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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Підвищення захищеності системи запобігання фродовим атакам за допомогою вдосконаленої комбінованої системи на основі SVM, логістичної регресії та вагового підходу»

Виконав: студент 2 курсу, групи КІТС-23м
спеціальності 125 — Кібербезпека та захист інформації
Освітня програма — Кібербезпека інформаційних технологій та систем
Карпілов В.І.

Керівник: професор д.т.н.
Яремчук Ю.Є.

«____» _____ 2024 р.

Опонент: к.т.н., доцент, доцент каф. ОТ
Черняк О.І

«____» _____ 2024 р.

Допущено до захисту
Голова секції УБ кафедри МБІС
Юрій ЯРЕМЧУК
«09» грудня 2024 р.

Вінниця ВНТУ — 2024

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

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

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС
_____ **Юрій ЯРЕМЧУК**

«20»вересня 2024 р.

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу студенту
Карпілову Віктору Івановичу

1 Тема роботи «Підвищення захищеності системи запобігання фродовим атакам за допомогою вдосконаленої комбінованої системи на основі SVM, логістичної регресії та вагового підходу»
керівник роботи професор д.т.н. Яремчук Ю.Є.,
затверджені наказом вищого навчального закладу від 17.09.2024 р. №310

2 Строк подання студентом роботи 29.11.2024 р.

3 Вихідні дані до роботи — стандарти, електронні джерела, підручники та наукові статті по темі, які стосуються теми магістерської кваліфікаційної роботи.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):Для досягнення мети було поставлено задачі: дослідити існуючі методи запобігання фродовим атакам; розробити архітектуру комбінованої системи захисту; реалізувати та протестувати розроблену систему; здійснити економічне обгрунтування. В першому розділі проведено аналіз теоретичних засад систем запобігання фродовим атакам та існуючих ріщень. В другому розділі розроблено архітектуру та алгоритми комбінованої системи захисту. В третьому розділі здійснено практичну реалізацію системи та її тестування. В четвертому розділі проаналізовано економічну доцільність розробки.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): В першому розділі магістерської роботи 7 рисунків, в другому розділі 2 рисунка, в третьому розділі 12 рисунків

6 Консультанти розділів роботи представлені в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Основна частина			
I	Яремчук Юрій Євгенович професор д.т.н.		
II	Яремчук Юрій Євгенович професор д.т.н.		
III	Яремчук Юрій Євгенович професор д.т.н.		
Економічна частина			
IV	Бурєннікова Наталія Вікторівна, професор кафедри ЕПВМ, д.е.н.		

7 Дата видачі завдання 20 вересня 2024 р.

8 Календарний план наведено в таблиці 2.

Таблиця 2 – Календарний план

№	Назва етапів виконання магістерської роботи	Строк виконання етапів роботи		Пр ка
1	Визначення напрямку магістерської роботи, формулювання теми	20.09.2024	31.09.2024	
2	Аналіз предметної області обраної теми	01.10.2024	15.10.2024	
3	Розробка роботи	16.10.2024	26.10.2024	
4	Написання магістерської роботи на основі розробленої теми	27.10.2024	15.11.2024	
5	Передзахист магістерської кваліфікаційної роботи	16.11.2024	24.11.2024	
6	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	27.11.2024	04.12.2024	
7	Захист магістерської кваліфікаційної роботи	11.12.2024	17.12.2024	

Студент Карпілов В.І.

Керівник роботи Яремчук Ю.Є.

АНОТАЦІЯ

УДК 004.056.5

Підвищення захищеності системи запобігання фродовим атакам за допомогою вдосконаленої комбінованої системи на основі SVM, логістичної регресії та вагового підходу. Магістерська кваліфікаційна робота зі спеціальності 125 - "Кібербезпека", освітня програма "Кібербезпека інформаційних технологій та систем". Вінниця: ВНТУ, 2024. 112 с.

На укр. мові. Бібліогр.: 40 назв; рис.: 31; табл. 8.

У магістерській кваліфікаційній роботі представлено розробку системи запобігання фродовим атакам з використанням комбінованої системи на основі SVM, логістичної регресії та вагового підходу.

В першому розділі роботи здійснено аналіз теоретичного матеріалу: досліджено методи машинного навчання для виявлення шахрайства, проведено аналіз існуючих систем захисту від фродових атак.

У другому розділі роботи описано розробку вдосконаленої комбінованої системи запобігання фродовим атакам та розроблено алгоритми її функціонування.

У третьому розділі роботи здійснено практичну реалізацію системи, проведено тестування та оцінку ефективності розробленого рішення. Реалізовано серверну частину системи та користувацький інтерфейс.

У четвертому розділі проаналізовано економічну доцільність розробки, продемонстровано її комерційний потенціал та можливості подальшого застосування.

Ключові слова: фродові атаки, запобігання шахрайству, підтримуючі векторні машини (SVM), логістична регресія, ваговий підхід, комбінована система, машинне навчання, кібербезпека, виявлення аномалій, фінансові транзакції.

ANNOTATION

Improving the security of a fraud prevention system using an improved combined system based on SVM, logistic regression and weighting approach. Master's qualification work in the specialty 125 - “Cybersecurity”, educational program “Cybersecurity of information technologies and systems”. Vinnytsia: VNTU, 2024. 112 c.

In Ukrainian. Bibliography: 40 titles; Figures: 31; Table 8.

The master's qualification work presents the development of a system for preventing fraud attacks using a combined system based on SVM, logistic regression and weighting approach.

The first chapter of the thesis analyzes the theoretical material: machine learning methods for fraud detection are studied, and existing systems for protecting against fraud attacks are analyzed.

The second section of the paper describes the development of an improved combined system for preventing fraud attacks and develops algorithms for its functioning.

The third section of the paper describes the practical implementation of the system, testing and evaluation of the effectiveness of the developed solution. The server part of the system and the user interface are implemented.

The fourth section analyzes the economic feasibility of the development, demonstrates its commercial potential and possibilities for further application.

Keywords: fraud attacks, fraud prevention, support vector machines (SVMs), logistic regression, weighted approach, combined system, machine learning, cybersecurity, anomaly detection, financial transactions.

ЗМІСТ

ВСТУП

Актуальність. На сьогоднішній день стрімкий розвиток цифрових технологій, особливо в фінансовому секторі, супроводжується зростанням загроз, пов'язаних із шахрайством. Фродові атаки у фінансових та інформаційних системах можуть спричиняти значні фінансові втрати, компрометацію особистих даних та завдавати шкоди репутації компаній. В умовах, коли кількість і складність фродових атак постійно збільшується, питання підвищення захищеності від них стає надзвичайно актуальним[1] .

Традиційні методи захисту часто виявляються недостатньо ефективними, адже вони не завжди здатні вчасно розпізнати складні та багатошарові атаки, які все частіше здійснюються кіберзлочинцями. Водночас методи на основі машинного навчання, зокрема підтримуючі векторні машини (SVM) та логістична регресія, демонструють перспективні результати в ідентифікації шахрайських дій завдяки здатності до аналізу великих масивів даних та виявлення прихованих закономірностей.

Проте ефективність окремих алгоритмів є обмеженою у випадках, коли шахрайство маскується під звичайні транзакції або коли необхідно враховувати різні вагові фактори ризику. Це обумовлює необхідність створення комбінованих моделей, що поєднують переваги кількох підходів. Впровадження комбінованої системи, яка поєднує SVM, логістичну регресію та ваговий підхід, дозволяє суттєво підвищити точність і надійність системи виявлення фроду, знижуючи кількість помилкових спрацювань і покращуючи адаптивність моделі до нових типів атак.

Таким чином, розробка ефективних методів протидії фродовим атакам за допомогою вдосконалених алгоритмів машинного навчання є актуальним і важливим завданням. Це дослідження спрямоване на створення більш захищеної та адаптивної системи, яка може сприяти зниженню рівня шахрайства в сучасних фінансових та інформаційних системах.

Зв’язок роботи з науковими програмами, планами, темами. Дисертаційна робота виконувалась згідно з планом науково-дослідних робіт кафедри

менеджменту та безпеки інформаційних систем Вінницького національного технічного університету.

Мета і задачі дослідження. Метою даного дослідження є розробка та впровадження вдосконаленої комбінованої системи для підвищення захищеності від фродових атак, що базується на використанні підтримуючих векторних машин (SVM), логістичної регресії та вагового підходу.

Об’єктом дослідження є системи запобігання фродовим атакам у фінансових та інформаційних середовищах, які використовують алгоритми машинного навчання для ідентифікації та блокування шахрайських транзакцій. Особливу увагу приділено комбінованим методам виявлення фроду, які базуються на поєднанні підтримуючих векторних машин (SVM), логістичної регресії та вагового підходу для підвищення ефективності захисту від кіберзагроз.

Досліджувані засоби — інструменти системної інженерії: Алгоритм підтримуючих векторних машин (SVM); інструменти системи прогнозування: інструменти представлення текстових даних в числовій формі, математичної статистики, класичного аналізу даних, машинного та глибинного навчання, великих даних; діаграми UML, нейронні мережі.

Предметом дослідження є методи підвищення ефективності систем запобігання фродовим атакам шляхом застосування комбінованих алгоритмів машинного навчання, зокрема підтримуючих векторних машин (SVM), логістичної регресії та вагового підходу. Увага зосереджена на аналізі ефективності цих методів, їхньої взаємодії у складі комбінованої моделі та оптимізації вагового підходу для покращення точності і надійності виявлення шахрайських транзакцій.

Методи дослідження. В роботі використовуються методи системного аналізу, методи моделювання, емпіричного аналізу, аналізу даних, теорії систем, машинного навчання та оптимізації.

Наукова новизна одержаних результатів складається з таких положень:

розроблена модель використовує переваги різних алгоритмів машинного навчання, що дозволяє ефективніше розпізнавати аномалії та знижувати ймовірність помилок у класифікації фродових і легітимних транзакцій;

запропонована система забезпечує більш високу точність у порівнянні з традиційними моделями, що ґрунтуються на одному алгоритмі, та дозволяє значно зменшити кількість хибнопозитивних і хибно-негативних результатів;

запропонована комбінована модель є інноваційним підходом до захисту фінансових систем і може сприяти підвищенню рівня безпеки інформаційних середовищ.

Практичне значення одержаних результатів. Отримані результати дослідження можуть бути використані для підвищення рівня кібербезпеки у фінансових системах, оптимізації роботи служб захисту від шахрайства та розробки нових рішень для автоматизованого виявлення загроз.

1 ТЕОРЕТИЧНІ ЗАСАДИ СИСТЕМ ЗАПОБІГАННЯ ФРОДОВИМ АТАКАМ

В даному розділі буде здійснено аналіз теоретичних засад побудови систем запобігання фродовим атакам. Буде розглянуто термінологію та визначення поняття фродової атаки, теоретичні принципи систем запобігання шахрайству та підходи до виявлення відхилень. Особливу увагу буде приділено аналізу існуючих рішень та методів, включаючи нейронні мережі, експертні системи, генетичні алгоритми, статистичний аналіз та інші підходи.

На основі проведеного аналізу буде здійснено порівняння існуючих рішень за ключовими характеристиками та сформовано висновки щодо їх ефективності та обмежень. Це дозволить обґрунтувати необхідність розробки вдосконаленої комбінованої системи та визначити вимоги до неї.

1.1 Термінологія та визначення поняття фродової атаки

Асоціація сертифікованих експертів з питань шахрайства (ACFE)[3] створила фундаментальний наріжний камінь для розуміння та боротьби з шахрайством у сучасних системах. По суті, ACFE визначає шахрайство як «використання своєї професії для особистого збагачення шляхом навмисного зловживання або застосування ресурсів чи активів організації, що працевлаштовує». Це визначення стало особливо актуальним у сучасному технологічному ландшафті, де шахрайська діяльність проникла в різні цифрові сфери, включаючи телекомунікаційні мережі, мобільний зв'язок, інтернет-банкінг та системи електронної комерції.

У контексті сучасних технологічних систем шахрайство проявляється в кількох різних формах. Шахрайство з кредитними картками є однією з найпоширеніших категорій, що трапляється як в офлайн, так і в онлайн середовищі. В офлайн-сценаріях шахраї зазвичай використовують викрадені фізичні картки на вітринах магазинів або в колл-центрах. Такі випадки часто виграють від механізмів швидкого виявлення, оскільки установи зазвичай можуть заблокувати картку до того, як відбудеться значна шахрайська активність. Однак

шахрайство в Інтернеті є більш складним завданням, оскільки зловмисникам потрібні лише реквізити картки, а не сама фізична картка. Цей вид шахрайства, що здійснюється за допомогою інтернет-транзакцій та покупок по телефону, особливо складно виявити та запобігти через відсутність вимог щодо фізичної перевірки.

Телекомунікаційне шахрайство є ще однією серйозною проблемою, яка проявляється у двох формах: шахрайство з підпискою та шахрайство, що накладається на підписку. Шахрайство з підпискою відбувається, коли особи отримують підписку на послуги, використовуючи фальшиві документи, не маючи наміру платити, часто із застосуванням схем навмисного створення безнадійних боргів. Накладне шахрайство, навпаки, передбачає несанкціоноване використання законних послуг, що, як правило, відображається у вигляді невідомих платежів у рахунках законних користувачів. Ця категорія охоплює різні складні технології, включаючи клонування мобільних телефонів, «примари» (коли маніпулюють мережами), інсайдерське шахрайство і «перекидання», коли на клонованих телефонах використовують фальшиві серійні номери, щоб приписувати послідовні дзвінки на різні легальні телефони.

