Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра програмних систем і технологій

УДК 004.8 На правах рукопису

ВИПУСКНА КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

Тема: “Розробка системи для виявлення та запобігання шахрайських дій при проведенні фінансових операцій”

Спеціальність 121 “Інженерія програмного забезпечення”

ПОЯСНЮВАЛЬНА ЗАПИСКА

Студент:
Підгорний Максим Васильович

Науковий керівник:
д.т.н.; професор Бичков Олексій Сергійович

Допускається до захисту
з питань нормоконтролю
Завідувач кафедри

_____
(підпис) (розшифровка підпису) (дата)

Київ - 2022

Рішенням Екзаменаційної комісії
випускна кваліфікаційна робота студента

_____

захищена з оцінкою

_____

Голова Екзаменаційної комісії
д.т.н., проф. Андрій БОНДАРЧУК

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

Спеціальність 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ:

Завідувач кафедри програмних систем і технологій

_____(О.С.Бичков)

“____”_____ 20___р.

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Підгорному Максиму Васильовичу

1. Тема випускної кваліфікаційної магістерської роботи
“Розробка системи для виявлення та запобігання шахрайських дій при проведенні фінансових операцій”

керівник проекту (роботи) к.т.н. Бичков Олексій Сергійович

затверджені наказом вищого навчального закладу від „__”_____20__р. №_____

2. Строк здачі студентом закінченої роботи „__” _____ 2021 р.

3. Вихідні дані до роботи: навчальні посібники, статті, Інтернет-ресурси

4. Зміст пояснювальної записки (перелік питань, що їх належить розробити)

Аналітична частина:

- обґрунтувати актуальність розробки системи для виявлення шахрайських транзакцій;
- виконати огляд підходів до розробки та методів для системи виявлення шахрайських транзакцій;

Практична частина:

- спроектувати архітектуру системи для виявлення шахрайських транзакцій;
- розробити систему для виявлення шахрайських транзакцій;
- запустити проект в роботу.

5. Консультанти з роботи із зазначенням розділів роботи, що їх стосуються

Розділ	Консультат	Підпис, дата	
		Завдання видав	Завдання прийняв
Розділ 1	О.С. Бичков		
Розділ 2	О.С. Бичков		
Розділ 3	О.С. Бичков		

6. Дата видачі завдання 11 січня 2022 р.

Керівник_____/ О.С. Бичков /

Завдання прийняв до виконання_____/ М.В. Підгорний /

КАЛЕНДАРНИЙ ПЛАН

Номер і назва етапів роботи	Термін виконання етапів роботи	Примітка
1) виконати огляд підходів до розробки та методів для системи виявлення шахрайських транзакцій;	01.11.2021 - 01.01.2022	Виконано
2) спроектувати архітектуру системи для виявлення шахрайських транзакцій;	01.01.2022 - 01.02.2022	Виконано
3) розробити систему для виявлення шахрайських транзакцій;	01.02.2022 - 20.05.2022	Виконано

Студент – магістр **Підгорний М.В.**

(підпис) **(розшифровка підпису)**

Керівник роботи **Бичков О.С.**

(підпис) **(розшифровка підпису)**

АНОТАЦІЯ

Робота виконана на 43 сторінках, містить 8 ілюстрацій, 3 додатків. При підготовці використовувалась інформація з 5 джерел.

Актуальність роботи полягає в тому, що дослідження та здобутки в сфері штучного інтелекту та, зокрема, виявлення шахрайських транзакцій вже зараз використовуються поза межами наукових (або просто робіт) робіт вчених та дослідників. Враховуючи всі фактори, подібна система здатна значно полегшити роботу систем та установ, що дають змогу проводити платежі. Зокрема, це дає змогу

Знання та навички, здобуті при дослідженні та розробці компонентів із застосуванням подібних технологій можуть знадобитись при комерційній розробці та / або впровадженні систем та підсистем, що потребують контролю за автентичністю фінансових переведень в своїй роботі.

Мета

Метою роботи є розробка системи для виявлення шахрайських транзакцій.

Завдання

Для досягнення мети роботи були поставлені наступні завдання:

- Виконати аналіз предметної області
- Зробити огляд методів вирішення проблеми
- Розробити список бізнес-вимог, функціональних та нефункціональних вимог
- Спроектувати архітектуру системи
- Розробити підсистему, що буде виконувати задачу виявлення шахрайських транзакцій

Об’єктом дослідження є методи виявлення шахрайських транзакцій та впровадження одного з методів у вигляді розробленого модуля для виявлення шахрайських транзакцій.

Предметом дослідження є розробка модуля для виявлення шахрайських транзакцій на основі вибраного методу.

Практична цінність

В результаті виконання роботи був розроблений прототип системи виявлення шахрайських транзакцій. Створений сервіс може використовуватись іншими системами для детекції шахрайських транзакцій.

Ключові слова: шахрайські транзакції, система на основі правил, нейронні мережі, моделі.

ABSTRACT

The work is done on 43 pages, contains 8 illustrations, 3 appendices. Information from 5 sources was used in the preparation.

The relevance of the work is that research and achievements in the field of artificial intelligence and, in particular, the detection of fraudulent transactions are already used outside the scientific (or simply the work) of scientists and researchers. Taking into account all the factors, such a system can greatly facilitate the work of systems and institutions that allow payments. In particular, it allows.

The knowledge and skills acquired in the research and development of components using similar technologies may be required in the commercial development and / or implementation of systems and subsystems that require control over the fraudulend financial transactions in their work.

The goal of the work is to develop a system for detecting fraudulent transactions.

Tasks

- To achieve the goal of the work the following tasks were set:
- Perform subject area analysis Review methods of solving the problem
- Develop a list of business requirements, functional and non-functional requirements
- Design a system architecture
- Develop a subsystem that will perform the task of detecting fraudulent transactions

The object of research is the methods of fraudulent transaction detection and the introduction of one of the methods in the form of a developed module for fraudulent transaction detection.

The subject of research is the development of a module for detecting fraudulent transactions based on the chosen method.

Practical value: As a result of the work, a prototype of a fraudulent transaction detection system was developed. The created service can be used by other systems for detection of fraudulent transactions.

Key words: fraudulent transactions, rule-based system, neural networks, models.

ЗМІСТ

ВСТУП

Машинне навчання (англ. *machine learning*) — це підгалузь штучного інтелекту в галузі інформатики, яка часто застосовує статистичні прийоми для надання комп'ютерам здатності «навчатися» (тобто, поступово покращувати продуктивність у певній задачі) з даних, без того, щоби бути програмованими явно.

Виявлення шахрайства – це комплекс заходів, які здійснюються для запобігання одержанню грошей або майна за допомогою неправдивого приводу. Виявлення шахрайства застосовується в багатьох галузях, таких як банківська справа або страхування. У банківській сфері шахрайство може включати підробку чеків або використання крадених кредитних карток. Інші форми шахрайства можуть включати перебільшення збитків або спричинення нещасного випадку з єдиним наміром виплати. З необмеженою та зростаючою кількістю способів, якими хтось може вчинити шахрайство, виявлення може бути складним. Такі дії, як реорганізація, скорочення, перехід на нові інформаційні системи або зіткнення з порушенням кібербезпеки, можуть послабити здатність організації виявляти шахрайство. Рекомендуються такі методи, як моніторинг у реальному часі на предмет шахрайства. Організації мають шукати випадки шахрайств у фінансових операціях враховуючи локації платежів, пристроях, що використовуються, сесіях та системах аутентифікації.

Онлайн-платежі на сьогоднішній день є найпопулярнішою формою транзакцій у світі. Згідно з дослідженням Experian, понад 90% споживачів у всьому світі покладаються на онлайн-платежі для покупки товарів і послуг. Однак це збільшення онлайн-платежів призводить до збільшення кількості шахрайських транзакцій.

Станом на червень 2018 року 63% компаній у всьому світі повідомили, що протягом останнього року зазнали принаймні стільки ж збитків через шахрайство через канали онлайн-платежів, скільки й минулого року. Зіткнувшись із все більш безпечними системами захисту

від шахрайства, впровадженими набагато більшими компаніями, шахраї починають націлюватися на менші, менш захищені підприємства щоб випробувати удачу. Однак є способи захистити свій малий бізнес від цих шахрайських атак.

Види шахрайств

Шахрайство може здійснюватися різними способами та за різних обставин. Наприклад, шахрайство може бути скоєно в банківській, страховій, державній та медичній сферах.

Поширеним видом банківського шахрайства є заволодіння рахунками клієнта. Це коли хтось незаконно отримує доступ до банківського рахунку жертви за допомогою ботів. Інші приклади шахрайства в банківській сфері включають використання шкідливих програм, використання фальшивих ідентифікаційних даних, відмивання грошей, шахрайство з кредитними картками та мобільне шахрайство.

