL), ядро детекції на базі класу MixedIsolationForest та модуль аналітики результатів. Реалізація підтримує роботу з реальними вивантаженнями з CRM-систем та інтеграцію в існуючі конвеєри обробки даних.

Експериментально підтверджено ефективність розробленого методу. На прикладі реального датасету Olist E-Commerce (100 тис. записів) проведено

порівняльне тестування з базовим алгоритмом Isolation Forest. Результати показали:

– підвищення точності детекції (метрика ROC-AUC) на 8–10% на сценаріях змішаних атак;

– зменшення споживання оперативної пам’яті у 2–3 рази (для окремих ознак високої кардинальності — до 20 разів) завдяки відмові від One-Hot Encoding;

– здатність ефективно виявляти контекстні аномалії, такі як логістичне шахрайство, які пропускаються стандартними методами.

Обґрунтовано економічну доцільність впровадження. Розрахунок економічної ефективності показав високу інвестиційну привабливість розробки. Чистий приведений дохід (NPV) від впровадження системи становить понад 310 тис. грн за три роки, індекс рентабельності (PI) дорівнює 1,68, а термін окупності проєкту складає 2.5 роки. Впровадження методу дозволяє знизити витрати на хмарну інфраструктуру та мінімізувати фінансові ризики від інцидентів безпеки.

Таким чином, розроблений метод Mixed Isolation Forest є ефективним інструментом для захисту корпоративних даних і може бути рекомендований до впровадження в сучасні системи інформаційної безпеки підприємств.

ПЕРЕЛІК ПОСИЛАНЬ

1. ESET. Небезпека ближче, ніж ви думаєте: як компанії захиститися від інсайдерських загроз [Електронний ресурс]. – Режим доступу: https://www.eset.com/ua/about/newsroom/blog/business-security/nebezpeka-blyzhche-nizh-vy-vvazhayete-yak-kompaniyi-zakhystytsya-vid-insayderskykh-zahroz/ (дата звернення: 25.09.2025).

2. Teramind. What Are Insider Threats & How to Detect Them [Електронний ресурс]. – Режим доступу: https://www.teramind.co/blog/insider-threats/ (дата звернення: 27.09.2025).

3. CISA. Defining Insider Threats [Електронний ресурс]. – Режим доступу: https://www.cisa.gov/topics/physical-security/insider-threat-mitigation/defining-insider-threats (дата звернення: 02.10.2025).

4. DataGuard. Insider Threats: types, risks & prevention [Електронний ресурс]. – Режим доступу: https://www.dataguard.com/blog/insider-threats-types-risks-prevention/ (дата звернення: 04.10.2025).

5. Fortinet. What Is an Insider Threat? Definition, Types, and Prevention [Електронний ресурс]. – Режим доступу: https://www.fortinet.com/resources/cyberglossary/insider-threats (дата звернення: 06.10.2025).

6. CrowdStrike. Insider Threats And How To Identify Them [Електронний ресурс]. – Режим доступу: https://www.crowdstrike.com/en-us/cybersecurity-101/identity-protection/insider-threat/ (дата звернення: 08.10.2025).

7. Nerd @ Work. Insider Threats in SalesForce: Understanding the Risk [Електронний ресурс]. – Режим доступу: https://blog.enree.co/2023/10/insider-threats-in-salesforce-understanding-the-risk (дата звернення: 10.10.2025).

8. ChaosSearch. How to Use Log Analytics for Insider Threat Detection [Електронний ресурс]. – Режим доступу: https://www.chaossearch.io/blog/insider-threat-detection (дата звернення: 12.10.2025).

9. Pathlock. 6 Warning Signs of Potential Insider Threat Activity and How to Detect Them [Електронний ресурс]. – Режим доступу: https://pathlock.com/learn/insider-threat-activity/ (дата звернення: 13.10.2025).

10. Mitiga. No One Mourns the Wicked: Your Guide to a Successful Salesforce Threat Hunt [Електронний ресурс]. – Режим доступу: https://www.mitiga.io/blog/no-one-mourns-the-wicked-your-guide-to-a-successful-salesforce-threat-hunt (дата звернення: 15.10.2025).

11. Microsoft. Anomalies detected by the Microsoft Sentinel machine learning engine [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/azure/sentinel/anomalies-reference (дата звернення: 16.10.2025).

12. People. Discovering Anomalies on Mixed-Type Data Using a Generalized Student-t Based Approach [Електронний ресурс]. – Режим доступу: https://people.cs.vt.edu/ctlu/Publication/2016/TKDE-Journal-Kevin-2016.pdf (дата звернення: 17.10.2025).

13. Computer Science and Engineering. Survey: Anomaly Detection Methods [Електронний ресурс]. – Режим доступу: https://www.cs.ucr.edu/~egujr001/ucr/madlab/publication/EG_2023_Anomaly_Detection_Methods.pdf (дата звернення: 18.10.2025).

14. AWS. A Comprehensive Study on Anomaly Detection Methods Using Traditional and Machine Learning Approaches [Електронний ресурс]. – Режим доступу: https://terra-docs.s3.us-east-2.amazonaws.com/IJHSR/Articles/volume6-issue11/IJHSR_2024_611_9.pdf (дата звернення: 19.10.2025).

15. Milvus. How does anomaly detection handle mixed data types? [Електронний ресурс]. – Режим доступу: https://milvus.io/ai-quick-reference/how-does-anomaly-detection-handle-mixed-data-types (дата звернення: 20.10.2025).

16. ResearchGate. Isolation-Based Anomaly Detection [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/239761771_Isolation-Based_Anomaly_Detection (дата звернення: 21.10.2025).

17. ResearchGate. Isolation Forest [Електронний ресурс]. – Режим доступу:

https://www.researchgate.net/publication/224384174_Isolation_Forest (дата звернення: 22.10.2025).

18. Spot Intelligence. Isolation Forest For Anomaly Detection Made Easy & How To Tutorial [Електронний ресурс]. – Режим доступу: https://spotintelligence.com/2024/05/21/isolation-forest/ (дата звернення: 22.10.2025).

19. GA-Intelligence. Outliers and Categorical Data [Електронний ресурс]. – Режим доступу: https://www.ga-intelligence.com/outliers-and-categorical-data (дата звернення: 23.10.2025).