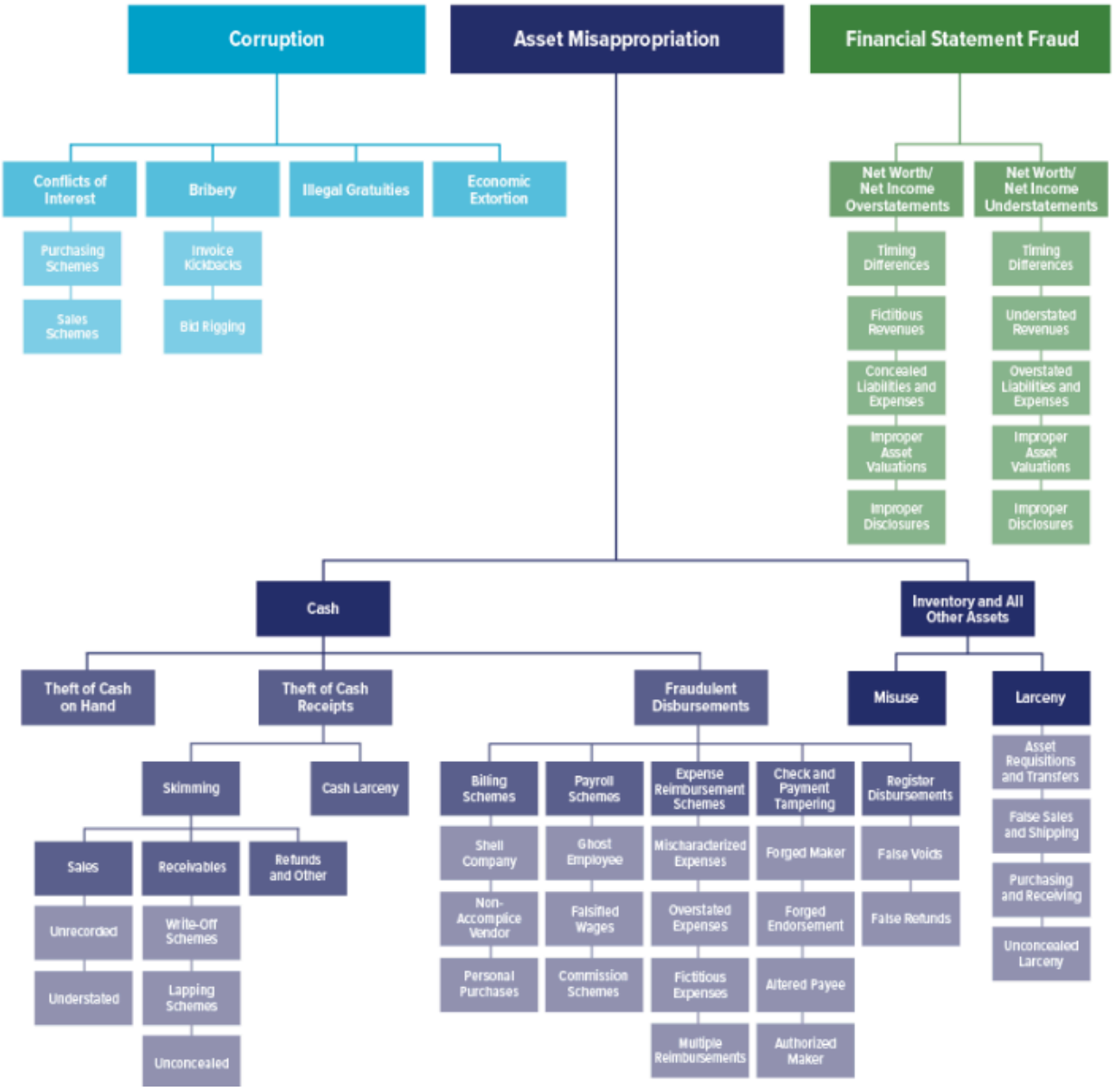


Рисунок 1.1 — Класифікація фродових атак згідно з ACFE[2]

Комп'ютерне вторгнення є третьою основною категорією шахрайства, визначеною ACFE, яка включає в себе як вторгнення з метою зловживання, так і вторгнення через аномалії. Зловмисні вторгнення включають в себе чітко визначені атаки, спрямовані на відомі вразливості системи, за впізнаваними шаблонами атак. Аномальні вторгнення, однак, ідентифікуються через відхилення від нормальних моделей використання системи, що охоплюють такі дії, як спроби злому, маскарадні атаки, витік інформації, відмова в обслуговуванні та різні форми зловмисного використання.

Система ACFE підкреслює важливість показників ефективності в системах виявлення шахрайства. Основна мета полягає в тому, щоб максимізувати правильні прогнози щодо шахрайства, зберігаючи при цьому прийнятний рівень хибних спрацьовувань. Цей делікатний баланс вимагає складних систем, здатних виявляти шахрайські дії швидко після їх вчинення, мінімізуючи помилкові спрацьовування, які можуть порушити законні бізнес-операції. Ключовими показниками ефективності є рівень хибних спрацьовувань, рівень виявлення шахрайства, точність виявлення та середній час виявлення.

Впровадження систем виявлення шахрайства стикається з кількома серйозними проблемами. Обмін ідеями щодо виявлення шахрайства часто обмежений через міркування безпеки, а обмеження конфіденційності даних можуть перешкоджати розвитку системи. Крім того, управління та аналіз великих масивів даних з одночасним розпізнаванням складних шаблонів є постійною технічною проблемою. Мабуть, найважливіше, що системи виявлення шахрайства повинні постійно адаптуватися до нових методів шахрайства, які постійно розвиваються.

Сучасні застосування системи ACFE дедалі більше інтегрують передові технології, включаючи інтелектуальний аналіз даних, статистичний аналіз, штучний інтелект і складні системи розпізнавання образів. Ці системи використовують моніторинг у реальному часі, автоматизовані оповіщення, комплексні системи оцінки ризиків і протоколи швидкого реагування. Концепція підкреслює важливість системної архітектури, яка підтримує модульність, масштабованість і безперешкодну інтеграцію з існуючими бізнес-системами.

Дивлячись у майбутнє, фреймворк ACFE продовжує розвиватися у відповідь на нові виклики. Оскільки методи шахрайства стають все більш витонченими, а системи — все більш складними, механізми запобігання та виявлення повинні адаптуватися відповідно. Ця еволюція вимагає постійного розвитку методів виявлення, систем реагування та методів запобігання, зберігаючи при цьому крихкий баланс між безпекою та операційною ефективністю.

Вимоги до даних є важливим аспектом системи, що підкреслює необхідність всебічного збору даних при одночасному забезпеченні їхньої якості та безпеки. Питання конфіденційності стають все більш важливими, особливо в світлі розширення глобального зв'язку і підвищення складності систем. Концепція містить рекомендації щодо розподілу ресурсів, моніторингу системи та вимог до технічного обслуговування, гарантуючи, що системи запобігання шахрайству залишатимуться ефективними впродовж тривалого часу.

Таким чином, концепція ACFE забезпечує всеосяжну основу для розуміння та боротьби з шахрайством у сучасних технологічних системах. Її принципи продовжують спрямовувати розвиток механізмів запобігання шахрайству, допомагаючи організаціям захищати свої активи та підтримувати цілісність системи в умовах дедалі складнішого цифрового ландшафту.

1.2 Теоретичні принципи систем запобігання шахрайству

Сучасні системи запобігання шахрайству працюють на кількох фундаментальних принципах[3], які поєднують статистичний аналіз, машинне навчання та розпізнавання шаблонів. По суті, ці системи спрямовані на максимізацію рівня виявлення шахрайства при мінімізації помилкових спрацьовувань, що математично виражається як:

$$\text{Мета оптимізації} = \max(TPR) \text{ при збереженні } FPR \leq \alpha,$$

де TPR (True Positive Rate) — це показник виявлення шахрайства;

FPR (False Positive Rate) — показник хибних спрацьовувань;

α — допустимий поріг хибних спрацьовувань.

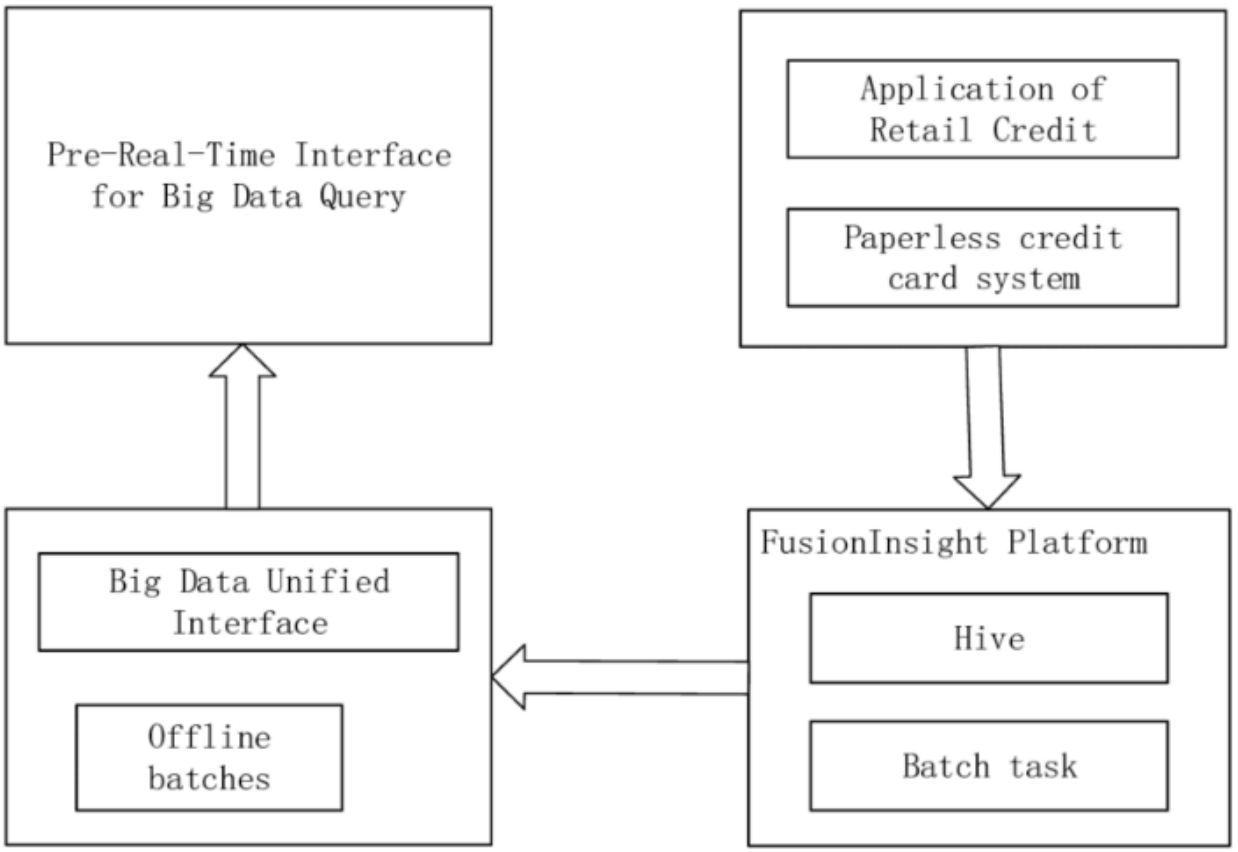


Рисунок 1.2 — Схема роботи анти фродової системи[5]

1.3 Підходи до виявлення відхилень

В основі методу виявлення відхилень у запобіганні шахрайству лежить статистичний принцип, згідно з яким шахрайські дії значно відхиляються від нормальних моделей поведінки. Використовуючи методи неконтрольованого навчання, ці системи моделюють базовий розподіл поведінки B(x) і позначають спостереження, які перевищують поріг відхилення τ. Спостереження x позначається як потенційно шахрайське, коли

$$|x - \mu| > \tau\sigma, \tag{1.1}$$

де μ — середнє значення нормальної поведінки;

σ — стандартне відхилення;

τ — зазвичай встановлюється між 2 і 3 для збалансованої чутливості.

Типовими статистичними алгоритмами для виявлення відхилень є Z-score метод[6], Modified Z-score[7], Local Outlier Factor[8], Elliptic Envelope (Відстань

Махаланобіса)[9], DBSCAN[10], Gaussian Mixture Models (GMM)[11] (Ймовірність належності до компоненти).

Z-score — вимірює скільки стандартних відхилень точка віддалена від середнього. Припускає нормальний розподіл даних.

$$Z = \frac{x - \mu}{\sigma} \tag{1.2}$$

Modified Z-score — Покращена версія Z-score, використовує медіану замість середнього.

$$Modified\,Z - score = \frac{0.6745(x - median)}{MAD} \tag{1.3}$$

$$MAD = median\{|x_i - \tilde{x}|\} \tag{1.4}$$

Local Outlier Factor — Порівнює локальну щільність точки з щільністю її сусідів.

$$LOF(x_i) = \frac{1}{k} * \frac{\sum_j d(x_i, x_j)}{\sum_j \sum_l d(x_j, x_l)} \tag{1.5}$$

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) — це алгоритм кластеризації, заснований на щільності точок даних.

Основні параметри:

— ε (eps) — радіус околу точки;

— MinPts — мінімальна кількість точок в околі.

