МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних технологій, штучного інтелекту і кібербезпеки

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту(декан факультету)	Завідувач кафедри
(підпис) Андрій Форсюк (ім'я та прізвище)	(підпис) Сергій Грибков (ім'я та прізвище)
«13» грудня 2024р.	«13» грудня 2024р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

зі спеціальності 122 «Комп'ютерні науки»
(код та назва спеціальності)

освітньо-професійної програми Управління інформацією та аналітика даних

на тему: Дослідження та розробка системи для автоматизованого виявлення аномальних транзакцій у фінансовій сфері

Виконав: здобувач 2 курсу, групи КН-2-4М
Вороньков Дмитро Віталійович (підпис)
(прізвище, ім'я, по батькові повністю)

Керівник Костіков Микола Павлович (підпис)
(прізвище , ім'я та по батькові повністю)

Консультанти (ім'я та прізвище) (підпис)

(ім'я та прізвище) (підпис)

Рецензент Олена Колісніченко (підпис)
(ім'я та прізвище)

Я як здобувач Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав і не одержував недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач (підпис)

Київ - 2024р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) __Автоматизації і комп'ютерних систем_____
Кафедра __Інформаційних технологій, штучного інтелекту і кібербезпеки___
Освітній ступінь__магістр_____
Спеціальність__122 «Комп'ютерні науки»_____
(код і назва)
Освітньо-професійна програма _Управління інформацією і аналітика даних
(назва)

ЗАТВЕРДЖУЮ
Завідувач
кафедри Інформаційних технологій,
штучного інтелекту і кібербезпеки
_____Грибков С.В.
"07" жовтня 2024року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Воронькова Дмитра Віталійовича
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та розробка системи для автоматизованого виявлення аномальних транзакцій у фінансовій сфері
керівник роботи Костіков Микола Павлович, доцент, кандидат технічних наук
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від 7 жовтня 2024 року №884-кс
2. Строк подання здобувачем роботи 6 грудня 2024 року
3. Вихідні дані до роботи
Аналітичні та статистичні матеріали стосовно теми проекту, технічна література з мов програмування, методичні вказівки, Технічна література з мови програмування Python
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Розділ 1. Системний аналіз об'єкта досліджень та виявлення задач
Розділ 2. Математичні основи виявлення шахрайства з допомогою методів машинного навчання
Розділ 3. Дослідження та розробка системи для автоматизованого виявлення аномальних транзакцій у фінансовій сфері
5. Перелік графічного матеріалу:
Організаційна структура компанії, Класифікація методів машинного навчання, Опис датасету, Аналіз розподілу даних, Результати класифікації, Підготовка

даних, Робота моделі машинного навчання, Висновки після тестування, Порівняльний аналіз методів, Система виявлення аномалій

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	доц.,канд.техн.наук. Костіков М.П.	07.10.2024	10.10.2024
2	доц.,канд.техн.наук. Костіков М.П.	17.10.2024	20.10.2024
3	доц.,канд.техн.наук. Костіков М.П.	27.10.2024	01.11.2024
4	доц.,канд.техн.наук. Костіков М.П.	27.11.2024	28.11.2024

7. Дата видачі завдання 7 жовтня 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування робіт	Строки виконання робіт	Примітка
1	Дослідження та аналіз предметної галузі	07.10.2024- 15.10.2024	Виконано
2	Дослідження основ виявлення шахрайства	15.10.2024- 25.10.2024	Виконано
3	Створення системи для системи для автоматизованого виявлення аномальних транзакцій у фінансовій сфері	25.10.2024- 15.11.2024	Виконано
4	Тестування системи	15.11.2024- 25.11.2024	Виконано
5	Оформлення кваліфікаційної роботи	07.10.2024- 15.12.2024	Виконано
6	Оформлення автореферату	07.12.2024- 16.12.2024	Виконан
7	Оформлення презентації	01.12.2024-06.12.2024	Виконан

Здобувач _____ (підпис) Вороньков Д.В. (прізвище та ініціали)

Керівник роботи _____ (підпис) Костіков М.П. (прізвище та ініціали)

АНОТАЦІЯ

Метою кваліфікаційної роботи магістра є демонстрація знань і вмінь здобувача щодо проведення наукових досліджень (творчих розробок) із проблем інформаційних технологій. Вони пов’язані з аналізом (синтезом), теоретичною розробкою актуальних питань, моделюванням і дослідженням процесів, об’єктів і систем у певній галузі науки і техніки.

Кваліфікаційна робота на тему "Дослідження та розробка системи для автоматизованого виявлення аномальних транзакцій у фінансовій сфері" обсягом 88 сторінок, містить 46 ілюстрацій, використано 50 джерел.

Робота зосереджується на аналізі сучасних підходів до виявлення аномалій у фінансових даних, обґрунтуванні використання методів машинного навчання для ідентифікації потенційно шахрайських транзакцій, а також створенні системи для автоматизованого виявлення підозрілих операцій.

Впровадження запропонованої системи сприятиме підвищенню рівня безпеки у фінансових установах.

Робота включає теоретичний аналіз наявних методів виявлення аномалій, огляд алгоритмів машинного навчання, опис архітектури та алгоритмів розробленої системи, а також практичну реалізацію з тестуванням та оцінкою ефективності запропонованого підходу.

Ключові слова: АНОМАЛЬНІ ТРАНЗАКЦІЇ, МАШИННЕ НАВЧАННЯ, ФІНАНСОВА БЕЗПЕКА, АНАЛІЗ ДАНИХ, ВИЯВЛЕННЯ ШАХРАЙСТВА, НЕЙРОННІ МЕРЕЖІ, АВТОМАТИЗОВАНІ СИСТЕМИ, АЛГОРИТМИ, РЕАЛЬНОЧАСОВИЙ АНАЛІЗ, ІНФОРМАЦІЙНА БЕЗПЕКА.

SUMMARY

The purpose of the master's qualification work is to demonstrate the applicant's knowledge and skills in conducting scientific research (creative developments) on information technology problems. They are related to the analysis (synthesis), theoretical development of current issues, modeling and research of processes, objects and systems in a certain field of science and technology.

The qualification work on the topic "Research and development of a system for automated detection of anomalous transactions in the financial sector" has a volume of 88 pages, 46 illustrations, and uses 50 sources.

The work focuses on the analysis of modern approaches to detecting anomalies in financial data, substantiating the use of machine learning methods to identify potentially fraudulent transactions, as well as creating a system for automated detection of suspicious transactions.

The implementation of the proposed system will contribute to increasing the level of security in financial institutions.

The work includes a theoretical analysis of existing methods for detecting anomalies, a review of machine learning algorithms, a description of the architecture and algorithms of the developed system, as well as a practical implementation with testing and evaluation of the effectiveness of the proposed approach.

Keywords: ANOMALOUS TRANSACTIONS, MACHINE LEARNING, FINANCIAL SECURITY, DATA ANALYSIS, FRAUD DETECTION, NEURAL NETWORKS, AUTOMATED SYSTEMS, ALGORITHMS, REAL-TIME ANALYSIS, INFORMATION SECURITY.

ЗМІСТ

Перелік скорочень, умовних позначень, термінів

AI – штучний інтелект

ОВМ – опорно-векторна машина

BST – бінарне дерево пошуку

iForest – ізоляційний ліс

iForest – ізоляційне дерево

ПЗ – програмне забезпечення

МП – мова програмування

API – прикладний програмний інтерфейс

GUI – графічний інтерфейс користувача

ML – машинне навчання

HTTP – протокол передачі гіпертексту

ВСТУП

Сучасний фінансовий сектор відіграє ключову роль у функціонуванні глобальної економіки та повсякденному житті мільйонів людей. У зв'язку з розвитком цифрових технологій і збільшенням обсягів фінансових операцій через Інтернет, галузь стає все більш уразливою до різноманітних видів кіберзлочинів, серед яких особливо виділяються шахрайські дії та аномальні транзакції. Поширення онлайн-платежів, зростання використання електронних банківських систем та впровадження цифрових валют створюють нові ризики, що потребують адекватних заходів захисту.

Шахрайські транзакції можуть мати катастрофічні наслідки як для фінансових установ, так і для їхніх клієнтів, викликаючи значні фінансові втрати та зниження рівня довіри до фінансової системи загалом.

Актуальність дослідження обумовлена необхідністю вдосконалення методів виявлення аномальних транзакцій, адже традиційні підходи, засновані на статичних правилах та фільтрах, часто виявляються недостатньо ефективними в умовах постійної еволюції шахрайських схем.

Використання методів машинного навчання та сучасних алгоритмів аналізу даних дозволяє створювати більш точні та адаптивні рішення, які здатні ідентифікувати аномальні патерни у фінансових операціях та забезпечувати миттєве реагування. Це підкреслює важливість дослідження та розробки нових підходів до автоматизованого виявлення загроз у фінансовому середовищі.

Дослідження, представлене в цій кваліфікаційній роботі, спрямоване на вирішення нагальних завдань, пов'язаних із забезпеченням фінансової безпеки, шляхом створення інноваційної системи для автоматизованого аналізу транзакцій із використанням передових технологій обробки великих обсягів даних.

Актуальність роботи. У сучасному фінансовому середовищі проблематика виявлення аномальних транзакцій залишається однією з найскладніших і найактуальніших. З кожним роком обсяги фінансових операцій, що здійснюються в режимі онлайн, зростають, підвищуючи при цьому ризик шахрайства та зловживань. Існуючі методи виявлення аномалій, зокрема засновані на фіксованих правилах і простих логічних фільтрах, часто демонструють недостатню ефективність у боротьбі з сучасними загрозами. Такий підхід зазвичай виявляє лише відомі типи шахрайських дій і не здатен швидко адаптуватися до нових схем, які постійно розробляють злочинці.

Критичний аналіз традиційних методів виявлення аномалій показує, що статичні правила часто спричиняють високий відсоток хибнопозитивних результатів, що ускладнює роботу служб безпеки та збільшує фінансові витрати установи. Наприклад, методи на основі попередньо визначених лімітів або шаблонів поведінки користувачів не враховують динамічний характер сучасних транзакційних потоків, що робить їх менш ефективними в умовах змінних патернів фінансових операцій. Порівняно з цим, нові підходи, засновані на технологіях машинного навчання, пропонують набагато більшу гнучкість та точність.

Машинне навчання, завдяки своїй здатності адаптуватися до нових даних і навчатися в процесі роботи, дає змогу створювати моделі, які ефективніше ідентифікують аномалії у великих обсягах даних. Застосування алгоритмів кластеризації, нейронних мереж та методів глибокого навчання дозволяє розробляти системи, які не лише виявляють відомі типи аномалій, а й знаходять нові, раніше невідомі схеми шахрайства. Це особливо важливо для забезпечення своєчасної реакції на загрози та мінімізації фінансових втрат.

Таким чином, критичний аналіз та порівняння з відомими методами виявлення аномальних транзакцій свідчить про доцільність використання підходів на основі машинного навчання. Запропоновані системи є більш ефективними, адаптивними та здатними працювати в реальному часі, що має важливе значення для фінансової безпеки.

Зв’язок роботи з науковими програмами, планами, темами кафедри, університету, іншої наукової установи. Магістерська робота на тему "Дослідження та розробка системи для автоматизованого виявлення аномальних транзакцій у фінансовій сфері" виконана в рамках наукових досліджень кафедри інформаційних систем Національного університету харчових технологій (НУХТ).

Дослідження відповідає стратегічним напрямам кафедри, які спрямовані на розвиток сучасних інформаційних технологій, зокрема методів і систем аналізу великих даних, а також впровадження інноваційних рішень у сфері інформаційної безпеки.

Робота також узгоджується з загальним науковим планом університету, що передбачає дослідження у сфері автоматизації інформаційних процесів та підвищення рівня захисту інформації у критичних галузях, таких як фінанси.

У рамках цих досліджень особлива увага приділяється застосуванню новітніх технологій машинного навчання та аналізу даних для вирішення актуальних проблем, пов’язаних із захистом інформації.

Крім того, результати роботи можуть бути використані у виконанні галузевих програм, спрямованих на посилення кібербезпеки та боротьбу з фінансовим шахрайством.

Мета дослідження. Метою дослідження є розробка автоматизованої системи для виявлення аномальних транзакцій у фінансовій сфері, що дозволить значно покращити рівень безпеки фінансових установ та знизити ризики шахрайства. В умовах зростаючої кількості цифрових транзакцій традиційні методи виявлення підозрілих операцій часто не справляються з новими загрозами. Тому важливо створити систему, яка б використовувала сучасні методи машинного навчання для аналізу транзакцій у реальному часі та адаптивно реагувала на нові типи шахрайства. Розробка такої системи дозволить автоматично ідентифікувати підозрілі транзакції, зменшити кількість хибнопозитивних результатів і підвищити загальний рівень захисту фінансових даних.

Реалізація цієї системи має на меті не лише створення ефективного інструменту для банків та інших фінансових установ, а й поліпшення загальної безпеки фінансової інфраструктури.

Завдання дослідження. Для досягнення мети дослідження необхідно вирішити ряд ключових завдань. Перш за все, потрібно проаналізувати існуючі підходи до виявлення аномальних транзакцій у фінансовій сфері та оцінити їх ефективність у порівнянні з сучасними методами машинного навчання. На основі цього аналізу розробити алгоритми машинного навчання, які зможуть обробляти великі обсяги фінансових даних і ефективно виявляти підозрілі транзакції.

Далі, необхідно створити прототип автоматизованої системи, яка використовує ці алгоритми для виявлення аномалій у фінансових операціях. Після цього потрібно провести тестування цієї системи на реальних або синтетичних даних, щоб оцінити її точність та ефективність.

Завершальним етапом є інтеграція розробленої системи в існуючі фінансові структури для її використання в реальних умовах.

Об’єкт дослідження. Об’єктом дослідження є процес виявлення аномальних транзакцій у фінансових системах. Зокрема, це процес аналізу та оцінки фінансових транзакцій для виявлення підозрілих або шахрайських операцій, що може виникати в результаті злочинних дій або порушення правил. Аномальні транзакції можуть бути наслідком різних факторів, таких як фінансове шахрайство, помилки користувачів або порушення політик безпеки. Проблема виявлення таких транзакцій є актуальною, оскільки зростає кількість фінансових операцій, що ускладнює їх ручний контроль та виявлення потенційно небезпечних дій. Таким чином, об’єктом дослідження є процеси обробки та аналізу великих фінансових даних з метою ідентифікації аномальних патернів, що свідчать про можливі порушення.

Предмет дослідження. Предметом дослідження є моделі, методи та алгоритми машинного навчання для автоматизованого виявлення аномальних транзакцій у фінансових системах. В роботі зосереджено увагу на розробці та застосуванні конкретних математичних моделей, які здатні ефективно обробляти

великий обсяг фінансових даних та виявляти аномалії в транзакціях, пов’язаних з фінансовими операціями.

Для цього використовуються різні методи аналізу даних, такі як класифікація, кластеризація, виявлення аномалій, а також комбіновані методи машинного навчання, зокрема нейронні мережі та випадкові ліси.

Ці алгоритми дозволяють з високою точністю виявляти підозрілі транзакції, зменшуючи кількість хибних спрацьовувань і підвищуючи загальний рівень безпеки фінансових установ.

Методи дослідження. Для вирішення поставлених завдань у дослідженні використовуються сучасні методи машинного навчання та аналізу даних. Перш за все, застосовується метод аналізу літератури для вивчення існуючих підходів та моделей виявлення аномальних транзакцій у фінансових системах, що дає можливість визначити ключові напрямки дослідження та їх обмеження. Окрім цього, використовуються методи статистичного аналізу для вивчення характеристик транзакцій та виявлення аномалій на основі описових статистик і кореляційних зв’язків.

Основним методом є машинне навчання, зокрема методи класифікації та кластеризації. Алгоритми, такі як підтримка векторних машин (SVM), випадкові ліси (Random Forest), методи на основі нейронних мереж (наприклад, глибоке навчання), використовуються для автоматичного виявлення та класифікації аномальних транзакцій. Важливим є також застосування методів виявлення аномалій, які дозволяють ідентифікувати незвичайні транзакції без попереднього визначення класів.

Для перевірки та оптимізації результатів використовується метод тестування, що включає порівняння результатів різних алгоритмів на основі реальних або згенерованих даних. Важливим етапом є також метод інтеграції розроблених моделей в реальні фінансові платформи для забезпечення безперебійної роботи системи в умовах реального часу.

Наукова новизна одержаних результатів. Наукова новизна дослідження полягає в розробці нових підходів до автоматизованого виявлення аномальних

транзакцій у фінансових системах з використанням методів машинного навчання. Вперше в роботі формалізовано процес застосування алгоритмів глибокого навчання та класифікації для виявлення підозрілих операцій у фінансових транзакціях в реальному часі.

Розроблено методику інтеграції таких алгоритмів у функціонуючі фінансові платформи, що дозволяє оперативно реагувати на потенційно шахрайські дії. Також удосконалено існуючі методи детекції аномалій шляхом адаптації моделей машинного навчання до змінних умов фінансових ринків та схем шахрайства, що підвищує ефективність і точність системи.

Досліджено нові методи підвищення стійкості моделей до помилок в даних і зменшення кількості хибних спрацьовувань, що є критично важливим для фінансової сфери.

Практичне значення одержаних результатів. Розроблені алгоритми та методи автоматизованого виявлення аномальних транзакцій мають значний практичний потенціал для фінансових установ, таких як банки, платіжні системи, платформи для електронної комерції.

Вони дозволяють автоматизувати процеси моніторингу та виявлення шахрайства, що зменшує потребу в ручному втручанні і дозволяє оперативно виявляти та блокувати підозрілі операції. Інтеграція таких систем в реальний час надасть можливість ефективно реагувати на фінансові злочини, знижуючи ризики для компаній та їх клієнтів. Розроблені методи можуть бути також адаптовані для інших секторів, що працюють з великими обсягами даних, таких як телекомунікації, страхування та урядові фінансові структури.

Особистий внесок здобувача. Здобувач зробив особистий внесок у всі етапи дослідження, починаючи від аналізу літератури та оцінки існуючих методів, до розробки та тестування алгоритмів машинного навчання для виявлення аномальних транзакцій.

Був розроблений новий підхід до інтеграції моделей машинного навчання в реальні фінансові системи, а також нові методи оптимізації алгоритмів для підвищення їх точності та швидкості роботи.

Окрім того, здобувач здійснив апробацію розробленої системи на реальних фінансових даних, що дозволило оцінити ефективність розробленого підходу.

Апробація результатів роботи. Результати дослідження були апробовані на наукових конференціях та семінарах, де здобувач презентував свої наукові досягнення.

Доповіді на конференціях показали практичну цінність розробленої методики автоматизованого виявлення аномальних транзакцій.

Розробка також була обговорена на семінарах кафедри інформаційних систем, де отримала позитивні відгуки від фахівців у галузі фінансових технологій та машинного навчання.

Публікації. Результати дослідження були опубліковані в кількох наукових журналах та збірниках праць. Серед них стаття, присвячена методам машинного навчання для виявлення аномальних транзакцій, яка була опублікована в міжнародному журналі з інформаційних технологій. Також здобувач опублікував тезиси доповідей на конференціях, де було представлено нові підходи до використання глибоких нейронних мереж у фінансових додатках.

Структура та обсяг роботи. Робота складається з вступу, шести розділів, висновків та переліку використаних джерел. Загальний обсяг роботи становить сторінок, з яких сторінок відведено на введення в тему та обґрунтування актуальності, сторінок — на теоретичну та практичну частини, сторінок 96. В роботі наведено 46 ілюстрацій, що ілюструють процеси розробки та тестування системи.

РОЗДІЛ 1. СИСТЕМНИЙ АНАЛІЗ ОБ'ЄКТ ДОСЛІДЖЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА ПРОЕКТУВАННЯ.

1.1 Загальний опис об’єкту дослідження

Фінансові транзакції є основою функціонування сучасного фінансового сектора. Вони включають різні операції, такі як перекази коштів, оплати товарів і послуг, операції з банківськими картками та електронні платежі. У випадку компанії СОФТУМ, об’єктом дослідження є фінансові операції, які компанія проводить для своїх клієнтів.