Страхове шахрайство включає шахрайство з перерахуванням премій, яке є розтратою страхових премій, або безкоштовним шахрайством, що є надмірною торгівлею біржовим брокером з метою максимізації комісійних. Інші форми страхового шахрайства включають вилучення активів, шахрайство з виплатою компенсацій працівникам, шахрайство з автомобільними аваріями, шахрайство з викраденим або пошкодженим автомобілем та шахрайство з пожежею в будинку. Мотивом усіх страхових шахрайств є фінансові прибутки.

Шахрайства, пов’язані з державними установами здійснюються проти державних установ, таких як Міністерство охорони здоров’я, Міністерство транспорту, Міністерство освіти, Міністерство енергетики тощо. Типи державних шахрайств включають виставлення рахунків за непотрібні процедури, переплату за товари, які коштують дешевше, надання старого обладнання під час виставлення рахунків за нове обладнання та звітування про відпрацьовані години працівника, який не існує.

В рамках роботи виконується дослідження шахрайств, пов’язаних з фінансовими платежами.

Це актуальна проблема, яка потребує уваги з боку сфери машинного навчання та науки про дані, де вирішення цієї проблеми може бути автоматизовано. Ця проблема особливо складна з погляду навчання, оскільки вона характеризується різними факторами, такими як дисбаланс класів. Кількість дійсних транзакцій значно перевищує кількість шахрайських. Крім того, моделі транзакцій часто змінюють свої статистичні властивості протягом час.

РОЗДІЛ 1

ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Аналіз предметної області

Шахрайство з оплатою відбувається, коли хтось викрадає платіжну інформацію іншої особи та використовує її для здійснення несанкціонованих транзакцій або покупок. Фактичний власник картки або власник платіжної інформації потім помічає, що його обліковий запис використовується для транзакцій або покупок, які він не авторизував, і порушує суперечку. Тут виникає проблема для власників бізнесу, оскільки їм доведеться вирішувати спір, сплачувати численні штрафи, такі як комісія за повернення платежів і плата за розслідування, а також загальна втрата часу та ресурсів. У деяких випадках клієнти самі можуть помилково ініціювати повернення платежу, заперечуючи, що коли-небудь отримували продукт. Це також форма шахрайства з оплатою.

Якщо постачальники торгових рахунків, такі як банки, вважають, що участь у транзакціях бізнесу стає все більш небезпечною, їхній обліковий запис компанії може бути деактивовано через ризик шахрайства. Легко побачити, наскільки проблематичним може бути шахрайство з оплатою для власників бізнесу.

Шахрайство в операціях з кредитними картками - це несанкціоноване та небажане використання рахунку кимось іншим, крім власника цього рахунку. Необхідні профілактичні заходи можуть бути вжиті для припинення цього зловживання, а поведінку таких шахраїв можна вивчити, щоб звести шахрайські транзакції до мінімуму та захистити від подібних випадків у майбутньому. Іншими словами, шахрайство з кредитними картками можна визначити як випадок, коли людина використовує чиюсь чужу кредитну картку для особистих цілей, у той час як власник і Власник та органи, що випустили картку, не знають про те, що картка використовується. Виявлення шахрайства включає моніторинг діяльності групи користувачів з метою оцінки, сприйняття чи

запобігання небажаної поведінки, що складається з шахрайства, вторгнення та невиконання обов'язків.

Шахрайство, як правило, включає кілька повторюваних методів, що робить пошук шаблонів загальним фокусом для виявлення шахрайства. Наприклад, аналітики даних можуть запобігти страховому шахрайству, створюючи алгоритми для виявлення закономірностей і аномалій.

Виявлення шахрайств можна розділити за допомогою методів статистичного аналізу даних або штучного інтелекту (ШІ).

Методи аналізу статистичних даних включають:

- обчислення статистичних параметрів
- регресійний аналіз
- розподіли ймовірностей і моделі
- відповідність даних

Методи штучного інтелекту, які використовуються для виявлення шахрайства, включають:

- Інтелектуальний аналіз даних, що класифікує, групує та сегментує дані для пошуку в мільйонах транзакцій, щоб знайти закономірності та виявити шахрайство.
- Нейронні мережі вивчають підозрілі моделі і використовують ці шаблони для подальшого їх виявлення.
- Машинне навчання автоматично визначає ознаки шахрайства.
- Розпізнавання шаблонів виявляє класи, кластери та моделі підозрілої поведінки.

Однак це не єдині проблеми при впровадженні системи виявлення шахрайства. У реальних прикладах, масивний потік запитів на оплату швидко сканується автоматичними інструментами, які визначають, які транзакції справжні.

Алгоритми машинного навчання використовуються для аналізу всіх санкціонованих транзакцій та повідомляють про підозрілі. Ці звіти вивчаються фахівцями, які зв'язуються з власниками карт, щоб підтвердити, чи була транзакція справжньою або шахрайської. Розслідувачі надають зворотний зв'язок автоматизованій системі. яка використовується для навчання та оновлення алгоритму, щоб в кінцевому підсумку покращити ефективність виявлення шахрайства з часом.

1.2. Вибір алгоритму роботи

Система розпізнавання шахрайських транзакцій на основі системи правил

Ці системи покладаються на жорстко закодовані правила, які встановлюються таким чином, щоб відзначати транзакції, якщо вони відповідають певним критеріям. Такі правила можуть бути розроблені шляхом:

- наслідуючи кращу галузеву практику - наприклад, блокуючи кілька транзакцій з одного рахунку за короткий проміжок часу або транзакції, що проходять через VPN або з ризикованих зон
- аналізу спійманих / запобігли шахрайським транзакціям і розробляти нові правила, щоб охопити всі їхні підозрілі характеристики.

Правила часто виражаються за допомогою операторів "if-else", присутніх майже у всіх імперативних мовах програмування, легко інтерпретуються. Вони відображають спосіб обробки транзакції людиною - механізм перевіряє, чи транзакція відповідає будь-якій з ризикованих моделей, виражених у правилах, і якщо так, то блокує її або відправляє на ручну перевірку людиною. Це одна з причин, чому їхня присутність все ще дуже сильна - зацікавлені сторони довіряють їм, бо вони імітують спосіб, за допомогою якого вони самі вирішували б це завдання.

Переваги

- Повна зрозумілість із коробки - якщо певне правило викликало попередження для конкретної транзакції, то на 100% зрозуміло, чому це сталося.
- Відсутність проблеми "холодного старту" - вони працюють з першого дня, не потрібно збирати набори навчальних даних, які потрібні для алгоритмів машинного навчання.
- Низький поріг входження – вам не потрібна команда data scientist'ів, інженерів машинного навчання або MLOps – перші правила можуть бути легко реалізовані командою бекенда, оскільки вони вже знайомі з переведенням бізнес-логіки в код.

Недоліки

- Постійна потреба в реінжинірингу атак шахраїв – нові правила мають розроблятися у міру появи нових моделей шахрайства.
- Збільшення кількості правил - вартість обслуговування зростає з часом (перекалібрування та адаптація до нових моделей шахрайства).
- Виявлення випадків шахрайства з обмеженою складністю - існує межа кількості правил і особливостей транзакцій. Системи, засновані на правилах, обмежені людським розумінням (через ручне розроблення правил та необхідного обслуговування).

Методи машинного навчання

Для вирішення даної проблеми можна використовувати велику кількість методів ML. Це відображено у вигляді величезної кількості опублікованих робіт на цю тему за останнє десятиліття.

При виявленні шахрайства з кредитними картками дані зазвичай складаються з даних про транзакції, зібрані, наприклад, платіжним процесором або банком. Дані про транзакції можна розділити на три групи

- Характеристики, пов'язані з рахунком - вони включають, наприклад, номер рахунку, дату відкриття рахунку, ліміт картки, дату закінчення терміну дії картки тощо.

- Характеристики, пов'язані з транзакціями - до них відносяться, наприклад номер посилання на операцію, номер рахунку, сума операції, номер терміналу (тобто POS), час операції і т.д. З терміналу можна отримати додаткову категорію інформації: характеристики, пов'язані з торгівлею, такі як код категорії (ресторан, супермаркет, ...) або місцезнаходження.
- Характеристики, пов'язані з клієнтом - вони включають, наприклад, номер клієнта, тип клієнта (низький профіль, високий профіль, ...) і т.д.

Переваги системі на основі машинного навчання:

- Автоматичне розпізнавання образів шахрайства – завдання з'ясувати, що саме є шахрайством, що вирішується алгоритмом. Наше завдання - надати йому якомога детальніший опис (у вигляді вектора ознак).
- Дрейф концепції, що визначається як зміна характеристик шахрайства в часі (нові методи шахрайства, нові інструменти, що використовуються шахраями), часто може бути вирішено шляхом перенавчання моделей на нових даних - немає потреби в реінжинірингу методів шахраїв.
- Менше ручної роботи – багато процесів можна автоматизувати. Компанії, що мають зрілі конвеєри машинного навчання, витрачають більшу частину часу на дослідження нових функцій та алгоритмів, стежачи за показниками продуктивності поточних моделей, доступних через програми для моніторингу.
- Економічна ефективність ML-моделей зростає разом із обсягом даних. Що більше даних і що вони складніше, то складніше розробляти системи, засновані на правилах. Таким чином, рентабельність розробки автоматизованих систем виявлення шахрайства з використанням ML-моделей зростає зі збільшенням обсягу даних.