Принцип роботи:

— для кожної точки визначається її ε-окіл;

— точки класифікуються на 3 типи:

1) Core points: мають ≥ MinPts сусідів;
2) Border points: мають < MinPts сусідів, але є в ε-околі core point;
3) Noise points: всі інші.

1.3.1 Реалізація нейронних мереж

Нейронні мережі для виявлення шахрайства зазвичай використовують багатошарові архітектури з наступними ключовими компонентами:

— вхідний шар: Характеристики транзакцій (f_1, f_2, ..., f_n);

— приховані шари: Нелінійні функції перетворення;

— вихідний шар: Оцінка ймовірності шахрайства P(шахрайство|транзакція).

Мережа обчислює ймовірність шахрайства за допомогою функції активації:

$$P(\text{шахрайство}) = \sigma(\Sigma_i w_i x_i + b),$$

де σ — є сигмоїдною функцією;

w_i — навчені ваги;

x_i — вхідні ознаки;

b — член зсуву.

Результат роботи нейронної мережі можна подати у вигляді:

$$z = (\sum_{i=1}^{n} x_i * w_i) + b, \quad (1.6)$$

де z — вихідне значення;

x — значення нейрону;

w — вага;

b — байяс.

1.3.2 Теоретична основа системи CARDWATCH

CARDWATCH[11] інтегрує аналіз часових патернів з нейронними мережами для створення профілів поведінки, орієнтованих на конкретного клієнта. Для аналізу послідовностей транзакцій використовується підхід ковзного вікна:

$$\text{Risk\_Score} = \Sigma_{t=1}^{t} (w_t \times \delta(t_t, p_t)),$$

де t — індекс транзакції;

w_t — вага для часової позиції;

$\delta(t_t, p_t)$ вимірює відхилення від прогнозованого шаблону;

p_t — вивчений нормальний шаблон.

1.3.3 Теорія виявлення комп'ютерних вторгнень

Системи виявлення вторгнень використовують як сигнатурні, так і аномальні методи виявлення. Математична модель поєднує в собі умовні ймовірності:

$$P(\text{вторгнення}|\text{поведінка}) = P(\text{поведінка}|\text{вторгнення}) \times P(\text{вторгнення}) / P(\text{поведінка})$$

Система підтримує як профілі нормальної поведінки N(b), так і сигнатури атак A(s), запускаючи оповіщення в разі виявлення будь-якої з них:

— поведінка відповідає відомим шаблонам атак: similarity(current_behavior, A(s)) > threshold_s;

— поведінка відхиляється від нормального профілю: deviation(current_behavior, N(b)) > threshold_n.

1.3.4 Фреймворк експертних систем

Експертні системи для виявлення шахрайства використовують представлення знань на основі правил у поєднанні з механізмами виведення для виявлення підозрілих дій. Ці системи імітують процес прийняття рішень експертами з виявлення шахрайства.

Структура бази знань. База знань складається з трьох основних компонентів:

— База фактів: Поточні транзакції та історичні дані

— База правил: Визначені експертами правила виявлення шахрайства

— Механізм логічної обробки

Правила структуровані в наступному форматі:

$$\text{Rule}(n) = \text{IF } (\text{Condition}_1 \text{ AND Condition}_2 \ldots \text{AND Condition}_n) \text{ THEN } (\text{Risk\_Score} = \text{Calculate\_Risk}), \quad (1.7)$$

де $\text{Calculate\_Risk} = \Sigma(\text{Rule\_Weight} \times \text{Condition\_Match\_Score})$.

Розрахунок достовірності правила. Впевненість кожного правила динамічно оновлюється на основі його продуктивності:

$$\text{Rule\_Confidence} = (\text{Successful\_Detections} / \text{Total\_Applications}) \times (1 - \text{False\_Positive\_Rate}) \times \text{Historical\_Effectiveness} \quad (1.8)$$

$$\text{Historical\_Effectiveness} = (\text{Past\_Success} \times 0.7 + \text{Recent\_Success} \times 0.3) \quad (1.9)$$

Процес виведення експертної системи. Система оцінює транзакції шляхом послідовного застосування правил:

Ризик_копіювання_транзакції = Σ(Впевненість_правила × Відповідність_правила × Суворість_правила) (1.10)

Transaction_Risk = Σ(Rule_Confidence × Rule_Match × Rule_Severity)

Final_Expert_Score = Transaction_Risk × (1 + Historical_Risk_Factor) × Pattern_Multiplier (1.11)

де Pattern_Multiplier = 1 + Σ(Matched_Known_Patterns × Pattern_Weight).

1.3.5 Система міркувань на основі моделей

Міркування на основі моделей створюють абстрактні моделі шахрайської поведінки і порівнюють поточну діяльність з цими моделями.

Модель переходу станів. Система відстежує стани та переходи транзакцій:

Current_State = { Transaction_Sequence: [T_1, T_2, ..., T_n], Time_Deltas: [Δt_1, Δt_2, ..., Δt_n], Amount_Pattern: [A_1, A_2, ..., A_n] }, (1.12)

State_Risk = Compare_State(Current_State, Fraud_Model_States) (1.13)

де Compare_State = Weighted_Match(Sequence) × Time_Pattern_Score × Amount_Pattern_Score.

Співставлення поведінкових моделей. Система підтримує поведінкові моделі для різних типів шахрайства:

— Pattern_Match_Score = Σ(Feature_Weight × Feature_Distance);

— Feature_Distance = |Current_Feature – Model_Feature| / Feature_Standard_Deviation;

— Behavioral_Risk = Pattern_Match_Score × Model_Confidence × (1 + Temporal_Factor).

Еволюція моделі. Моделі постійно оновлюються на основі нових шаблонів шахрайства:

Model_Update = {

Weight_Adjustment: Previous_Weight + Learning_Rate × Detection_Success,

Pattern_Refinement: (Old_Pattern × 0.8 + New_Pattern × 0.2),

Confidence_Update: (Historical_Confidence × 0.7 + Recent_Performance × 0.3)
}

Інтеграція експертних та модельних систем. Обидва підходи доповнюють один одного через:

$$Combined\_Risk\_Score = (Expert\_System\_Weight \times Final\_Expert\_Score) + (Model\_Based\_Weight \times Behavioral\_Risk), \quad (1.13)$$

де Expert_System_Weight = 0.6 + (Expert_System_Confidence × 0.4);

Model_Based_Weight = 0.4 + (Model_Performance_Score × 0.6).

Процес прийняття рішення. Остаточне визначення шахрайства поєднує обидва підходи:

$$Fraud\_Probability = Normalize(Combined\_Risk\_Score) \times System\_Confidence \times (1 + Context\_Risk\_Factor) \quad (1.14)$$

$$Alert\_Decision = Fraud\_Probability > Dynamic\_Threshold, \quad (1.15)$$

де Dynamic_Threshold = Base_Threshold × (1 + Risk_Tolerance) × (1 + Time_Factor).

Показники ефективності. Ефективність системи вимірюється за допомогою:
System_Effectiveness = {
Detection_Rate: True_Positives / (True_Positives + False_Negatives),
Precision: True_Positives / (True_Positives + False_Positives),
F1_Score: 2 × (Precision × Detection_Rate) / (Precision + Detection_Rate)
}

Цей гібридний підхід використовує сильні сторони як експертних систем (чіткі правила та людський досвід), так і міркувань на основі моделей (розпізнавання образів та поведінковий аналіз) для створення надійної системи виявлення шахрайства. Експертна система забезпечує негайну реакцію на основі відомих шахрайських схем, тоді як компонент на основі моделей адаптується до нових схем шахрайства, що розвиваються.

1.3.6 Генетичні алгоритми

Генетичні алгоритми[13] пропонують інноваційний підхід до виявлення шахрайства, імітуючи біологічну еволюцію. Ці алгоритми підтримують

популяцію потенційних рішень (правил виявлення шахрайства), які еволюціонують з часом через процеси природного відбору.

Система починає з набору базових правил і поступово вдосконалює їх. Цей підхід особливо ефективний для адаптації до нових шахрайських схем, що робить його цінним для виявлення нових схем шахрайства, які розвиваються.

1.3.7 Статистичний аналіз

Статистичні методи[14] є основою багатьох систем виявлення шахрайства, оскільки вони виявляють аномалії в даних про транзакції. Ці методи особливо цінні, оскільки вони можуть працювати як з міченими, так і з немічеными даними.

До традиційних статистичних підходів належать:

— аналіз стандартного відхилення;

— ковзні середні;

— декомпозиція часових рядів;

— аналіз розподілу.

Коли транзакція значно відхиляється від нормальних шаблонів, її позначають для перевірки. Основною перевагою статистичних методів є їхня міцна математична основа та можливість інтерпретації.

1.3.8 Приховані марковські моделі (HMM)

HMM[15] особливо ефективні в моделюванні послідовних моделей поведінки клієнтів. Вони працюють шляхом створення моделей переходу станів нормальної поведінки, визначення ймовірностей послідовності виявлення незвичайних переходів станів.

Вірогідність стану визначається як:

$$P(s_{i1}, s_{i2},...,s_{ik}) = P(s_{ik} \mid s_{ik-1})P(s_{ik-1} \mid s_{ik-2})...P(s_{i2} \mid s_{i1})P(s_{i1}),$$

де s_{ij} — стан j в момент часу i; (1.16)

$P(s_{ik} \mid s_{ik-1})$ — ймовірність переходу в стан k, знаючи попередній стан k-1.

Тоді ймовірність спостереження це:

$$P(O|\lambda) = \Sigma P(O|S,\lambda)P(S|\lambda), \quad (1.17)$$

де O — послідовність спостережень;

λ — параметри моделі (A, B, π);

S — послідовність станів.

Ймовірність переходу це:

$$a\_ij = P(s\_t+1 = j \mid s\_t = i), \quad (1.18)$$

де a_ij — ймовірність переходу зі стану i в стан j;

s_t — стан в момент часу t.

Ймовірність емісії (ймовірність того, що в певному прихованому стані модель згенерує конкретне спостереження):

$$b\_j(k) = P(o\_t = k \mid s\_t = j), \quad (1.19)$$

де b_j(k) — ймовірність спостереження k в стані j;

o_t — спостереження в момент t;

s_t — стан в момент t.

1.3.9 Машини опорних векторів (SVM)

SVM[16] є потужним інструментом для виявлення шахрайства завдяки своїй здатності створювати оптимальні межі рішень, обробляти дані високої розмірності.

Метод заснований на графічному представленні даних та ділення їх лінією, площиною, гіперплощиною в залежності від розмірності даних.

В основі лежить визначення цієї лінії, площини або гіперплощини.

Представляються вони ядрами:

— лінійне: $K(x,y) = xy$;

— поліноміальне: $K(x,y) = (xy + 1)^d$.

— RBF: $K(x,y) = \exp(-\|x-y\|^2/2\sigma^2)$,

де x,y — значення відповідного розкладу на параметри;

d — степінь полінома в ядрі для підвищення складності моделі;

σ — параметр ширини ядра.

1.3.10 Байєсівські мережі

Байєсівські мережі[17] забезпечують імовірнісну основу для виявлення шахрайства за допомогою моделювання взаємозв'язків між змінними, оновлення переконань на основі нових доказів, ефективної обробки невизначеності

Байєсівські мережі є моделлю для представлення ймовірнісних відношень між змінними через направлений ациклічний граф.

Засновані та теоремі Байєса:

$$P(A|B) = P(B|A) * P(A) / P(B) \quad (1.20)$$

Звідси:

$$P(X_1,..., X_n) = \prod_i P(X_i|Parents(X_i) \quad (1.21)$$

1.3.11 Автокодери

Автокодери[18] працюють за принципом вивчення нормальних шаблонів транзакцій, реконструкції вхідних даних, позначення транзакцій, які не відповідають вивченим шаблонам.Їхньою головною перевагою є здатність виявляти нові шаблони шахрайства

1.3.12 Обґрунтування вибору характеристик для порівняння методів

При порівнянні методів виявлення фродових атак важливо оцінити їх за характеристиками, які безпосередньо впливають на ефективність впровадження та експлуатації в реальних умовах [19]. Обрані характеристики можна розділити на три основні групи:

Продуктивність:

- Швидкість навчання та класифікації - критичні для обробки транзакцій в реальному часі
- Можливості паралелізації - важливі для масштабування системи
- Вимоги до обчислювальних ресурсів (CPU, RAM) - визначають вартість інфраструктури

Практичність використання:

- Простота перенавчання - необхідна для адаптації до нових видів шахрайства

- Інтерпретованість результатів - важлива для аналізу та обґрунтування рішень
- Вимоги до розміру навчальної вибірки - впливають на можливість впровадження
- Стійкість до шуму в даних - визначає надійність роботи в реальних умовах

Економічні аспекти:

- Складність впровадження - впливає на початкові витрати
- Можливість роботи в реальному часі - критична для бізнес-процесів
- Вартість впровадження та підтримки - визначає загальну економічну ефективність

Саме ці характеристики дозволяють комплексно оцінити придатність методів для практичного використання в системах виявлення шахрайства.