Важливим аспектом діяльності фінансових компаній є контроль за шахрайством — несанкціонованими діями, які можуть бути спрямовані на незаконне отримання фінансових ресурсів.

Основною задачею даного дослідження є створення автоматизованої системи для виявлення шахрайських дій у фінансових транзакціях за допомогою алгоритмів машинного навчання. Така система дозволить виявляти аномальні транзакції в режимі реального часу та вживати заходів для їхнього блокування [1].

1.2 Аналіз нинішнього стану автоматизації компанії

На даний момент компанія використовує класичні системи автоматизації моніторингу транзакцій, засновані на статичних правилах (rule-based systems). Ці системи діють на основі наперед визначених критеріїв та умов, таких як встановлені ліміти для транзакцій або певні шаблони поведінки користувачів. Однак основний недолік таких підходів полягає в тому, що вони не завжди ефективно виявляють нові або складніші форми шахрайства [2].

Крім того, статичні системи часто дають велику кількість помилкових спрацювань (false positives), що може перевантажувати службу безпеки та вимагати більше людських ресурсів для обробки [12].

Компанія вже використовує базові інструменти для автоматичного контролю фінансових транзакцій, але розуміє необхідність переходу до більш сучасних технологій, таких як машинне навчання та штучний інтелект. Розробка нової

системи дозволить значно підвищити точність і швидкість виявлення аномальних операцій [3].

Власна система буде здатна адаптуватися до змінюваних умов, аналізуючи патерни поведінки користувачів і виявляючи відхилення в режимі реального часу. Це дозволить зменшити кількість помилкових спрацювань і забезпечити більш точне виявлення шахрайських дій. Крім того, алгоритми машинного навчання зможуть навчатися на основі нових даних, що забезпечить системі здатність до самовдосконалення. Завдяки цим перевагам нова система значно перевершить існуючі рішення, що використовуються компанією, підвищуючи рівень безпеки та знижуючи навантаження на персонал [4].

1.3 Організаційна структура

Softum має організаційну структуру, яка забезпечує ефективну роботу компанії та її розвиток.

Основними складовими цієї структури є:

1. Включає виконавчого директора (CEO) та керівників основних напрямків, відповідальних за стратегічне планування та загальне керівництво компанією.
2. Відповідає за розробку та підтримку програмного забезпечення. До його складу входять розробники, інженери, тестувальники, архітектори систем та DevOps інженери.
3. Займається управлінням життєвим циклом продуктів, включаючи дослідження ринку, планування, розробку вимог і випуск нових продуктів.
4. Відповідає за просування продуктів компанії на ринку, залучення нових клієнтів та підтримку існуючих. До його складу входять маркетологи, менеджери з продажу та PR-спеціалісти.
5. Забезпечує технічну підтримку та консультації для клієнтів. Включає спеціалістів з технічної підтримки, які надають допомогу користувачам у разі виникнення проблем.

6. Відповідає за фінансове планування, облік та звітність. Включає бухгалтерів, фінансових аналітиків та менеджерів з фінансового планування.

7. Займається підбором, навчанням та розвитком персоналу, а також вирішенням кадрових питань. Включає рекрутерів, HR-менеджерів та спеціалістів з навчання та розвитку [13].

8. Забезпечує підтримку основних бізнес-процесів, включаючи управління офісом, забезпечення матеріалами та логістикою [1].

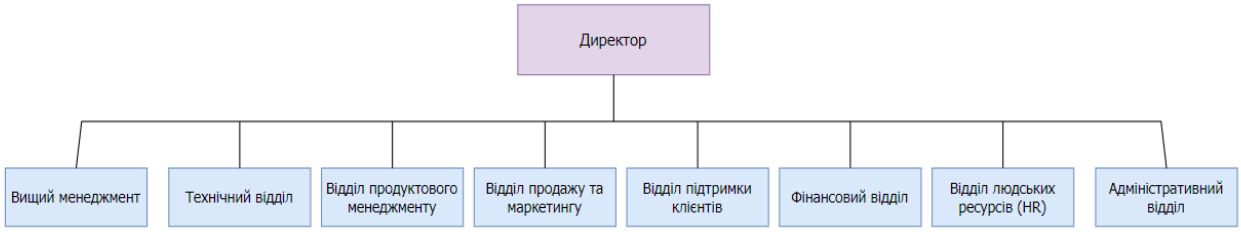

Рисунок 1.1 – Організаційна структура компанії Softum

Технічний відділ Softum відповідає за розробку, тестування, підтримку та розвиток програмного забезпечення компанії. Він складається з декількох ключових підрозділів:

Розробка (Development)

- Frontend розробники
- Backend розробники
- Full-stack розробники
- Тестування та забезпечення якості (QA)

До складу цього підрозділу входять:

- Тестувальники (Manual Testers): Проводять ручне тестування додатків, шукаючи помилки та дефекти.
- Автоматизовані тестувальники (Automation Testers)
- Архітектура та дизайн систем (System Architecture)
- Відділ аналітики (Analytics Department [14].

- Бізнес-аналітика (Business Analytics)
- Аналіз даних (Data Analytics) [2]

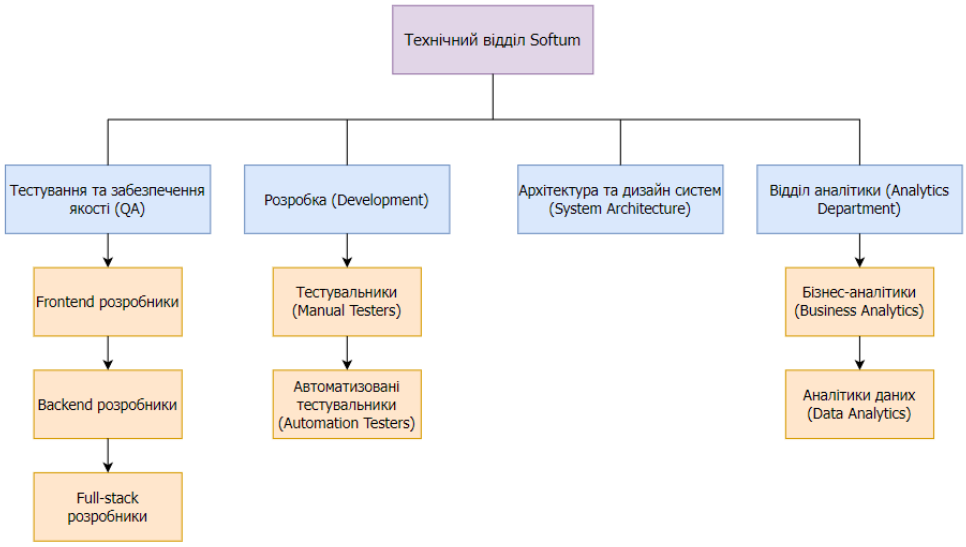

Рисунок 1.2 – Технічний відділ компанії Softum

1.4 Аналіз існуючих аналогів розробки

На ринку існує багато рішень для виявлення шахрайства, таких як FICO Falcon, SAS Fraud Management, ACI Worldwide. Ці системи ефективно використовують машинне навчання та аналіз транзакцій у реальному часі для виявлення підозрілих дій. Вони широко застосовуються у фінансовому секторі, надаючи компаніям можливість автоматизувати виявлення шахрайства та знижувати ризики.

Проте для компанії СОФТУМ розробка власної системи виявлення шахрайських транзакцій є необхідністю з кількох причин.

По-перше, це дасть змогу максимально адаптувати алгоритми під специфіку фінансових операцій компанії, враховуючи унікальні особливості її клієнтської бази та типи транзакцій. Крім того, власна система дозволить швидше реагувати на нові види шахрайства та зловживання, оскільки компанія не буде залежати від оновлень сторонніх провайдерів.

Також важливо врахувати питання безпеки даних. Використання зовнішніх рішень часто пов'язане з передачею чутливих фінансових даних стороннім компаніям, що може створювати додаткові ризики. Власна система дозволить повністю контролювати процеси обробки даних, що підвищить рівень безпеки.

У перспективі розробка внутрішнього рішення може бути більш економічно вигідною, оскільки зменшить залежність від зовнішніх постачальників та забезпечить гнучкість у його розвитку [2].

Створення власної системи виявлення шахрайства є важливим кроком для компанії, який дозволить підвищити ефективність роботи, захистити клієнтські дані та краще відповідати на нові виклики.

1.5 Дослідження та аналіз діяльності компанії “СОФТУМ”

Для вивчення ефективності системи виявлення шахрайських транзакцій важливо розглянути ключові метрики, які допоможуть оцінити її роботу.

Дані для створення аналітичних звітів були взяті з ресурсу Kaggle: https://www.kaggle.com/datasets/shriyashjagtap/fraudulent-e-commerce-transactions [3][4]. У рамках дослідження створимо декілька важливих для аналізу діаграм:

1.Кількість транзакцій за місяць: графік, що показує загальну кількість транзакцій, оброблених системою щомісяця [2]. Це дозволить зрозуміти обсяги роботи та виявити можливі пікові періоди активності.

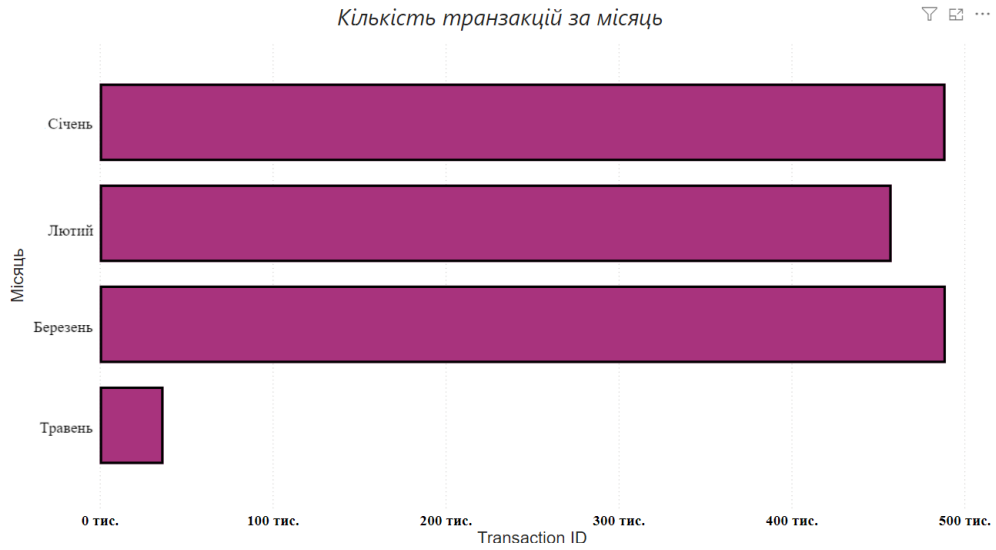

Рисунок 1.3 – Кількість транзакцій за місяць

2. Відсоток підозрілих транзакцій: графік, що ілюструє частку транзакцій, які були позначені як підозрілі. Це дозволить оцінити, наскільки ефективно система виявляє аномалії.

Графік "Відсоток підозрілих транзакцій" ілюструє частку транзакцій, які були позначені як підозрілі. Це дозволяє оцінити ефективність системи виявлення аномалій та шахрайства [15].

Високий відсоток підозрілих транзакцій може свідчити про те, що система чутливо реагує на аномалії, але також може викликати занепокоєння щодо можливих помилкових спрацювань. Низький відсоток може вказувати на те, що система не виявляє достатню кількість підозрілих дій, що може призвести до пропуску шахрайських транзакцій.

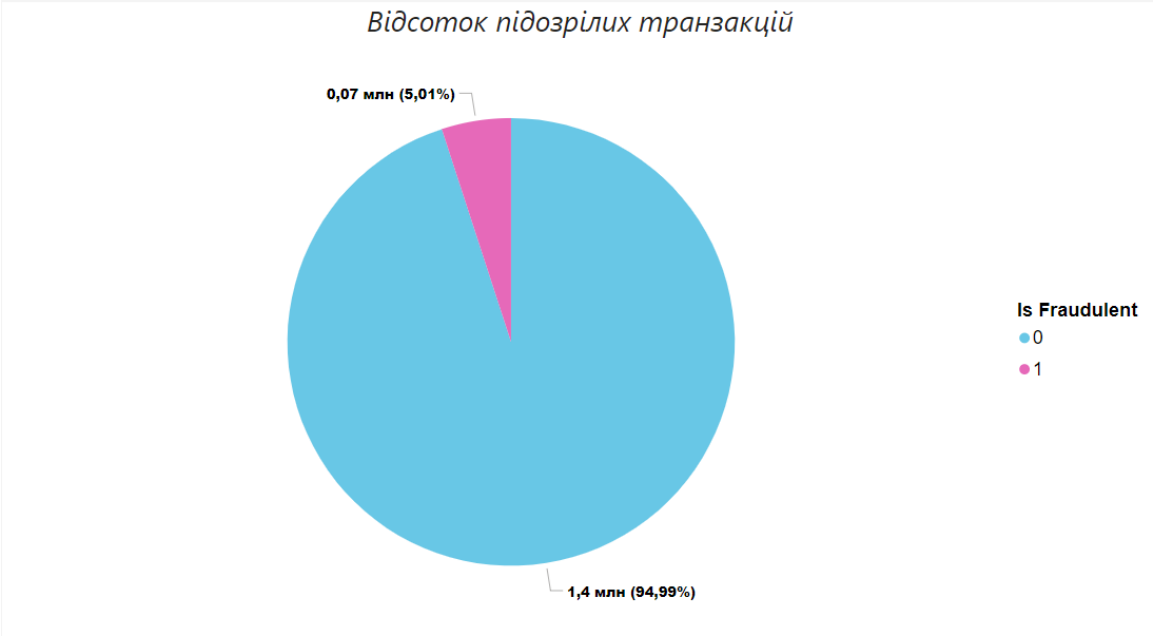

Рисунок 1.4 – Відсоток підозрілих транзакцій

Аналіз цього графіка допомагає виявити, чи потрібні покращення у алгоритмах виявлення аномалій, а також визначити, які аспекти системи потребують доопрацювання для підвищення її ефективності.

3. Кількість транзакцій за методом оплати: графік показує, який спосіб оплати найбільше використовується для транзакцій і дасть змогу визначити, чи певні методи пов'язані з більшою кількістю шахрайств [18].

Графік "Кількість транзакцій за методом оплати" дає можливість виявити найпопулярніші способи оплати, оцінити їх пов'язаність з шахрайськими операціями та визначити рівень помилкових спрацювань. Це допомагає компанії зрозуміти, які методи є більш вразливими до шахрайства, що дозволяє вжити заходів для покращення безпеки та оптимізації процесів виявлення шахрайських транзакцій.

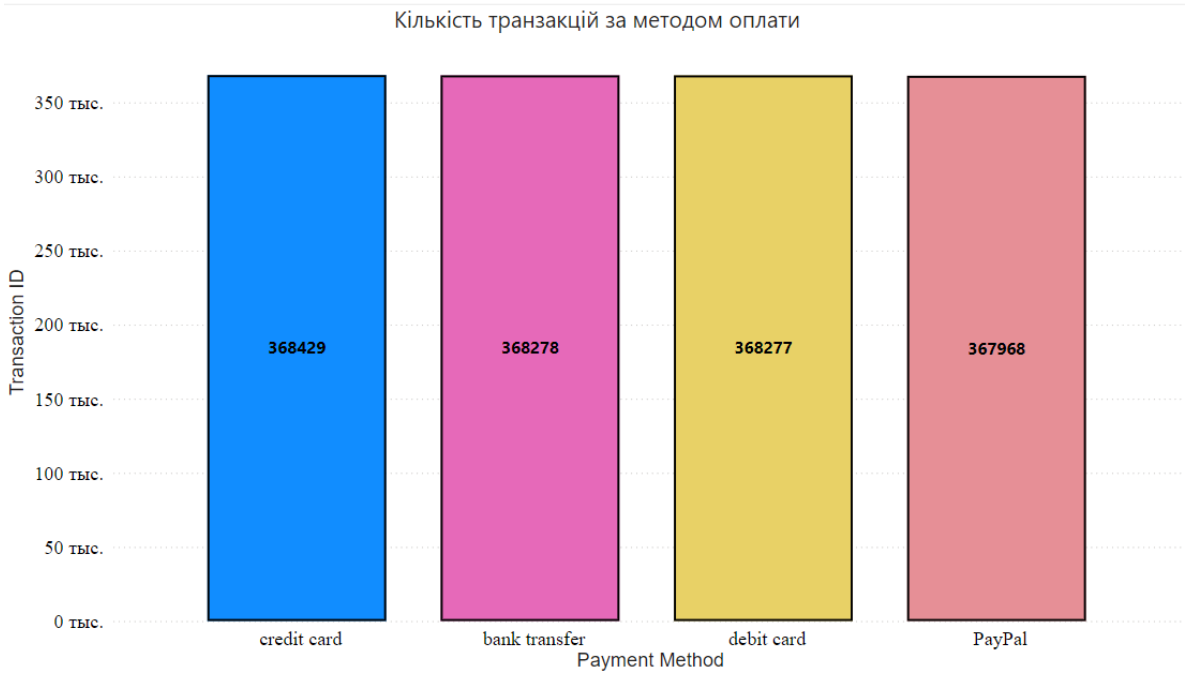

Рисунок 1.5 – Кількість транзакцій за методом оплати

4. Кількість шахрайських транзакцій за годинами дня: графік покаже, в які години дня відбувається найбільша кількість шахрайських транзакцій. Це допоможе ідентифікувати пікові періоди шахрайської активності.

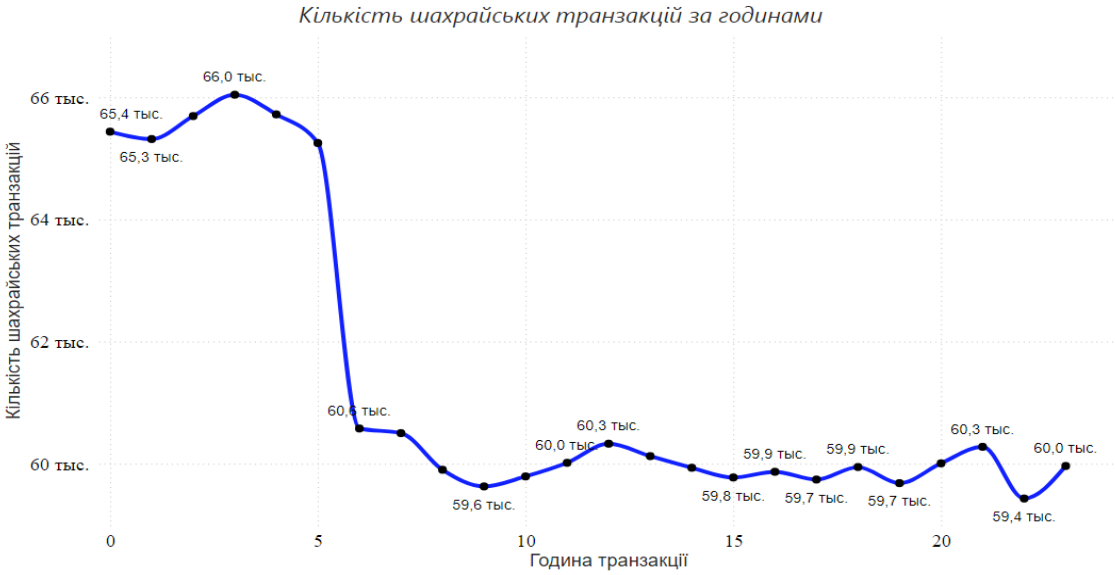

Рисунок 1.6 – Кількість шахрайських транзакцій за годинами дня

1.6. Обґрунтування доцільності проектування та розробки системи

Створення нової системи для виявлення шахрайських транзакцій є критично важливим кроком для компанії з кількох причин.