Недоліки

- Проблема "холодного старту" - для запуску ML-моделей потрібна значна кількість історичних даних.
- Відсутність з'ясовності з коробки - не всі прогнози алгоритмів можна легко пояснити, деякі з них є "чорними ящиками", для яких немає простих пояснень між входами та виходами.

Оскільки, на мою думку, для вирішення даної проблеми краще підходить система на основі машинного навчання, фундаментом системи був обраний саме цей підхід.

У своїй найпростішій формі транзакція платіжною карткою складається з будь-якої суми, сплаченої клієнтом продавцю в певний час. Набір історичних даних про транзакції може бути представлений у вигляді таблиці. Для виявлення шахрайства зазвичай передбачається, що легітимність всіх транзакцій відома (тобто була транзакція справжньою чи шахрайською). Це зазвичай є двійковою міткою, значення 0 якої означає справжню транзакцію, а значення 1 - шахрайську.

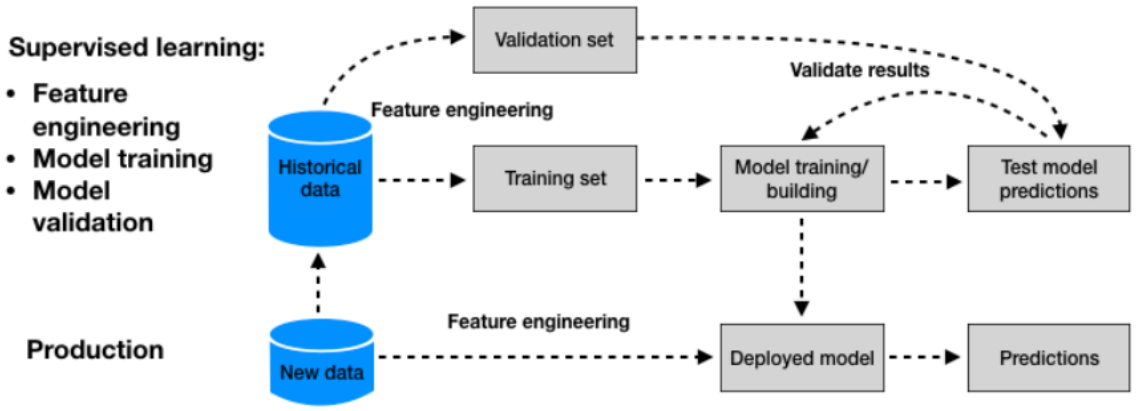


Рис. 1.1.. Приклад схеми роботи з моделлю машинного навчання при використанні навчання з учителем.

У своїй найпростішій формі транзакція платіжною карткою складається з будь-якої суми, сплаченої клієнтом продавцю в певний час. Набір історичних даних про транзакції може бути представлений у вигляді таблиці, файлу тощо. Для виявлення шахрайства зазвичай передбачається, що легітимність усіх транзакцій відома (тобто була транзакція справжньою

чи шахрайською). Це зазвичай є двійковою міткою, значення 0 якої означає справжню транзакцію, а значення 1 - шахрайську.

У розробці системи виявлення шахрайства на основі ML можна виділити два етапи. Перший етап полягає у побудові моделі прогнозування на основі набору помічених історичних даних (рис. 1, верхня частина). Цей процес називається контрольованим навчанням, оскільки мітка транзакції (1/0, справжня або шахрайська) відома для всіх записів. На другому етапі модель прогнозування, отримана у процесі контрольованого навчання, використовується для прогнозування значень міток нових транзакцій (рис. 1, нижня частина).

Формально модель прогнозування - це параметрична функція з параметрами θ, також звана гіпотезою, яка приймає вхідні дані χ з вхідної області $\chi \subset R^n$, і видає прогноз $y = h(\chi, \theta)$ по вихідній області $Y \subset R$:

$$h(x, \theta): X \to Y \qquad [1.1]$$

Вхідна область χ зазвичай відрізняється від простору необроблених даних про транзакції з двох причин. По-перше, з математичних причин більшість алгоритмів контрольованого навчання вимагають, щоб вхідна область мала реальні значення, тобто, $\chi \subset R^n$, що вимагає перетворення ознак транзакції, які не є речовими числами (наприклад, тимчасові мітки, категоріальні змінні тощо). По-друге, зазвичай корисно збагатити дані про транзакції іншими змінними, які можуть покращити ефективність виявлення моделі прогнозування. Цей процес називається інженерією ознак (також відомою як перетворення ознак, вилучення ознак або попередня обробка даних).

Для виявлення шахрайства вихідна область зазвичай є передбачений клас Y для даного входу χ , тобто $y = \{0, 1\}$. Зважаючи на те, що вихідний клас є бінарним, такі моделі прогнозування також називаються бінарними класифікаторами. В якості альтернативи, вихідний сигнал може бути

виражений як ймовірність шахрайства, з $y = \{0, 1\}$, або в більш загальному випадку як оцінка ризику, з $Y = R$, де більш високі значення виражають більш високий ризик шахрайства.

Навчання (або побудова) моделі $y = h(\chi, \theta)$ передбачення полягає у пошуку параметрів θ, що забезпечують найкращу продуктивність. Ефективність моделі передбачення оцінюється за допомогою функції втрат, яка порівнює справжню мітку *y* із передбаченою міткою $y = h(\chi, \theta)$ для вхідного сигналу *x*. У завданнях двійкової класифікації поширеною функцією втрат є функція втрат $L_{\frac{0}{1}}$, яка привласнює втрату, рівну одиниці у разі неправильного передбачення або нуль в іншому випадку:

$$L_{\frac{0}{1}}: Y \times Y \to \{0, 1\}\, y, y\hat{} = \{1, if y \neq y\hat{}\}\, y, y\hat{} = \{0, if y = y\hat{}\}$$

Для отримання справедливої оцінки ефективності моделі прогнозування важливою методологічною практикою, відомою як валідація є оцінка ефективності моделі прогнозування на даних, які не використовувалися для навчання. Це досягається шляхом поділу набору даних перед навчанням на навчальну та перевірочну множини. Навчальна множина використовується для навчання моделі прогнозування (тобто для пошуку параметрів θ, які мінімізують втрати на навчальній множині). Після того, як параметри θ визначені, втрати оцінюються на перевірочній множині, що дає найкращу оцінку продуктивності, яку модель прогнозування повинна мати на майбутніх (та невидимих) транзакціях.

Процедура контрольованого навчання зазвичай складається з навчання набору моделей прогнозування та оцінки їхньої ефективності на перевірочній множині. Наприкінці процедури вибирається модель, яка, як передбачається, забезпечує найкращу продуктивність (тобто найменші втрати на перевірочній множині), і використовується для прогнозування нових транзакцій.

Огляд проблем

Дисбаланс класів: Дані про транзакції містять набагато більше законних транзакцій, ніж шахрайських: Відсоток шахрайських транзакцій у реальному наборі даних зазвичай становить менше 1%. Навчання на

незбалансованих даних є складним завданням, оскільки більшість алгоритмів навчання погано долають великі відмінності між класами. Для вирішення проблеми дисбалансу класів потрібно використовувати додаткові стратегії навчання, такі як вибірка або зважування втрат, що відоме як дисбалансне навчання.

Дрейф концепцій: Моделі транзакцій та шахрайства змінюються з часом. З одного боку, видаткові звички користувачів кредитних карток різняться у будні та вихідні дні, у період відпусток і в цілому змінюються з часом. З іншого боку, шахраї запроваджують нові методи, оскільки старі старіють. Ці залежні від часу зміни у розподілі транзакцій та шахрайства називаються дрейфом концепцій. Дрейф понять вимагає розробки стратегій навчання, які можуть упоратися з тимчасовими змінами статистичних розподілів, що відоме як онлайн-навчання. Насправді проблема дрейфу понять загострюється через затримки зворотний зв'язок (див. розділ Система виявлення шахрайства з кредитними картами).

Вимоги близькі до реального часу: Системи виявлення шахрайства мають бути здатні швидко виявляти шахрайські операції. Враховуючи потенційно великий обсяг даних про транзакції (мільйони транзакцій на день), час класифікації може становити десятки мілісекунд. Це завдання тісно пов'язане з розпаралелюванням та масштабованістю систем виявлення шахрайства.

Категоріальні ознаки: Транзакційні дані зазвичай містять безліч категоріальних ознак, таких як ідентифікатор клієнта, термінал, тип картки тощо. Категоріальні ознаки погано піддаються обробці алгоритмами машинного навчання, тому їх необхідно перетворювати на числові ознаки. Загальні стратегії перетворення категоріальних ознак включають в себе агрегування ознак, перетворення на основі графів або підходи глибокого навчання, такі як вбудовування ознак.