Таблиця 1.1 — Порівняння методів виявлення фродової атаки

Метод	Аномалії	Нейромережі	HMM	SVM	Баєсівські мережі	Експертні системи	Генетичні алгоритми	Автокодери
Швидкість навчання	Висока	Низька	Середня	Середня	Висока	Дуже висока	Низька	Середня
Швидкість класифікації	Висока	Середня	Низька	Висока	Середня	Висока	Низька	Середня
Паралелізація	Висока	Дуже висока	Середня	Висока	Середня	Низька	Висока	Висока
Перенавчання	Легке	Складне	Середнє	Середнє	Легке	Легке	Складне	Середнє
Інтерпретованість	Висока	Низька	Середня	Середня	Висока	Дуже висока	Низька	Низька
Розмір вибірки	Малий	Великий	Середній	Середній	Малий	Дуже малий	Середній	Великий
Стійкість до шуму	Середня	Висока	Середня	Висока	Середня	Низька	Середня	Висока
CPU	Низька	Висока	Середня	Середня	Низька	Низька	Висока	Висока
RAM	Низька	Висока	Середня	Середня	Низька	Низька	Середня	Висока
Складність впровадження	Низька	Висока	Середня	Середня	Середня	Низька	Висока	Висока
Real-time	Так	Обмежено	Так	Так	Так	Так	Обмежено	Обмежено

Вартість впровадження	Низька	Висока	Середня	Середня	Середня	Низька	Середня	Висока
Вартість підтримки	Низька	Висока	Середня	Середня	Низька	Низька	Середня	Висока

1.4 Аналіз існуючих рішенньь

На даний момент усі системи запобігання фродовим атакам можна поділити на дві великі категорії. А саме: системи на основі правил та системи на основі штучного інтелекту. Також існують деякі комбіновані системи. Основними характеристиками є: спектр можливих використаннь, ціна, надійність та вимоги до кваліфікації. Оскільки надійність та точність даних систем є конфіденційною інформацією кожної конкретної системи — у подальшому розгляді вона не буде прийматись до уваги

1.4.1 Кардвотч

Рисунок 1.3 — логотип кардвотч[18]

— спеціалізація на роздрібній торгівлі;

— використовується в основному для офлайн бізнесів;

— висока ціна використання;

— ручне налаштування.

1.4.2 Страйп радар

Stripe Radar — це інтегрована система запобігання шахрайству в платіжній платформі Stripe, яка використовує алгоритми машинного навчання, навчені на мільярдах глобальних транзакцій. Вона забезпечує виявлення шахрайства в режимі реального часу, аналізуючи шаблони в широкій мережі продавців Stripe, автоматично адаптуючись до конкретних бізнес-шаблонів і галузевих норм.

Система поєднує в собі складні моделі машинного навчання з правилами, що налаштовуються, дозволяючи компаніям тонко налаштовувати свою стратегію запобігання шахрайству, зберігаючи при цьому безперебійний клієнтський досвід для законних транзакцій.

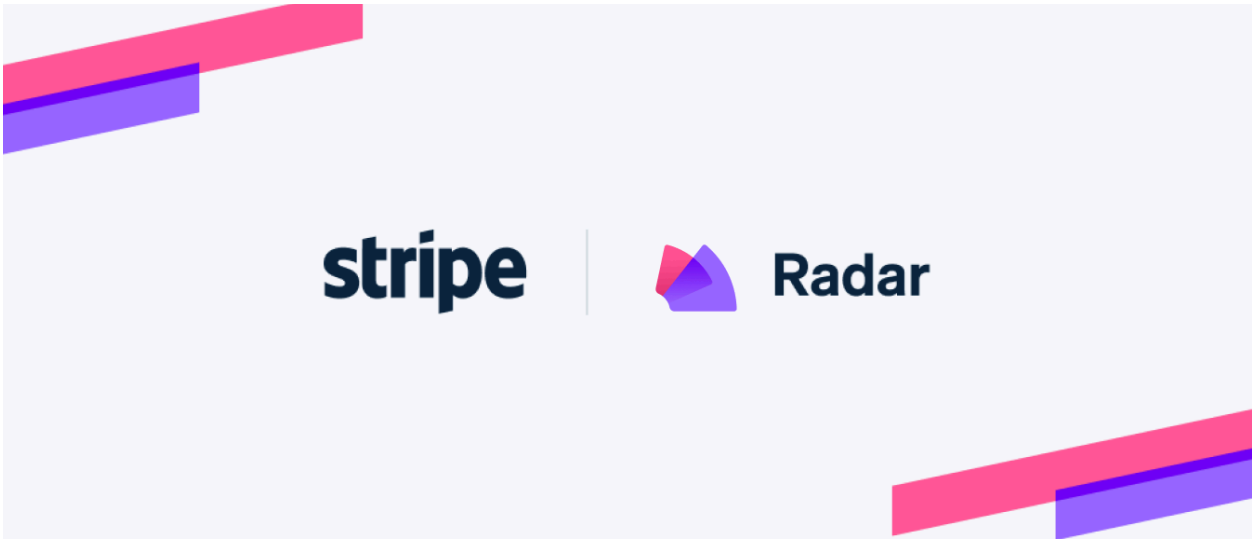


Рисунок 1.4 — логотип страйп радар[20]

Особливості:

— широке використання для онлайн бізнесів;

— використовує нейронні мережі;

— надає власні правила для використання та широкі можливості налаштування;

— адаптивні моделі нейронних мереж.

1.4.3 HICE Актімайз

NICE Actimize[21] — це комплексне рішення для боротьби з фінансовими злочинами та комплаєнсом, що використовується найбільшими фінансовими установами по всьому світу. Платформа поєднує в собі передову аналітику, штучний інтелект і традиційні підходи, засновані на правилах, для виявлення і запобігання різним видам фінансових злочинів, включаючи шахрайство, відмивання грошей і маніпулювання ринком. Її перевага полягає в здатності обробляти величезні обсяги фінансових даних у режимі реального часу, зберігаючи при цьому відповідність нормативним вимогам і надаючи детальні аудиторські сліди для проведення розслідувань.

Рисунок 1.5 — логотип HICE ACTIMIZE

Особливості:

— використовується великими банківськими компаніями;

— використовує комбінований підхід;

— спеціалізація на протистоянні “відмиванні” коштів (AML);

— моніторинг транзакцій в реальному часі.

1.4.4 SAS Система

SAS Fraud Management[22] являє собою складне поєднання традиційних систем, заснованих на правилах, і сучасних підходів машинного навчання, розроблених спеціально для фінансових установ і страхових компаній. Система відмінно справляється з обробкою складних потоків даних в режимі реального часу, пропонуючи можливості адаптивного навчання, які допомагають організаціям випереджати нові шахрайські схеми. Її потужний аналітичний

механізм може обробляти тисячі транзакцій в секунду, зберігаючи високу точність виявлення шахрайства та мінімізуючи помилкові спрацьовування.

Рисунок 1.6 — логотип SAS Fraud Management

Особливості:

— використовується страховими компаніями;

— використовує гібридний підхід;

— можливості адаптивного навчання;

— фокус на регуляторному секторі.

1.4.5 Інструменти AWS

AWS Fraud Detector[23] — це хмарний сервіс виявлення шахрайства від Amazon, який використовує ту саму технологію, що й Amazon для захисту своїх масштабних операцій електронної комерції. Сервіс дозволяє організаціям створювати власні моделі виявлення шахрайства за допомогою машинного навчання, не вимагаючи глибокої експертизи в галузі ML. Він пропонує безперешкодну інтеграцію з іншими сервісами AWS і може бути швидко масштабований для обробки різних робочих навантажень, що робить його особливо привабливим для організацій, які вже працюють в екосистемі AWS, зберігаючи при цьому економічно ефективну модель ціноутворення за принципом «платити по мірі використання».

Рисунок 1.7 — логотип AWS Fraud Detector

Особливості:

— використовує Amazon SageMaker;

— має наявні приклади та датасети;

— вимагає дуже високої кваліфікації спеціаліста.

1.5 Порівняльний аналіз існуючих рішень

Очевидно що для великого бізнесу з доступом до кваліфікованих співробітників існує доволі широкий спектр можливих рішень, проте якщо проводити аналіз з оглядом на малий бізнес без доступу до кваліфікованого персоналу — доступним є не такий великий спектр можливих рішень.

Для порівняльного аналізу було обрано наступні характеристики: доступність для малого бізнесу, вартість впровадження, необхідність спеціалістів, інтеграція з локальними системами, вимоги до інфраструктури (табл. 1.1).

Таблиця 1.2 Порівняння існуючих рішення

Характеристика	CARDWATCH	Stripe Radar	NICE Actimize	SAS	AWS
Доступність для малого бізнесу	Обмежена	Середня	Низька	Низька	Середня

Вартість впровадження	Висока	Низька	Дуже висока	Дуже висока	Середня
Необхідність спеціалістів	Так	Ні	Так	Так	Так
Інтеграція з локальними системами	Складна	Проста	Складна	Складна	Середня
Вимоги до інфраструктури	Високі	Низькі	Високі	Високі	Середні

1.6 Висновки до розділу

У першому розділі було проведено комплексний аналіз теоретичних засад систем запобігання фродовим атакам. Розглянуто базову термінологію та визначення поняття фродової атаки, що заклало фундамент для подальшого дослідження. Детально проаналізовано теоретичні принципи систем запобігання шахрайству та різноманітні підходи до виявлення відхилень.

Особливу увагу приділено дослідженню різних методів виявлення шахрайства, включаючи нейронні мережі, експертні системи, генетичні алгоритми, статистичний аналіз, приховані марковські моделі, машини опорних векторів, байєсівські мережі та автокодери. Кожен з цих методів має свої переваги та обмеження, що впливає на їх ефективність у конкретних сценаріях застосування.

Проведений аналіз існуючих комерційних рішень, включаючи CARDWATCH, Stripe Radar, NICE Actimize, SAS System та інструменти AWS, виявив їх основні характеристики, переваги та недоліки. Порівняльний аналіз показав, що існуючі рішення, хоча і є ефективними, часто виявляються занадто дорогими або складними для впровадження в малому та середньому бізнесі.

Дослідження теоретичних основ та практичних реалізацій систем запобігання фродовим атакам дозволило виявити потребу в розробці більш доступного та гнучкого рішення, яке б поєднувало переваги різних підходів. Зокрема, перспективним напрямком є комбінування методів машинного навчання (SVM та логістичної регресії) з елементами блокчейн-технології для забезпечення надійної системи виявлення шахрайства.

На основі проведеного аналізу можна стверджувати, що розробка вдосконаленої комбінованої системи, яка враховує особливості та обмеження існуючих рішень, є актуальним напрямком досліджень. Така система повинна забезпечувати високу точність виявлення шахрайства при збереженні доступності та простоти впровадження для широкого кола користувачів.

2 РОЗРОБКА ЗАПОБІГАННЯ ФРОДОВИМ АТАКАМ ЗА ДОПОМОГОЮ ВДОСКОНАЛЕНОЇ КОМБІНОВАНОЇ СИСТЕМИ НА ОСНОВІ SVM, ЛОГІСТИЧНОЇ РЕГРЕСІЇ ТА ВАГОВОГО ПІДХОДУ

В даному розділі буде представлено розробку системи запобігання фродовим атакам за допомогою вдосконаленої комбінованої системи. Буде розглянуто особливості використання SVM та логістичної регресії для виявлення шахрайських транзакцій, а також описано проектування архітектури системи. Значну увагу буде приділено розробці алгоритмів роботи системи та обґрунтуванню вибору технічних засобів для її реалізації.

2.1 Особливості запобігання фродовим атакам за допомогою вдосконаленої комбінованої системи на основі SVM

Розвиток технологій та постійна еволюція шахрайських схем створюють нові виклики у сфері запобігання фродовим атакам. Традиційні методи виявлення шахрайства часто виявляються недостатньо ефективними через їх обмежену здатність до адаптації та високу вартість впровадження. У відповідь на ці виклики пропонується вдосконалена комбінована система, що базується на використанні машин опорних векторів (SVM) як основного методу класифікації.

Вибір SVM як базового методу обумовлений його потужними математичними властивостями та здатністю ефективно працювати з нелінійними залежностями у даних. Доповнення SVM логістичною регресією та системою вагових коефіцієнтів дозволяє створити більш гнучку та адаптивну систему виявлення шахрайства. Особлива увага приділяється механізмам роботи з незбалансованими даними, що є критичним аспектом у контексті виявлення фродових транзакцій.

Обґрунтування вибору SVM як базового методу

Support Vector Machine демонструє значні переваги при вирішенні задач класифікації шахрайських транзакцій. Ключовою особливістю є ефективність роботи у просторах високої розмірності, що особливо важливо при аналізі транзакцій з великою кількістю характеристик. Використання різних ядерних

функцій дозволяє ефективно працювати з нелінійними залежностями, які характерні для фродових патернів. Принцип максимізації зазору між класами забезпечує оптимальну границю розділення, що підвищує якість класифікації.

У контексті виявлення шахрайства дані характеризуються значною незбалансованістю, де кількість легітимних транзакцій суттєво перевищує кількість шахрайських. SVM ефективно вирішує цю проблему через можливість встановлення різних вагових коефіцієнтів для класів. Метод опорних векторів дозволяє зосередитись на граничних випадках, що критично важливо при виявленні аномалій. Адаптивне налаштування параметрів моделі забезпечує оптимальну чутливість системи.

Значною перевагою SVM є висока здатність до узагальнення, що дозволяє ефективно класифікувати нові, раніше невідомі патерни шахрайства. Це досягається завдяки принципу структурної мінімізації ризику, який закладено в основу алгоритму. SVM здатна виявляти складні взаємозв'язки в даних та застосовувати отримані знання до нових випадків, що особливо важливо в умовах постійної еволюції шахрайських схем.