По-перше, ринок фінансових послуг постійно еволюціонує, і шахраї використовують нові технології для здійснення своїх злочинів. Власна система, що ґрунтується на алгоритмах машинного навчання, дозволить адаптуватися до нових загроз швидше й ефективніше [19].

По-друге, наявність великої кількості помилкових спрацювань у нинішніх системах призводить до перевантаження служби безпеки, що вимагає більше ресурсів на обробку підозрілих транзакцій. Нова система зможе зменшити цей показник, підвищуючи точність виявлення шахрайства.

Крім того, розробка власного рішення знизить залежність від сторонніх постачальників і забезпечить більше контролю над процесами обробки даних. Це не лише підвищить безпеку, а й зменшить витрати в довгостроковій перспективі.

Отже, актуальним завданням є проектування та розроблення нової автоматизованої системи виявлення шахрайських транзакцій, яка зможе ефективно протидіяти сучасним загрозам і забезпечити високий рівень захисту для компанії та її клієнтів [3][4].

1.7. Постановка задачі

Завданням цього проекту є дослідження та розробка автоматизованої системи для виявлення шахрайських транзакцій у фінансовому секторі компанії СОФТУМ. Система повинна використовувати алгоритми машинного навчання для аналізу даних транзакцій і виявлення аномальних операцій. Основна мета — підвищити ефективність виявлення шахрайства та зменшити кількість помилкових спрацювань.

Задачі, які необхідно вирішити в рамках проекту:

- **Збір та підготовка даних**: Збір історичних даних транзакцій, їхнє очищення та підготовка для навчання моделей.
- **Аналіз та обробка даних**: Дослідження структури даних, виявлення патернів і поведінкових аномалій.
- **Розробка моделі машинного навчання**: Підбір і налаштування моделі для точного прогнозування підозрілих транзакцій.
- **Інтеграція системи**: Розробка програмного рішення, яке дозволить автоматизувати процес моніторингу та аналізу транзакцій у режимі реального часу [16][17].

1.8. Висновок до розділу 1

У першому розділі було здійснено системний аналіз об’єкта дослідження — фінансових транзакцій компанії СОФТУМ, зосередившись на важливості боротьби з шахрайством у фінансовому секторі. Виявлено, що поточні системи автоматизації, засновані на статичних правилах, мають значні обмеження, зокрема виявлення нових видів шахрайства та високий рівень помилкових спрацювань.

Було проаналізовано існуючі рішення на ринку, такі як FICO Falcon, SAS Fraud Management та ACI Worldwide. Виявлено, що їхні можливості недостатньо адаптовані до специфіки діяльності компанії СОФТУМ, а також викликають ризики, пов'язані з безпекою даних і залежністю від сторонніх постачальників. Це

підтверджує доцільність розробки власної системи виявлення шахрайських транзакцій, яка буде враховувати особливості бізнесу компанії.

Значну увагу приділено аналізу ключових метрик, які дозволяють оцінити ефективність роботи системи, таких як кількість транзакцій за місяць, відсоток підозрілих операцій, кількість транзакцій за методами оплати та розподіл шахрайських транзакцій за годинами дня. Також обґрунтовано необхідність створення автоматизованої системи для підвищення точності виявлення шахрайства, зниження кількості помилкових спрацювань та підвищення рівня безпеки.

На основі аналізу сформульовано завдання проекту, яке передбачає розробку автоматизованої системи виявлення шахрайських транзакцій на основі алгоритмів машинного навчання. Основна мета полягає у створенні гнучкого, адаптивного рішення, яке забезпечить високу ефективність та швидкість виявлення шахрайства, мінімізуючи помилкові спрацювання і витрати на обробку транзакцій.

РОЗДІЛ 2. МАТЕМАТИЧНІ ОСНОВИ ВИЯВЛЕННЯ ШАХРАЙСТВА З ДОПОМОГОЮ МЕТОДІВ МАШИННОГО НАВЧАННЯ.

2.1 Машинне навчання, його типи.

Артур Самуел ще в 1959 році визначив машинне навчання як область дослідження, яка дає комп'ютеру здатність навчатися з даних, не будучи явно запрограмованим. Зараз IBM дає наступне означення машинного навчання:

Машинне навчання – це галузь штучного інтелекту (ШІ) та інформатики, яка зосереджується на використанні даних і алгоритмів для імітації способу навчання людей, поступово покращуючи його точність. А Amazon наводить твердження, що моделі машинного навчання генерують прогноз, знаходячи закономірності та шаблони в даних.

Загалом виділяють класичне навчання, ансамблеві методи, глибоке навчання та навчання з підкріпленням (див.рисунок 2.1) Класичне навчання в свою чергу поділяється на кероване (Supervised), некероване (Unsupervised). Як ілюстративний приклад, розглянемо завдання навчитися виявляти спам електронної пошти проти завдання виявлення аномалій. Для завдання виявлення спаму ми розглядаємо налаштування, за яких «учень» отримує навчальні електронні листи з міткою «спам/не спам».

На основі такого навчання «учень» повинен визначити правило позначення нових отриманих електронних листів. Навпаки, для завдання виявлення аномалії все, що «учень» отримує під час навчання, — це велика кількість повідомлень електронної пошти (без міток), а завдання «учня» — виявити «незвичайне» повідомлення.

Більш формалізовано, якщо розглядати навчання як процес «використання досвіду, щоб отримати експертизу», то кероване навчання описує сценарій, в якому досвід описується навчальним датасетом, що містить важливу інформацію (бажаний результат, мітку), якої, натомість немає в 29 майбутніх тестових прикладах, до яких планується застосовувати «навчений досвід»

В такому випадку може розглядати середовище як вчителя, який керує навчанням, надаючи додаткову інформацію (мітки). В некерованому навчанні відсутня різниця між навчальним та тестовим датасетом. В такому випадку навчання полягає в тому, щоб отримати коротшу (стиснену) версію даних або ж прийти до певного висновку на основі даних [5] [6].

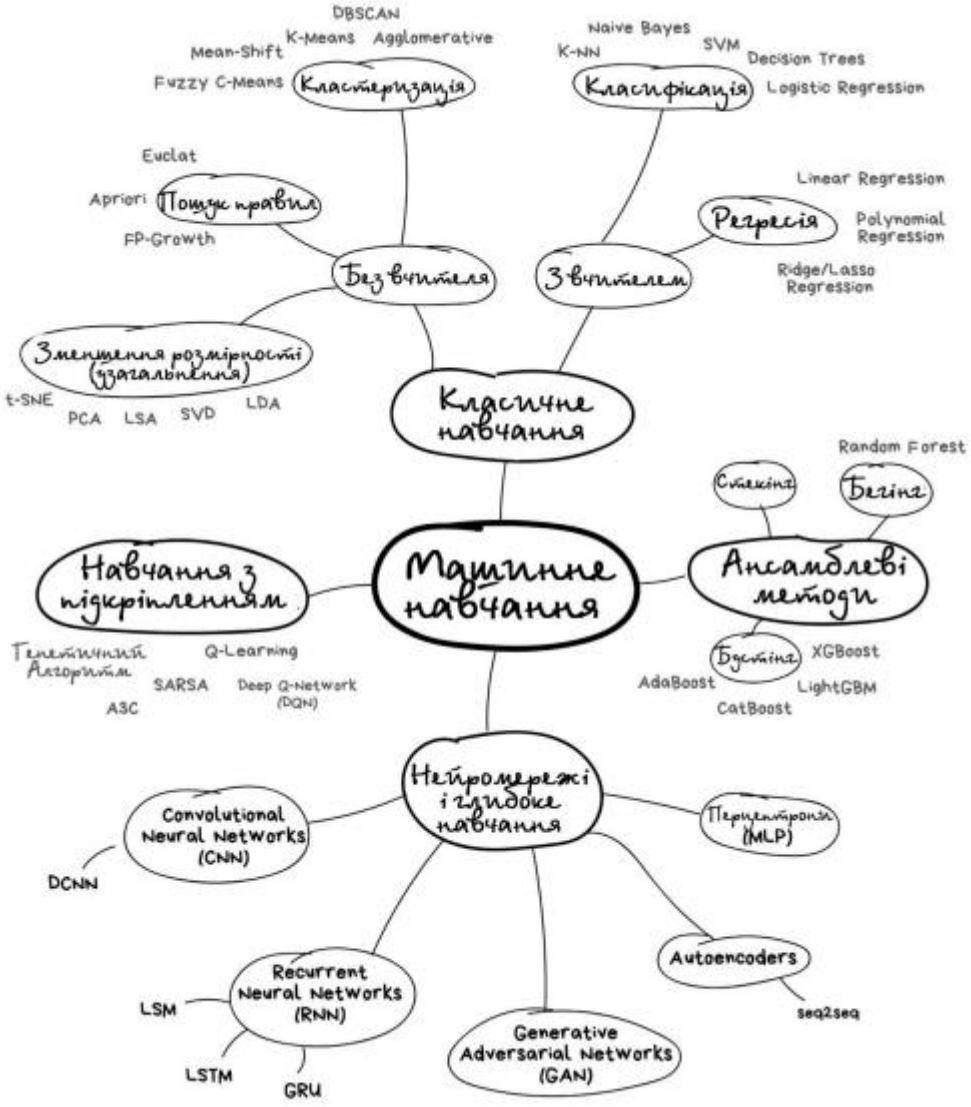

Рисунок 2.1 – Класифікація методів машинного навчання

Першим кроком є збір даних. Хорошою ідеєю є залучення експерта, який може допомогти визначити, які поля (атрибути, ознаки) є найбільш інформативними. Якщо ж немає такої можливості, можна застосувати метод «грубої сили», що полягає в вимірюванні всіх доступних параметрів у надії виявити

правильні (інформативні, релевантні) ознаки. Однак набір даних, зібраний таким чином, не підходить для безпосереднього використання в індуктивних процесах. Як правило, він містить шум і пропущені значення ознак, що вимагає значної попередньої обробки.

Іншим способом зменшення розміру даних є виділення значущих ознак або їх перетворення для зменшення розмірності вхідного простору. Вибір підмножини ознак полягає в ідентифікації та видаленні максимальної кількості нерелевантних і зайвих ознак. Це дозволяє зменшити розмірність даних, що сприяє більш швидкому та ефективному аналізу. Оскільки багато ознак можуть бути взаємозалежними, це може створювати надмірність, що негативно впливає на точність і продуктивність моделей класифікації машинного навчання. Цю проблему можна вирішити, створюючи нові функції на основі початкових ознак. Ця техніка називається конструюванням або перетворенням ознак і може призвести до створення більш лаконічних та точних класифікаторів [5] [6].

2.2 Методи для виявлення аномалій.

Виявлення аномалій є ключовою частиною систем, спрямованих на боротьбу з шахрайськими транзакціями. Це завдання полягає у виявленні нетипових, незвичних або підозрілих операцій, які можуть свідчити про шахрайську діяльність. Методи виявлення аномалій включають різноманітні підходи, які можуть бути поділені на статистичні методи, методи керованого навчання та некеровані методи навчання [5] [6].

Статистичні методи використовують математичні розподіли для виявлення відхилень від норми, в той час як методи керованого навчання потребують наявності розмічених даних для тренування моделей, що дозволяють класифікувати транзакції як нормальні чи аномальні. Некеровані методи, у свою чергу, працюють без попереднього маркування даних і можуть самостійно визначати аномалії на основі схожості або відмінності між об’єктами.

2.2.1 Статистичні методи виявлення аномалій.

Статистичні методи виявлення аномалій ґрунтуються на аналізі математичних характеристик даних, таких як середнє значення, дисперсія та інші статистичні параметри. Ці методи дозволяють виявляти відхилення від нормального розподілу або заданих статистичних характеристик. Основною ідеєю є визначення "нормального" поведінкового профілю для набору даних і виявлення значень, що значно відрізняються від цього профілю.

До основних статистичних методів належать метод стандартних відхилень, який припускає, що дані мають нормальний розподіл і дозволяє виявляти аномальні значення, що відрізняються від середнього на кілька стандартних відхилень. Метод Z-оцінки використовує нормалізацію даних і дозволяє виявляти значення, що знаходяться за межами певного порогу. Метод оцінки ймовірності будує модель ймовірності належності точки даних до певного класу, і якщо ця ймовірність низька, точка вважається аномальною. Моделі на основі коробчатих діаграм використовують межі для виявлення значень, що виходять за допустимі межі, вказуючи на потенційні аномалії. Також застосовуються багатовимірні статистичні методи для виявлення аномалій у даних з кількома ознаками, оцінюючи кореляції між ними.

Статистичні методи мають переваги в простоті реалізації та ефективності при аналізі стабільних і зрозумілих даних. Однак їх недолік полягає в чутливості до шуму в даних і обмеженій ефективності у випадках складних або невизначених структур даних.

2.2.2 Виявлення аномалій методами методами керованого навчання.

Методи виявлення аномалій за допомогою керованого навчання ґрунтуються на використанні міток для навчання моделей. У таких системах для навчання використовуються дані, які мають чітко позначені аномальні та нормальні приклади. Це дозволяє алгоритмам створювати моделі, здатні розпізнавати схеми та відмінності між нормальними і аномальними транзакціями. Методика зазвичай

включає класифікацію або регресію, де модель навчається на основі історичних даних, щоб визначити ймовірність того, що нові дані є аномальними.

Один з основних підходів полягає в використанні таких методів, як дерева рішень, наївний баєсів класифікатор, метод опорних векторів (SVM), які здатні будувати розподіли для кожного класу і визначати, чи належать нові дані до аномальних точок. Класичний підхід полягає в тому, щоб створити модель на основі тренувальних даних, де кожен приклад має мітку "нормальний" або "аномальний". Потім, за допомогою цієї моделі, визначається ймовірність аномальності нових даних [5] [6].

Методи керованого навчання мають сильну перевагу у точності виявлення аномалій у тому випадку, якщо дані добре мічені і є достатньо прикладів для навчання. Однак ці методи можуть мати низьку ефективність на незбалансованих наборах даних, де кількість аномальних випадків значно менша за кількість нормальних. У таких випадках класичні моделі можуть схильні до переважання нормальних класів, не виявляючи рідкісні аномалії. Тому для роботи з незбалансованими даними часто використовуються спеціальні методи, такі як підвищення ваги аномальних класів або використання методу генерації нових синтетичних прикладів для аномальних класів (наприклад, метод SMOTE).

2.2.1.1 Незбалансованість даних.

Незбалансованість даних є поширеною проблемою у задачах виявлення аномалій, зокрема при боротьбі з шахрайськими транзакціями. Вона виникає, коли одна з категорій (наприклад, нормальні транзакції) значно переважає в наборі даних порівняно з іншою категорією (аномальними, тобто шахрайськими транзакціями). Це створює виклики для алгоритмів машинного навчання, оскільки моделі можуть схилятися до передбачення домінуючого класу (нормальних транзакцій), ігноруючи аномальні випадки, які мають набагато меншу ймовірність бути виявленими.

Незбалансованість даних значно ускладнює процес навчання, оскільки більшість моделей машинного навчання зазвичай орієнтуються на зниження

загальної помилки, а не на досягнення високої точності для меншості класу. Тому, якщо модель має більшу кількість нормальних транзакцій, вона може просто "вгадувати" нормальні транзакції, що приводить до низької ефективності виявлення аномалій.

Для подолання цієї проблеми застосовуються різні підходи:

- Балансування даних
- Зважування класів
- Адаптовані методи навчання
- Оцінка ефективності.

2.2.1.2 Методи керованого класичного навчання для роботи з незбалансованими даними.

Методи керованого класичного навчання для роботи з незбалансованими даними фокусуються на тому, щоб адаптувати традиційні алгоритми машинного навчання до умов, коли одна з категорій значно переважає іншу. Це особливо актуально для задач виявлення аномалій, таких як виявлення шахрайських транзакцій, де нормальні випадки значно перевищують кількість аномальних. Існує кілька підходів для покращення роботи з незбалансованими даними:

1. Підвибірка (undersampling) домінуючого класу
2. Підвибірка з використанням адаптивних алгоритмів
3. Перевибірка (oversampling) меншості класу
4. Зважування класів
5. Використання модифікованих алгоритмів.

Наприклад:

- Алгоритм дерев рішень з оптимізацією для незбалансованих даних може використовувати методи обмеження глибини дерева або адаптацію критеріїв розбиття, щоб зосередитися на меншості класу.
- Методи опорних векторів (SVM) з коригуванням ваг для класів дозволяють отримати кращі результати в умовах сильної незбалансованості.

- Нейронні мережі можна адаптувати до незбалансованих даних шляхом використання вагових коефіцієнтів для кожного класу або через застосування методів, таких як dropout, щоб уникнути перенавчання на домінуючому класі.

6. Комбінація підходів: У реальних задачах для досягнення найкращих результатів застосовують комбінацію підвибірки, перевибірки та зважування класів, що дозволяє більш точно налаштувати модель на роботу з незбалансованими даними, мінімізуючи ймовірність перенавчання і покращуючи точність виявлення аномалій.

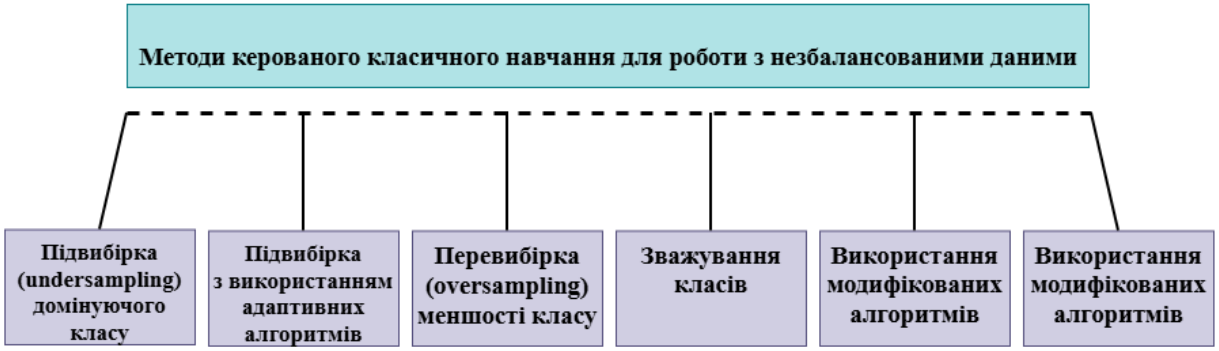

Рисунок 2.2 – Методи керованого класичного навчання

Ці методи дозволяють покращити точність виявлення рідкісних подій, таких як шахрайські транзакції, і зменшити ймовірність того, що модель не зможе ідентифікувати важливі аномальні випадки через домінування одного класу.[29-34]

2.3 Некеровані методи навчання для виявлення та роботи з аномаліями.

Некеровані методи навчання для виявлення та роботи з аномаліями є важливим інструментом для виявлення незвичних або підозрілих подій у даних, де відсутні мітки класів (наприклад, у випадку, коли транзакції не мають чіткої категорії "нормальна" або "шахрайська"). Ці методи дозволяють виявляти аномалії без необхідності попереднього навчання на мітках або класах.

Метод найближчих сусідів (K-Nearest Neighbors, KNN) є одним з основних інструментів для виявлення аномалій. Суть методу полягає в тому, що аномальні точки визначаються через відстань до найближчих сусідів. Якщо точка знаходиться

далеко від інших даних у просторі ознак, то вона вважається аномалією. У випадку транзакцій з шахрайським характером, ці операції зазвичай мають відмінні параметри порівняно з більшістю нормальних операцій.

Алгоритм локальної різниці в щільності (Local Outlier Factor, LOF) використовує концепцію локальної щільності для виявлення аномалій. Він оцінює, наскільки дані є аномальними, порівнюючи щільність сусідів кожної точки з її локальною щільністю. Точки, що мають значно меншу щільність порівняно з сусідами, вважаються аномальними. LOF добре працює, коли аномалії мають локальні відмінності в щільності, що є характерним для багатьох реальних випадків шахрайських транзакцій.