Послідовне моделювання: Кожен термінал та/або клієнт генерує потік послідовних даних з унікальними характеристиками. Важливе завдання виявлення шахрайства полягає у моделюванні цих потоків, щоб краще охарактеризувати їх очікувану поведінку та виявити, коли

відбувається аномальна поведінка. Моделювання може здійснюватись шляхом агрегування характеристик у часі (наприклад, відстеження середньої частоти або суми транзакцій клієнта) або за допомогою моделей послідовного прогнозування (наприклад, прихованих моделей Маркова або рекурентних нейронних мереж).

Перекриття класів: Останні дві проблеми можуть бути пов'язані з загальною проблемою перекриття двох класів. Маючи лише необроблену інформацію про транзакцію, відрізнити шахрайську транзакцію від справжньої практично неможливо. Ця проблема зазвичай вирішується за допомогою методів розробки ознак, що додають контекстну інформацію до необробленої платіжної інформації.

Показники ефективності: Стандартні показники систем класифікації, такі як середня помилка неправильної класифікації або AUC ROC, не дуже добре підходять для проблем виявлення через проблему дисбалансу класів і складної структури витрат при виявленні шахрайства. Система виявлення шахрайства має бути здатною максимізувати виявлення шахрайських транзакцій при мінімізації кількості невірно передбачених шахрайств (хибних спрацьовувань). Для оцінки загальної ефективності системи виявлення шахрайства часто необхідно враховувати кілька показників. Незважаючи на те, що показники ефективності відіграють центральну роль у розробці системи виявлення шахрайства, нині немає єдиної думки про те, який набір показників ефективності слід використовувати.

Відсутність відкритих наборів даних: З очевидних міркувань конфіденційності набір даних з реальними транзакціями знайти неможливо.

РОЗДІЛ 2
ПРОЕКТУВАННЯ СИСТЕМИ

2.1. Аналіз вимог до системи

Функціональні вимоги

1) Система повинна надавати API для перевірки транзакцій

Взаємодія з системою має бути побудованою за рахунок REST API, яке надасть змогу оцінювати кожну наступну транзакцію за допомогою натренованої моделі на ступінь шахрайності.

Нефункціональні вимоги

1) Сумісність системи з вимогами GDPR

Загальний регламент про захист даних (англ. *General Data Protection Regulation, GDPR*; Regulation (EU) 2016/679) — регламент в межах законодавства Європейського Союзу щодо захисту персональних даних усіх осіб у межах Європейського Союзу та Європейської економічної зони. Він також стосується експорту персональних даних за межі ЄС і Європейської Економічної Зони. GDPR покликаний насамперед надати громадянам та резидентам ЄС контроль за їхніми персональними даними та спростити регуляторне середовище для міжнародного бізнесу шляхом уніфікації регулювання в межах ЄС.

Оскільки продуктом потенційно можуть користуватись громадяни ЄС, продукт повинен дотримуватись вимог GDPR для уникнення потенційних судових процесів та стягнень.

Список вимог, які можуть бути порушені системою та шляхи їх вирішення:

- Використання псевдонімів - замість будь-яких даних реальних осіб необхідно використовувати виключно ідентифікатори користувачів (які мають сенс лише всередині системи).
- Право на стирання - необхідно надати функціонал, що дозволяє видаляти дані користувача, що робить запит на видалення своїх даних, з набору даних, що використовується для навчання моделі.

- Співробітник з питань захисту даних - ця умова повинна бути виконуватись з боку клієнта, тобто юридичної або фізичної особи, що володіє ліцензією на користування даною системою. Це ж стосується і пункту про записи опрацювання даних (відповідальна особа повинна вести записи про про цілі опрацювання даних, залучені категорії та передбачені терміни зберігання)

2) Запит на перевірку транзакції повинен виконуватись менше двох секунд

Оскільки дана підсистема, скоріш за все, буде використовуватись як частина багатокрокових складних операцій, вона не повинна занадто гальмувати виконання багатокрокової операції загалом.

3) SLA системи має перевищувати 95%

Підсистема повинна перебувати у робочому стані більш, ніж 95% часу роботи системи, в яку вона буде інтегруватись.

4) Розробка CI/CD пайплайну для системи

Як один з факторів підтримання високої якості та стабільності системи, необхідно розробити пайплайн, який дасть змогу запускати тести та збірку проекту кожного разу, коли створюються запити на додавання нових змін в системі контролю версій.

2.2. Модель даних

Набір даних складається з трьох моделей:

- Термінал
- Користувач
- Транзакція

Генерація даних

Модель терміналу складається з полів TERMINAL_ID, TERMINAL_X, TERMINAL_Y.

TERMINAL_ID - унікальний ідентифікатор платіжного пристрою, число, якісний, діапазон можливих значень - 0..4999 (вказується при генерації даних, в даному наборі даних використовується 5000 терміналів).

TERMINAL_X - геолокація платіжного пристрою за віссю X, число, якісний, діапазон можливих значень - 0..1000,000000.

TERMINAL_Y - геолокація платіжного пристрою за віссю Y, число, якісний, діапазон можливих значень - 0..1000,000000.

Таблиця №2.1. Приклад записів в таблиці терміналів

TERMINAL_ID	TERMINAL_X	TERMINAL_Y
1	76.308289	779.918792
2	438.409231	723.465178

Модель користувача складається з полів USER_ID, USER_X, USER_Y, MEAN_AM, STD_AM, MEAN_TX_PER_DAY

USER_ID - унікальний ідентифікатор користувача, число, якісний, діапазон можливих значень - 0..3999 (число користувачів вказується при генерації даних, в даному випадку використовується набір даних з 4000 користувачів).

USER_X - відцентроване локація користувача за віссю X, число з плаваючою крапкою, якісний, діапазон можливих значень - 0..1000,000000.

USER_Y - відцентроване локація користувача за віссю Y, число з плаваючою крапкою, якісний, діапазон можливих значень - 0..1000,000000.

MEAN_AM - медіана розміру транзакції користувача, число з плаваючою крапкою, кількісний, діапазон можливих значень - 100,000000..5000,000000.

STD_AM - стандартне відхилення медіани транзакції, в конкретному наборі даних - половина медіани, число з плаваючою крапкою, кількісний, діапазон можливих значень - 50,000000..2500,000000.

MEAN_TX_PER_DAY - середня число транзакцій, здійснених користувачем за день, число з плаваючою крапкою, кількісний, діапазон можливих значень - 0,000000..4,000000.

Таблиця №2.2. Приклад записів в таблиці користувачів

USER_ID	USER_X	USER_Y	MEAN_AM	STD_AM	MEAN_TX_PER_DAY
1	548.813504	715.189366	3053.540543	1526.770271	2.179533

2	423.654799	645.894113	1122.088668	1122.088668	3.567092

Модель транзакції складається з полів DATETIME, USER_ID, TERMINAL_ID, AMOUNT, TIME_SECONDS, TIME_DAYS

DATETIME - дата та час проведення транзакції в форматі YYYY-MM-DD HH:mm:ss, якісний, діапазон значень - 2021-01-01 00:00:00 .. 2021-06-31 23:59:59 (початкова дата та кількість днів вказується при генерації даних).

USER_ID - ідентифікатор користувача, що виконав транзакцію, число, якісний, діапазон можливих значень 0..3999 (вказується при генерації даних, даних набір даних містить дані про операції 4000 користувачів).

TERMINAL_ID - ідентифікатор терміналу, що був використаний для проведення транзакції, число, якісний, діапазон значень - 0..4999 (вказується при генерації даних, даний набір даних містить операції, виконані за допомогою 5000 унікальних терміналів).

AMOUNT - сума транзакції, число з плаваючою крапкою, кількісний, діапазон значень 0..1.7976931348623157e+308 (для транзакцій, що не є шахрайськими, максимальне значення обмежене сумою медіани та стандартного відхилення, для шахрайських - максимальним значенням числа з плаваючою крапкою в межах мови програмування (або іншої утиліти), що використовується для генерації даних, в даному випадку використовується python).

TIME_SECONDS - кількість секунд від початкового дня (відносно стартової точки генерації даних, якщо дані генеруються починаючи з 2021-01-01, то значення цього поля дорівнюватиме 0 + час в секундах) - число, кількісний, діапазон значень - 0..1.6*10^7 (значення округлене, фактично - дорівнює різниці між стартовою датою генерації даних та кількістю днів, для якої генеруються дані).

TIME_DAYS - порядковий номер дня з початку генерації даних, число, якісний, діапазон значень - 0..182 (кількість днів вказується при генерації даних).

Таблиця №2.3. Приклад записів в таблиці транзакцій

DATETIME	USER_ID	TERMINAL_ID	AMOUNT	TIME_SECONDS	TIME_DAYS
2021-01-01 07:19:05	0	29	6061.16	26345	0
2021-01-02 15:13:02	0	20	1586.69	141182	1

2.3. Опис згенерованих даних

Який би з алгоритмів оцінки транзакції не був обраним, ключові характеристики даних, які будуть враховуватись при оцінці транзакції - локація користувача, локація терміналу та термінали, наближені до користувача.