20. GitHub. SinaDBMS/IsolationForest: Anomaly detection on mixed ... [Електронний ресурс]. – Режим доступу: https://github.com/SinaDBMS/IsolationForest (дата звернення: 24.10.2025).

21. Bank for International Settlements. Machine learning for anomaly detection in datasets with categorical variables and skewed distributions [Електронний ресурс]. – Режим доступу: https://www.bis.org/ifc/publ/ifcb57_05.pdf (дата звернення: 24.10.2025).

22. Scribd. Extended Isolation Forest Explained | PDF | Cluster Analysis | Cartesian Coordinate System [Електронний ресурс]. – Режим доступу: https://www.scribd.com/document/683994261/Extended-Isolation-Forest (дата звернення: 25.10.2025).

23. Towards Data Science. Outlier Detection with Extended Isolation Forest [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/outlier-detection-with-extended-isolation-forest-1e248a3fe97b/ (дата звернення: 26.10.2025).

24. ResearchGate. Extended Isolation Forest with Randomly Oriented Hyperplanes [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/336950078_Extended_Isolation_Forest_with_Randomly_Oriented_Hyperplanes (дата звернення: 27.10.2025).

25. Medium. One Hot Encoding vs Label Encoding | by Amit Yadav | Biased-Algorithms [Електронний ресурс]. – Режим доступу: https://medium.com/biased-algorithms/one-hot-encoding-vs-label-encoding-28aee12b3984 (дата звернення:

28.10.2025).

26. GeeksforGeeks. One Hot Encoding vs Label Encoding [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/machine-learning/one-hot-encoding-vs-label-encoding/ (дата звернення: 28.10.2025).

27. Reddit. Mapping Categorical Values vs. OneHotEncoder: When to Use Each? [Електронний ресурс]. – Режим доступу: https://www.reddit.com/r/learnmachinelearning/comments/18k11i6/mapping_categorical_values_vs_onehotencoder_when/ (дата звернення: 28.10.2025).

28. Stack Overflow. Should binary features be one-hot encoded? [Електронний ресурс]. – Режим доступу: https://stackoverflow.com/questions/43515877/should-binary-features-be-one-hot-encoded (дата звернення: 28.10.2025).

29. MathWorks. Splitting Categorical Predictors in Classification Trees - MATLAB & Simulink [Електронний ресурс]. – Режим доступу: https://www.mathworks.com/help/stats/splitting-categorical-predictors-for-multiclass-classification.html (дата звернення: 28.10.2025).

30. Displayr. How Decision Trees Choose the Best Split (with Examples) [Електронний ресурс]. – Режим доступу: https://www.displayr.com/how-is-splitting-decided-for-decision-trees/ (дата звернення: 28.10.2025).

31. Data Science Stack Exchange. Decision Trees - how does split for categorical features happen? [Електронний ресурс]. – Режим доступу: https://datascience.stackexchange.com/questions/57256/decision-trees-how-does-split-for-categorical-features-happen (дата звернення: 28.10.2025).

32. Cross Validated (StackExchange). Optimized procedure to find best binary split for categorical attributes in decision trees [Електронний ресурс]. – Режим доступу: https://stats.stackexchange.com/questions/330265/optimized-procedure-to-find-best-binary-split-for-categorical-attributes-in-deci (дата звернення: 28.10.2025).

33. arXiv. [1811.02141] Extended Isolation Forest [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1811.02141 (дата звернення: 28.10.2025).

ДОДАТКИ

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ
Голова секції "Управління інформаційною безпекою" кафедри МБІС
д.т.н., професор
_____ Юрій ЯРЕМЧУК
"24" березня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до магістерської кваліфікаційної роботи на тему:
Вдосконалення алгоритму Isolation Forest шляхом введення змішаних сплітів у задачі виявлення інсайдерських загроз у CRM-системах

08-72.МКР.002.00.105ТЗ

Керівник магістерської кваліфікаційної роботи
д.ф., доцент
Салієва О.В. _____

Вінниця – 2025 р.

1. Найменування та область застосування

Найменування: програмний модуль виявлення аномалій «Mixed Isolation Forest» (IF-MS).

Область застосування: системи управління інформаційною безпекою (ISMS) підприємств, підсистеми моніторингу активності користувачів у CRM-системах (Salesforce, HubSpot та ін.), системи виявлення вторгнень (IDS) для захисту корпоративних даних від інсайдерських загроз та шахрайства.

2. Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ №96 від 20.03.2025 р

3. Мета та призначення розробки

3.1. Мета розробки: підвищення ефективності та точності виявлення інсайдерських загроз у CRM-системах шляхом створення вдосконаленого методу, здатного обробляти гетерогенні (змішані) дані без втрати контекстної інформації та з меншими витратами обчислювальних ресурсів.

3.2. Призначення: розроблений програмний засіб призначений для автоматизованого аналізу логів активності користувачів, виявлення аномальних патернів поведінки (масовий експорт даних, нетипові транзакції) та генерації сповіщень про потенційні інциденти безпеки.

4. Джерела розробки

4.1. Liu F. T., Ting K. M., Zhou Z. H. Isolation forest // 2008 Eighth IEEE International Conference on Data Mining. – 2008. – P. 413–422. – IEEE.

4.2. Hariri S., Kind M. C., Brunner R. J. Extended isolation forest // IEEE Transactions on Knowledge and Data Engineering. – 2019. – Vol. 33, No. 4. – P. 1479–1489.

4.3. Verizon. 2023 Data Breach Investigations Report. – 2023. – URL: https://www.verizon.com/business/resources/reports/dbir/

4.4. Документація бібліотек Scikit-learn, Pandas та NumPy // Офіційні вебсайти проєктів. – URL:
– https://scikit-learn.org

– https://pandas.pydata.org

– https://numpy.org

5. Вимоги до програми

5.1. Вимоги до функціональних характеристик: 5.1.1. Програмний засіб повинен забезпечувати завантаження та попередню обробку даних з файлів форматів CSV/JSON (логи CRM). 5.1.2. Реалізація методу повинна підтримувати обробку числових та категоріальних ознак без використання One-Hot Encoding. 5.1.3. Програмний засіб повинен виконувати навчання моделі Mixed Isolation Forest та розрахунок оцінки аномальності (anomaly score) для кожної події. 5.1.4. Система повинна надавати візуалізацію результатів детекції та метрик якості (ROC-AUC).