SVM демонструє високу стійкість до перенавчання завдяки використанню регуляризації та принципу максимізації зазору. Це дозволяє моделі зберігати ефективність при роботі з новими даними, навіть якщо вони дещо відрізняються від тренувальних. Механізм м'якої границі (soft margin) забезпечує баланс між точністю на тренувальних даних та здатністю до узагальнення, що критично важливо для практичного застосування в системах виявлення шахрайства.

2.1.1 Особливості комбінованого підходу

Комбінований підхід базується на паралельному використанні двох методів класифікації. Кожен метод має свої переваги: SVM ефективно працює з нелінійними залежностями та добре узагальнює дані, тоді як логістична регресія забезпечує стабільні імовірнісні оцінки та простіше навчання. Паралельне використання цих методів дозволяє компенсувати індивідуальні недоліки кожного підходу та підвищити загальну надійність системи.

Фінальна оцінка транзакції формується через зважену комбінацію результатів обох класифікаторів. Вагові коефіцієнти визначають вплив кожної моделі на кінцеве рішення. Початкові значення ваг встановлюються на основі історичної ефективності кожного методу, з можливістю подальшого автоматичного коригування на основі якості прийнятих рішень.

Система використовує два рівні порогових значень для класифікації транзакцій. Пороги визначаються на основі статистичного аналізу історичних даних та постійно коригуються відповідно до поточної ситуації. Це забезпечує гнучкість системи та її адаптацію до змін у патернах шахрайства.

Процес прийняття рішень включає три основні сценарії:

1) блокування підозрілих транзакцій при перевищенні високого порогу;

2) позначення транзакцій для додаткової перевірки при середньому рівні ризику;

3) автоматичне прийняття транзакцій з низьким рівнем ризику.

Система враховує додаткові фактори, такі як історичний профіль користувача та поточні тренди шахрайства. Кожне рішення супроводжується відповідним рівнем сповіщень для команди безпеки та зберігається для подальшого аналізу та вдосконалення системи.

2.2 Проектування архітектури системи запобігання фродовим атакам

Система повинна забезпечувати високу доступність та масштабованість для обробки великої кількості транзакцій в режимі реального часу. Архітектура базується на мікросервісному підході[24], де кожен модуль є незалежним компонентом з чітко визначеними інтерфейсами взаємодії. Всі модулі мають бути слабко зв'язаними для забезпечення можливості незалежного масштабування та оновлення.

Була розробленна архітектура системи, що складається з наступних модулів.

API шар повинен забезпечувати стандартизований інтерфейс для взаємодії із зовнішніми системами, викристовуючи REST[25] протокол. Основна задача

цього компонента — ізоляція системи та зручний інтерфейс використання системи користувачем.

Шар попередньої обробки даних повинен забезпечувати надійне отримання та валідацію вхідних даних транзакцій, використовуючи механізми буферизації та чергування. Основна задача цього компонента — гарантована доставка даних в систему та їх первинна валідація, забезпечуючи стійкість до пікових навантажень.

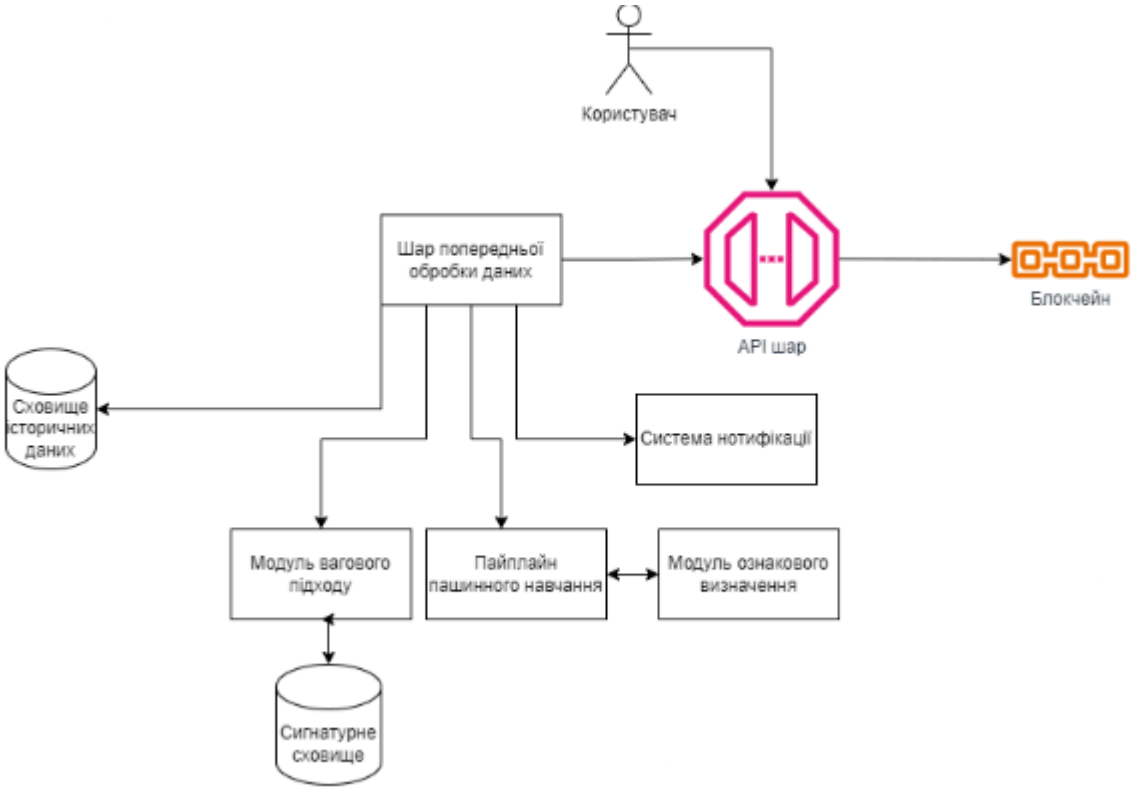


Рисунок 2.1 — Архітектура системи запобігання фродовим атакам

Блокчейн модуль представляє собою зв’язок з існуючими блокчейн системами заради отримання додаткової інформації про транзакції [26] з ціллю валідації та верифікації [27] .

Система нотифікації повинна забезпечувати гнучку систему сповіщень про підозрілі транзакції, використовуючи різні канали комунікації та рівні пріоритетності. Основна задача цього компонента — своєчасне інформування відповідальних осіб про потенційні загрози та забезпечення швидкого реагування.

Модуль ознакового визначення повинен забезпечувати генерацію та обробку ознак транзакцій в режимі реального часу, використовуючи оптимізовані алгоритми обчислень. Основна задача цього компонента — підготовка даних для моделей машинного навчання та забезпечення якісних вхідних характеристик.

Пайплайн машинного навчання [28] повинен забезпечувати паралельну роботу SVM та логістичної регресії, використовуючи механізми балансування навантаження та кешування. Основна задача цього компонента — ефективне застосування моделей машинного навчання та забезпечення їх оптимальної продуктивності.

Модуль вагового підходу повинен забезпечувати комбінування результатів різних моделей, використовуючи налаштовувані вагові коефіцієнти та пороги прийняття рішень. Основна задача цього компонента — формування фінального рішення щодо транзакції на основі зважених результатів всіх моделей.

Сховище історичних даних сховище історичних аналізів та тимчасових даних.

Сигнатурне сховище сховище калькульованих сигнатур.

2.3 Розробка алгоритму запобігання фродовим атакам

Розробка ефективного алгоритму запобігання фродовим атакам є критичним етапом у створенні системи виявлення шахрайства. Основною метою розробки є створення надійного та ефективного механізму, який би забезпечував високу точність виявлення шахрайських транзакцій при збереженні прийнятного рівня помилкових спрацювань.

Запропонований алгоритм базується на комбінованому підході, який об'єднує переваги машин опорних векторів (SVM) та логістичної регресії. Такий підхід дозволяє компенсувати недоліки кожного окремого методу та забезпечити більш надійну класифікацію транзакцій. Важливим аспектом алгоритму є використання системи вагових коефіцієнтів, яка дозволяє динамічно адаптувати вплив кожної моделі на фінальне рішення.

Алгоритм обробки транзакції для виявлення шахрайства

Крок 1. Початок:

Отримання транзакції на вхід системи

Крок 2. Попередня валідація:

Попередня валідація транзакції (перевірка формату даних)

Крок 3. Валідація даних:

Перевірка валідності даних — узгодження нетворку, за наявності, з токеном. Валідація даних згідно бізнес логіки (перевірка наявності підписки, використання сторонніх засобів)

Якщо НІ

Крок 4.1 Помилка

Якщо ТАК

Крок 4.1 Криптографічний підпис

Перевірка криптографічного підпису транзакції

Якщо ТАК

Крок 5.1 Отримання інформації з блокчейну

Якщо НІ

Крок 5.1 Перевірка наявності даних в системі

Крок 5.2 Перевірка отриманого результату

Перевірка успішності отримання даних

Якщо НІ

Крок 6.1 Помилка

Крок 6.2 Кінець

Якщо ТАК

Крок 7. Перевірка наявності історичних даних

Якщо ТАК

Крок 8.1 Отримання історичних даних

Якщо НІ

Крок 9. Перехід до формування вектору ознак вектору ознак

Формування вектору ознак F

Крок 10. Класифікація SVM моделлю:

Отримання P_SVM (ймовірності)

Крок 11. Збереження результату

Збереження результату P_SVM для забезпечення збереження мінімально-корисної сигнатури користувача

Крок 12. Перевірка порогового значення P_SVM

Якщо ТАК

Крок 13.1 Логування проміжного результату та позначення транзакції як небезпечної

Крок 13.2 Кінець

Якщо НІ

Крок 13.1 Класифікація логістичною регресією

Отримання P_Reg

Крок 14 Перевірка порогового значення P_Reg

Якщо ТАК

Крок 15.1 Збереження результуючих даних

Крок 15.2 Логування проміжного результату та позначення транзакції як небезпечної

Крок 15.3 Кінець

Якщо НІ

Крок 15.1 Розрахунок зваженої оцінки

Розрахунок зваженої оцінки, отримання S_Res

Крок 16. Перевірка порогового значення

Якщо ТАК

Крок 17.1 Логування проміжного результату та позначення транзакції як небезпечної

Крок 17.2 Кінець

Якщо НІ

17.1 Збереження результуючих даних

18. Кінець обробки

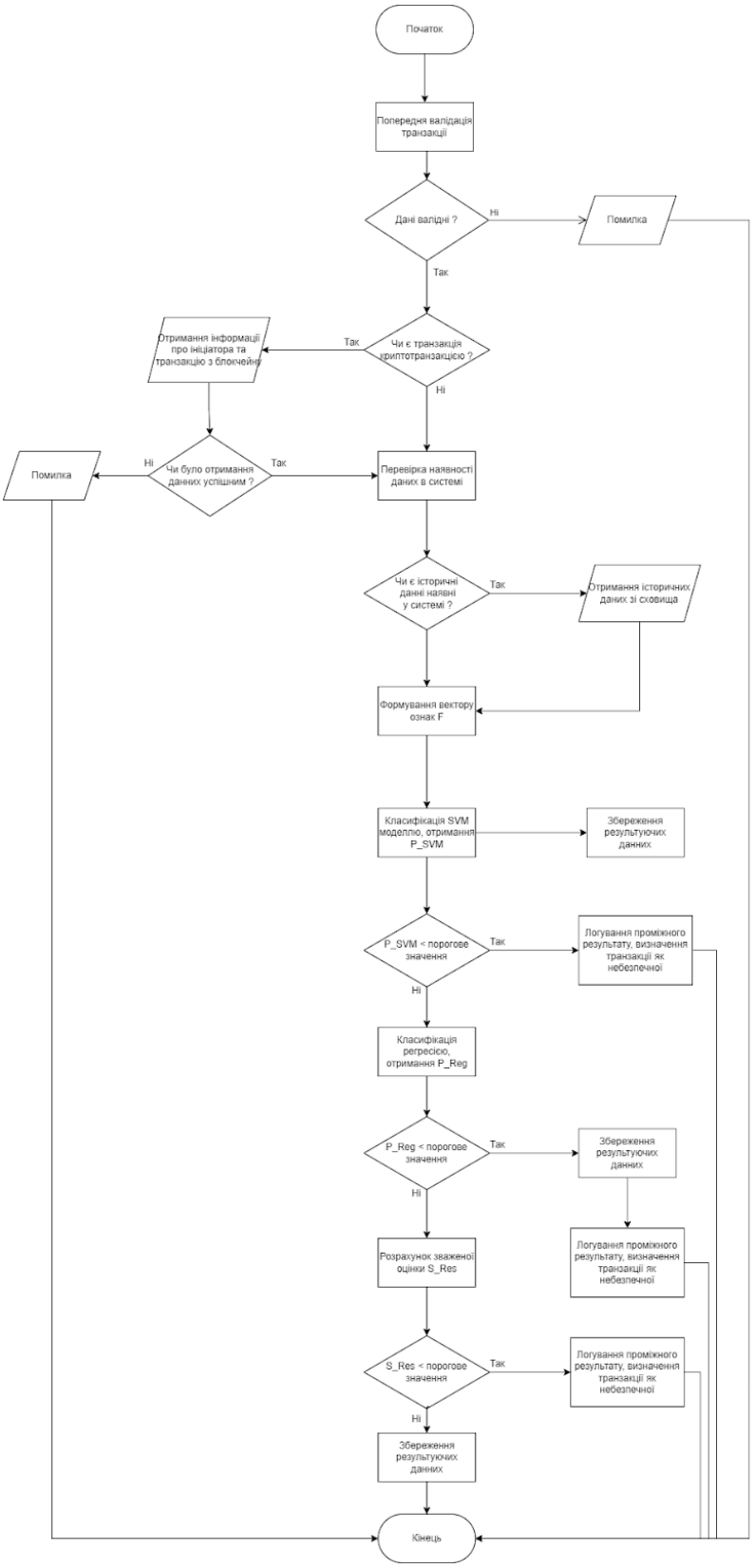


Рисунок 2.2 — алгоритм аналізу транзакції

2.4 Висновки до розділу 2

У другому розділі представлено розробку вдосконаленої комбінованої системи запобігання фродовим атакам, яка базується на використанні машин опорних векторів (SVM), логістичної регресії та вагового підходу. Запропоноване рішення враховує недоліки існуючих систем та пропонує інноваційний підхід до виявлення шахрайських транзакцій.