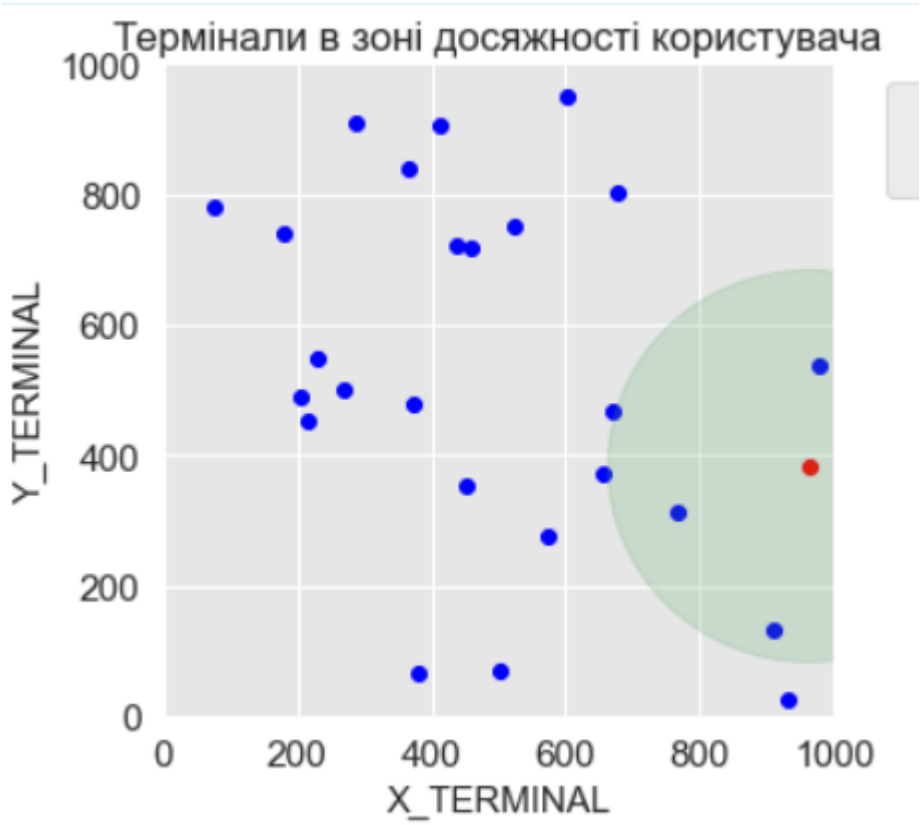


Рис. 2.1. Термінали в зоні досяжності користувача

Розподіл даних за сумою транзакції зображений на Рис. 2.2.

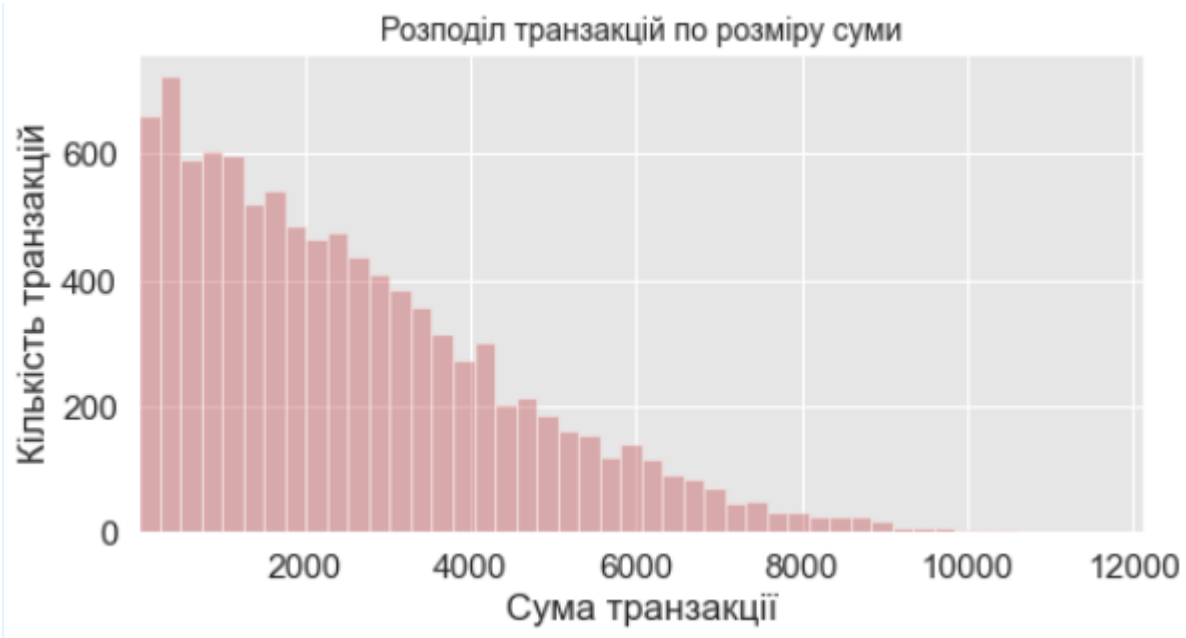


Рис. 2.2. Розподіл транзакцій за розміром суми

Розподіл транзакцій за днями зображений на Рис. 2.3. (перші 10 днів).

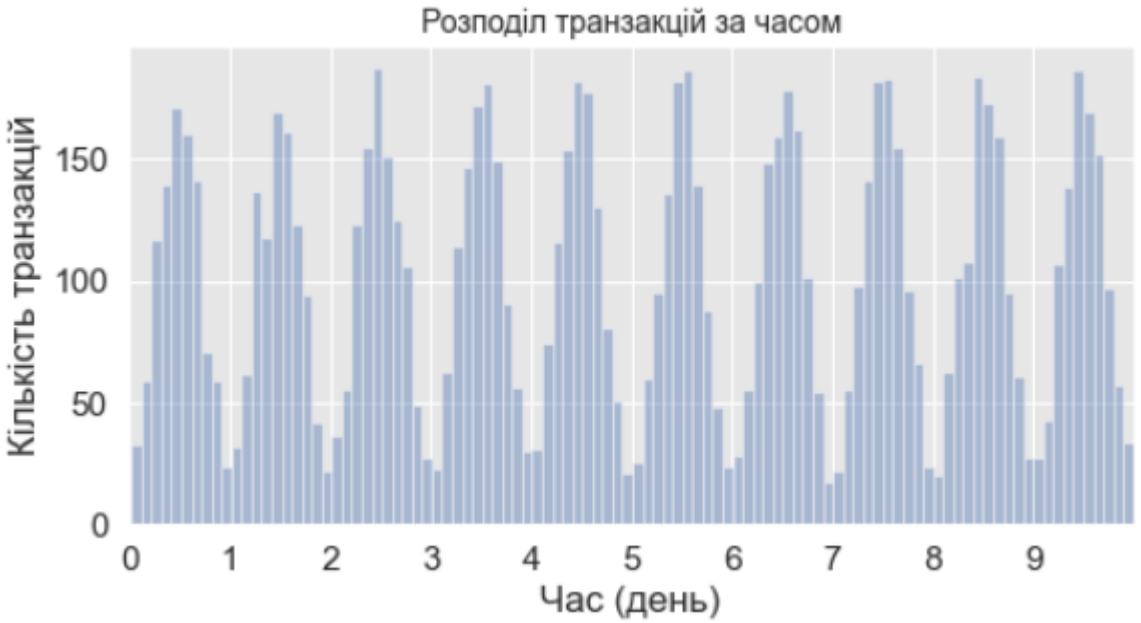


Рис. 2.3. Розподіл транзакцій за часом (перші 10 днів)

РОЗДІЛ 3
РОЗРОБКА СИСТЕМИ

3.1. Метрики навчання моделі

Для оцінки якості згенерованих даних та навчання моделі необхідно скористатись метриками, що допоможуть в оцінці отриманої моделі

Перш за все, варто навести графік втрат при тренуванні та тестуванні моделі відносно кількості епох

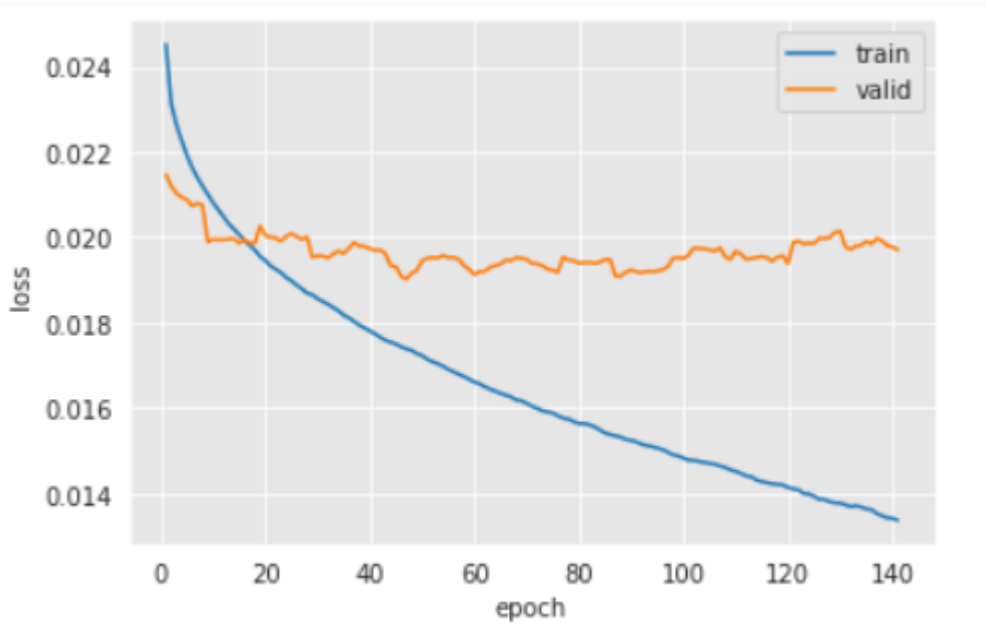


Рис. 3.1. Графік втрат при навчанні та тренуванні відносно епох

На графіку втрат можна помітити, як втрати при навчанні поступово зменшуються. Це означає, що оптимізація пройшла добре: обрана швидкість навчання дозволяє оновлювати модель в бік кращого рішення для навчального набору даних.