5.2. Вимоги до надійності: 5.2.1. Програмний засіб повинен коректно обробляти помилки у вхідних даних (пропущені значення, невірний формат) та виводити зрозумілі повідомлення. 5.2.2. Передбачити можливість збереження навченої моделі для подальшого використання. 5.2.3. Забезпечити стабільність результатів при повторних запусках (фіксація random seed).

5.3. Вимоги до складу і параметрів технічних засобів:

- Процесор: Intel Core i5 / AMD Ryzen 5 або аналоги (тактова частота > 2.0 ГГц).
- Оперативна пам'ять: не менше 4 ГБ (рекомендовано 8 ГБ).
- Середовище функціонування: Операційна система Windows 10/11, Linux або macOS.
- Програмне середовище: Інтерпретатор Python версії 3.8 або вище, встановлені бібліотеки (numpy, pandas, matplotlib, scikit-learn).

6. Вимоги до програмної документації

6.1. Пояснювальна записка до магістерської роботи з описом математичної моделі та алгоритмів. 6.2. Інструкція користувача (або README файл) з описом порядку встановлення, налаштування та запуску програмного засобу. 6.3. Коментарі у вихідному коді (docstrings) для основних класів та функцій.

7. Вимоги до технічного захисту інформації

7.1. Забезпечити цілісність програмного коду та захист від несанкціонованої модифікації (використання систем контролю версій). 7.2. Програмний засіб не повинен передавати оброблювані дані на зовнішні сервери без відома користувача. 7.3. Обробка персональних даних (якщо такі є у логах) повинна відповідати вимогам законодавства (деперсоналізація/анонімізація перед аналізом).

8. Техніко-економічні показники

8.1. **Цінність результатів:** впровадження розробки повинно зменшити витрати на серверну інфраструктуру (за рахунок економії пам'яті у 2-3 рази порівняно зі стандартними методами) та знизити фінансові ризики від витоку даних. 8.2. **Доступність:** програмний засіб має бути реалізований з використанням відкритих технологій (Open Source), що дозволяє його впровадження без витрат на ліцензійне ПЗ.

9. Стадії та етапи розробки

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Початок	Кінець
1	Визначення напрямку магістерської роботи, формулювання теми	24.09.2025	30.09.2025
2	Аналіз предметної області обраної теми	01.10.2025	10.10.2025
3	Апробація отриманих результатів	11.10.2025	20.10.2025
4	Розробка алгоритму роботи	21.10.2025	03.11.2025
5	Написання магістерської роботи на основі розробленої теми	04.11.2025	10.11.2025
6	Розробка економічної частини	11.11.2025	15.11.2025
7	Передзахист магістерської кваліфікаційної роботи	16.11.2025	22.11.2025
8	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	23.11.2025	25.11.2025
9	Захист магістерської кваліфікаційної роботи	26.11.2025	28.11.2025

10. Порядок контролю та прийому

86

10.1 До приймання магістерської кваліфікаційної роботи надається:

- ПЗ до магістерської кваліфікаційної роботи;
- програмний додаток;
- презентація;
- відзив керівника роботи;
- відзив опонента

Технічне завдання до виконання прийняв _____ Вахній Д.Д.