Ключовою відмінністю запропонованого алгоритму є динамічне налаштування ваг для кожної моделі на основі їх ефективності у виявленні конкретних типів шахрайства. Додатково, використання трирівневої системи порогових значень для класифікації транзакцій (автоматичне прийняття, додаткова перевірка, блокування) дозволяє знизити кількість помилкових спрацювань порівняно з бінарною класифікацією, що використовується в більшості існуючих рішень. Впроваджений механізм адаптивного коригування порогових значень забезпечує кращу пристосованість системи до нових патернів шахрайства.

Детально розглянуто особливості використання SVM як базового методу класифікації, обґрунтовано його переваги у контексті виявлення шахрайства, зокрема здатність до роботи з незбалансованими даними та високу стійкість до перенавчання. Представлено комбінований підхід, який доповнює SVM логістичною регресією та системою вагових коефіцієнтів, що дозволяє підвищити точність класифікації та зменшити кількість помилкових спрацювань.

Розроблено архітектуру системи, яка включає дев'ять ключових компонентів: API Layer, Data Ingestion Layer, Blockchain network, Alert Management System, Feature Engineering Module, ML Pipeline, Weighted Decision Module, History Storage та Signature Storage. Така структура забезпечує необхідну гнучкість, масштабованість та надійність системи.

Запропоновано детальний алгоритм обробки транзакцій, який включає етапи валідації даних, feature engineering, класифікації за допомогою двох моделей та прийняття зваженого рішення. Особливу увагу приділено механізмам адаптивного налаштування порогових значень та системі прийняття рішень, що дозволяє системі ефективно адаптуватися до змін у патернах шахрайства.

Результати теоретичного аналізу та проектування системи вказують на потенційну ефективність запропонованого рішення для виявлення фродових атак. Комбінований підхід дозволяє досягти балансу між точністю класифікації та обчислювальною складністю, що робить систему придатною для практичного впровадження в умовах обмежених ресурсів.

3 РЕАЛІЗАЦІЯ ПРОГРАМНИХ ЗАСОБІВ

В даному розділі буде представлено практичну реалізацію розробленої системи запобігання фродовим атакам. Буде здійснено опис та аналіз вхідних даних, реалізацію серверної частини системи та розробку користувацького інтерфейсу. Окрему увагу буде приділено тестуванню системи та оцінці її ефективності на реальних даних. За результатами тестування буде проведено порівняльний аналіз з існуючими рішеннями та зроблено висновки щодо практичної цінності розробленої системи.

3.1 Опис та аналіз вхідних даних

В рамках дослідження для розробки системи запобігання фродовим атакам використовується набір даних банківських транзакцій Fraudulent Transactions Data[29], який містить інформацію про фінансові операції за певний період часу. Кожен запис у наборі даних представляє окрему транзакцію та містить детальну інформацію про її характеристики, що дозволяє здійснювати глибокий аналіз та виявлення потенційних шахрайських операцій.

Датасет складається з записів транзакцій, де кожен запис характеризується часовим кроком, що дозволяє відстежувати послідовність операцій. Один часовий крок відповідає одній годині, що надає можливість аналізувати патерни активності користувачів та виявляти аномальну поведінку в часі.

Кожна транзакція в наборі даних класифікується за одним з п'яти типів: PAYMENT (платіж), TRANSFER (переказ), CASH_OUT (зняття готівки), CASH_IN (внесення готівки) та DEBIT (дебетова операція). Такий розподіл дозволяє враховувати специфіку різних типів операцій при аналізі на предмет шахрайства.

Важливою характеристикою транзакції є її сума (amount), яка варіюється від невеликих повсякденних платежів до значних грошових переказів. Аналіз розподілу сум транзакцій показує значну варіативність, що відображає реальну картину фінансових операцій.

Для кожної транзакції зберігається інформація про початковий та кінцевий баланс як відправника (oldbalanceOrg, newbalanceOrig), так і отримувача коштів (oldbalanceDest, newbalanceDest). Ця інформація є критично важливою для виявлення підозрілих операцій, оскільки дозволяє відстежувати аномальні зміни балансів та невідповідності у фінансових потоках.

Кожен учасник транзакції ідентифікується унікальним ідентифікатором: nameOrig для відправника та nameDest для отримувача. Ці ідентифікатори дозволяють відстежувати взаємозв'язки між користувачами та виявляти підозрілі схеми передачі коштів.

Особливо важливим аспектом датасету є наявність мітки isFraud для кожної транзакції, яка вказує, чи була операція визначена як шахрайська (1) або легітимна (0). Ця інформація використовується як цільова змінна при навчанні моделей машинного навчання.

Загальний обсяг набору даних складає 6,3 мільйона транзакцій, що забезпечує достатню кількість прикладів для навчання моделей. При цьому важливою особливістю є значна незбалансованість класів — частка шахрайських транзакцій складає лише близько 0,17% від загального обсягу. Така незбалансованість є типовою для задач виявлення шахрайства та потребує спеціальних підходів при навчанні моделей.

Статистичний аналіз датасету показує, що розподіл сум транзакцій має правосторонню асиметрію, що є характерним для фінансових даних. Також спостерігається значна варіація в балансах користувачів, що відображає реальну різноманітність фінансових можливостей клієнтів банку.

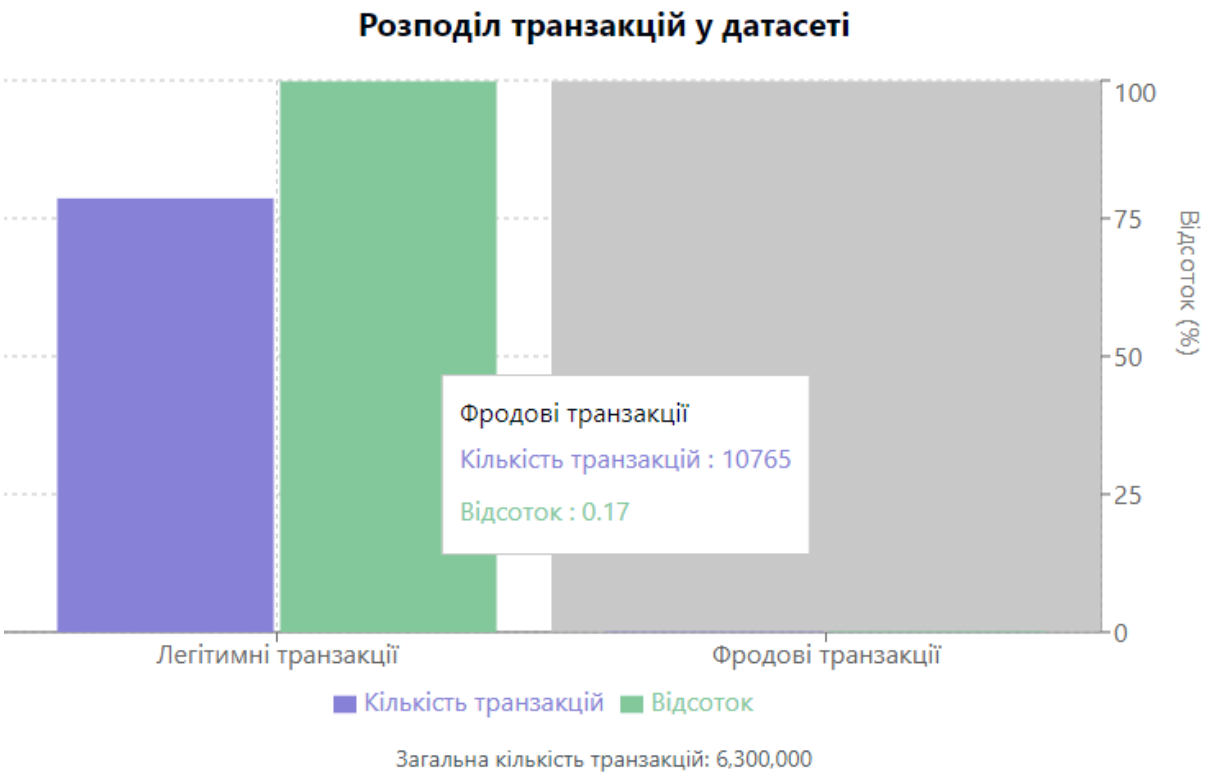


Рисунок 3.1 — розподіл фродових транзакцій у датасеті

Важливою характеристикою набору даних є наявність часової складової, що дозволяє аналізувати не лише окремі транзакції, але й послідовності операцій, виявляючи підозрілі патерни в часовому розрізі. Це особливо важливо для виявлення складних схем шахрайства, які можуть розгортатися протягом певного періоду часу.

3.1.1 Підготовка та очищення даних

У процесі підготовки даних було проведено комплексний аналіз вхідного набору даних та реалізовано ряд кроків для забезпечення якості даних для подальшого аналізу. При обробці транзакційних даних особлива увага приділялася збереженню цілісності фінансової інформації та виявленню потенційних аномалій.

Першим етапом стала перевірка на наявність пропущених значень у ключових полях. Виявлено, що поля, які відповідають за баланси рахунків, мали незначну кількість пропусків. Враховуючи специфіку банківських операцій, було

прийнято рішення заповнити пропуски нульовими значеннями, оскільки це може відповідати реальній ситуації з новими або неактивними рахунками.

Наступним кроком стала нормалізація числових значень. Для полів, пов'язаних із сумами транзакцій та балансами рахунків, було застосовано StandardScaler, що дозволило привести всі значення до єдиного масштабу. Особлива увага приділялася полю amount, де через значну правосторонню асиметрію розподілу було застосовано логарифмічну трансформацію.

У процесі очищення даних також було виявлено та оброблено аномальні значення. Транзакції з надзвичайно великими сумами (що перевищують 99.9 перцентиль розподілу) були позначені як потенційні викиди та потребували додаткового аналізу. Крім того, було проведено валідацію послідовності балансів, щоб виявити некоректні зміни балансу в рамках однієї транзакції.

3.1.2 Аналіз розподілу класів та балансування даних

Аналіз розподілу класів у наборі даних виявив значну незбалансованість між легітимними та шахрайськими транзакціями. З загальної кількості 6,3 мільйона транзакцій лише 0,17% були позначені як шахрайські, що створює серйозний виклик для навчання моделей машинного навчання.

Для вирішення проблеми незбалансованості було застосовано комбінований підхід. По-перше, використано метод SMOTE (Synthetic Minority Over-sampling Technique)[30] для генерації синтетичних прикладів шахрайських транзакцій. Цей метод дозволяє створювати нові приклади меншинного класу, зберігаючи при цьому статистичні характеристики оригінальних даних.

Додатково було застосовано Random Under-sampling[31] для більшинного класу, що дозволило зменшити загальний обсяг даних та наблизити співвідношення класів до більш збалансованого стану. В результаті було досягнуто співвідношення приблизно 1:10 між шахрайськими та легітимними транзакціями в навчальному наборі даних.

3.2 Реалізація серверної частини системи

В якості основного фреймворку для розробки API було обрано FastAPI[32]. Цей вибір обумовлений тим, що FastAPI надає можливість створювати високопродуктивні асинхронні веб-додатки з використанням сучасних можливостей Python. Важливою перевагою FastAPI є автоматична генерація OpenAPI[33] документації, що значно спрощує процес інтеграції та тестування. Фреймворк також забезпечує вбудовану валідацію даних через Pydantic[34], що робить роботу з вхідними даними більш надійною та безпечною.

Для взаємодії з базою даних використовується SQLAlchemy[35] — потужний інструмент об'єктно-реляційного відображення. SQLAlchemy надає гнучкий та функціональний підхід до роботи з базами даних, дозволяючи абстрагуватися від конкретної СУБД та працювати з даними на рівні Python-об'єктів. Це особливо важливо в контексті розробки системи виявлення фроду, де потрібна ефективна обробка та аналіз великих обсягів транзакційних даних.

В якості системи управління базами даних було обрано SQLite[36]. Це рішення дозволило забезпечити необхідний рівень надійності та продуктивності без надмірних витрат на розгортання та обслуговування окремого сервера баз даних. SQLite забезпечує атомарність транзакцій та надійне зберігання даних, при цьому залишаючи можливість легкого переходу на більш потужні рішення, такі як PostgreSQL, при зростанні навантаження на систему.

Для забезпечення високої продуктивності веб-серверу використовується Uvicorn — легкий та швидкий ASGI-сервер[37]. Uvicorn відмінно інтегрується з FastAPI та забезпечує ефективну обробку HTTP-запитів[38], що критично важливо для системи реального часу, якою є система виявлення фродових транзакцій.