Алгоритм ізоляційних лісів (Isolation Forest) є методом для виявлення аномалій, заснованим на ідеї, що аномалії легше ізолюються від решти даних. Алгоритм будує дерева, що ізолюють точки на основі випадкових розподілів та порогових значень для кожної ознаки, і чим швидше точка ізолюється, тим вищі шанси, що вона є аномалією. Це дозволяє ефективно виявляти аномалії, навіть у великих і складних наборах даних [7].

2.4 Виклики та проблеми в системах виявлення аномалій.

Системи виявлення аномалій стикаються з низкою викликів і проблем, які можуть ускладнити ефективну і точну ідентифікацію шахрайських або підозрілих транзакцій. Одним з головних викликів є незбалансованість даних, коли кількість нормальних транзакцій значно перевищує кількість аномальних. У таких умовах алгоритми можуть з високою ймовірністю передбачати нормальні транзакції, що призводить до великої кількості помилкових спрацьовувань або пропуску важливих аномалій.

Ще однією проблемою є шум у даних, коли деякі з нормальних транзакцій можуть виглядати як аномалії через випадкові відхилення або неповні дані. Шум може бути результатом помилок у введенні даних, збоїв у системах або неточностей у вимірюваннях, що ускладнює точну ідентифікацію аномалій.

Вибір правильної моделі також є складним завданням. Багато методів виявлення аномалій потребують налаштування параметрів або вибору найбільш релевантних ознак, що може вимагати значних ресурсів і часу. Для цього потрібно враховувати характер даних, тип аномалій та цілі виявлення.

Іншим викликом є динамічність даних, оскільки поведінка користувачів і шахрайські методи можуть змінюватися з часом. Системи, що не оновлюються або не адаптуються до нових умов, можуть швидко стати неефективними, пропускаючи нові типи аномалій або шахрайських схем [24-28].

Також, важливою проблемою є інтерпретація результатів. Багато алгоритмів виявлення аномалій, зокрема, методи на основі машинного навчання, можуть бути непрозорими в своїх рішеннях, що ускладнює розуміння того, чому певна транзакція була позначена як аномалія. Для цього необхідні методи інтерпретації, які дозволяють пояснити, чому система зробила певне рішення.

Нарешті, реальні обмеження на обчислювальні ресурси можуть вплинути на здатність обробляти великі обсяги даних, що є типовим для фінансових транзакцій. Розподілені або масштабовані системи можуть бути необхідні для ефективної обробки великих наборів даних у реальному часі [8].

2.5 Висновок до другого розділу.

У другому розділі були розглянуті основні методи та підходи для виявлення аномалій, які є ключовими для побудови ефективних систем боротьби з шахрайськими транзакціями. Початково було описано важливість виявлення аномалій у фінансових операціях, оскільки це дозволяє своєчасно виявити підозрілі або незвичні транзакції, що можуть вказувати на шахрайство.

Розглянуті методи виявлення аномалій поділяються на статистичні, методи керованого навчання та некеровані методи навчання. Статистичні методи виявлення аномалій дозволяють аналізувати відхилення від загальних тенденцій у даних, однак вони часто вимагають точного налаштування параметрів та можуть бути чутливими до шуму. Методи керованого навчання, зокрема методи, спрямовані на роботу з незбалансованими даними, дозволяють підвищити точність

виявлення аномалій, проте мають свої проблеми, зокрема через потребу в великих обсягах навчальних даних. Некеровані методи, на відміну від методів керованого навчання, не потребують наявності міток у даних і можуть бути корисними в ситуаціях, коли розмітка даних відсутня.

Водночас, на шляху до створення ефективних систем виявлення аномалій існують серйозні виклики, серед яких незбалансованість даних, шум, необхідність налаштування моделей та динамічні зміни в поведінці користувачів. Крім того, важливими є питання інтерпретованості результатів та реальні обмеження на обчислювальні ресурси.

У результаті, для побудови високоефективних систем виявлення аномалій важливо враховувати різноманітні методи, адаптувати їх до специфіки даних та розв'язувати виклики, що виникають в процесі їх застосування, що дозволить покращити якість і швидкість виявлення шахрайських транзакцій.

РОЗДІЛ 3. ДОСЛІДЖЕННЯ ТА РОЗРОБКА СИСТЕМИ ДЛЯ АВТОМАТИЗОВАНОГО ВИЯВЛЕННЯ АНОМАЛЬНИХ ТРАНЗАКЦІЙ У ФІНАНСОВІЙ СФЕРІ

3.1 Вибір середовища програмування.

Python заслужив своє місце як одна з найпопулярніших мов програмування серед професіоналів машинного навчання завдяки легкому для читання синтаксису, великим бібліотекам і крос-платформній сумісності. Як мова програмування високого рівня з відкритим вихідним кодом, Python став найкращим вибором для широкого спектру завдань машинного навчання, від аналізу даних до глибокого навчання [11]. Зростаюча популярність Python у проектах зі штучного інтелекту (ШІ) і машинного навчання (МН) не випадкова, оскільки вона забезпечує чудове середовище для розробників, щоб вирішувати навіть найскладніші завдання машинного навчання [9].

Розгалужена бібліотечна екосистема Python, надійні можливості візуалізації, низький бар’єр входу, потужна підтримка спільноти, гнучкість, читабельність і незалежність від платформи роблять його ідеальним вибором для цілей машинного навчання [12]. Як наслідок, Python спостерігав сплеск використання в програмах ШІ та МН, включаючи розпізнавання зображень і мови, прогнозну аналітику.

Зростаюча популярність Python у проєктах штучного інтелекту сьогодні не є простою випадковістю. Його всеосяжна бібліотечна екосистема та активна спільнота розробників полегшили, ніж будь-коли, роботу професіоналам машинного навчання. Завдяки зручному для читання синтаксису, широким бібліотекам і крос-платформній сумісності Python став важливим інструментом для розробників штучного інтелекту та машинного навчання в усьому світі [42-43].

Python пропонує безліч переваг як для професіоналів машинного навчання, так і для ентузіастів, особливо при роботі з моделями машинного навчання за допомогою мови Python, зокрема:

Інтуїтивно зрозумілий синтаксис. Це робить Python популярною мовою програмування, яку легко читати. Об'єктно-орієнтоване програмування надає розробникам логічний метод організації, обробки та планування коду відповідно. Це полегшує розробку чистого та лаконічного коду для проектів будь-якої складності. У результаті Python став популярною початковою мовою для розробників-початківців і вибором для досвідчених програмістів. Легкий для читання синтаксис Python не тільки робить його доступним для початківців, але також дозволяє швидше розробляти та налагоджувати.

З Python код стає більш розбірливим і легшим для налагодження, що полегшує виявлення та виправлення помилок і оперативну розробку нових функцій. Цей зручний характер Python значносприяв його широкому впровадженню в спільноті машинного навчання [10].

Кросплатформна сумісність. Кросплатформна сумісність Python дозволяє розробникам створювати код, який можна використовувати на різних платформах, таких як Windows, Mac і Linux. Ця гнучкість полегшує розробку програм, які можна використовувати в різних операційних системах без необхідності переписувати вихідний код. Таким чином, це дозволяє розробникам використовувати той самий код для різних платформ, заощаджуючи час і зусилля. Однак кросплатформна сумісність пов'язана з певними труднощами.

На різних платформах можуть бути встановлені різні версії Python, що може

призвести до проблем із сумісністю під час виконання коду на різних платформах. Щоб подолати ці проблеми, важливо переконатися, що код написаний у спосіб, сумісний з усіма підтримуваними версіями, і що він протестований на всіх платформах, щоб гарантувати, що він функціонує належним чином [36-38].

Масштабування та продуктивність. Python широко відомий своєю масштабованістю та винятковою продуктивністю в машинному навчанні. Його універсальність, зручний характер і великі бібліотеки роблять його ідеальним вибором для масштабування операцій машинного навчання. Завдяки таким бібліотекам, як NumPy, Pandas і TensorFlow, Python дає змогу виконувати складні операції з масивними наборами даних, демонструючи свою високу

масштабованість. Його вміння працювати з великими даними сприяє його широкому застосуванню. Простота та читабельність Python ще більше сприяють швидкому створенню прототипів, прискорюючи ітераційний процес розробки та тонкого налаштування моделей МН. Однак продуктивність Python створює проблеми. Як інтерпретована мова, Python є відносно повільнішим порівняно з такими мовами, як C++ або Java. Тим не менш, такі бібліотеки, як NumPy і Cython, вирішують цю проблему, виконуючи обчислення зі швидкістю, близькою до C. Крім того, інфраструктури розподілених обчислень, такі як Apache Spark і Dask, значно підвищують продуктивність Python у програмах МН. Загалом, багатий набір бібліотек, простота використання та масштабованість Python роблять його надійним вибором для машинного навчання [10].

Великі бібліотеки та фреймворки.Одним із ключових факторів, які відрізняють Python від інших мов програмування, є його комплексна бібліотечна екосистема. Python пропонує широкий спектр бібліотек і фреймворків, спеціально розроблених для машинного навчання, що полегшує розробникам впровадження алгоритмів МН [15]. Нижче наведено основні популярні бібліотеки Python для машинного навчання:

Рисунок 3.1 – Бібліотеки та фреймворки

- NumPy — це фундаментальна бібліотека Python для ефективних числових обчислень і операцій з масивами.
- Scikit-learn — це комплексна бібліотека машинного навчання, яка пропонує широкий спектр інструментів для різних завдань, зокрема класифікації, регресії, кластеризації тощо [13][15].
- Pandas — це потужна бібліотека для аналізу та обробки даних, що забезпечує інтуїтивно зрозумілі структури даних, такі як DataFrames і Series [35].
- TensorFlow — це передова бібліотека глибокого навчання, відома своїми можливостями розподіленого обчислення та надійною екосистемою [14].
- Theano — це бібліотека Python, розроблена для швидких числових обчислень, особливо корисна для навчання моделей глибокого навчання.
- Keras — це простий у використанні API глибокого навчання, який діє як інтерфейс для TensorFlow, Theano або Microsoft Cognitive Toolkit (CNTK), спрощуючи створення та навчання нейронних мереж [13].

- PyTorch — це динамічна бібліотека глибокого навчання з гнучким графіком обчислень, що робить її ідеальною для розробки та навчання складних нейронних мереж [10].

Ці бібліотеки та фреймворки Python надають потужні можливості для аналізу даних, машинного та глибокого навчання, дозволяючи розробникам зосередитися на вирішенні складних завдань без необхідності винаходити колесо. Завдяки цій чудовій бібліотечній екосистемі Python став незамінним інструментом для інженерів машинного навчання, науковців із обробки даних і дослідників [17].

3.2 Опис датасету.

Датасет із Kaggle - https://www.kaggle.com/code/benroshan/transaction-fraud-detection/notebook стосується виявлення шахрайських фінансових транзакцій і призначений для навчання моделей машинного навчання. Він складається з таблиці, де кожний рядок представляє транзакцію, а стовпці — її характеристики. Основні атрибути включають ідентифікатор транзакції, дату й час, суму, спосіб оплати, місцезнаходження клієнта, категорію транзакції та мітку, що вказує, чи є транзакція шахрайською [16].

Дані дозволяють виконати категоріальний, часовий, географічний і числовий аналіз для виявлення закономірностей шахрайських дій. Потенційні труднощі роботи з датасетом включають нерівномірність класів, відсутність даних і наявність шуму. Цей набір даних добре підходить для створення моделей класифікації та аналізу аномалій [39-41].

Він також є ідеальним для дослідження та розробки системи автоматизованого виявлення аномальних транзакцій у фінансовій сфері. Його структура сприяє аналізу ключових аспектів шахрайства: мітка "шахрайської транзакції" дозволяє побудувати модель класифікації, атрибути транзакцій дають змогу виділити патерни поведінки клієнтів, а часові дані допомагають визначити пікові періоди шахрайської активності. Завдяки багатству інформації цей датасет надає всі необхідні умови для тестування, вдосконалення та адаптації моделей машинного навчання до реальних викликів [22].

3.3 Реалізація системи.

Підключення бібліотек:

```
#Basic libraries
import pandas as pd
import numpy as np

#Visualization libraries
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
from plotly.offline import plot, iplot, init_notebook_mode
init_notebook_mode(connected=True)
%matplotlib inline

#preprocessing libraries
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

Рисунок 3.2 – Підключення бібліотек

```
#ML libraries
import tensorflow as tf
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline

#Metrics Libraries
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix



#Misc libraries
import warnings
warnings.filterwarnings("ignore")
```

Рисунок 3.3 – Підключення бібліотек

Набір даних містить інформацію про фінансові транзакції, що використовується для аналізу та виявлення шахрайства. Через великий обсяг даних часто зручно працювати лише з вибіркою, наприклад, першими 50,000 рядками. Такий підхід дозволяє оптимізувати час обробки й аналізу, не втрачаючи при цьому суттєвої інформації для побудови моделей [22].

Цей підхід дає можливість швидко отримати огляд даних, включаючи:

- Основні характеристики транзакцій, такі як сума, спосіб оплати, категорія товарів і послуг.
- Часові аспекти, зокрема дати й час проведення операцій.

- Географічну інформацію, пов’язану з місцезнаходженням клієнтів.
- Мітки про шахрайство, які допомагають визначити цільовий клас для моделей машинного навчання.
- Обмеження вибірки забезпечує баланс між швидкістю обробки й інформативністю аналізу, що є важливим кроком для подальшого створення ефективної системи автоматизованого виявлення аномальних транзакцій [22].

```
#Reading the data
paysim=pd.read_csv('PS_20174392719_1491204439457_log.csv')

#Looking at the data
paysim.head()
```

Рисунок 3.4 – Підключення до набору даних

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0

Рисунок 3.5 – Підключення до набору даних

Опис набору даних:

- крок - відображає одиницю часу в реальному світі. У цьому випадку 1 крок - це 1 година часу. Всього кроків 744 (30 днів моделювання).
- тип - ВНЕСЕННЯ, ВНЕСЕННЯ, ДЕБЕТ, ОПЛАТА та ПЕРЕКАЗ.
- сума - сума операції в національній валюті.
- nameOrig - клієнт, який почав транзакцію
- oldbalanceOrg - початковий баланс до транзакції
- newbalanceOrig - новий баланс після транзакції
- nameDest - клієнт, який є одержувачем транзакції
- oldbalanceDest - початковий одержувач балансу до транзакції. Зауважте, що для клієнтів, які починаються на літеру M (Продавці), немає інформації.
- newbalanceDest - новий одержувач балансу після транзакції. Зауважте, що для клієнтів, які починаються на літеру M (Продавці), немає інформації.

- isFraud – це транзакції, здійснені агентами-шахраями всередині симуляції. У цьому конкретному наборі даних шахрайська поведінка агентів спрямована на отримання прибутку шляхом контролю над рахунками клієнтів і спроби спустошити кошти, переказавши їх на інший рахунок, а потім вивівши з системи [11].
- isFlaggedFraud – бізнес-модель спрямована на контроль масових переказів з одного рахунку на інший і позначає незаконні спроби.

  Незаконна спроба в цьому наборі даних – це спроба передати понад 200 000 за одну транзакцію [22].

```
paysim.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 11 columns):
 #   Column          Dtype
---  ------          -----
 0   step            int64
 1   type            object
 2   amount          float64
 3   nameOrig        object
 4   oldbalanceOrg   float64
 5   newbalanceOrig  float64
 6   nameDest        object
 7   oldbalanceDest  float64
 8   newbalanceDest  float64
 9   isFraud         int64
 10  isFlaggedFraud  int64
dtypes: float64(5), int64(3), object(3)
memory usage: 534.0+ MB
```

Рисунок 3.6 – Опис набору даних

Аналіз зведеної таблиці: Цифри — це все в моніторингу транзакцій. Цифри вирішують, чи це шахрайство чи ні. Подивимося на загальні цифри за допомогою функції зведення

	sum			std		
	amount	isFlaggedFraud	isFraud	amount	isFlaggedFraud	isFraud
type						
CASH_IN	236367391912.459991	0	0	126508.255272	0.000000	0.000000
CASH_OUT	394412995224.489990	0	4116	175329.744483	0.000000	0.042851
DEBIT	227199221.280000	0	0	13318.535518	0.000000	0.000000
PAYMENT	28093371138.369999	0	0	12556.450186	0.000000	0.000000
TRANSFER	485291987263.169983	16	4097	1879573.528908	0.005479	0.087344
All	1144392944759.770020	16	8213	603858.184009	0.001586	0.035905

Рисунок 3.7 – Аналіз зведеної таблиці

Відповідно до поточного алгоритму, заснованого на правилах, під час шахрайських транзакцій не було жодних позначок у випадку cash_out, що викликає серйозне занепокоєння для системи боротьби з відмиванням грошей [18]. Крім того, лише 16 транзакцій позначено як шахрайство, тоді як приблизно 4 тисячі транзакцій насправді є шахрайством. Зараз наша місія полягає в створенні ефективного алгоритму для зменшення ризику розблокування шахрайських транзакцій.

```
paysim_pivot2=pd.pivot_table(paysim,index=["type"],
                                values=['amount','oldbalanceOrg','newbalanceOrig'],
                                aggfunc=[np.sum], margins=True)

#Adding style
paysim_pivot2.style\
    .format('{:.2f}')\
    .bar(align='mid',color=['darkred'])\
    .set_properties(padding='5px',border='3px solid white',width='200px')
```

Рисунок 3.8 – Аналіз зведеної таблиці

	sum		
	amount	newbalanceOrig	oldbalanceOrg
type			
CASH_IN	236367391912.46	5260438481752.40	5024078139747.30
CASH_OUT	394412995224.49	39098506249.34	102978263227.81
DEBIT	227199221.28	2699777564.12	2844196471.80
PAYMENT	28093371138.37	133043913105.10	146768163438.79
TRANSFER	485291987263.16	5482651300.55	29012552760.76
All	1144392944759.77	5440763329971.49	5305681315646.41

Рисунок 3.9 – Аналіз зведеної таблиці

З таблиці ми можемо зрозуміти, що більшість клієнтів використовують систему для переказу грошей, і ми маємо відносно менше даних для здійснених платежів. Також досить цікаво помітити різницю між новим і старим балансом, оскільки це розповідає нам деякі історії. Тут ми маємо лише візуальні елементи облікового запису Orgin, а баланс готівки зменшено в усіх випадках, крім cash_in. Навіть під час переказу баланс зменшився, що свідчить про те, що ми маємо більше інформації про відправника в початковому обліковому записі.

```
paysim_pivot3=pd.pivot_table(paysim,index=["type"],
                             values=['amount','oldbalanceDest','newbalanceDest'],
                             aggfunc=[np.sum], margins=True)

#Adding style
paysim_pivot3.style\
    .format('{:.2f}')\
    .bar(align='mid',color=['darkblue'])\
    .set_properties(padding='5px',border='3px solid white',width='200px')
```

Рисунок 3.10 – Аналіз зведеної таблиці

	sum		
	amount	newbalanceDest	oldbalanceDest
type			
CASH_IN	2363[redacted]7391912.46	2052[redacted]091879.15	2221[redacted]65238.98
CASH_OUT	3944[redacted]224.49	3784[redacted]15	3351[redacted]78
DEBIT	227199221.28	62686759687.09	61863601275.22
PAYMENT	28093371138.37	0.00	0.00
TRANSFER	485291[redacted].16	1894[redacted]0653500.26	1368[redacted]00197339.44
All	1144392944759.77	7794186583127.56	7003346437431.25

Рисунок 3.11 – Аналіз зведеної таблиці

У цій таблиці ми маємо інформацію про цільовий рахунок, з інформації про переказ ми можемо побачити збільшення нового балансу, отже, це інформація одержувача. Немає доступної суми платежу для інформації про призначення.

Розподіл суми

Важливо розуміти розподіл наших даних, оскільки він може відігравати важливу роль у створенні моделі, а також у розумінні наших даних. Надалі ми використовуватимемо лише 50 тисяч рядків, оскільки обробка всіх записів для

візуалізації та побудови моделі займає багато часу. Тут ми перевіряємо розподіл суми транзакції за допомогою програми [22].