Додаток Б. Лістинг коду

```
================================================================================
СТРУКТУРА ПРОЄКТУ (prog)
================================================================================
prog
 ├── anomaly_detector.py
 ├── config.py
 ├── crm_analytics.py
 ├── custom_forest.py
 ├── data_loader.py
 ├── data_processor.py
 ├── main.py
 ├── real_data_experiment.py
 └── veryfication_experiment.py


================================================================================
ФАЙЛ: prog\anomaly_detector.py
================================================================================
import pandas as pd
import numpy as np
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import StandardScaler, OneHotEncoder
import matplotlib.pyplot as plt
import seaborn as sns
import config
import os
from custom_forest import MixedIsolationForest

class AnomalyDetector:
    def __init__(self, master_df):
        self.df = master_df.copy()
        self.X_numeric = None # Для стандартного методу (тільки числа або OHE)
        self.X_mixed = None   # Для мого методу (числа + рядки)

    def prepare_data(self):
        """Готує два набори даних: для Sklearn і для мого методу"""
        print("Підготовка даних для ML...")

        # 1. Формуємо набір фіч (включаючи категоріальні!)
        # Ми беремо: Ціну, Доставку (числа) + Категорію товару, Тип оплати (текст)
        features = self.df[['price', 'freight_value', 'product_category_name', 'payment_type']].copy()

        # Заповнюємо пропуски
        features['price'] = features['price'].fillna(0)
        features['freight_value'] = features['freight_value'].fillna(0)
```

```
    features['product_category_name'] = features['product_category_name'].fillna('unknown')
    features['payment_type'] = features['payment_type'].fillna('unknown')

    # --- НАБІР А (Для мого методу) ---
    # Мій метод любить сирі категорії :)
    self.X_mixed = features.copy()
    print(f"   Дані для Mixed IF підготовлено (з текстовими категоріями): {self.X_mixed.shape}")

    # --- НАБІР Б (Для стандартного sklearn) ---
    # Sklearn не розуміє текст, треба One-Hot Encoding
    print("   Кодування One-Hot для стандартного методу...")
    ohe = OneHotEncoder(sparse_output=False, handle_unknown='ignore')
    cat_encoded = ohe.fit_transform(features[['product_category_name', 'payment_type']])
    cat_df = pd.DataFrame(cat_encoded, columns=ohe.get_feature_names_out())

    num_df = features[['price', 'freight_value']].reset_index(drop=True)
    self.X_numeric = pd.concat([num_df, cat_df], axis=1)
    print(f"   Дані для Standard IF підготовлено (роздуті OHE): {self.X_numeric.shape}")

  def run_standard_isolation_forest(self, contamination=0.01):
    print("   Запуск Standard Isolation Forest (Sklearn)...")
    # Використовуємо дані з One-Hot Encoding
    iso = IsolationForest(n_estimators=100, contamination=contamination, random_state=42,
   n_jobs=-1)
    preds = iso.fit_predict(self.X_numeric)
    self.df['anomaly_standard'] = [1 if x == -1 else 0 for x in preds]
    print(f"   Знайдено аномалій (Standard): {self.df['anomaly_standard'].sum()}")

  def run_custom_method(self):
    print("   Запуск Mixed Isolation Forest...")

    # Використовуємо МІЙ КЛАС на змішаних даних
    my_model = MixedIsolationForest(n_estimators=100, sample_size=256)
    my_model.fit(self.X_mixed)

    # Отримуємо score. У коді чим вище score, тим більше це аномалія [0.5 ... 1.0]
    # Зазвичай поріг це > 0.6 або беремо топ %
    scores = my_model.anomaly_score(self.X_mixed)
    self.df['score_custom'] = scores

    # Визначаємо поріг для топ 1% аномалій
    threshold = np.percentile(scores, 99)
    self.df['anomaly_custom'] = [1 if x > threshold else 0 for x in scores]

    print(f"   Знайдено аномалій (Custom): {self.df['anomaly_custom'].sum()}")

  def compare_and_visualize(self):
```

```
        print("   Генерація графіків порівняння...")

        # Графік
        plt.figure(figsize=(14, 6))

        # Standard
        plt.subplot(1, 2, 1)
        # Малюємо тільки "нормальні" точки (сині) і "аномалії" (червоні)
        # Щоб графік не був кашею, беремо семпл
        plot_data = self.df.sample(2000)
        sns.scatterplot(data=plot_data,       x='price',       y='freight_value',       hue='anomaly_standard',
      palette={0:'blue', 1:'red'})
        plt.title('Standard IF')

        # Custom
        plt.subplot(1, 2, 2)
        sns.scatterplot(data=plot_data,       x='price',       y='freight_value',       hue='anomaly_custom',
      palette={0:'green', 1:'orange'})
        plt.title('Mixed IF')

        plt.tight_layout()
        plt.savefig(os.path.join(config.OUTPUT_DIR, 'final_comparison.png'))

        # Зберігаємо файл із результатами
        self.df[['order_id',  'price',  'product_category_name',  'anomaly_standard',  'anomaly_custom',
      'score_custom']].head(1000).to_csv(
            os.path.join(config.OUTPUT_DIR, 'final_results.csv'), index=False
        )
        print("   Результати збережено!")

================================================================================
ФАЙЛ: prog\config.py
================================================================================
import os

# Визначаємо шлях до кореневої папки проєкту
BASE_DIR = os.path.dirname(os.path.abspath(__file__))

# Шлях до датасетів (на рівень вище від папки crm_project)
DATA_DIR = os.path.join(os.path.dirname(BASE_DIR), 'Data')

# Словник файлів, щоб легко звертатися за ключами
FILES = {
    'customers': 'olist_customers_dataset.csv',
    'orders': 'olist_orders_dataset.csv',
    'items': 'olist_order_items_dataset.csv',
    'products': 'olist_products_dataset.csv',
```

```
    'payments': 'olist_order_payments_dataset.csv',
    'reviews': 'olist_order_reviews_dataset.csv',
    'sellers': 'olist_sellers_dataset.csv',
    'geo': 'olist_geolocation_dataset.csv',
    'translation': 'product_category_name_translation.csv'
}

# Шлях куди зберігати результати
OUTPUT_DIR = os.path.join(BASE_DIR, 'output')
if not os.path.exists(OUTPUT_DIR):
    os.makedirs(OUTPUT_DIR)
```

```
================================================================================
ФАЙЛ: prog\crm_analytics.py
================================================================================
import pandas as pd
import config
import os

class CRMAnalyzer:
    def __init__(self, master_df):
        self.df = master_df

    def calculate_rfm(self):
        """Розрахунок RFM метрик для сегментації клієнтів"""
        print("   Розрахунок RFM аналізу...")

        # Беремо тільки доставлені замовлення
        delivered = self.df[self.df['order_status'] == 'delivered'].copy()

        # Визначаємо дату "сьогодні" як останню дату в датасеті + 1 день
        latest_date = delivered['order_purchase_timestamp'].max() + pd.Timedelta(days=1)

        # Групуємо по унікальному клієнту
        rfm = delivered.groupby('customer_unique_id').agg({
            'order_purchase_timestamp': lambda x: (latest_date - x.max()).days, # Recency (днів з