Ключовим компонентом системи є бібліотека машинного навчання scikit-learn. Вона забезпечує необхідний функціонал для реалізації алгоритмів класифікації, попередньої обробки даних та оцінки якості моделей. Інтеграція

scikit-learn з іншими компонентами системи дозволяє ефективно виконувати аналіз транзакцій та виявляти потенційні шахрайські операції.

Обраний технологічний стек формує єдину, добре інтегровану систему, де кожен компонент виконує свою роль, забезпечуючи необхідну функціональність та продуктивність. Всі використані технології мають активну спільноту розробників та детальну документацію, що значно спрощує процес розробки та подальшої підтримки системи. Важливо відзначити, що обрані технології дозволяють легко масштабувати систему при збільшенні навантаження та додавати нову функціональність без значних змін в архітектурі.

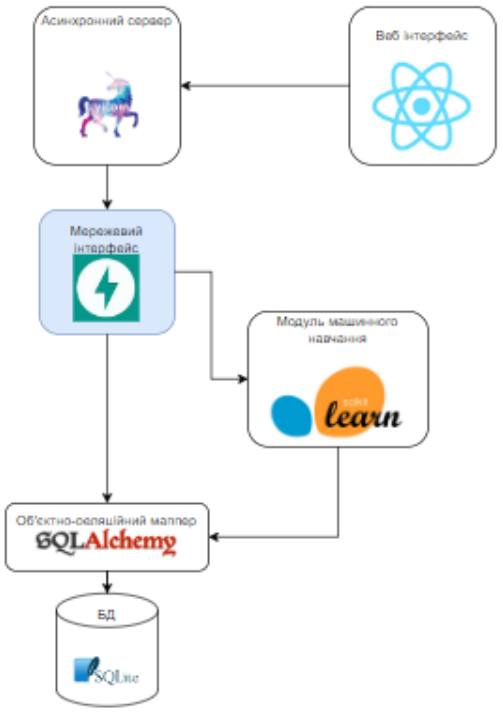


Рисунок 3.2 — архітектура програмної реалізації

3.3 Розробка користувацького інтерфейсу

При розробці користувацького інтерфейсу системи запобігання фродовим атакам основним пріоритетом була простота та інтуїтивність використання при збереженні всієї необхідної функціональності. Інтерфейс реалізовано за допомогою бібліотеки React з використанням сучасних підходів до розробки веб-додатків.

Головний інтерфейс системи розділений на чотири основні функціональні сторінки, кожна з яких відповідає за певний аспект роботи з системою.

Статусу додатку відображає загальну інформацію про поточний стан системи.

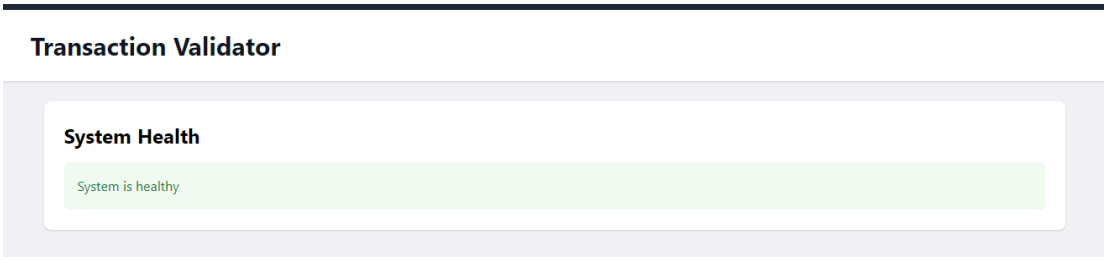


Рисунок 3.3 — інтерфейс сторінки статусу додатку

Сторінка модулю завантаження транзакцій надає інтерфейс для поповнення бази даних користувацькими транзакціями.

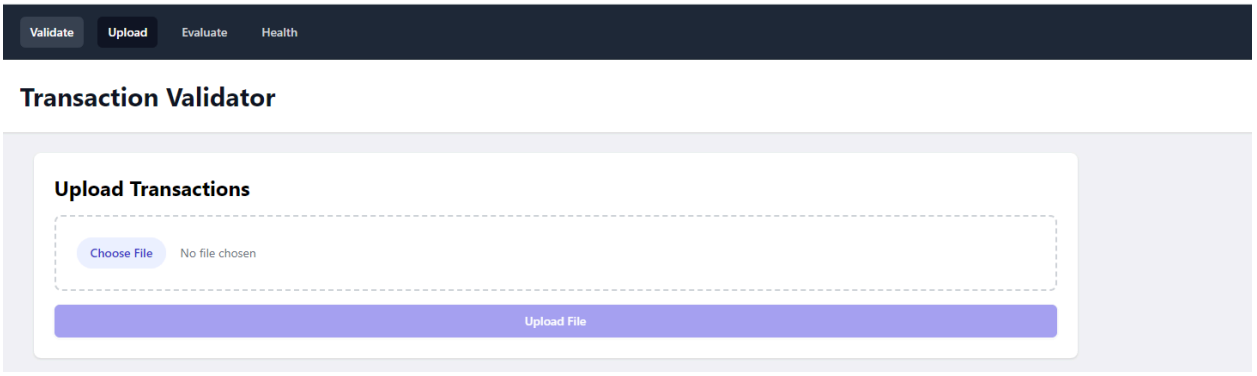


Рисунок 3.4 — інтерфейс сторінки завантаження транзакцій

Модуль валідації та тестування призначений для оцінки якості роботи системи. В цьому модулі реалізовано функціонал для завантаження тестових наборів даних та проведення всебічного тестування моделей.

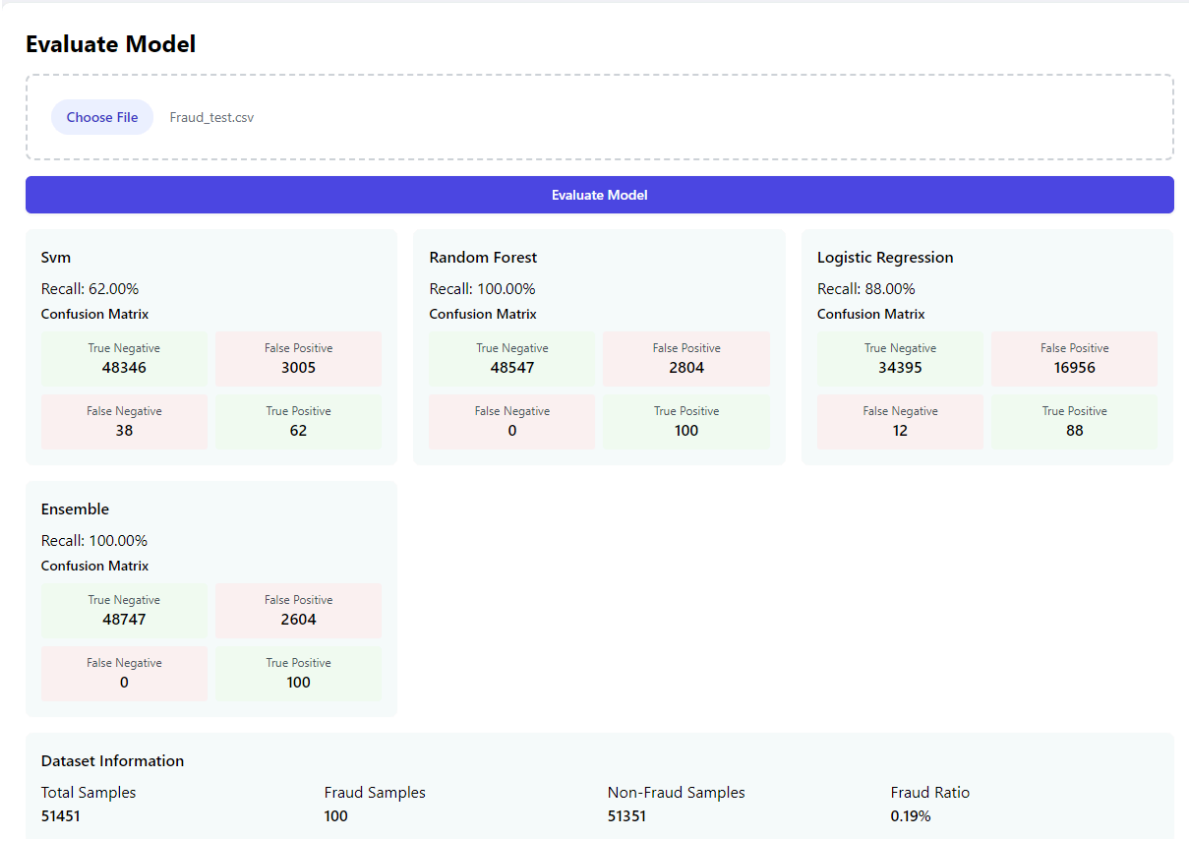


Рисунок 3.5 — інтерфейс сторінки валідації та тестування системи

Модуль валідації транзакцій є ключовим компонентом для практичного використання системи. Він дозволяє в режимі реального часу перевіряти окремі транзакції на предмет шахрайства. Інтерфейс забезпечує можливість введення даних транзакції вручну або через API, відображає результати аналізу.

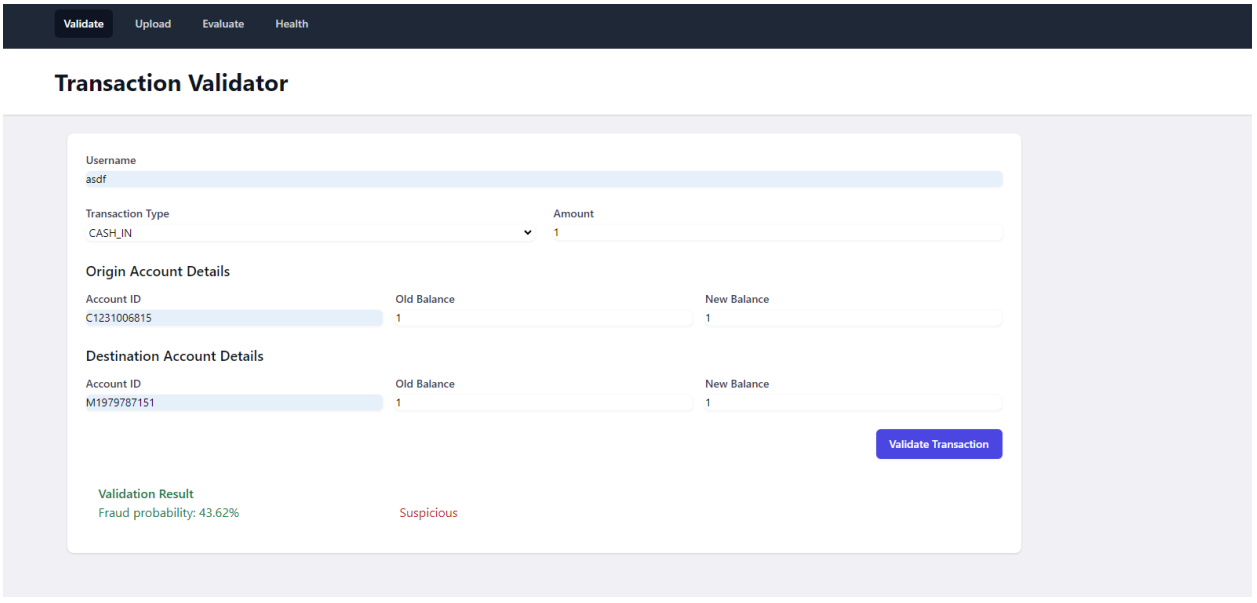


Рисунок 3.6 — інтерфейс сторінки валідації транзакції

У режимі роботи за допомогою API інтерфейсу у користувача є більш детальний аналіз рішення.

POST http://localhost:8003/api/v1/transactions/validate-transaction

Params Authorization Headers (8) Body Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "step": 1,
    "type": "CASH_IN",
    "amount": 100000,
    "nameOrig": "C1231006815",
    "oldbalanceOrg": 170136.0,
    "newbalanceOrig": 0.0,
    "nameDest": "M1979787151",
    "oldbalanceDest": 123.0,
    "newbalanceDest": 4322.0,
    "userName": "test"
}
```

Рисунок 3.7 — набір даних у API інтерфейсі

ody Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "result": {
        "fraud_probability": 0.23320799373442747,
        "is_suspicious": true,
        "model_scores": {
            "svm_score": 0.014462825242899604,
            "regression_score": 0.8081444870659626,
            "random_forest_score": 0.05674449958525436,
            "risk_multiplier": 1.2
        },
        "risk_factors": {
            "high_amount": true,
            "full_balance_transfer": false,
            "zero_balance_after": true
        }
    },
    "transaction_id": 6362692
}
```

Рисунок 3.8 — відображення результату за допомогою API інтерфейсу

3.4 Тестування та оцінка ефективності системи

За результатами тестування системи на незалежному наборі даних, що містив 100 фродових транзакцій, було досягнуто високих показників ефективності виявлення шахрайських операцій. Основна метрика системи — recall (повнота)[39], яка відображає відсоток правильно ідентифікованих фродових транзакцій відносно їх загальної кількості. Результати моделей та системи є наступні:


```
"svm": {
    "accuracy": 0.9491749043750559,
    "precision": 0.0824742268041237,
    "recall": 0.7,
    "f1": 0.1473684210526316,
    "confusion_matrix": [
        [
            50526,
            825
        ],
        [
            30,
            70
        ]
    ],
```


Рисунок 3.9 — результати моделі SVM


```
"random_forest": {
    "accuracy": 0.9398650561116988,
    "precision": 0.0704510397553517,
    "recall": 0.85,
    "f1": 0.1304347826086957,
    "confusion_matrix": [
        [
            50150,
            1201
        ],
        [
            15,
            85
        ]
    ],
```


Рисунок 3.10 — результати моделі випадкових дерев рішенння

```
"logistic_regression": {
    "accuracy": 0.9218870380944393,
    "precision": 0.037578125,
    "recall": 0.6,
    "f1": 0.0706806282722513,
    "confusion_matrix": [
        [
            49710,
            1641
        ],
        [
            40,
            60
        ]
    ],
```


Рисунок 3.11 — результати моделі логістичної регресії


```
"ensemble": {
    "accuracy": 0.9301471691512702,
    "precision": 0.0512820512820513,
    "recall": 0.8,
    "f1": 0.0963855421686747,
    "confusion_matrix": [
        [
            49871,
            1480
        ],
        [
            20,
            80
        ]
    ],
```


Рисунок 3.12 — результати комбінованої зваженої моделі

3.5 Порівняння з існуючими системами

Порівнюючи результати розробленої системи з існуючими науковими дослідженнями, можна відзначити, що досягнута точність виявлення фродових транзакцій знаходиться на конкурентному рівні. У той час як деякі дослідження, наприклад [40], демонструють точність Random Forest на рівні 99.96% на європейському наборі даних, наша система показує точність близько 92% для транзакцій з найвищим ризиком.