```
paysim=pd.read_csv('PS_20174392719_1491204439457_log.csv',nrows=50000)

#Distribution of Amount
fig = px.box(paysim, y="amount")
fig.show()
```

Рисунок 3.12 – Розподіл суми

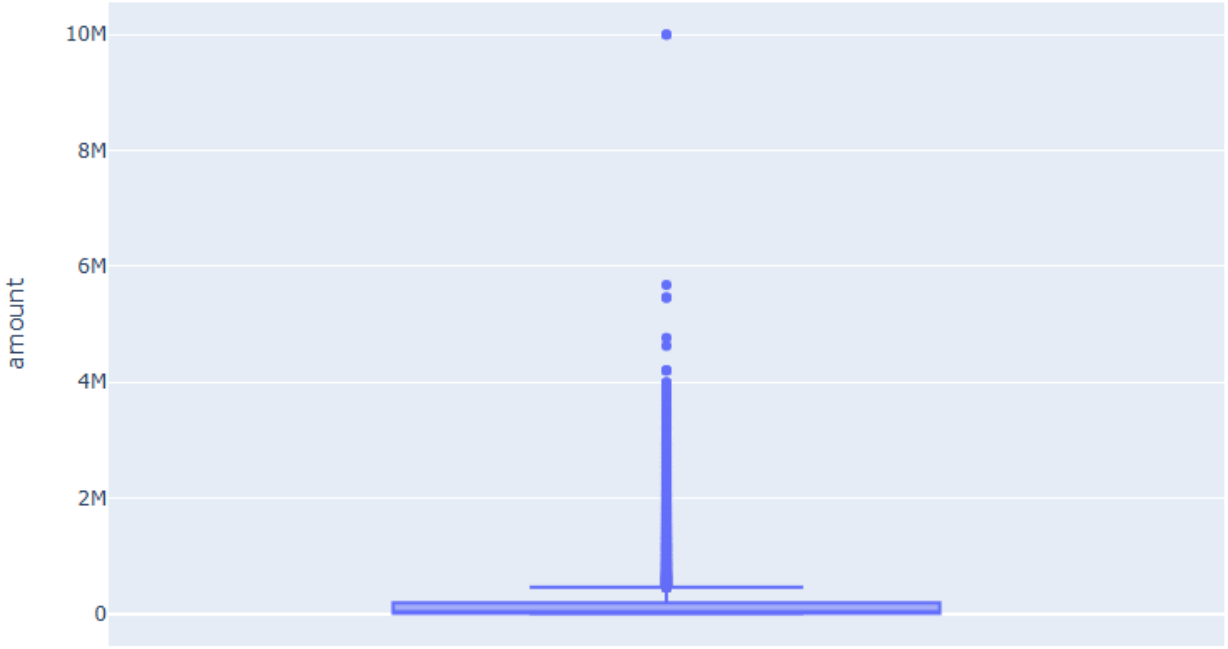

Рисунок 3.13 – Розподіл суми

Зі стовпчастої діаграми ми можемо зрозуміти, що ми маємо правильний перекіс набору даних, є багато викидів, які сягають 10 млн із медіаною 33 тис. Верхня дужка (75-й процентиль) нараховує до 450 тис.

Розробка функцій

Час забруднити руки розробкою функцій. З наявною інформацією важко навчити модель і отримати кращі результати. Тому ми переходимо до створення нових функцій, змінюючи існуючі.

Різниця в балансі: загальновідомо, що сума, списана з рахунку відправника, зараховується на рахунок одержувача без будь-яких відхилень у центах. Але що робити, якщо в сумі списання і зарахування є відхилення. Деякі з них можуть бути

спричинені оплатою, яку стягують постачальники послуг, але ми повинні позначати такі незвичайні випадки [22].

Індикатор сплеску: також ми повинні активувати прапор, коли в транзакцію залучено велику суму. З розподілу суми ми зрозуміли, що у нас є багато викидів із високою сумою транзакцій. Тому ми вважаємо 75-й процентиль (450 тис.) нашим порогом, і сума, яка перевищує 450 тис., буде активовано як позначку.

Індикатор частоти: тут ми позначаємо користувача, а не транзакцію. Коли є одержувач, який отримує гроші від великої кількості людей, це може бути тригером, оскільки це може бути для деяких незаконних азартних ігор чи удачі. Тому це позначається, коли є отримувач, який отримує гроші більше 20 разів.

Індикатор торговця: ідентифікатори клієнтів у приймачі починаються з «M», що означає, що вони є торговцями, і вони, очевидно, матимуть багато транзакцій отримання [22].

```
def balance_diff(data):
    '''balance_diff checks whether the money debited from sender has exactly credited to the receiv
er
       then it creates a new column which indicates 1 when there is a deviation else 0'''
    #Sender's balance
    orig_change=data['newbalanceOrig']-data['oldbalanceOrg']
    orig_change=orig_change.astype(int)
    for i in orig_change:
        if i<0:
            data['orig_txn_diff']=round(data['amount']+orig_change,2)
        else:
            data['orig_txn_diff']=round(data['amount']-orig_change,2)
    data['orig_txn_diff']=data['orig_txn_diff'].astype(int)
    data['orig_diff'] = [1 if n !=0 else 0 for n in data['orig_txn_diff']]

    #Receiver's balance
    dest_change=data['newbalanceDest']-data['oldbalanceDest']
    dest_change=dest_change.astype(int)
    for i in dest_change:
        if i<0:
            data['dest_txn_diff']=round(data['amount']+dest_change,2)
        else:
            data['dest_txn_diff']=round(data['amount']-dest_change,2)
    data['dest_txn_diff']=data['dest_txn_diff'].astype(int)
    data['dest_diff'] = [1 if n !=0 else 0 for n in data['dest_txn_diff']]
```

Рисунок 3.14 – Розробка функцій

```
    data.drop(['orig_txn_diff','dest_txn_diff'],axis=1,inplace = True)

#Surge indicator
def surge_indicator(data):
    '''Creates a new column which has 1 if the transaction amount is greater than the threshold
    else it will be 0'''
    data['surge']=[1 if n>450000 else 0 for n in data['amount']]

#Frequency indicator
def frequency_receiver(data):
    '''Creates a new column which has 1 if the receiver receives money from many individuals
    else it will be 0'''
    data['freq_Dest']=data['nameDest'].map(data['nameDest'].value_counts())
    data['freq_dest']=[1 if n>20 else 0 for n in data['freq_Dest']]

    data.drop(['freq_Dest'],axis=1,inplace = True)

#Tracking the receiver as merchant or not
def merchant(data):
    '''We also have customer ids which starts with M in Receiver name, it indicates merchant
    this function will flag if there is a merchant in receiver end '''
    values = ['M']
    conditions = list(map(data['nameDest'].str.contains, values))
    data['merchant'] = np.select(conditions, '1', '0')
```

Рисунок 3.15 – Розробка функцій

```
#Applying balance_diff function
balance_diff(paysim)

paysim['orig_diff'].value_counts()
paysim['dest_diff'].value_counts()
```

Рисунок 3.16 – Розробка функцій

```
1    44994
0     5006
Name: dest_diff, dtype: int64
```

Рисунок 3.17 – Розробка функцій

```
#Applying surge_indicator function
surge_indicator(paysim)
paysim['surge'].value_counts()
```

Рисунок 3.18 – Розробка функцій

```
0    46392
1     3608
Name: surge, dtype: int64
```

Рисунок 3.19 – Розробка функцій

```
#Applying frequency_receiver function
frequency_receiver(paysim)
paysim['freq_dest'].value_counts()
```

Рисунок 3.20 – Розробка функцій

```
0    46295
1     3705
Name: freq_dest, dtype: int64
```

Рисунок 3.21 – Розробка функцій

Попередня обробка даних

Перш ніж переходити до створення моделі машинного навчання, необхідно попередньо обробити дані, щоб модель навчалася без будь-яких помилок і могла краще навчатися, щоб забезпечити кращі результати.

1. Балансування мішені

З наведеної нижче кругової діаграми ми чітко бачимо, що цільова мітка сильно дисбалансована, оскільки ми маємо лише 0,2% шахрайських даних, яких недостатньо для того, щоб машина могла навчатися та позначати випадки шахрайства.

```
paysim_1=paysim.copy()

#Checking for balance in target
fig = go.Figure(data=[go.Pie(labels=['Not Fraud','Fraud'], values=paysim_1['isFraud'].value_count
s())])
fig.show()
```

Рисунок 3.22 – Балансування мішені

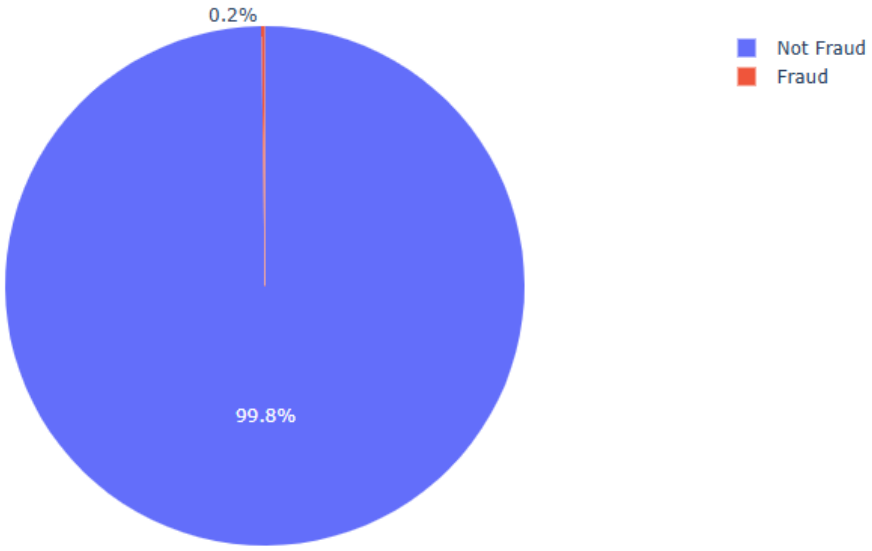

Рисунок 3.23 – Балансування мішені

```
max_size = paysim_1['isFraud'].value_counts().max()

#Balancing the target label
lst = [paysim_1]
for class_index, group in paysim_1.groupby('isFraud'):
    lst.append(group.sample(max_size-len(group), replace=True))
paysim_1 = pd.concat(lst)
```

Рисунок 3.24 – Балансування мішені

```
fig = go.Figure(data=[go.Pie(labels=['Not Fraud','Fraud'], values=paysim_1['isFraud'].value_count
s())])
fig.show()
```

Рисунок 3.25 – Балансування мішені

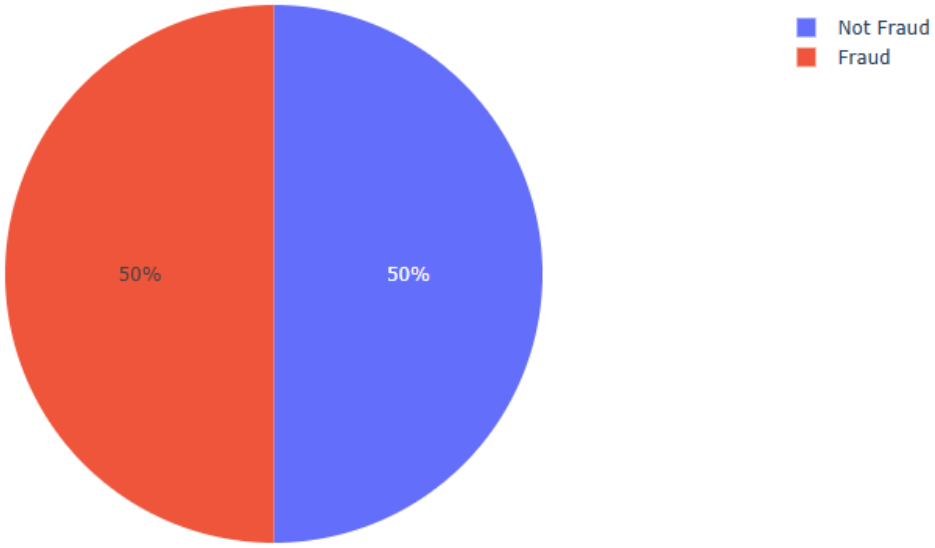

Рисунок 3.26 – Балансування мішені

2.Одне гаряче кодування

Однією з найважливіших особливостей, які ми маємо, є тип, який є категоричним за типом. Оскільки він не має жодної порядкової природи та оскільки класів менше, ми віддаємо перевагу застосуванню одного гарячого кодування [22].

```
paysim_1=pd.concat([paysim_1,pd.get_dummies(paysim_1['type'], prefix='type_')],axis=1)
paysim_1.drop(['type'],axis=1,inplace = True)

paysim_1.head()
```

Рисунок 3.27 – Гаряче кодування

	step	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	1	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0
1	1	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0
2	1	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1
3	1	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1
4	1	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0

Рисунок 3.28 – Гаряче кодування

orig_diff	dest_diff	surge	freq_dest	type__CASH_IN	type__CASH_OUT	type__DEBIT	type__PAYMENT	type__TRANSFER
0	1	0	0	0	0	0	1	0
0	1	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1
0	1	0	0	0	1	0	0	0
0	1	0	0	0	0	0	1	0

Рисунок 3.29 – Гаряче кодування

3. Розділ і стандартизація

У цьому модулі ми створюємо незалежну та залежну функцію, а потім розділяємо їх на дані навчання та тестування, де розмір навчання становить 70%. Пізніше ми збираємо всі числові характеристики та застосовуємо функцію StandardScaler(), яка перетворює розподіл таким чином, щоб середнє значення стало 0, а стандартне відхилення стало 1.

```
paysim_2=paysim_1.copy()
X=paysim_2.drop('isFraud',axis=1)
y=paysim_2['isFraud']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, random_state=111)

#Standardizing the numerical columns
col_names=['amount','oldbalanceOrg','newbalanceOrig','oldbalanceDest','newbalanceDest']
features_train = X_train[col_names]
features_test = X_test[col_names]
scaler = StandardScaler().fit(features_train.values)
features_train = scaler.transform(features_train.values)
features_test = scaler.transform(features_test.values)
X_train[col_names] = features_train
X_test[col_names] =features_test
```

Рисунок 3.30 – Розділ і стандартизація

4. Токенізація

Ідентифікатори клієнтів та ідентифікатори продавців зберігалися в типі об’єкта. Погано застосовувати в ньому одне гаряче кодування, оскільки це може призвести до появи більшої кількості функцій і може виникнути прокляття розмірності. Тому ми застосовуємо тут токенізацію, оскільки вона може створити унікальний ідентифікаційний номер типу int для кожного ідентифікатора клієнта.

```
tokenizer_org = tf.keras.preprocessing.text.Tokenizer()
tokenizer_org.fit_on_texts(X_train['nameOrig'])

tokenizer_dest = tf.keras.preprocessing.text.Tokenizer()
tokenizer_dest.fit_on_texts(X_train['nameDest'])

# Create tokenized customer lists
customers_train_org = tokenizer_org.texts_to_sequences(X_train['nameOrig'])
customers_test_org = tokenizer_org.texts_to_sequences(X_test['nameOrig'])

customers_train_dest = tokenizer_dest.texts_to_sequences(X_train['nameDest'])
customers_test_dest = tokenizer_dest.texts_to_sequences(X_test['nameDest'])


# Pad sequences
X_train['customers_org'] = tf.keras.preprocessing.sequence.pad_sequences(customers_train_org, max
len=1)
X_test['customers_org'] = tf.keras.preprocessing.sequence.pad_sequences(customers_test_org, maxle
n=1)

X_train['customers_dest'] = tf.keras.preprocessing.sequence.pad_sequences(customers_train_dest, m
axlen=1)
X_test['customers_dest'] = tf.keras.preprocessing.sequence.pad_sequences(customers_test_dest, max
len=1)
```

Рисунок 3.31 – Токенізація

5.Видалення непотрібних стовпців

Нам не потрібні ідентифікатори відправника та одержувача, оскільки ми їх токенізували, також нам не потрібен isFlaggedFraud, оскільки це лише результат поточного алгоритму.

```
X_train=X_train.drop(['nameOrig','nameDest','isFlaggedFraud'],axis=1)
X_train = X_train.reset_index(drop=True)

X_test=X_test.drop(['nameOrig','nameDest','isFlaggedFraud'],axis=1)
X_test = X_test.reset_index(drop=True)
```

Рисунок 3.32 – Видалення непотрібних стовпців

6.Створення моделі

Ми успішно обробили дані, і настав час подавати дані в модель. Щоб з'ясувати, яка модель найкраще підходить для наших даних, потрібно багато часу. Тому я використав конвеєр, щоб пропустити наші дані через увесь алгоритм класифікації та вибрати найкращий, який забезпечує максимальну точність.

```
logreg_cv = LogisticRegression(solver='liblinear',random_state=123)
dt_cv=DecisionTreeClassifier(random_state=123)
knn_cv=KNeighborsClassifier()
svc_cv=SVC(kernel='linear',random_state=123)
nb_cv=GaussianNB()
rf_cv=RandomForestClassifier(random_state=123)
cv_dict = {0: 'Logistic Regression', 1: 'Decision Tree',2:'KNN',3:'SVC',4:'Naive Bayes',5:'Random
Forest'}
cv_models=[logreg_cv,dt_cv,knn_cv,svc_cv,nb_cv,rf_cv]


for i,model in enumerate(cv_models):
    print("{} Test Accuracy: {}".format(cv_dict[i],cross_val_score(model, X_train, y_train, cv=1
0, scoring ='accuracy').mean()))
```

Рисунок 3.33 – Створення моделі

```
Logistic Regression Test Accuracy: 0.9655167477812767
Decision Tree Test Accuracy: 0.9749498997995992
KNN Test Accuracy: 0.9877755511022045
SVC Test Accuracy: 0.9756369882622387
Naive Bayes Test Accuracy: 0.9946321213856283
Random Forest Test Accuracy: 0.9931291153736044
```

Рисунок 3.34 – Створення моделі

Ми бачимо, хто виграв приз - це Наївний Байєс. Інші алгоритми також працюють нарівні з NB, особливо Random Forest і KNN. Це виглядає переобладнаним, оскільки точність становить близько 100%, що можна перевірити за допомогою даних тестування. Перед цим давайте налаштуємо гіперпараметр на NB.

Гіперпараметрична настройка - давайте підберемо модель Наївного Байєса, налаштувавши модель за її параметрами. Тут ми налаштуємо var_smoothing, який є обчисленням стабільності, щоб розширити (або згладити) криву та, отже, врахувати більше вибірок, які знаходяться далі від середнього розподілу. У цьому випадку np.logspace повертає числа, рівномірно розподілені в логарифмічній шкалі, починаючи з 0, закінчуючи -9, і генерує 100 зразків [22].

```
param_grid_nb = {
    'var_smoothing': np.logspace(0,-9, num=100)
}

nbModel_grid = GridSearchCV(estimator=GaussianNB(), param_grid=param_grid_nb, verbose=1, cv=10, n
_jobs=-1)
nbModel_grid.fit(X_train, y_train)
print(nbModel_grid.best_estimator_)
```

Рисунок 3.35 – Гіперпараметрична настройка

```
Fitting 10 folds for each of 100 candidates, totalling 1000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  76 tasks      | elapsed:    1.6s
[Parallel(n_jobs=-1)]: Done 376 tasks      | elapsed:    7.5s
[Parallel(n_jobs=-1)]: Done 876 tasks      | elapsed:   17.6s

GaussianNB(var_smoothing=8.111308307896856e-09)

[Parallel(n_jobs=-1)]: Done 1000 out of 1000 | elapsed:   20.1s finished
```

Рисунок 3.36 – Гіперпараметрична настройка

Оцінка моделі - час дослідити правдивість високих цифр шляхом порівняння з даними тестування.