    останньої покупки)
            'order_id': 'count',                                    # Frequency (кількість покупок)
            'payment_value': 'sum'                                  # Monetary (сума грошей)
        }).reset_index()

        rfm.rename(columns={
            'order_purchase_timestamp': 'Recency',
            'order_id': 'Frequency',
            'payment_value': 'Monetary'
        }, inplace=True)
```

```
        return rfm

    def save_results(self, rfm_df, master_df):
        """Зберігає результати в CSV"""
        print("   Збереження результатів...")

        # Зберігаємо RFM звіт
        rfm_path = os.path.join(config.OUTPUT_DIR, 'crm_rfm_analysis.csv')
        rfm_df.to_csv(rfm_path, index=False)

        # Зберігаємо частину Master Table (перші 1000 рядків для прикладу, щоб не забити диск)
        master_sample_path = os.path.join(config.OUTPUT_DIR, 'crm_master_sample.csv')
        master_df.head(1000).to_csv(master_sample_path, index=False)

        print(f"   Готово! Файли збережено в: {config.OUTPUT_DIR}")
```

```
================================================================================
ФАЙЛ: prog\custom_forest.py
================================================================================
import numpy as np
import pandas as pd

class MixedIsolationTree:
    def __init__(self, height_limit):
        self.height_limit = height_limit
        self.split_attr = None
        self.split_val = None
        self.left = None
        self.right = None
        self.size = 0
        self.node_type = None
        self.split_type = None

    def fit(self, X, current_height):
        self.size = len(X)
        if current_height >= self.height_limit or self.size <= 1:
            self.node_type = 'external'
            return self

        columns = X.columns
        self.split_attr = np.random.choice(columns)
        col_data = X[self.split_attr]

        # Перевірка типу: Число чи Категорія
        if pd.api.types.is_numeric_dtype(col_data):
            self.split_type = 'numerical'
            min_val = col_data.min()
```

```
            max_val = col_data.max()
            if min_val == max_val:
                self.node_type = 'external'
                return self
            self.split_val = np.random.uniform(min_val, max_val)
            mask = col_data < self.split_val
        else:
            # Categorical Subset Split
            self.split_type = 'categorical'
            unique_vals = col_data.unique()
            if len(unique_vals) <= 1:
                self.node_type = 'external'
                return self

            # Випадкова підмножина категорій
            choices = list(unique_vals)
            np.random.shuffle(choices)
            # Беремо випадкову кількість елементів (мінімум 1, максимум всі-1)
            split_idx = np.random.randint(1, len(choices)) if len(choices) > 1 else 1
            self.split_val = set(choices[:split_idx])
            mask = col_data.isin(self.split_val)

        X_left = X[mask]
        X_right = X[~mask]

        self.node_type = 'internal'
        self.left = MixedIsolationTree(self.height_limit).fit(X_left, current_height + 1)
        self.right = MixedIsolationTree(self.height_limit).fit(X_right, current_height + 1)
        return self

    def path_length(self, x, current_height):
        if self.node_type == 'external':
            return current_height + self._c(self.size)

        attr_val = x[self.split_attr]

        if self.split_type == 'numerical':
            if attr_val < self.split_val:
                return self.left.path_length(x, current_height + 1)
            else:
                return self.right.path_length(x, current_height + 1)
        else: # categorical
            if attr_val in self.split_val:
                return self.left.path_length(x, current_height + 1)
            else:
                return self.right.path_length(x, current_height + 1)
```

```
    def _c(self, n):
        if n <= 1: return 0
        return 2.0 * (np.log(n - 1) + 0.5772156649) - (2.0 * (n - 1) / n)

class MixedIsolationForest:
    def __init__(self, n_estimators=100, sample_size=256):
        self.n_estimators = n_estimators
        self.sample_size = sample_size
        self.trees = []

    def fit(self, X):
        for _ in range(self.n_estimators):
            subset = X.sample(n=min(self.sample_size, len(X)))
            height_limit = int(np.ceil(np.log2(self.sample_size)))
            tree = MixedIsolationTree(height_limit).fit(subset, 0)
            self.trees.append(tree)
        return self

    def anomaly_score(self, X):
        scores = []
        c_norm = self._c(self.sample_size)
        for i in range(len(X)):
            x = X.iloc[i]
            avg_path = np.mean([tree.path_length(x, 0) for tree in self.trees])
            score = 2 ** (-avg_path / c_norm)
            scores.append(score)
        return np.array(scores)

    def _c(self, n):
        if n <= 1: return 1
        return 2.0 * (np.log(n - 1) + 0.5772156649) - (2.0 * (n - 1) / n)

================================================================================
ФАЙЛ: prog\data_loader.py
================================================================================
import pandas as pd
import os
import config

class DataLoader:
    def __init__(self):
        self.data_path = config.DATA_DIR
        self.files = config.FILES

    def load_all_data(self):
        """Завантажує всі CSV файли у словник DataFrame-ів"""
        datasets = {}
```

```
        print(f"   Починаю завантаження даних з: {self.data_path}...")

        for key, filename in self.files.items():
            file_path = os.path.join(self.data_path, filename)
            if os.path.exists(file_path):
                datasets[key] = pd.read_csv(file_path)
                print(f"     {key}: завантажено {datasets[key].shape[0]} рядків")
            else:
                print(f"     Файл не знайдено: {filename}")

        return datasets

================================================================================
ФАЙЛ: prog\data_processor.py
================================================================================
import pandas as pd

class DataProcessor:
    def __init__(self, datasets):
        self.data = datasets
        self.master_table = None

    def clean_dates(self):
        """Перетворює рядкові дати у формат datetime для таблиці orders"""
        print("   Очищення форматів дат...")
        date_cols = ['order_purchase_timestamp', 'order_approved_at',
                     'order_delivered_carrier_date', 'order_delivered_customer_date',
                     'order_estimated_delivery_date']

        for col in date_cols:
            self.data['orders'][col] = pd.to_datetime(self.data['orders'][col])

    def translate_categories(self):
        """Перекладає категорії товарів з португальської на англійську"""
        print("BR -> US Переклад категорій...")