ROC-крива

Крива операційної характеристики приймача, або ROC-крива - це графічний графік, що ілюструє діагностичну здатність системи бінарного класифікатора при зміні порога дискримінації. Спочатку метод був розроблений для операторів військових радіолокаційних приймачів, починаючи з 1941 року, що й зумовило його назву.

Найкращий класифікатор відповідає точці (0,1) у просторі ROC (немає хибнонегативних і хибнопозитивних результатів), у той час як

класифікатор, що передбачає випадковим чином, мав би показники вздовж діагоналі, що з'єднує лівий нижній кут з правим верхнім. Якщо немає явного переможця (наприклад, класифікатор K домінує над W лише в одній частині простору ROC), порівняння зазвичай проводиться шляхом обчислення площі під кривою ROC (AUC ROC). AUC ROC зазвичай обчислюється за допомогою правила трапеції (лінійно інтерполяції), яке забезпечує очікувану ймовірність для TPR та FPR у разі рівності передбачуваних значень.

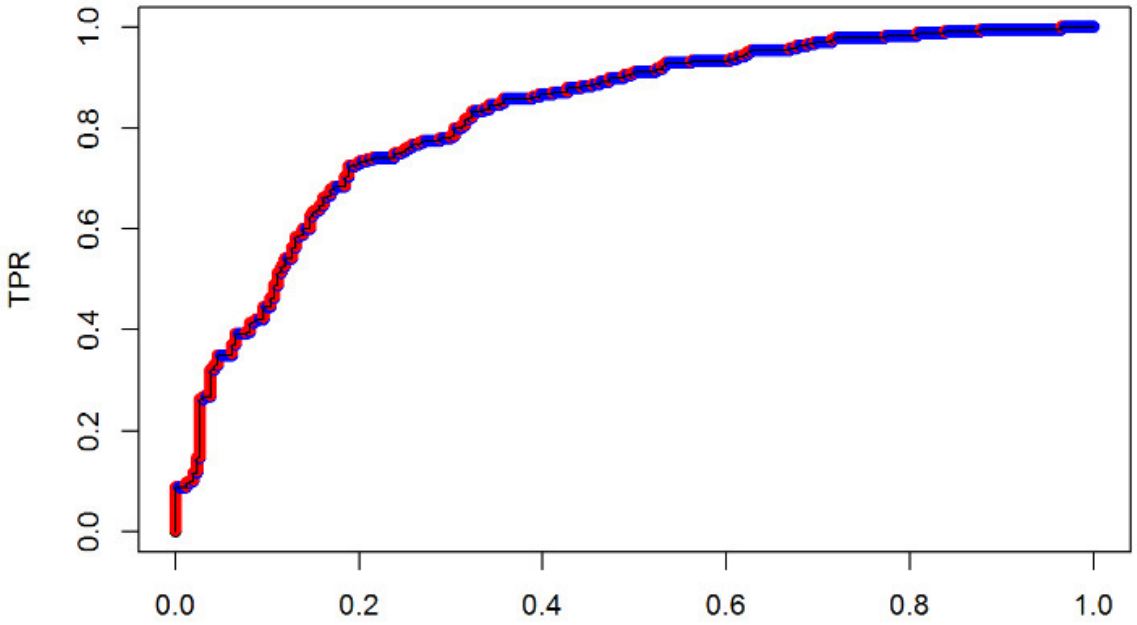


Рис. 3.2. ROC-крива

3.2. Демонстрація роботи

Оскільки взаємодія з системою організована у вигляді REST API, для тестування системи використаємо Postman.

Для того, щоб перевірити транзакцію на ступінь її “справжньості”, необхідно надіслати запит на відповідний ендпоїнт, в тілі запиту має бути вказана сума транзакції, ідентифікатор терміналу та ідентифікатор користувача, тобто необхідний набір параметрів, необхідних для оцінки транзакції.

У випадку, якщо транзакція буде оцінена, як шахрайська, відповідь на HTTP-запит буде з кодом 406 (NOT_ACCEPTABLE), в іншому випадку повернеться відповідь з кодом 202 (ACCEPTED). В тілі буде міститись

детальна інформація про транзакцію, це дасть змогу використати дані на UI іншої системи, що надсилає запит на перевірку.

Результат виконання запиту з успішним результатом зображене на Рис. 3.3.

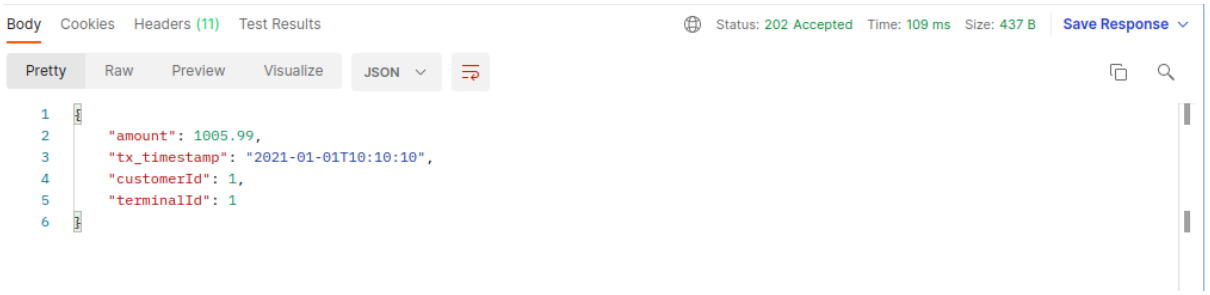


Рис. 3.3. Приклад тіла з позитивним результатом перевірки відповіді на HTTP-запит

Результат виконання запиту з негативним результатом зображене на Рис. 3.4.

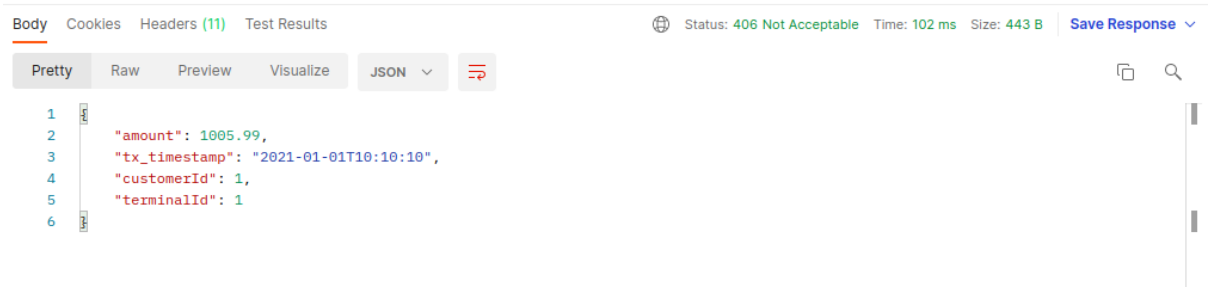


Рис. 3.4. Приклад тіла з негативним результатом перевірки відповіді на HTTP-запит

ВИСНОВКИ

В результаті виконання проекту на тему “Розробка системи для виявлення шахрайських транзакцій” було спроектовано та створено систему для виявлення шахрайських транзакцій.

Переваги системи:

1) Cloud-ready – система спроектована таким чином, що може бути розміщена в хмарі. Це дозволить позбутись з точки зору компанії-провайдера послуг видатків на купівлю та підтримку фізичних серверів, на яких може розміщуватись система

2) Інтегрованість з клієнтськими сервісами – система працює як окремий модуль, вона не потребує жорсткої зв’язаності з сервісами та/або різними компонентами інформаційної системи клієнта

3) Масштабованість – завдяки обраній мікросервісній та Cloud-ready архітектурі ми маємо змогу масштабувати кожен сервісний компонент окремо один від одного, в залежності від підвищення або зниження навантаження на сервер

Майбутні шляхи розвитку системи

Створена система є лише закладеним фундаментом для подальшого розвитку системи виявлення шахрайських транзакцій. Після впровадження системи у production варто продовжувати розвиток та покращення продукту як з технічного боку, так і з боку user experience.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Andrea Dal Pozzolo. *Adaptive machine learning for credit card fraud detection*. Université libre de Bruxelles, 2015.

2. Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, 233–240. 2006.

3. Kang Fu, Dawei Cheng, Yi Tu, and Liqing Zhang. Credit card fraud detection using convolutional neural networks. In *International conference on neural information processing*, 483–490. Springer, 2016.

4. Imane Sadgali, Nawal Sael, and Faouzia Benabbou. Detection of credit card fraud: state of art. *International Journal of computer science and network security*, 18(11):76–83, 2018.

5. Statistic Brain Research Institute. Credit card fraud statistics. April 2018. [Електронний ресурс]. URL: https://www.statisticbrain.com/credit-card-fraud-statistics/.

6. General Data Protection Regulation [Електронний ресурс], URL: https://gdpr-info.eu/

ДОДАТКИ

ДОДАТОК А

Структура тіла запиту до API системи виявлення шахрайських транзакцій

```
{
    "transactionAmount": "1000.59",
    "customerId": 123,
    "terminalId": 123
}
```

ДОДАТОК Б

Структура відповіді на запит про оцінку ранзакції


```
{
    "tx_timestamp": "2021-04-23T18:25:43.511Z",
    "transactionAmount": "1000.59",
    "customerId": 123,
    "terminalId": 123
}
```

ДОДАТОК В

Fraud detection system

Software Architecture Document (SAD)

CONTENT OWNER: Maksym Pidhornyi

TABLE OF CONTENTS

1. Introduction

In a modern globalized world electronic payment is not a rocket science, you can pay in the grocery store using your phone, you can use your credit card in foreign online stores to pay for some useless goods etc. But it also brings up a number of problems. As an online shop owner or a bank mobile app developer you should always consider each transaction as fraudulent - either a physical credit card can be stolen or its digital analog (the user's mobile banking account can be hacked) but the result is the same - money can be stolen from its owner. The result is loss of trust to the bank in case its online systems have low level of security or direct loss of financial assets in case you're the owner of an online shop.

Therefore, the development of a fraud detection system based on AI will increase precision of fraudulent financial transaction detection based on the history of financial operations of the user, location of transaction, size of transaction etc.

1.1 Purpose

1.1.1 Problem statement

The more time passes the more new users and financial transactions are processed by POS, online stores, banks etc. There are many mobile applications and web applications in the field of healthcare on the market today, so there is a need to clearly classify such applications, as well as to formulate the main problems they solve.

1.1.2 Stakeholders

Name	Description Concerns

Project team members End user	People who will implement the system System that processes financial transactions	Up-to-date programming tools and libs, frameworks, cloud deployment, accuracy and completeness of the requirements Potential performance issues (transactions should be processed fast to not impact general flow), data security, efforts to deploy application to environments.
Project manager	Project leader Team requests fulfillment, staffing, resource management	team
Project sponsor	The person who finances project	Budget

1.1.3 Non-functional requirements

- Security integration with other systems using SSO (Okta, Duo security, Microsoft authenticator) or OAuth token
- SLA > 90%
- Max. response time for transaction analysis - 2 sec.
- Server should be able to handle 5000 requests simultaneously ● DB

server must support transactions to handle financial operations correctly

- The application should not break other systems PCI DSS complicance

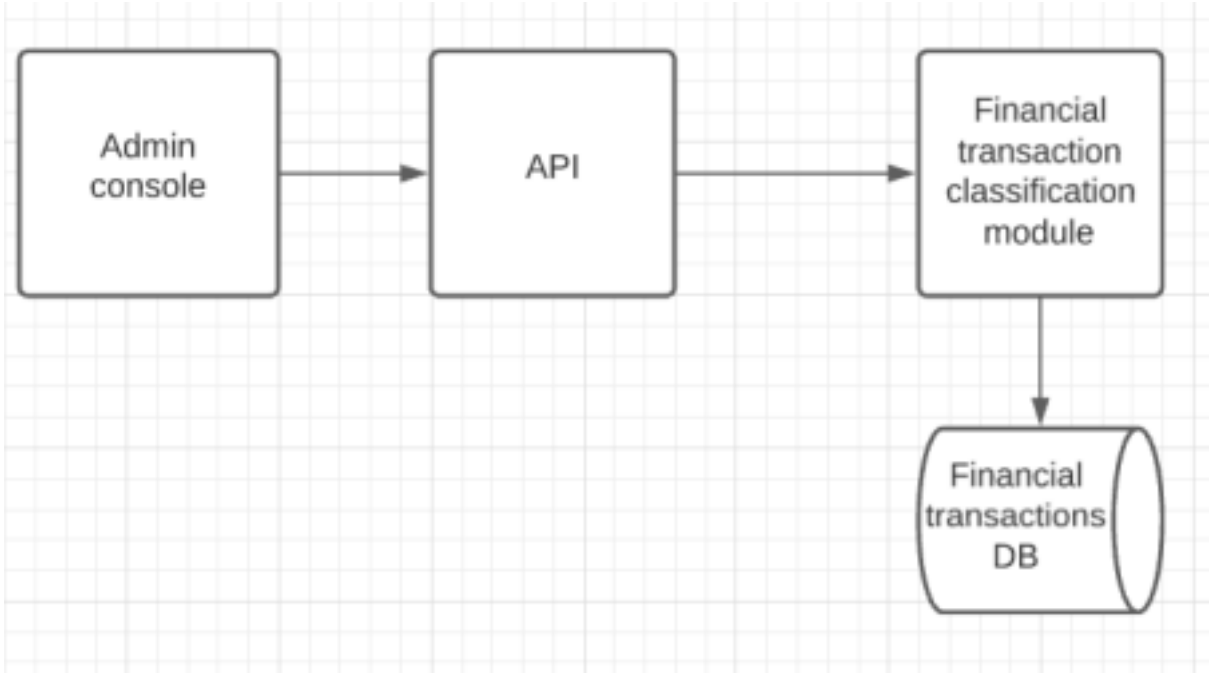
1.1.4 Glossary

Term Definition
Java high-level, class-based, object-oriented programming language
DB Database
REST a software architectural style that was created to guide the design and development of the architecture for the World Wide Web.
Stakeholder any person that is involved into project implementation or any person that is affected by product API Application Programming Interface ML Machine learning Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small- and large-scale projects.
SLA is a commitment between a service provider and a client. Particular aspects of the service – quality, availability, responsibilities – are agreed between the service provider and the service user.

2. Architecture overview

2.1 Architecture summary

The system is implemented using client-server architecture that consists of multiple microservices. Each microservice has its own scope of responsibilities and serves its own group of tasks. Each service is deployed as an independent application. Such approach allows to develop these services separately and use different developers or teams to work on each subpart simultaneously.

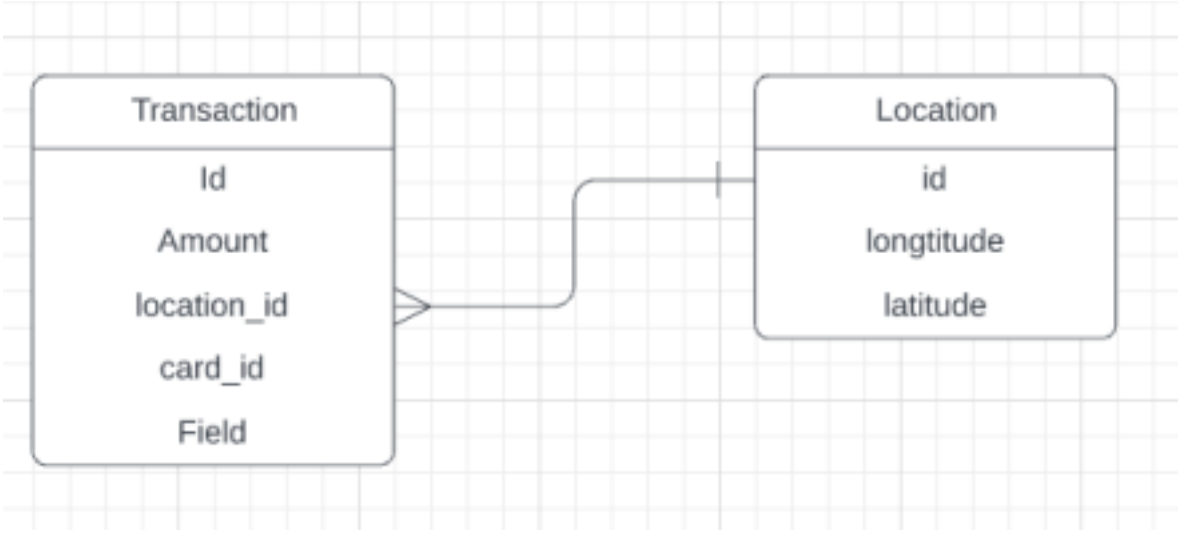


The system includes 4 components:

- Admin console - allows authorized user to track application health metrics,
- API - the access point of the system - allows the integrated system to send requests to fraudulent transaction detection system and receive response with the result of analysis
- Financial transaction classification module - the main module of the system, it performs analysis of each financial transaction and sends the response to the system that requested it.
- Financial transaction DB - DB that contains all financial transaction that will be used to teach the ML model

2.2 Database

Database should contain all data required for transaction analysis



Database can include more fields, tables data etc but the data above is a required minimum for analysis

2.3 Concerns and decisions

As a solution for implementation of API and admin console backend was chosen Java 11 programming language, as the most popular enterprise programming language. The main features that made an influence on this decision are:

- Cross-platform features - you can run java application on each machine that has jre or jdk (it has built-in jre) installed
- Huge amount of libraries and frameworks - team will be to cover every possible requirement to the system and speed up development with help of a big number of ready to use solutions.
- Wide community - there are numbers of java developers in the world that can help you with the issues you can possibly get

For web-oriented features I've chosen Spring Boot framework, Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run". It helps to speed up development and makes it easier to implement cloud-ready microservices.