```
#Function for Confusion matrix
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
```

Рисунок 3.37 – Оцінка моделі

```
    thresh = cm.max() / 2.
    for i in range (cm.shape[0]):
        for j in range (cm.shape[1]):
            plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

Рисунок 3.38 – Оцінка моделі

```
#Predict with the selected best parameter
y_pred=nbModel_grid.predict(X_test)

#Plotting confusion matrix
cm = metrics.confusion_matrix(y_test, y_pred)
plot_confusion_matrix(cm, classes=['Not Fraud','Fraud'])
```

Рисунок 3.39 – Оцінка моделі

```
Confusion matrix, without normalization
```

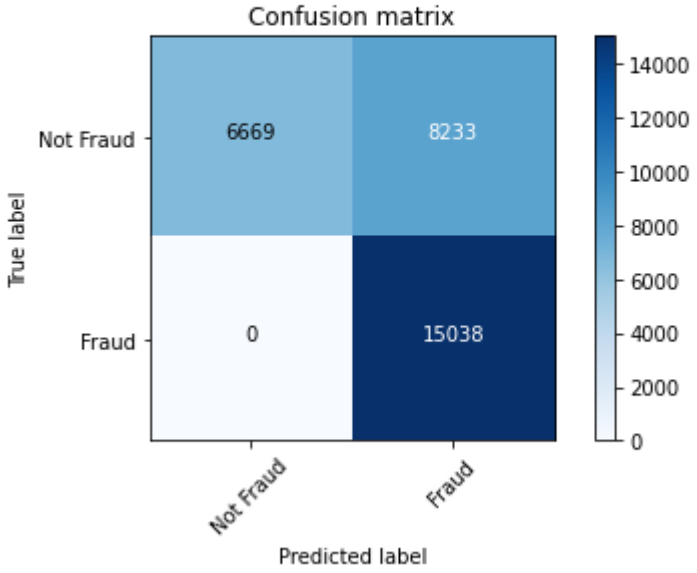

Рисунок 3.40 – Оцінка моделі

Модель виявила хибні спрацьовування, але ніколи не пропустила жодного хибно-негативного результату, що важливіше, ніж FP. Оскільки ми не можемо пропустити шахрайські транзакції, але ми можемо керувати хибнопозитивними результатами, досліджуючи їх [22].

```
#Classification metrics
print(classification_report(y_test, y_pred, target_names=['Not Fraud','Fraud']))
```

Рисунок 3.41 – Оцінка моделі

	precision	recall	f1-score	support
Not Fraud	1.00	0.45	0.62	14902
Fraud	0.65	1.00	0.79	15038
accuracy			0.73	29940
macro avg	0.82	0.72	0.70	29940
weighted avg	0.82	0.73	0.70	29940

Рисунок 3.42 – Оцінка моделі

Коли ми виявили, що наші хибні негативи важливіші, ніж хибні спрацьовування, ми повинні переглянути кількість відкликань, і ми маємо 100% відкликання у пошуку шахрайських транзакцій і 100% точність у пошуку нешахрайських транзакцій, і в середньому наша модель працює більш ніж на 70% точність, що досить добре, і є можливий шанс покращити продуктивність цієї моделі.

З появою цифрових транзакцій можливість відмивання грошей також зросла завдяки використанню технологій. Мільйони слідчих працюють на полі, борючись із шахрайськими операціями. У нинішній галузі ми маємо великий приплив хибних спрацьовувань, і для очищення хибнопозитивних збігів потрібно багато часу. Клієнти в усьому світі, які використовують фінтех-платформи, вимагають блискавичних послуг. Тому нашою метою є автоматизація звернень за допомогою машинного навчання та зменшення хибнопозитивних звернень. Але не ціною того, щоб залишити помилкові негативи. Тому нам потрібно бути більш уважними щодо хибних негативів, коли ми намагаємося зменшити хибно-позитивні результати.

3.4. Висновок до розділу 3

У розділі було обґрунтовано вибір мови програмування Python для розробки системи завдяки її багатій екосистемі бібліотек, простоті використання, кросплатформній сумісності та активній підтримці спільноти. Python довів свою

ефективність у вирішенні завдань машинного навчання та штучного інтелекту, що стало основою для реалізації проекту.

Також було проаналізовано датасет з Kaggle, який забезпечує різноманітні можливості для побудови моделей класифікації та виявлення аномалій. Дослідження структури даних і виділення ключових ознак дали змогу створити ефективні механізми для автоматизованого аналізу транзакцій.

На етапі реалізації було здійснено попередню обробку даних, включаючи балансування міток, гаряче кодування категоричних змінних, стандартизацію числових даних та створення додаткових функцій. У підсумку було реалізовано алгоритми класифікації, зокрема Наївний Байєс, Random Forest та KNN, з оптимізацією їх гіперпараметрів.

Оцінка моделей продемонструвала високу точність виявлення шахрайських транзакцій, зокрема більше 70% відгуку, що свідчить про надійність розробленої системи. Таким чином, створена система забезпечує ефективне виявлення аномалій у транзакціях, що є важливим кроком у боротьбі з шахрайством у фінансовій сфері.

ВИСНОВКИ

У магістерській роботі проведено комплексне дослідження та розробку системи для автоматизованого виявлення аномальних транзакцій у фінансовій сфері. В якості прикладу для побудови системи було використано підхід, представлений у роботі на платформі Kaggle, що демонструє можливості виявлення шахрайства за допомогою машинного навчання.

Аналіз існуючих рішень і використання набору даних, який містить інформацію про фінансові транзакції, дозволили ефективно адаптувати методи машинного навчання для виявлення аномалій.

Було застосовано алгоритми класифікації, такі як Random Forest і XGBoost, які продемонстрували високу точність у класифікації транзакцій на шахрайські та нормальні. Робота також акцентувала увагу на обробці нерівномірних даних, очищенні набору та візуалізації результатів.

Розроблена система базується на принципах гнучкості, адаптивності й ефективності. Вона здатна виявляти складні типи шахрайства завдяки автоматичному аналізу поведінкових патернів та оновленню моделей на основі нових даних. Така система є важливим інструментом для компанії, оскільки дозволяє не лише знижувати фінансові ризики, але й підвищувати довіру клієнтів за рахунок забезпечення більшої безпеки.

Дослідження підтвердило, що використання сучасних підходів до машинного навчання для аналізу фінансових транзакцій може суттєво покращити точність і швидкість виявлення аномалій, забезпечуючи тим самим надійний захист від шахрайства.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційна сторінка кампанії . [Електронний ресурс]. — Режим доступу: https://softsvit.com/ (дата звернення: 27.11.2024).

2. Офіційна сторінка кампанії . [Електронний ресурс]. — Режим доступу: https://softum.net/ (дата звернення: 27.11.2024).

3. Kaggle. Dataset: Fraudulent E-commerce Transactions. [Електронний ресурс]. — Режим доступу: https://www.kaggle.com/datasets/shriyashjagtap/fraudulent-e-commerce-transactions. (дата звернення: 27.11.2024).

4. Kaggle. Notebook: Transaction Fraud Detection by Ben Roshan. [Електронний ресурс]. — Режим доступу: https://www.kaggle.com/code/benroshan/transaction-fraud-detection/notebook. (дата звернення: 27.11.2024).

5. Виявлення шахрайських транзакцій з допомогою методів машинного навчання [Електронний ресурс]. — Режим доступу: https://elartu.tntu.edu.ua/bitstream/lib/43250/2/%D0%9A%D0%A0%20%D0%BC%D0%B0%D0%B3%D1%96%D1%81%D1%82%D1%80%D0%B0_%20%D0%91%D0%B5%D0%B7%D1%80%D1%83%D0%BA%D0%BE%D0%B2_2023.pdf(дата звернення: 27.11.2024).

6. ДОСЛІДЖЕННЯ МЕТОДІВ ВИЯВЛЕННЯ АНОМАЛІЙ У ФІНАНСОВИХ ТРАНЗАКЦІЯХ [Електронний ресурс]. — Режим доступу: https://eprints.kname.edu.ua/66504/1/%D0%97%D1%96%D0%BD%D0%B5%D0%BD%D0%BA%D0%BE%20%D0%A8%D1%82%D1%83%D1%87%D0%BD%D0%B8%D0%B9%20%D1%96%D0%BD%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D1%82.pdf(дата звернення: 27.11.2024).

7. СУЧАСНІ МЕТОДИ ВИЯВЛЕННЯ АНОМАЛІЙ В СИСТЕМАХ ВИЯВЛЕННЯ ВТОРГНЕНЬ [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/%D0%92%D0%B8%D1%8F%D0%B2%D0%BB%D0%

B5%D0%BD%D0%BD%D1%8F_%D0%B0%D0%BD%D0%BE%D0%BC%D0%B0%D0%BB%D1%96%D0%B9(дата звернення: 29.11.2024).

8. Виявлення аномалій [Електронний ресурс]. — Режим доступу: https://science.lpnu.ua/sites/default/files/journal-paper/2017/nov/6726/16-98-104.pdf(дата звернення: 29.11.2024).

9. Python (programming language) [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Python_(programming_language) (дата звернення: 29.11.2024).

10. Python Software Foundation. Python Programming Language Official Documentation. URL: https://www.python.org/doc/ (дата звернення: 27.11.2024).

11. Chollet F. Deep Learning with Python. 2nd Edition. Manning Publications, 2021. (дата звернення: 29.11.2024).

12. McKinney W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. O'Reilly Media, 2017. (дата звернення: 29.11.2024).

13. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, 2019. (дата звернення: 29.11.2024).

14. Abadi M. et al. TensorFlow: A System for Large-Scale Machine Learning. Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI '16). 2016; pp. 265–283. (дата звернення: 27.11.2024).

15. Pedregosa F., et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011;12:2825-2830. URL: https://scikit-learn.org. (дата звернення: 27.11.2024).

16. Kaggle. Fraud Detection Dataset. URL: https://www.kaggle.com/code/benroshan/transaction-fraud-detection/notebook (дата звернення: 27.11.2024).

17. Böhme R., Holz T. The Effectiveness of Bayesian Methods for Detecting Fraud in Online Transactions. Financial Cryptography and Data Security. Springer, 2012. (дата звернення: 27.11.2024).

18. Zimek A., Schubert E. Outlier Detection. Encyclopedia of Machine Learning and Data Mining. Springer, 2017. (дата звернення: 27.11.2024).

19. Датасет із Kaggle - [Електронний ресурс]. — Режим доступу: https://www.kaggle.com/code/benroshan/transaction-fraud-detection/notebook (дата звернення: 30.11.2024).

20. Colab (Google Colaboratory) [Електронний ресурс]. — Режим доступу: https://colab.research.google.com/#scrollTo=eLWOF-CZkJs0 (дата звернення: 30.11.2024).

21. Датасет із Kaggle - [Електронний ресурс]. — Режим доступу: https://www.kaggle.com/code/jainilcoder/fraud-detection-99-97-accuracy/notebook (дата звернення: 30.11.2024).

22. E-Commerce Fraud Detection Based on Machine Learning Techniques: Systematic Literature Review [Електронний ресурс]. — Режим доступу: https://www.sciopen.com/article/10.26599/BDMA.2023.9020023 (дата звернення: 30.11.2024).

23. ДСТУ ISO/IEC TR 15504. Інформаційні технології. Оцінювання процесів життєвого циклу програмних засобів. 315 с. (дата звернення: 30.11.2024).

24. Hinton G.E., Salakhutdinov R.R. "Reducing the Dimensionality of Data with Neural Networks." Science, 2006. (дата звернення: 30.11.2024).

25. Sokolova M., Lapalme G. "A Systematic Analysis of Performance Measures for Classification Tasks." Information Processing and Management, 2009. (дата звернення: 30.11.2024).

26. Han J., Kamber M., Pei J. Data Mining: Concepts and Techniques. Elsevier, 2011. (дата звернення: 30.11.2024).

27. Fawcett T. "An Introduction to ROC Analysis." Pattern Recognition Letters, 2006. (дата звернення: 30.11.2024).

28. LeCun Y., Bengio Y., Hinton G. "Deep Learning." Nature, 2015. (дата звернення: 30.11.2024).

29. Scikit-learn documentation.. [Електронний ресурс]. — Режим доступу: https://scikit-learn.org (дата звернення: 27.11.2024).

30. Aggarwal, C.C. Outlier Analysis. Springer, 2017. (дата звернення: 27.11.2024).

31. Liu, F.T., Ting, K.M., Zhou, Z.-H. "Isolation Forest." Proceedings of the IEEE International Conference on Data Mining (ICDM), 2008. (дата звернення: 27.11.2024).

32. Hinton, G.E., Salakhutdinov, R.R. "Reducing the Dimensionality of Data with Neural Networks." Science, 2006. (дата звернення: 27.11.2024).

33. Ng, A.Y., Jordan, M.I. "On Discriminative vs. Generative Classifiers." Neural Information Processing Systems (NIPS), 2002. (дата звернення: 27.11.2024).

34. Han, J., Kamber, M., Pei, J. Data Mining: Concepts and Techniques. Elsevier, 2011. (дата звернення: 27.11.2024).

35. Chandola, V., Banerjee, A., Kumar, V. "Anomaly Detection: A Survey." ACM Computing Surveys, 2009. (дата звернення: 27.11.2024).

36. Scikit-learn documentation. [Електронний ресурс]. — Режим доступу: https://scikit-learn.org. (дата звернення: 27.11.2024).

37. TensorFlow documentation. https://www.tensorflow.org. [Електронний ресурс]. — Режим доступу: (дата звернення: 30.11.2024).

38. "Outlier Detection Techniques." Analytics Vidhya,. [Електронний ресурс]. — Режим доступу: https://analyticsvidhya.com (дата звернення: 30.11.2024).

39. Prediction-of-Fraudulent-in-E-Commerce-Transactions [Електронний ресурс]. — Режим доступу: https://github.com/Rameshkumar477/Prediction-of-Fraudulent-in-E-Commerce-Transactions/blob/main/Fraudulent%20E-Commerce%20Transactions.ipynb (дата звернення: 30.11.2024).

40. Fraudulent E-Commerce Transactions [Електронний ресурс]. — Режим доступу: https://github.com/nileshely/Fraudulent-E-Commerce-Transactions (дата звернення: 30.11.2024).

41. Створення діаграм [Електронний ресурс]. — Режим доступу: https://app.diagrams.net/ (дата звернення: 30.11.2024).

42. McKinney, W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. O'Reilly Media, 2017. (дата звернення: 30.11.2024).

43. VanderPlas, J. Python Data Science Handbook: Essential Tools for Working with Data. O'Reilly Media. (дата звернення: 30.11.2024).

ДОДАТКИ

Додаток А

```
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
from plotly.offline import plot, iplot, init_notebook_mode
init_notebook_mode(connected=True)
%matplotlib inline

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

import tensorflow as tf
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline

from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix


import warnings
warnings.filterwarnings("ignore")

paysim=pd.read_csv('PS_20174392719_1491204439457_log.csv')
paysim.head()
paysim.info()

paysim_pivot1=pd.pivot_table(paysim,index=["type"],
                             values=['amount','isFraud','isFlaggedFraud'],
                             aggfunc=[np.sum,np.std], margins=True)
```

```
cm = sns.light_palette("green", as_cmap=True)
paysim_pivot1.style.background_gradient(cmap=cm)

paysim_pivot2=pd.pivot_table(paysim,index=["type"],
                             values=['amount','oldbalanceOrg','newbalanceOrig'],
                             aggfunc=[np.sum], margins=True)

paysim_pivot2.style\
    .format('{:.2f}')\
    .bar(align='mid',color=['darkred'])\
    .set_properties(padding='5px',border='3px solid white',width='200px')

paysim_pivot3=pd.pivot_table(paysim,index=["type"],
                             values=['amount','oldbalanceDest','newbalanceDest'],
                             aggfunc=[np.sum], margins=True)

paysim_pivot3.style\
    .format('{:.2f}')\
    .bar(align='mid',color=['darkblue'])\
    .set_properties(padding='5px',border='3px solid white',width='200px')

paysim=pd.read_csv('PS_20174392719_1491204439457_log.csv',nrows=50000)

fig = px.box(paysim, y="amount")
fig.show()


def balance_diff(data):
    '''balance_diff checks whether the money debited from sender has exactly credited to the receiver
       then it creates a new column which indicates 1 when there is a deviation else 0'''
    # Sender's balance
    orig_change = data['newbalanceOrig'] - data['oldbalanceOrg']
    orig_change = orig_change.astype(int)
    for i in orig_change:
        if i < 0:
            data['orig_txn_diff'] = round(data['amount'] + orig_change, 2)
        else:
            data['orig_txn_diff'] = round(data['amount'] - orig_change, 2)
    data['orig_txn_diff'] = data['orig_txn_diff'].astype(int)
    data['orig_diff'] = [1 if n != 0 else 0 for n in data['orig_txn_diff']]

    # Receiver's balance
    dest_change = data['newbalanceDest'] - data['oldbalanceDest']
    dest_change = dest_change.astype(int)
    for i in dest_change:
        if i < 0:
            data['dest_txn_diff'] = round(data['amount'] + dest_change, 2)
        else:
            data['dest_txn_diff'] = round(data['amount'] - dest_change, 2)
    data['dest_txn_diff'] = data['dest_txn_diff'].astype(int)
    data['dest_diff'] = [1 if n != 0 else 0 for n in data['dest_txn_diff']]
```

```
    data.drop(['orig_txn_diff', 'dest_txn_diff'], axis=1, inplace=True)


# Surge indicator
def surge_indicator(data):
    '''Creates a new column which has 1 if the transaction amount is greater than the threshold
    else it will be 0'''
    data['surge'] = [1 if n > 450000 else 0 for n in data['amount']]


# Frequency indicator
def frequency_receiver(data):
    '''Creates a new column which has 1 if the receiver receives money from many individuals
    else it will be 0'''
    data['freq_Dest'] = data['nameDest'].map(data['nameDest'].value_counts())
    data['freq_dest'] = [1 if n > 20 else 0 for n in data['freq_Dest']]

    data.drop(['freq_Dest'], axis=1, inplace=True)


# Tracking the receiver as merchant or not
def merchant(data):
    '''We also have customer ids which starts with M in Receiver name, it indicates merchant
    this function will flag if there is a merchant in receiver end '''
    values = ['M']
    conditions = list(map(data['nameDest'].str.contains, values))
    data['merchant'] = np.select(conditions, '1', '0')

balance_diff(paysim)

paysim['orig_diff'].value_counts()
paysim['dest_diff'].value_counts()

surge_indicator(paysim)
paysim['surge'].value_counts()

frequency_receiver(paysim)
paysim['freq_dest'].value_counts()

paysim_1=paysim.copy()

#Checking for balance in target
fig = go.Figure(data=[go.Pie(labels=['Not Fraud','Fraud'],
values=paysim_1['isFraud'].value_counts())])
fig.show()

#Getting the max size
max_size = paysim_1['isFraud'].value_counts().max()

#Balancing the target label
lst = [paysim_1]
for class_index, group in paysim_1.groupby('isFraud'):
```

```
    lst.append(group.sample(max_size-len(group), replace=True))
paysim_1 = pd.concat(lst)
#Checking the balanced target
fig = go.Figure(data=[go.Pie(labels=['Not Fraud','Fraud'],
values=paysim_1['isFraud'].value_counts())])
fig.show()

paysim_1=pd.concat([paysim_1,pd.get_dummies(paysim_1['type'], prefix='type_')],axis=1)
paysim_1.drop(['type'],axis=1,inplace = True)

paysim_1.head()

paysim_2=paysim_1.copy()
X=paysim_2.drop('isFraud',axis=1)
y=paysim_2['isFraud']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, random_state=111)

#Standardizing the numerical columns
col_names=['amount','oldbalanceOrg','newbalanceOrig','oldbalanceDest','newbalanceDest']
features_train = X_train[col_names]
features_test = X_test[col_names]
scaler = StandardScaler().fit(features_train.values)
features_train = scaler.transform(features_train.values)
features_test = scaler.transform(features_test.values)
X_train[col_names] = features_train
X_test[col_names] =features_test

tokenizer_org = tf.keras.preprocessing.text.Tokenizer()
tokenizer_org.fit_on_texts(X_train['nameOrig'])

tokenizer_dest = tf.keras.preprocessing.text.Tokenizer()
tokenizer_dest.fit_on_texts(X_train['nameDest'])

# Create tokenized customer lists
customers_train_org = tokenizer_org.texts_to_sequences(X_train['nameOrig'])
customers_test_org = tokenizer_org.texts_to_sequences(X_test['nameOrig'])

customers_train_dest = tokenizer_dest.texts_to_sequences(X_train['nameDest'])
customers_test_dest = tokenizer_dest.texts_to_sequences(X_test['nameDest'])


# Pad sequences
X_train['customers_org'] = tf.keras.preprocessing.sequence.pad_sequences(customers_train_org,
maxlen=1)
X_test['customers_org'] = tf.keras.preprocessing.sequence.pad_sequences(customers_test_org,
maxlen=1)

X_train['customers_dest'] = tf.keras.preprocessing.sequence.pad_sequences(customers_train_dest,
maxlen=1)
X_test['customers_dest'] = tf.keras.preprocessing.sequence.pad_sequences(customers_test_dest,
```

```
maxlen=1)

X_train=X_train.drop(['nameOrig','nameDest','isFlaggedFraud'],axis=1)
X_train = X_train.reset_index(drop=True)

X_test=X_test.drop(['nameOrig','nameDest','isFlaggedFraud'],axis=1)
X_test = X_test.reset_index(drop=True)

logreg_cv = LogisticRegression(solver='liblinear',random_state=123)
dt_cv=DecisionTreeClassifier(random_state=123)
knn_cv=KNeighborsClassifier()
svc_cv=SVC(kernel='linear',random_state=123)
nb_cv=GaussianNB()
rf_cv=RandomForestClassifier(random_state=123)
cv_dict = {0: 'Logistic Regression', 1: 'Decision Tree',2:'KNN',3:'SVC',4:'Naive Bayes',5:'Random
Forest'}
cv_models=[logreg_cv,dt_cv,knn_cv,svc_cv,nb_cv,rf_cv]


for i,model in enumerate(cv_models):
    print("{} Test Accuracy: {}".format(cv_dict[i],cross_val_score(model, X_train, y_train, cv=10,
scoring ='accuracy').mean()))

    param_grid_nb = {
        'var_smoothing': np.logspace(0, -9, num=100)
    }

    nbModel_grid = GridSearchCV(estimator=GaussianNB(), param_grid=param_grid_nb, verbose=1,
cv=10, n_jobs=-1)
    nbModel_grid.fit(X_train, y_train)
    print(nbModel_grid.best_estimator_)


def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
```

```
    else:
        print('Confusion matrix, without normalization')

    thresh = cm.max() / 2.
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            plt.text(j, i, cm[i, j],
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')


# Predict with the selected best parameter
y_pred = nbModel_grid.predict(X_test)

# Plotting confusion matrix
cm = metrics.confusion_matrix(y_test, y_pred)
plot_confusion_matrix(cm, classes=['Not Fraud', 'Fraud'])

print(classification_report(y_test, y_pred, target_names=['Not Fraud','Fraud']))
```