        products = self.data['products']
        translations = self.data['translation']

        merged = products.merge(translations, on='product_category_name', how='left')
        # Замінюємо оригінальну колонку на англійську, якщо є переклад
        merged['product_category_name'] =
       merged['product_category_name_english'].combine_first(merged['product_category_name'])
        self.data['products'] = merged.drop(columns=['product_category_name_english'])

    def create_master_view(self):
        """Зшиває всі таблиці в одну велику CRM-таблицю"""
        print("   Об'єднання таблиць у Master CRM View...")
```

```
        # 1. Orders + Items (щоб знати, що купили)
        df = self.data['orders'].merge(self.data['items'], on='order_id', how='left')

        # 2. + Products (щоб знати деталі товару)
        df = df.merge(self.data['products'], on='product_id', how='left')

        # 3. + Customers (щоб знати, хто купив)
        # Увага: customer_id змінюється з кожним замовленням,
        # customer_unique_id - це унікальний клієнт.
        df = df.merge(self.data['customers'], on='customer_id', how='left')

        # 4. + Payments (сума оплати)
        df = df.merge(self.data['payments'], on='order_id', how='left')

        self.master_table = df
        print(f"   Master Table створено. Розмір: {self.master_table.shape}")
        return self.master_table

================================================================================
ФАЙЛ: prog\main.py
================================================================================
from data_loader import DataLoader
from data_processor import DataProcessor
from crm_analytics import CRMAnalyzer
from anomaly_detector import AnomalyDetector
import time

def main():
    start_time = time.time()
    print("   Запуск CRM Data Pipeline (Master's Thesis)...")

    # 1. Завантаження даних
    loader = DataLoader()
    datasets = loader.load_all_data()

    # 2. Обробка та об'єднання (створення Master View)
    processor = DataProcessor(datasets)
    processor.clean_dates()
    processor.translate_categories()
    master_df = processor.create_master_view()

    # 3. Базова аналітика CRM (RFM)
    analyzer = CRMAnalyzer(master_df)
    rfm_table = analyzer.calculate_rfm()

    # Вивід топ-5 кращих клієнтів
```

```
    print("\n   ТОП-5 Клієнтів за витратами (Monetary):")
    print(rfm_table.sort_values(by='Monetary', ascending=False).head(5))

    analyzer.save_results(rfm_table, master_df)

    # ============================================================
    # 4. ПОШУК АНОМАЛІЙ (Наукова частина)
    # ============================================================
    print("\n   === ПОЧАТОК ЕКСПЕРИМЕНТУ З АНОМАЛІЯМИ ===")

    # Ініціалізація детектора
    detector = AnomalyDetector(master_df)

    # Підготовка даних (створення окремих наборів для OHE та Mixed підходів)
    detector.prepare_data()

    # Етап А: Запуск стандартного методу Isolation Forest
    detector.run_standard_isolation_forest(contamination=0.01)

    # Етап Б: Запуск методу Mixed Isolation Forest
    detector.run_custom_method()

    # Етап В: Порівняння результатів та генерація графіків
    detector.compare_and_visualize()
    # ============================================================

    print(f"\n   Роботу завершено за {round(time.time() - start_time, 2)} секунд.")

if __name__ == "__main__":
    main()

================================================================================
ФАЙЛ: prog\real_data_experiment.py
================================================================================
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import roc_auc_score, roc_curve
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import OneHotEncoder
import os
import config

# Імпорти ваших модулів
from data_loader import DataLoader
from data_processor import DataProcessor
from custom_forest import MixedIsolationForest
```

```
def prepare_semi_synthetic_data(master_df, n_anomalies=500):
    """
    Бере реальні дані Olist і підмішує туди штучні аномалії,
    щоб ми могли порахувати точність (ROC-AUC).
    """
    print("   Підготовка Semi-Synthetic даних на базі Olist...")

    # 1. Відбираємо фічі (2 числові, 2 категоріальні)
    # Це відповідає структурі мого методу
    features = master_df[['price', 'freight_value', 'product_category_name', 'payment_type']].copy()

    # Очистка
    features = features.fillna({
        'price': 0,
        'freight_value': 0,
        'product_category_name': 'unknown',
        'payment_type': 'unknown'
    })

    # Вважаємо всі реальні дані "нормальними" (label = 0)
    features['label'] = 0

    # Щоб не чекати вічність, візьмемо вибірку (наприклад, 10 000 записів)
    # Якщо комп'ютер потужний, можна взяти більше
    df_normal = features.sample(n=10000, random_state=42)

    # 2. Генеруємо АНОМАЛІЇ (label = 1)
    # Логіка: створюємо дані, які виглядають "дивно" для e-commerce

    # Аномалія 1: Величезна ціна з дивною категорією
    anom_1 = pd.DataFrame({
        'price': np.random.uniform(2000, 5000, n_anomalies // 2), # Зменшили розрив
        'freight_value': np.random.uniform(0, 5, n_anomalies // 2),
        # Замість 'hacked_item' візьмемо рідкісну, але реальну категорію, або лишимо 'unknown'
        'product_category_name': ['art'] * (n_anomalies // 2),
        'payment_type': ['voucher'] * (n_anomalies // 2),
        'label': 1
    })

    # Аномалія 2: Нормальна ціна, але аномально висока доставка (шахрайство на доставці)
    anom_2 = pd.DataFrame({
        'price': np.random.uniform(50, 150, n_anomalies // 2),
        'freight_value': np.random.uniform(300, 600, n_anomalies // 2), # Все ще багато, але ближче до
      реальності
        'product_category_name':      np.random.choice(df_normal['product_category_name'].unique(),
      n_anomalies // 2),
```

```
        'payment_type': ['boleto'] * (n_anomalies // 2),
        'label': 1
    })

    # Зшиваємо все разом
    df_final = pd.concat([df_normal, anom_1, anom_2], ignore_index=True)

    # Перемішуємо
    df_final = df_final.sample(frac=1, random_state=42).reset_index(drop=True)

    print(f"   Готово! Розмір: {df_final.shape}, Аномалій: {df_final['label'].sum()}")
    return df_final

def run_real_experiment():
    # 1. Завантажуємо реальні дані через пайплайн
    loader = DataLoader()
    datasets = loader.load_all_data()

    processor = DataProcessor(datasets)
    # Нам не обов'язково робити повний clean_dates, бо ми юзаємо тільки категорії і ціни
    # Але для коректності create_master_view може знадобитись
    processor.clean_dates()
    processor.translate_categories()
    master_df = processor.create_master_view()

    # 2. Готуємо дані з аномаліями
    df = prepare_semi_synthetic_data(master_df)
    y_true = df['label']

    # Видаляємо label для навчання
    X = df.drop('label', axis=1)

    # ========================================
    # МІЙ МЕТОД (Mixed IF)
    # ========================================