As a solution for implementation of Financial transaction classification module was chosen python and flask framework. Python has a wide range of available tools and libraries for machine learning.

As a solution for admin console ui was chosen TypeScript and

Angular. TypeScript is a strongly typed programming language that builds on JavaScript, giving you better tooling at any scale. Angular is a TypeScript-based free and open-source web application framework led by the Angular Team at Google and by a community of individuals and corporations.

As a CI/CD solution was chosen Jenkins. Jenkins is an open source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery. It is a server-based system that runs in servlet containers such as Apache Tomcat. It supports version control tools, including AccuRev, CVS, Subversion, Git, Mercurial, Perforce, ClearCase and RTC, and can execute Apache Ant, Apache Maven and sbt based projects as well as arbitrary shell scripts and Windows batch commands. It is a pretty old solution but a classic one, it is still widely used by a huge number of software projects.

For development of application components I've used Jetbrains Intellij IDEA and VSCode. IntelliJ IDEA is an integrated development environment (IDE) written in Java for developing computer software. It is developed by JetBrains (formerly known as IntelliJ), and is available as an Apache 2 Licensed community edition, and in a proprietary commercial edition. Both can be used for commercial development. Even though IDEA is a commercial software with
pretty high license cost, they offer an amazing program for students - you can receive a free license until you will graduate (but of course you can't use it for commercial projects development).

As a DB server I've used PostgreSQL, also known as Postgres, is a free and open-source relational database management system (RDBMS) emphasizing extensibility and SQL compliance. PostgreSQL features transactions with Atomicity, Consistency, Isolation, Durability (ACID) properties, automatically updatable views, materialized views, triggers, foreign keys, and stored procedures. It is designed to handle a range of workloads, from single machines to data warehouses or Web services with many concurrent users. It is the default database for macOS Server and is

also available for Windows, Linux, FreeBSD, and OpenBSD.

3. Components

The Api component - spring boot application implemented using layered architecture and consists of three layers:

- Representation layer
- Business logic layer
- Data layer

The representation layer is implemented as a REST interface with JSON as a response format. This component is a main access point of the system, every possible operation will be pre-processed by API first and then redirected to service that will process it.

The financial transaction classification module - flask application that includes trained ML model.

- Representation layer
- Business logic layer
- Data layer

The representation layer is implemented as a REST interface with JSON as a response format. Data layer contains all data related to transactions that will be used in ML model training.

The financial transactions DB - database that contains all data required for ML model training,

Admin console - component used to track system health metrics and all other features available to administrators only (the list of features is extending).

4. Testing

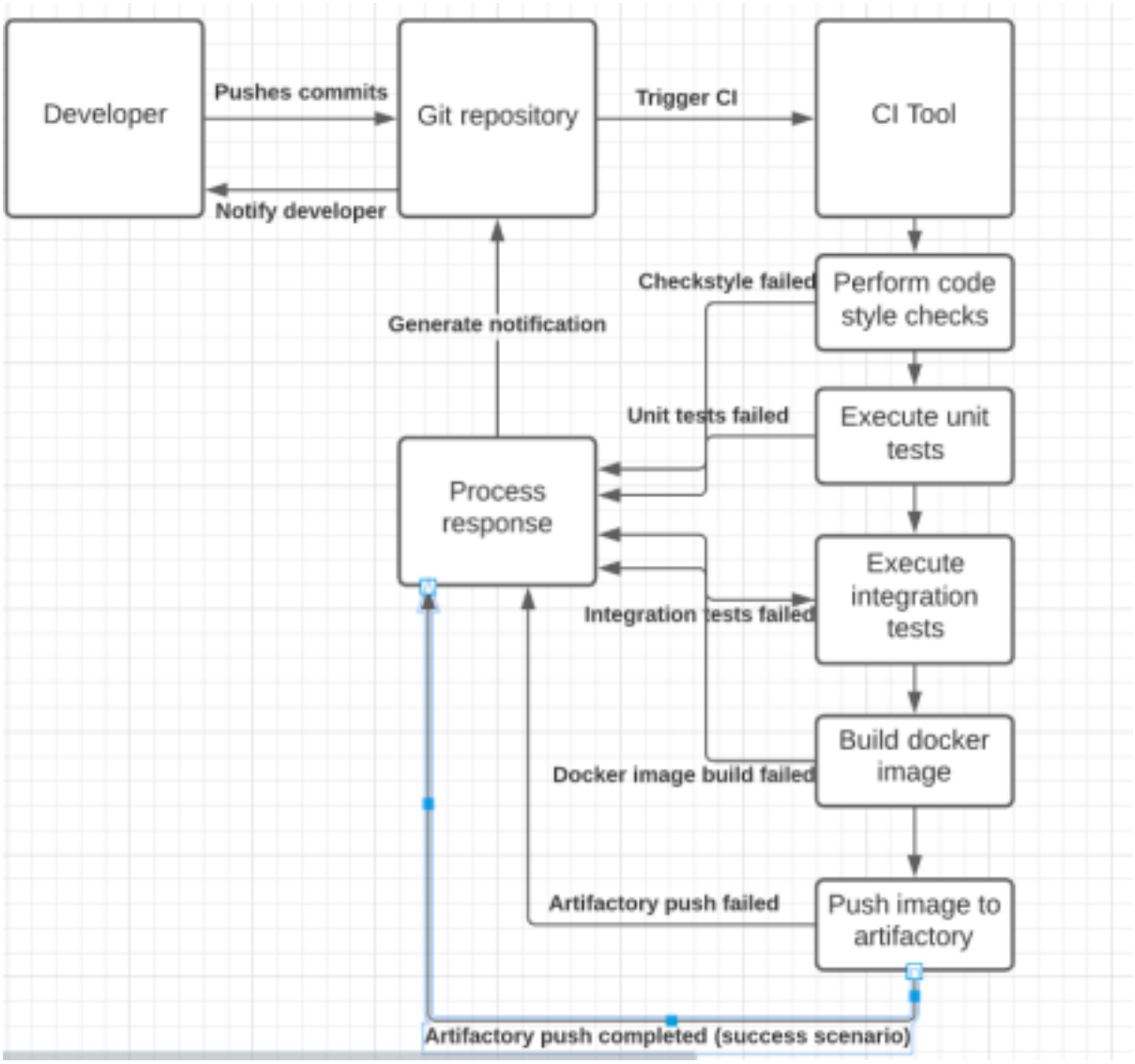
Testing should be performed in accordance with the pyramid of test automation. Each test should follow the FIRST principles. First principles of testing stand for:

- Fast - The developer shouldn’t hesitate to run the run the unit tests at any point of their development cycle, even if there are thousands

of unit tests. They should run and show you the desired output in a matter of seconds

- Isolated/Independent - For any given unit test, for its environment variables or for its setup. It should be independent of everything else should so that it results is not influenced by any other factor.
- Repeatable - tests should be repeatable and deterministic, their values shouldn't change based on being run on different environments; Each test should set up its own data and should not depend on any external factors to run its test
- Self-validating - you shouldn't need to check manually, whether the test passed or not;
- thorough - should cover all the happy paths; should cover all the happy paths; test for illegal arguments and variables; test for security and other issues; test for large values, what would a large input do their program; should try to cover every use case scenario and not just aim for 100% code coverage.

On the lowest level there are unit tests that should be executed on each CI/CD run to ensure that each commit in master branch can be compiled and new changes don't break existing functionality. Integration tests should cover as much as possible interactions between system components but at the same time number of tests will be smaller due to their nature (it takes more time to prepare resources for each of them, they take more time to execute). It is better to
execute integration tests on each build in CI/CD but at the same time to reduce build time we can limit it to execution on pull requests only. Api tests are executed as part of automation suite on a deployed application.


Developer
Pushes commits
Git repository
Trigger CI
CI Tool
Notify developer
Checkstyle failed
Perform code style checks
Generate notification
Unit tests failed
Execute unit tests
Process response
Execute integration tests
Integration tests failed
Build docker image
Docker image build failed
Artifactory push failed
Push image to artifactory
Artifactory push completed (success scenario)