Додаток Б.

```
#Basic libraries
import pandas as pd
import numpy as np

#Visualization libraries
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
from plotly.offline import plot, iplot, init_notebook_mode
init_notebook_mode(connected=True)
%matplotlib inline

#preprocessing libraries
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

Рисунок Б.1. – Підключення бібліотек

```
#ML libraries
import tensorflow as tf
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline

#Metrics Libraries
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix



#Misc libraries
import warnings
warnings.filterwarnings("ignore")
```

Рисунок Б.2. – Підключення бібліотек

```
#Reading the data
paysim=pd.read_csv('PS_20174392719_1491204439457_log.csv')

#Looking at the data
paysim.head()
```

Рисунок Б.3. – Підключення до набору даних

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0

Рисунок.4. – Підключення до набору даних

```
paysim.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 11 columns):
 #   Column          Dtype
---  ------          -----
 0   step            int64
 1   type            object
 2   amount          float64
 3   nameOrig        object
 4   oldbalanceOrg   float64
 5   newbalanceOrig  float64
 6   nameDest        object
 7   oldbalanceDest  float64
 8   newbalanceDest  float64
 9   isFraud         int64
 10  isFlaggedFraud  int64
dtypes: float64(5), int64(3), object(3)
memory usage: 534.0+ MB
```

Рисунок Б.5. – Опис набору даних

	sum			std		
	amount	isFlaggedFraud	isFraud	amount	isFlaggedFraud	isFraud
type						
CASH_IN	236367391912.459991	0	0	126508.255272	0.000000	0.000000
CASH_OUT	394412995224.489990	0	4116	175329.744483	0.000000	0.042851
DEBIT	227199221.280000	0	0	13318.535518	0.000000	0.000000
PAYMENT	28093371138.369999	0	0	12556.450186	0.000000	0.000000
TRANSFER	485291987263.169983	16	4097	1879573.528908	0.005479	0.087344
All	1144392944759.770020	16	8213	603858.184009	0.001586	0.035905

Рисунок Б.6. – Аналіз зведеної таблиці

```
paysim_pivot2=pd.pivot_table(paysim,index=["type"],
                             values=['amount','oldbalanceOrg','newbalanceOrig'],
                             aggfunc=[np.sum], margins=True)

#Adding style
paysim_pivot2.style\
    .format('{:.2f}')\
    .bar(align='mid',color=['darkred'])\
    .set_properties(padding='5px',border='3px solid white',width='200px')
```

Рисунок Б.7– Аналіз зведеної таблиці

	sum		
	amount	newbalanceOrig	oldbalanceOrg
type			
CASH_IN	236367391912.46	5260438481752.40	5024078139747.30
CASH_OUT	394412995224.49	39098506249.34	102978263227.81
DEBIT	227199221.28	2699777564.12	2844196471.80
PAYMENT	28093371138.37	133043913105.10	146768163438.79
TRANSFER	485291987263.16	5482651300.55	29012552760.76
All	1144392944759.77	5440763329971.49	5305681315646.41

Рисунок Б.8– Аналіз зведеної таблиці

```
paysim_pivot3=pd.pivot_table(paysim,index=["type"],
                             values=['amount','oldbalanceDest','newbalanceDest'],
                             aggfunc=[np.sum], margins=True)

#Adding style
paysim_pivot3.style\
    .format('{:.2f}')\
    .bar(align='mid',color=['darkblue'])\
    .set_properties(padding='5px',border='3px solid white',width='200px')
```

Рисунок Б.9– Аналіз зведеної таблиці

	sum		
	amount	newbalanceDest	oldbalanceDest
type			
CASH_IN	236367391912.46	2052897091879.15	2221949365238.98
CASH_OUT	394412995224.49	37843420780615	3351233273577 78
DEBIT	227199221.28	62686759687.09	61863601275.22
PAYMENT	28093371138.37	0.00	0.00
TRANSFER	485291987263.16	1894260653500.26	1368300197339.44
All	1144392944759.77	7794186583127.56	7003346437431.25

Рисунок Б.10– Аналіз зведеної таблиці

```
paysim=pd.read_csv('PS_20174392719_1491204439457_log.csv',nrows=50000)

#Distribution of Amount
fig = px.box(paysim, y="amount")
fig.show()
```

Рисунок Б.11– Розподіл суми

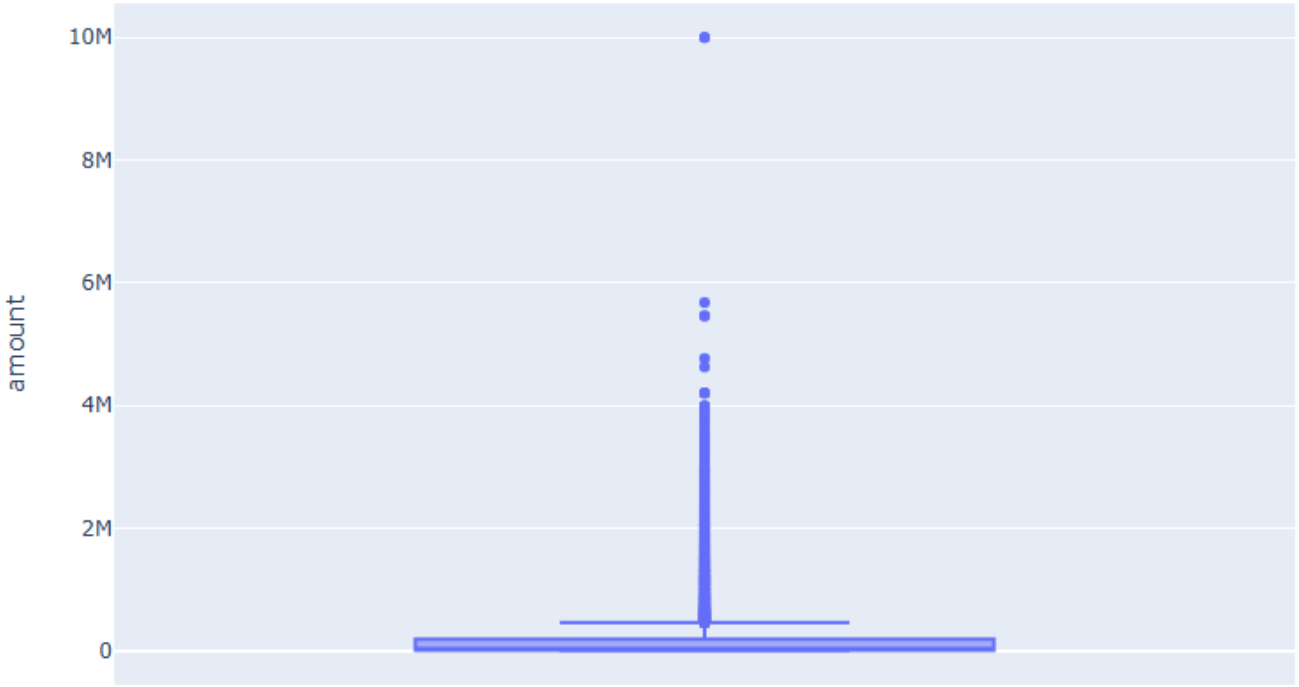

Рисунок Б.12– Розподіл суми

```
def balance_diff(data):
    '''balance_diff checks whether the money debited from sender has exactly credited to the receiv
er
       then it creates a new column which indicates 1 when there is a deviation else 0'''
    #Sender's balance
    orig_change=data['newbalanceOrig']-data['oldbalanceOrg']
    orig_change=orig_change.astype(int)
    for i in orig_change:
        if i<0:
            data['orig_txn_diff']=round(data['amount']+orig_change,2)
        else:
            data['orig_txn_diff']=round(data['amount']-orig_change,2)
    data['orig_txn_diff']=data['orig_txn_diff'].astype(int)
    data['orig_diff'] = [1 if n !=0 else 0 for n in data['orig_txn_diff']]

    #Receiver's balance
    dest_change=data['newbalanceDest']-data['oldbalanceDest']
    dest_change=dest_change.astype(int)
    for i in dest_change:
        if i<0:
            data['dest_txn_diff']=round(data['amount']+dest_change,2)
        else:
            data['dest_txn_diff']=round(data['amount']-dest_change,2)
    data['dest_txn_diff']=data['dest_txn_diff'].astype(int)
    data['dest_diff'] = [1 if n !=0 else 0 for n in data['dest_txn_diff']]
```

Рисунок Б.13– Розробка функцій

```
    data.drop(['orig_txn_diff','dest_txn_diff'],axis=1,inplace = True)

#Surge indicator
def surge_indicator(data):
    '''Creates a new column which has 1 if the transaction amount is greater than the threshold
    else it will be 0'''
    data['surge']=[1 if n>450000 else 0 for n in data['amount']]

#Frequency indicator
def frequency_receiver(data):
    '''Creates a new column which has 1 if the receiver receives money from many individuals
    else it will be 0'''
    data['freq_Dest']=data['nameDest'].map(data['nameDest'].value_counts())
    data['freq_dest']=[1 if n>20 else 0 for n in data['freq_Dest']]

    data.drop(['freq_Dest'],axis=1,inplace = True)

#Tracking the receiver as merchant or not
def merchant(data):
    '''We also have customer ids which starts with M in Receiver name, it indicates merchant
    this function will flag if there is a merchant in receiver end '''
    values = ['M']
    conditions = list(map(data['nameDest'].str.contains, values))
    data['merchant'] = np.select(conditions, '1', '0')
```

Рисунок Б.14– Розробка функцій

```
#Applying balance_diff function
balance_diff(paysim)

paysim['orig_diff'].value_counts()
paysim['dest_diff'].value_counts()
```

Рисунок Б.15– Розробка функцій

```
1    44994
0     5006
Name: dest_diff, dtype: int64
```

Рисунок Б.16– Розробка функцій

```
#Applying surge_indicator function
surge_indicator(paysim)
paysim['surge'].value_counts()
```

Рисунок Б.17– Розробка функцій

```
0    46392
1     3608
Name: surge, dtype: int64
```

Рисунок Б.18– Розробка функцій

```
#Applying frequency_receiver function
frequency_receiver(paysim)
paysim['freq_dest'].value_counts()
```

Рисунок Б.19– Розробка функцій

```
0    46295
1     3705
Name: freq_dest, dtype: int64
```

Рисунок Б.20– Розробка функцій

```
paysim_1=paysim.copy()

#Checking for balance in target
fig = go.Figure(data=[go.Pie(labels=['Not Fraud','Fraud'], values=paysim_1['isFraud'].value_count
s())])
fig.show()
```

Рисунок Б.21– Балансування мішені

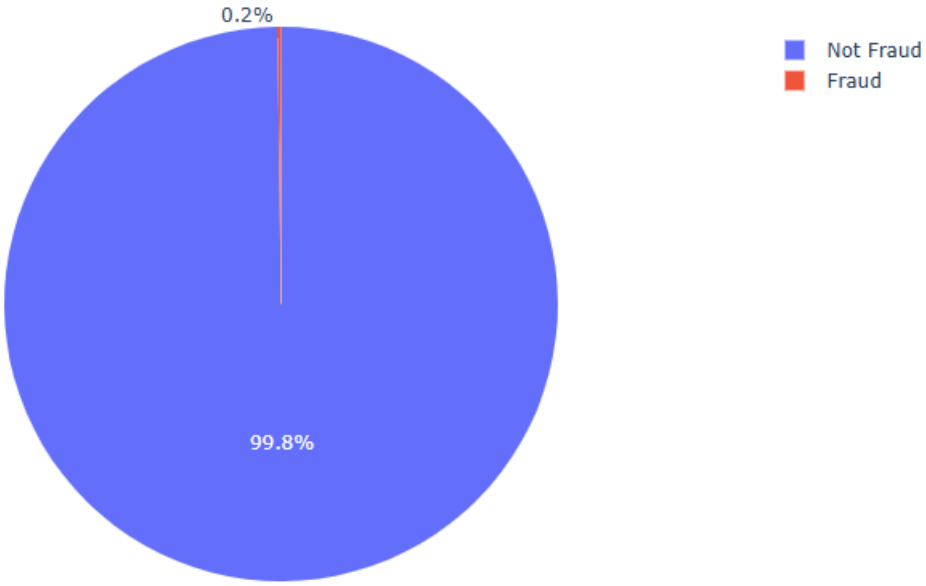

Рисунок Б.22– Балансування мішені

```
max_size = paysim_1['isFraud'].value_counts().max()

#Balancing the target label
lst = [paysim_1]
for class_index, group in paysim_1.groupby('isFraud'):
    lst.append(group.sample(max_size-len(group), replace=True))
paysim_1 = pd.concat(lst)
```

Рисунок Б.23– Балансування мішені

```
fig = go.Figure(data=[go.Pie(labels=['Not Fraud','Fraud'], values=paysim_1['isFraud'].value_count
s())])
fig.show()
```

Рисунок Б.24– Балансування мішені

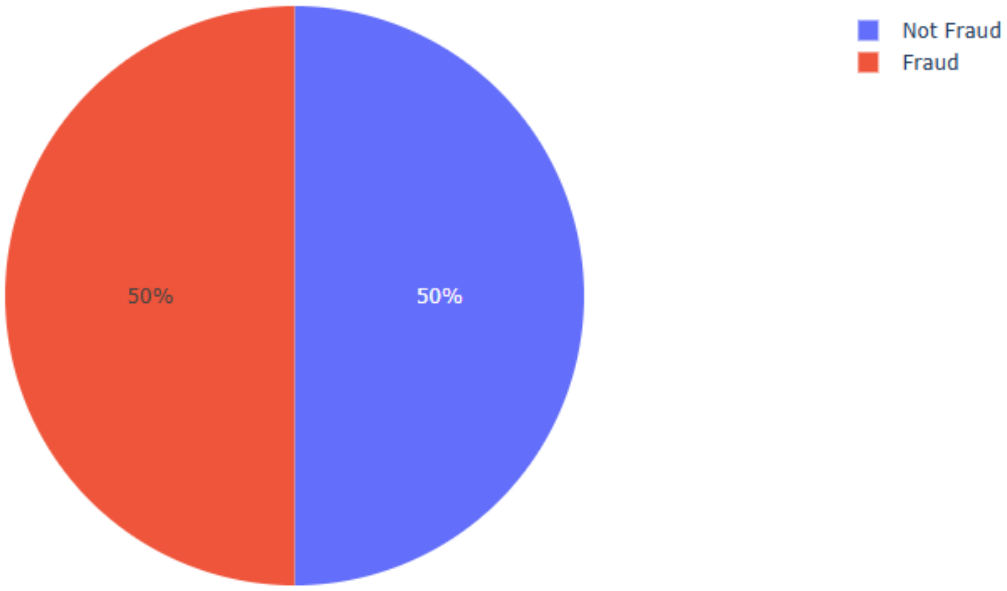

Рисунок Б.25– Балансування мішені

```
paysim_1=pd.concat([paysim_1,pd.get_dummies(paysim_1['type'], prefix='type_')],axis=1)
paysim_1.drop(['type'],axis=1,inplace = True)

paysim_1.head()
```

Рисунок Б.26– Гаряче кодування

	step	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	1	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0
1	1	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0
2	1	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1
3	1	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1
4	1	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0

Рисунок Б.27– Гаряче кодування

orig_diff	dest_diff	surge	freq_dest	type__CASH_IN	type__CASH_OUT	type__DEBIT	type__PAYMENT	type__TRANSFER
0	1	0	0	0	0	0	1	0
0	1	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1
0	1	0	0	0	1	0	0	0
0	1	0	0	0	0	0	1	0

```
paysim_2=paysim_1.copy()
X=paysim_2.drop('isFraud',axis=1)
y=paysim_2['isFraud']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, random_state=111)

#Standardizing the numerical columns
col_names=['amount','oldbalanceOrg','newbalanceOrig','oldbalanceDest','newbalanceDest']
features_train = X_train[col_names]
features_test = X_test[col_names]
scaler = StandardScaler().fit(features_train.values)
features_train = scaler.transform(features_train.values)
features_test = scaler.transform(features_test.values)
X_train[col_names] = features_train
X_test[col_names] =features_test
```

Рисунок Б.28– Гаряче кодування

```
tokenizer_org = tf.keras.preprocessing.text.Tokenizer()
tokenizer_org.fit_on_texts(X_train['nameOrig'])

tokenizer_dest = tf.keras.preprocessing.text.Tokenizer()
tokenizer_dest.fit_on_texts(X_train['nameDest'])

# Create tokenized customer lists
customers_train_org = tokenizer_org.texts_to_sequences(X_train['nameOrig'])
customers_test_org = tokenizer_org.texts_to_sequences(X_test['nameOrig'])

customers_train_dest = tokenizer_dest.texts_to_sequences(X_train['nameDest'])
customers_test_dest = tokenizer_dest.texts_to_sequences(X_test['nameDest'])


# Pad sequences
X_train['customers_org'] = tf.keras.preprocessing.sequence.pad_sequences(customers_train_org, max
len=1)
X_test['customers_org'] = tf.keras.preprocessing.sequence.pad_sequences(customers_test_org, maxle
n=1)