    print("\n   Навчання Mixed Isolation Forest...")
    my_if = MixedIsolationForest(n_estimators=100, sample_size=256)

    # Моєму методу ми даємо 'X' як є (з текстовими категоріями)
    my_if.fit(X)
    y_scores_my = my_if.anomaly_score(X)
    auc_my = roc_auc_score(y_true, y_scores_my)
    print(f"   ROC-AUC (Mixed IF): {auc_my:.4f}")

    # ========================================
    # СТАНДАРТНИЙ МЕТОД (Sklearn IF)
    # ========================================
```

```
print("\n   Навчання Standard Isolation Forest (Sklearn)...")

# Стандартному методу треба One-Hot Encoding
enc = OneHotEncoder(handle_unknown='ignore', sparse_output=False)

# Кодуємо категорії
cat_cols = ['product_category_name', 'payment_type']
num_cols = ['price', 'freight_value']

X_encoded = pd.DataFrame(enc.fit_transform(X[cat_cols]))
X_encoded.columns = enc.get_feature_names_out()

# Об'єднуємо числа + закодовані категорії
X_final = pd.concat([X[num_cols].reset_index(drop=True), X_encoded], axis=1)

iso_sklearn = IsolationForest(n_estimators=100, max_samples=256, random_state=42)
iso_sklearn.fit(X_final)

# Sklearn decision_function повертає від'ємні значення для аномалій, тому інвертуємо (-)
y_scores_sklearn = -iso_sklearn.decision_function(X_final)
auc_sklearn = roc_auc_score(y_true, y_scores_sklearn)
print(f"   ROC-AUC (Standard IF): {auc_sklearn:.4f}")

# ========================================
# ВІЗУАЛІЗАЦІЯ
# ========================================
plt.figure(figsize=(10, 6))

fpr_my, tpr_my, _ = roc_curve(y_true, y_scores_my)
plt.plot(fpr_my, tpr_my, label=f'Mixed IF (Olist Data), AUC={auc_my:.3f}', linewidth=2,
  color='green')

fpr_sk, tpr_sk, _ = roc_curve(y_true, y_scores_sklearn)
plt.plot(fpr_sk, tpr_sk, label=f'Standard IF (Olist Data), AUC={auc_sklearn:.3f}', linestyle='--',
  color='blue')

plt.plot([0, 1], [0, 1], 'k--', alpha=0.5)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Real Data Experiment: Anomaly Detection on Olist')
plt.legend(loc="lower right")
plt.grid(True)

output_path = os.path.join(config.OUTPUT_DIR, 'real_data_roc_auc.png')
plt.savefig(output_path)
print(f"\n   Графік збережено: {output_path}")
```

```
if __name__ == "__main__":
    run_real_experiment()

================================================================================
ФАЙЛ: prog\veryfication_experiment.py
================================================================================
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import IsolationForest
from sklearn.metrics import roc_auc_score, roc_curve
from sklearn.preprocessing import OneHotEncoder
import os
import config # Для шляху збереження

# Імпортуємо мій алгоритм із сусіднього файлу
from custom_forest import MixedIsolationForest

# ========================================
# БЛОК 1: Генерація синтетичних даних CRM
# ========================================
def generate_crm_data(n_samples=1000, outlier_fraction=0.05):
    """
    Генерує лог дій користувачів CRM.
    """
    np.random.seed(42)
    n_outliers = int(n_samples * outlier_fraction)
    n_inliers = n_samples - n_outliers

    # --- Нормальна поведінка ---
    # User roles: 1=Manager, 2=Sales, 3=Support (спрощено для прикладу)
    # Action types: View, Edit, Create

    data = {
        'num_records': np.random.normal(50, 10, n_inliers),
        'duration': np.random.normal(300, 60, n_inliers),
        'user_role': np.random.choice(['Manager', 'Sales', 'Support'], n_inliers, p=[0.6, 0.3, 0.1]),
        'action_type': np.random.choice(['View', 'Edit', 'Create'], n_inliers, p=[0.7, 0.2, 0.1]),
        'label': 0  # 0 = норма
    }
    df_norm = pd.DataFrame(data)

    # --- Аномальна поведінка ---
    # Сценарій 1: Масовий експорт
    outliers1 = pd.DataFrame({
        'num_records': np.random.normal(5000, 500, n_outliers // 2),
        'duration': np.random.normal(100, 20, n_outliers // 2),
```

```
        'user_role': np.random.choice(['Sales', 'Support'], n_outliers // 2),
        'action_type': ['Export'] * (n_outliers // 2),
        'label': 1
    })

    # Сценарій 2: Нетипова дія
    outliers2 = pd.DataFrame({
        'num_records': np.random.normal(10, 2, n_outliers - (n_outliers // 2)),
        'duration': np.random.normal(20, 5, n_outliers - (n_outliers // 2)),
        'user_role': ['Support'] * (n_outliers - (n_outliers // 2)),
        'action_type': ['Delete'] * (n_outliers - (n_outliers // 2)),
        'label': 1
    })

    df = pd.concat([df_norm, outliers1, outliers2], ignore_index=True)
    df = df.sample(frac=1, random_state=42).reset_index(drop=True)
    return df

# ========================================
# БЛОК 2: Експеримент
# ========================================
def run_experiment():
    print("   Запуск синтетичного експерименту...")

    # 1. Підготовка даних
    df = generate_crm_data(n_samples=2000, outlier_fraction=0.1)
    y_true = df['label']
    X = df.drop('label', axis=1)

    print(f"   Дані згенеровано: {X.shape}. Аномалій: {y_true.sum()}")

    # 2. Запуск методу Mixed Isolation Forest
    print("      Навчання Mixed Isolation Forest (Ваш метод)...")
    my_if = MixedIsolationForest(n_estimators=100, sample_size=256)
    my_if.fit(X)
    y_scores_my = my_if.anomaly_score(X)
    auc_my = roc_auc_score(y_true, y_scores_my)
    print(f"      ROC-AUC (Mixed IF): {auc_my:.4f}")

    # 3. Запуск методу Sklearn IF + OneHotEncoding
    print("      Навчання Standard Isolation Forest (Sklearn)...")
    enc = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
    X_encoded = pd.DataFrame(enc.fit_transform(X[['user_role', 'action_type']]))
    X_encoded.columns = enc.get_feature_names_out()
    X_final = pd.concat([X[['num_records', 'duration']], X_encoded], axis=1)

    iso_sklearn = IsolationForest(n_estimators=100, max_samples=256, random_state=42)
```

```
    iso_sklearn.fit(X_final)
    y_scores_sklearn = -iso_sklearn.decision_function(X_final)
    auc_sklearn = roc_auc_score(y_true, y_scores_sklearn)
    print(f"      ROC-AUC (Standard IF): {auc_sklearn:.4f}")

    # ========================================
    # БЛОК 3: Візуалізація
    # ========================================
    plt.figure(figsize=(10, 6))

    fpr_my, tpr_my, _ = roc_curve(y_true, y_scores_my)
    plt.plot(fpr_my, tpr_my, label=f'Proposed Method (Mixed IF), AUC={auc_my:.3f}', linewidth=2,
      color='green')

    fpr_sk, tpr_sk, _ = roc_curve(y_true, y_scores_sklearn)
    plt.plot(fpr_sk, tpr_sk, label=f'Standard Method (IF + OHE), AUC={auc_sklearn:.3f}', linestyle='--',
      color='blue')

    plt.plot([0, 1], [0, 1], 'k--', alpha=0.5)
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Detection Performance: Mixed vs Standard Approach')
    plt.legend(loc="lower right")
    plt.grid(True)

    # Збереження в папку output
    output_path = os.path.join(config.OUTPUT_DIR, 'roc_auc_experiment.png')
    plt.savefig(output_path)
    print(f"   Графік збережено: {output_path}")

if __name__ == "__main__":
    run_experiment()
```

Додаток Б. Ілюстративний матеріал

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
КАФЕДРА МЕНЕДЖМЕНТУ ТА БЕЗПЕКИ ІНФОРМАЦІЙНИХ СИСТЕМ

Вдосконалення алгоритму Isolation Forest шляхом введення змішаних сплітів

у задачі виявлення інсайдерських загроз у CRM-системах

Виконав:
магістрант гр. 2КІТС-24м
Вахній Д.Д.

Керівник:
д.ф., доц.
Салієва О.В.

Вінниця - 2025

Рис Б.1

Актуальність проблеми

CRM-системи є центральними сховищами критичних даних бізнесу, що робить їх головною мішенню.

- **19%** усіх витоків даних спричинені інсайдерами (зловмисними або недбалими).
- Середня вартість інциденту: **$15.4 млн** на рік.
- **Проблема:** Логи активності містять гетерогенні дані (числа + категорії), які складно обробляти класичними методами без втрати контексту.

Рис Б.2

Мета та задачі дослідження

Мета роботи

Підвищення ефективності виявлення інсайдерських загроз у CRM-системах шляхом розробки модифікованого алгоритму **Isolation Forest**, адаптованого для роботи з даними змішаного типу.

Основні задачі:

- Проаналізувати специфіку інсайдерських загроз та обмеження існуючих методів.
- Розробити механізм **"Mixed Splits"** для обробки категоріальних даних без кодування.
- Здійснити програмну реалізацію методу (Python).
- Провести експериментальне дослідження на реальному датасеті (Olist).

Рис Б.3

Наукова проблема: Обмеження класичного Isolation Forest

One-Hot Encoding

Стандартний метод вимагає перетворення категорій у бінарні вектори. Для CRM-систем (User_ID, IP, Product) це призводить до вибухового зростання розмірності.

Прокляття розмірності

Матриця ознак стає розрідженою (sparse). Алгоритм витрачає ресурси на "порожні" ознаки, втрачаючи здатність ізолювати реальні аномалії.

Втрата семантики

Штучна метрика відстані між категоріями (всі рівновіддалені) знищує контекстні зв'язки, необхідні для виявлення складних шахрайств.

Рис Б.4

Запропонований метод: Mixed Splits

Адаптивна логіка розбиття

Метод динамічно визначає тип ознаки у вузлі дерева та застосовує відповідне правило:

$$\text{Split}(x, q) = \begin{cases} x^{(q)} < p & \text{if Numerical} \\ x^{(q)} \in S & \text{if Categorical} \end{cases}$$

Де **S** — випадкова непуста підмножина унікальних значень категорії. Це дозволяє уникнути кодування.

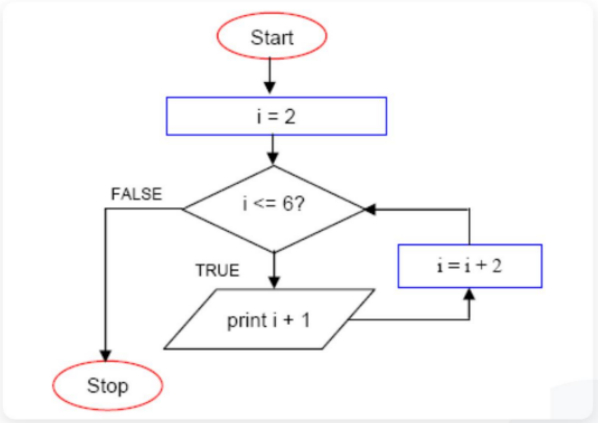


Рис Б.5

Архітектура програмного комплексу

Технологічний стек: Python 3.9

- **Data Processor (ETL):** Денормалізація реляційних таблиць CRM, очищення даних.
- **MixedIsolationTree (Core):** Клас, що реалізує поліморфні вузли дерева.
- **Scikit-learn:** Базові метрики та порівняння.
- **NumPy/Pandas:** Векторизовані операції для швидкодії.

Особливість реалізації: Використання хеш-множин (Set) для перевірки входження категорії $O(1)$, що забезпечує високу швидкість.

Рис Б.6

Експериментальне дослідження

Вхідні дані: Olist E-Commerce Dataset

- **Обсяг:** 100,000+ реальних транзакцій.
- **Ознаки:** Ціна, Вартість доставки (числові) + Категорія товару, Тип оплати (категоріальні, 74 унікальних значення).

Сценарії атак (Synthetic Injection):

1. **Фінансові махінації:** Дорогі товари + оплата ваучером (відмивання).
2. **Логістичне шахрайство:** Дешевий товар + аномально висока доставка (контекстна аномалія).

Рис Б.7

Результати: Точність детекції (ROC-AUC)

Ефективність пам'яті

Споживання RAM зменшено у **2.3 рази** завдяки відмові від One-Hot Encoding.

Якість кластеризації

Метод чітко виділив кластери контекстних аномалій, які "розмивалися" у стандартному підході.

Рис Б.8

Висновки

- **Наукова новизна:** Обгрунтовано та реалізовано метод "Mixed Splits", що дозволяє обробляти гетерогенні дані без зміни метричного простору.
- **Практичний результат:** Розроблено програмний модуль, який переважає стандартні аналоги на **8-10%** за точністю (ROC-AUC) на змішаних типах атак.
- **Оптимізація ресурсів:** Вирішено проблему "прокляття розмірності", що дозволяє використовувати систему на менш потужному обладнанні.
- **Впровадження:** Проект є економічно доцільним та готовим до інтеграції в корпоративні системи безпеки.

Рис Б.9

Додаток Г. Протокол перевірки на антиплагіат

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Вдосконалення алгоритму Isolation Forest шляхом введення зважених сплітів у задачі виявлення інсайдерських загроз у CRM-системах

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра менеджменту та безпеки інформаційних систем
факультет менеджменту та інформаційної безпеки
гр.2КІТС-24м

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) **0,71 %**

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту**
- У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

к.т.н., доцент, зав. каф. МБІС Карпінець В.В.

к.ф.-м.н., доцент каф. МБІС Шиян А.А.

Особа, відповідальна за перевірку Коваль Н.П.

З висновком експертної комісії ознайомлений(-на)

Керівник Салієва О.В.

Здобувач Вахній Д.Д.