X_train['customers_dest'] = tf.keras.preprocessing.sequence.pad_sequences(customers_train_dest, m
axlen=1)
X_test['customers_dest'] = tf.keras.preprocessing.sequence.pad_sequences(customers_test_dest, max
len=1)
```

Рисунок Б.29– Токенізація

```
X_train=X_train.drop(['nameOrig','nameDest','isFlaggedFraud'],axis=1)
X_train = X_train.reset_index(drop=True)

X_test=X_test.drop(['nameOrig','nameDest','isFlaggedFraud'],axis=1)
X_test = X_test.reset_index(drop=True)
```

Рисунок Б.30– Видалення непотрібних стовпців

```
logreg_cv = LogisticRegression(solver='liblinear',random_state=123)
dt_cv=DecisionTreeClassifier(random_state=123)
knn_cv=KNeighborsClassifier()
svc_cv=SVC(kernel='linear',random_state=123)
nb_cv=GaussianNB()
rf_cv=RandomForestClassifier(random_state=123)
cv_dict = {0: 'Logistic Regression', 1: 'Decision Tree',2:'KNN',3:'SVC',4:'Naive Bayes',5:'Random
Forest'}
cv_models=[logreg_cv,dt_cv,knn_cv,svc_cv,nb_cv,rf_cv]


for i,model in enumerate(cv_models):
    print("{} Test Accuracy: {}".format(cv_dict[i],cross_val_score(model, X_train, y_train, cv=1
0, scoring ='accuracy').mean()))
```

Рисунок Б.31– Створення моделі

```
Logistic Regression Test Accuracy: 0.9655167477812767
Decision Tree Test Accuracy: 0.9749498997995992
KNN Test Accuracy: 0.9877755511022045
SVC Test Accuracy: 0.9756369882622387
Naive Bayes Test Accuracy: 0.9946321213856283
Random Forest Test Accuracy: 0.9931291153736044
```

Рисунок Б.32– Створення моделі

```
param_grid_nb = {
    'var_smoothing': np.logspace(0,-9, num=100)
}

nbModel_grid = GridSearchCV(estimator=GaussianNB(), param_grid=param_grid_nb, verbose=1, cv=10, n
_jobs=-1)
nbModel_grid.fit(X_train, y_train)
print(nbModel_grid.best_estimator_)
```

Рисунок Б.33– Гіперпараметрична настройка

```
#Function for Confusion matrix
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
```

Рисунок Б.34– Оцінка моделі

```
    thresh = cm.max() / 2.
    for i in range (cm.shape[0]):
        for j in range (cm.shape[1]):
            plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

Рисунок Б.35– Оцінка моделі

```
#Predict with the selected best parameter
y_pred=nbModel_grid.predict(X_test)

#Plotting confusion matrix
cm = metrics.confusion_matrix(y_test, y_pred)
plot_confusion_matrix(cm, classes=['Not Fraud','Fraud'])
```

Рисунок Б.36– Оцінка моделі

```
Confusion matrix, without normalization
```

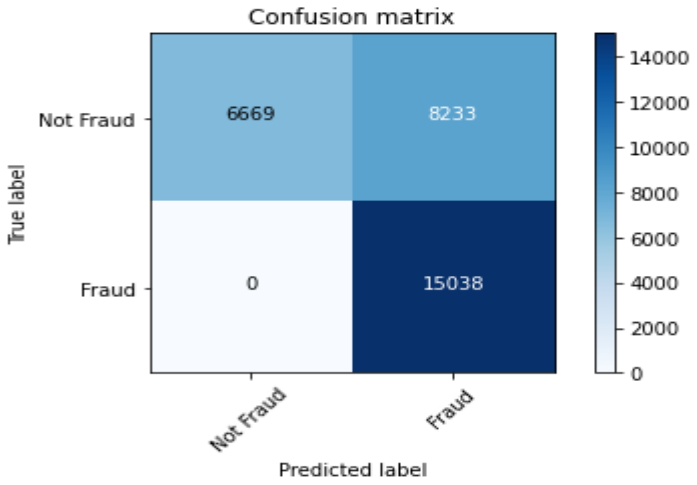

Рисунок Б.37– Оцінка моделі

```
#Classification metrics
print(classification_report(y_test, y_pred, target_names=['Not Fraud','Fraud']))
```

Рисунок Б.38– Оцінка моделі

```
                precision    recall  f1-score   support

     Not Fraud       1.00      0.45      0.62     14902
         Fraud       0.65      1.00      0.79     15038

      accuracy                           0.73     29940
     macro avg       0.82      0.72      0.70     29940
  weighted avg       0.82      0.73      0.70     29940
```

Рисунок Б.39– Оцінка моделі

Додаток В.

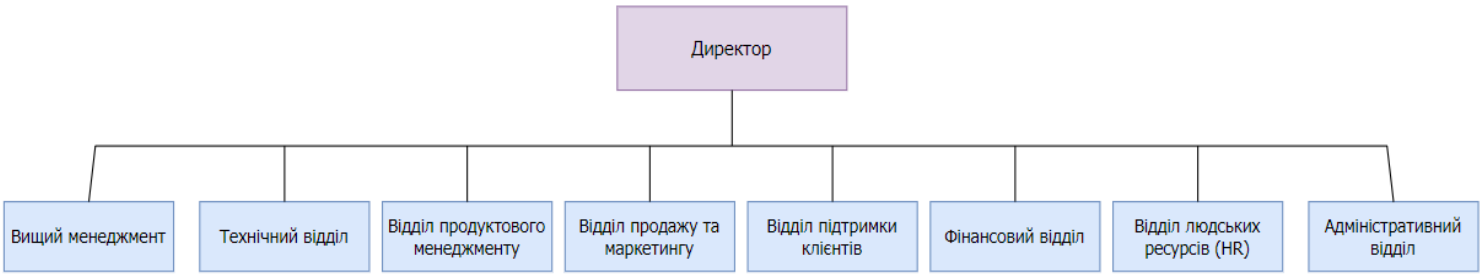

Рисунок В.1 – Організаційна структура компанії Softum

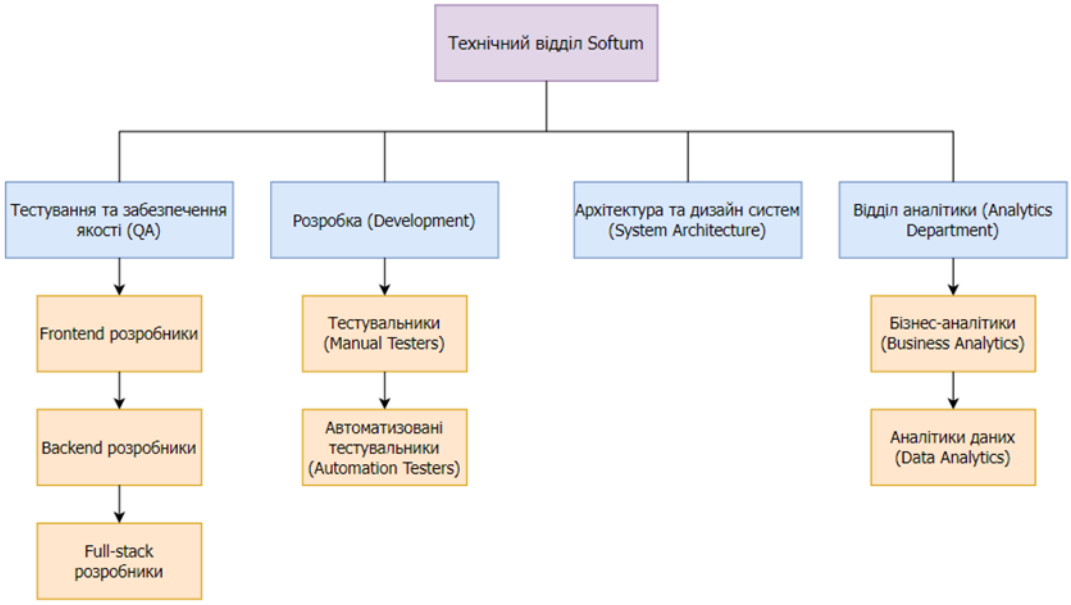

Рисунок В.2 – Технічний відділ компанії Softum

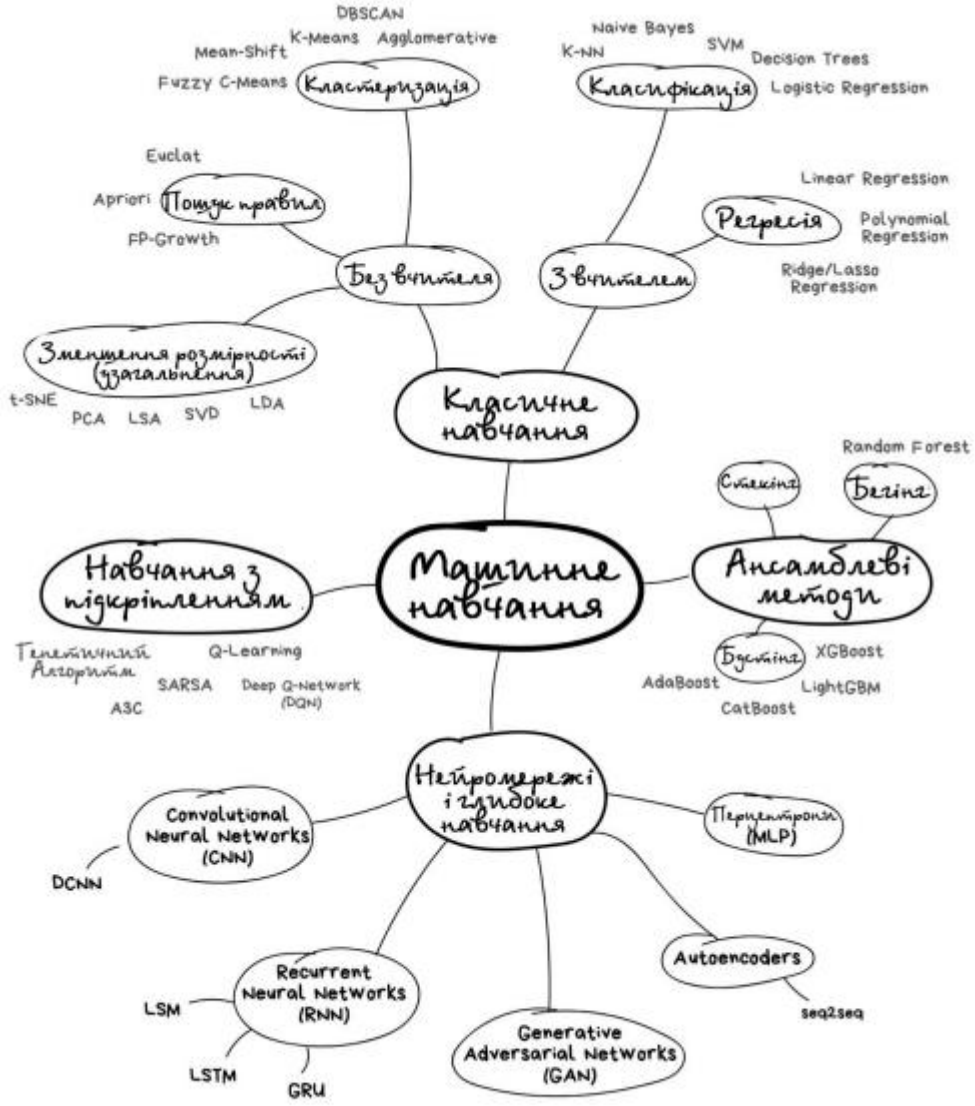

Рисунок В.3 – Класифікація методів машинного навчання

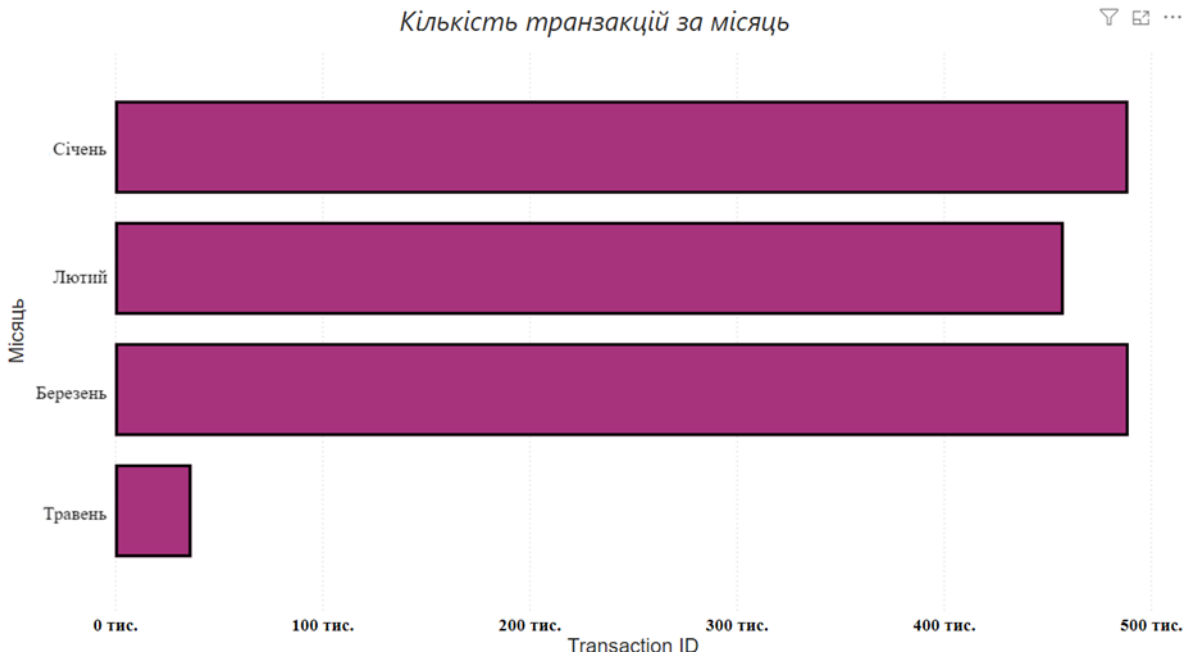

Рисунок В.4 – Кількість транзакцій за місяць

Рисунок В.5 – Відсоток підозрілих транзакцій

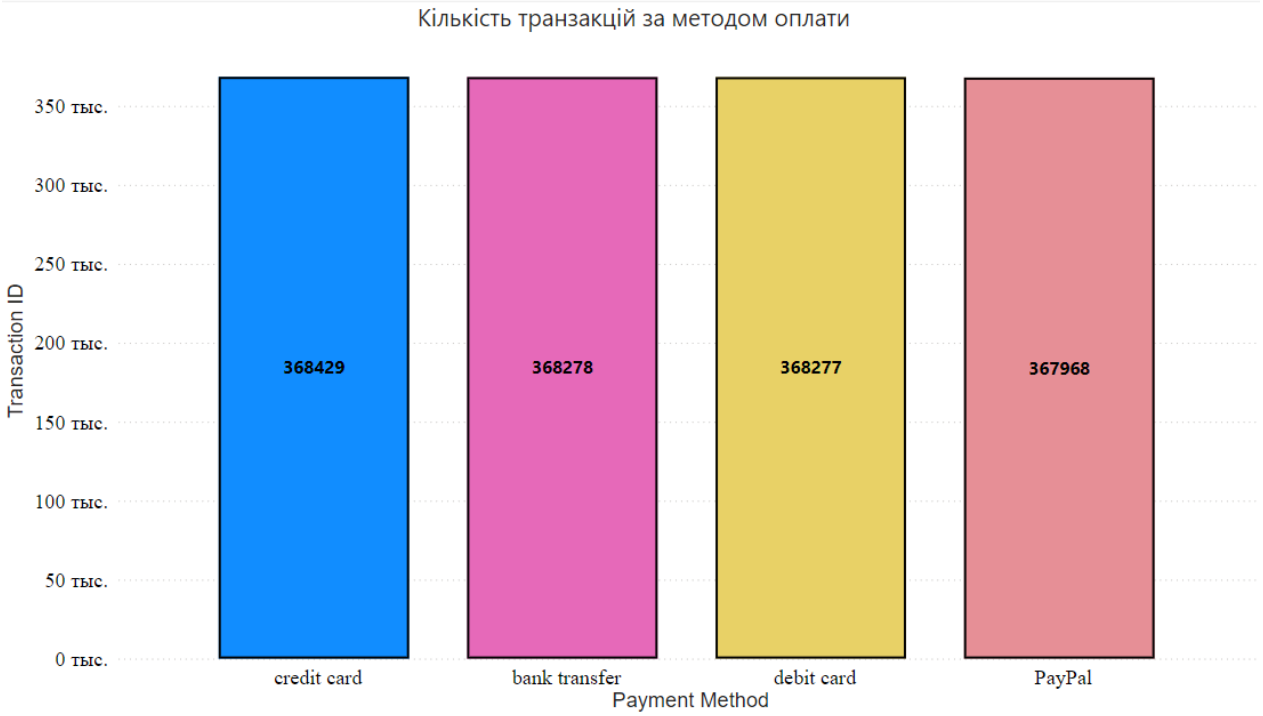

Рисунок В.6 – Кількість транзакцій за методом оплати

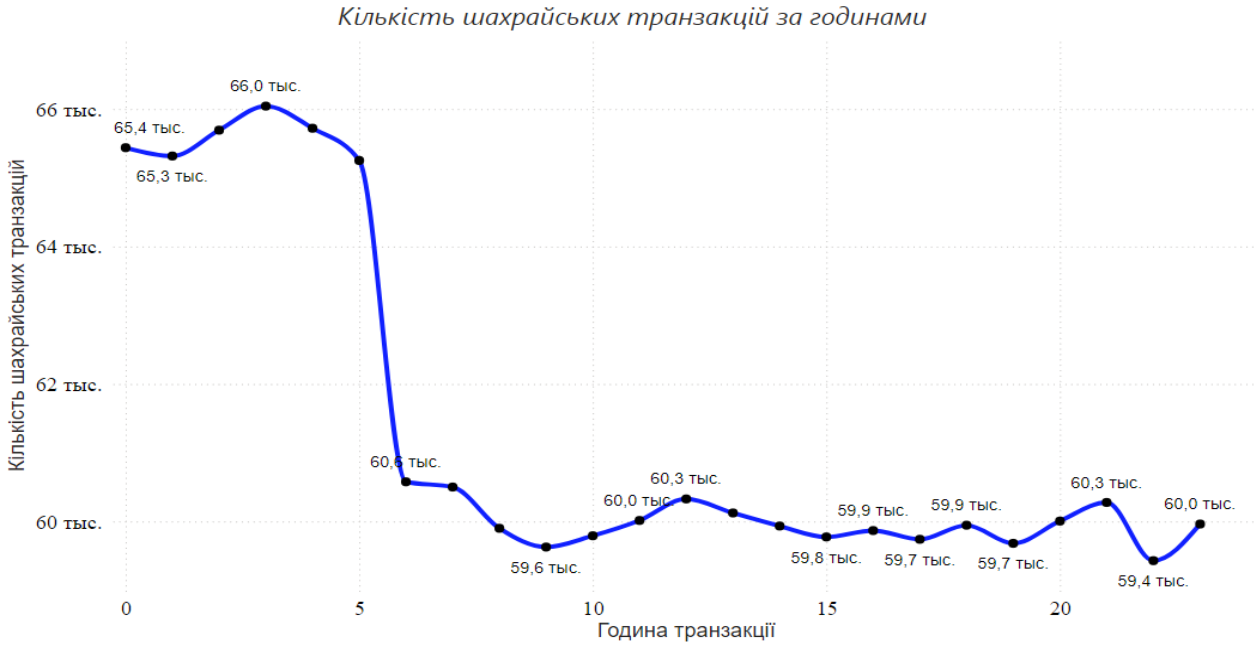

Рисунок В.7 – Кількість шахрайських транзакцій за годинами