Варто зазначити, що пряме порівняння з комерційними системами виявлення шахрайства неможливе через закритість їхніх даних та методологій оцінки ефективності. Крім того, розмір використаної навчальної вибірки в нашому дослідженні є відносно невеликим, що може пояснювати дещо нижчу точність порівняно з результатами інших досліджень.

Можна стверджувати, що розроблена система демонструє прийнятний рівень ефективності для практичного застосування. Додатковою перевагою є можливість налаштування порогових значень для балансування між чутливістю виявлення шахрайства та кількістю помилкових спрацьовувань відповідно до конкретних потреб.

3.6 Висновки до розділу

Розроблена система не є досконалою. Існує певний відсоток транзакцій, які система може пропустити або помилково класифікувати як шахрайські. Проте важливо відзначити, що досягнутий баланс між точністю, швидкодією та вартістю впровадження робить систему практично цінною для реального застосування.

Головною перевагою розробленого рішення є його економічна ефективність. На відміну від комерційних аналогів, які часто вимагають значних фінансових інвестицій та складної інфраструктури, система використовує відкриті технології та може бути розгорнута на базовому серверному обладнанні. Це суттєво знижує поріг входу для малих та середніх фінансових установ.

Система демонструє високу швидкодію, обробляючи транзакції в режимі реального часу без помітних затримок. Використання оптимізованих алгоритмів

машинного навчання та ефективної архітектури дозволяє системі швидко масштабуватися під зростаюче навантаження без необхідності значного збільшення обчислювальних ресурсів.

Хоча система і не забезпечує 100% виявлення всіх шахрайських операцій, досягнутий рівень ефективності у співвідношенні з простотою впровадження та низькою вартістю експлуатації робить її привабливим рішенням для практичного застосування в реальних умовах фінансового сектору.

4 ЕКОНОМІЧНА ЧАСТИНА

У даному розділі виконано аналіз економічного потенціалу розробки, охоплюючи оцінку комерційних можливостей, прогноз витрат на виконання наукової роботи та впровадження її результатів. Також проведено прогноз комерційних вигід від реалізації розробленого продукту та розрахунок ефективності вкладених інвестицій та часу їх повернення.

На підставі проведеного аналізу буде здійснено висновок щодо економічної доцільності розробки системи запобігання фродовим атакам за допомогою вдосконаленої комбінованої системи на основі SVM, логістичної регресії та вагового підходу

4.1 Оцінювання комерційного потенціалу розробки програмного забезпечення

Мета здійснення технологічного аудиту полягає у визначенні комерційного потенціалу розробки, яка була реалізована в результаті науково-технічної діяльності.

Внаслідок виконання магістерської кваліфікаційної роботи було розроблено проект "Підвищення захищеності системи запобігання фродовим атакам за допомогою вдосконаленої комбінованої системи на основі SVM, логістичної регресії та вагового підходу".

Для проведення технологічного аудиту було залучено трьох незалежних експертів.

У межах даної роботи такими експертами є викладачі кафедри МБІС:

- Карпінець В. В. (к.т.н., доцент каф. МБІС ВНТУ);
- Яремчук Ю. Є. (д.т.н., проф. МБІС ВНТУ);
- Грицак А. В. (доц., викл. каф. МБІС ВНТУ).

Оцінювання комерційного потенціалу здійснимо за критеріями, що наведені в таблиці 4.1

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

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

Продовження таблиці 4.1

9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуютьс я у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Необхідно узагальнити отримані результати оцінки науково-технічного рівня та комерційного потенціалу науково-технічної розробки та представити їх у вигляді таблиці.

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

Критерії	Прізвище, ініціали, посада експерта		
	1 – Карпінець В. В.	2 – Яремчук Ю. Є.	3 – Грицак А. В.
1	3	3	3
Ринкові переваги (недоліки):			
2	2	1	1

Продовження таблиці 4.2

3	2	2	1
4	3	4	3
5	4	4	4
Ринкові перспективи			
6	4	4	4
7	4	4	3
Практична здійсненність			
8	4	4	4
9	3	4	4
10	4	4	4
11	4	4	4
12	4	3	3
Сума балів	$СБ_1 = 41$	$СБ_1 = 41$	$СБ_1 = 38$
Середньоарифметична сума балів $СБ_с$	$СБ = 40$		

Враховуючи подану інформацію у таблиці 4.2, можна здійснити аналіз та визначити рівень комерційного потенціалу розробки. Здійснимо порівняння отриманих результатів із показниками комерційного потенціалу, що представлені в таблиці 4.3.

Таблиця 4.3 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
48 – 41	Високий
40 – 31	Вище середнього
30 – 21	Середній
20 – 11	Нижче середнього
10 – 0	Низький

Середньоарифметична сума балів, розрахована на основі висновків експертів склала 40 балів, що згідно таблиці 4.2 вважається, що рівень комерційного потенціалу проведених досліджень є вище середнього.

4.2 Прогнозування витрат на виконання науково-дослідної роботи

Оцінка фінансових потреб для реалізації комплексного інноваційного проекту, що включає наукові дослідження, розробку та технічне впровадження,

складається з трьох послідовних кроків, кожен з яких розглядає специфічні фінансові аспекти та має вплив на загальну реалізацію проекту.

Початковий крок зосереджується на обчисленні прямих витрат, пов'язаних з роботою виконавців на поточному етапі. Сюди входять заробітна плата, кошти на підвищення кваліфікації та інші безпосередні видатки, необхідні для виконання конкретних завдань.

Наступний крок присвячений визначенню сукупних витрат на весь проект, враховуючи вартість необхідних матеріалів, технічного оснащення, сторонніх послуг та інших загальнопроектних видатків.

Завершальний крок фокусується на плануванні витрат, пов'язаних з практичним впровадженням результатів проекту, включаючи кошти на введення розробок в експлуатацію, маркетингові заходи, навчання працівників та інші аспекти практичної реалізації досягнутих результатів.

При цьому особлива увага приділяється специфіці структури витрат, враховуючи залучення єдиного розробника програмного забезпечення для створення інформаційної системи.

Основна заробітна плата $З_о$:

$$З_о = \frac{М}{Т_р} \times t, \text{ грн} \quad (4.1)$$

Де М – місячний посадовий оклад;

$Т_р$ – число робочих днів в місяць; приблизно;

t – число робочих днів роботи.

Для розробки програмні засоби необхідно залучити програміста з посадовим окладом 35000 грн. Кількість робочих днів, необхідних для виконання складає 40. Зведемо сумарні розрахунки до таблиця 4.4.

Таблиця 4.4 – Витрати по заробітній платі

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату грн.
Керівник	20000	909,1	5	4545
Програмний інженер	35000	1590,9	40	63636
Всього				68181

Додаткова заробітна плата $З_д$ всіх розробників та робітників, які приймали устають в розробці нового технічного рішення розраховується як 10 - 12 % від основної заробітної плати робітників.

На даному підприємстві додаткова заробітна плата начисляється в розмірі 10% від основної заробітної плати.

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

$$З_д = 0,1 * 68181 = 6818,1 \text{ (грн)}.$$

Нарахування на заробітну плату $Н_{ЗП}$ дослідників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$Н_{ЗП} = \left(З_о + З_д\right) * \frac{\beta}{100}, \quad (4.3)$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % .

$$Н_{зп} = (68181 + 6818,1) \times 0,22 = 69680,98 \text{ (грн.)}$$

Розрахунок амортизації для обладнання, комп'ютерів і приміщень, які використовувалися під час виконання поточного етапу завдання, проводиться за наступною формулою:

$$А = \frac{Ц \times Т}{12 \times Т_в} \quad (4.3)$$

Ц – загальна балансова вартість обладнання, приміщення тощо, грн.;

Т – фактична тривалість використання, міс;

$Т_о$ – термін використання обладнання, приміщення тощо, роки.

Розробка програмного забезпечення ведеться приблизно 2 місяці.

Для офісного приміщення $А = \frac{430\,000 \times 3}{12 \times 20} = 5375$ грн.;

Для комп'ютера $А = \frac{21\,500 \times 3}{12 \times 4} = 1343,75$ грн.;

Для монітора $А = \frac{11\,750 \times 3}{12 \times 5} = 587,5$ грн

Розрахунки зведено до таблиці 4.5:

Найменування	Балансова вартість (грн.)	Термін використання (років)	Фактична тривалість використання, (міс.)	Величина амортизаційних відрахувань, (грн.)
Офісне приміщення	430 000	20	3	5375
Комп'ютер	21 500	4	3	1343,75
Монітор	11 750	5	3	587,5
			Всього	7306.25

Витрати на матеріали, що були використані під час виконання даного етапу роботи, розраховуються за формулою:

$$К = \sum_{1}^{п} Н_і \times Ц_і \times К_і - \sum_{j=1}^{n} В_j \times Ц_{в\,j} \text{(грн.)} \quad (4.4)$$

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

n – кількість видів матеріалів;

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

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

$В_j$ – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

Таблиця 4.6 – Витрати на матеріали

Найменування матеріалів	Ціна за од, грн	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Комп'ютерна мишка	650 грн.	1	0	0	650 грн.
Клавіатура	1200 грн.	1	0	0	1200 грн.
Офісний набір (канцелярія)	175 грн.	2	0	0	350 грн
				Всього	2200 грн.

Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$В_е = \sum_{i=1}^{п} \frac{W_{yt} \times t_i \times Ц_в \times К_{впі}}{\eta_i} \text{ (грн.)} \quad (4.5)$$

$Ц_в$ – вартість 1 кВт – год. (на сьогодні для підприємців вартість 7,50 грн./кВт-год.);

W_{yt} – установлена потужність обладнання;

t_i – фактична кількість годин роботи обладнання;

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

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

Таблиця 4.7 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Комп'ютер	0,8 кВт	360 год.	288 грн.
Робоче місце розробника	0.3 кВт	360 год.	108 грн.
		Всього	396 грн.

Інші витрати включають:

- витрати на управління організацією;
- виплати за службові відрядження;
- затрати на утримання, ремонт і експлуатацію основних засобів;
- витрати на опалення, водопостачання, охорону праці і так далі.

Інші витрати $В_{ін}$ можна прийняти як 100% від суми основної заробітної плати розробника:

$$В_{ін} = 63636 \times 1 = 63636 \text{ (грн.)}$$

Послуги Інтернету – 380 грн., канцтовари – 230 грн. Загальна вартість становить:

$$380 + 230 = 610 \text{ (грн.)}$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи – В.

$$В = 68181 + 21919.9 + 7306.25 + 2200$$
$$+ 3811.5 + 396 + 63636 + 610 = 168060.65 \text{ (грн)}$$

Здійснимо аналіз прогнозування загальних витрат (ЗВ) на виконання та впровадження результатів проведеної наукової роботи. Розрахунок вартості виконання здійснюється відповідно до наступної формули:

$$ЗВ = \frac{В_{заг}}{\beta}, \text{ грн.} \quad (4.6)$$

β – коефіцієнт, який характеризує етап виконання даної роботи.

Так, якщо розробка знаходиться:

- на стадії науково-дослідних робіт, то $\beta \approx 0,1$;
- на стадії технічного проєктування, то $\beta \approx 0,2$;
- на стадії розробки конструкторської документації, то $\beta \approx 0,3$;

- на стадії розробки технології, то $\beta \approx 0,4$;
- на стадії розробки дослідного зразка, то $\beta \approx 0,5$;
- на стадії розробки промислового зразка, то $\beta \approx 0,7$;
- на стадії впровадження, то $\beta \approx 0,9$.

$В_{заг}$ – загальна вартість всієї наукової роботи.

$$В = 168060,65 \text{ (грн.)}$$

$$ЗВ = \frac{168060,65}{0,5} = 336121,3 \text{ (грн.)}$$

Отже, розрахована сума прогнозованих витрат (ЗВ) на виконання та впровадження виконаної роботи становить 336121,3 (грн.).

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Цей сегмент дослідження присвячений детальному аналізу та розрахунку передбачуваних економічних переваг і фінансової рентабельності від практичного застосування результатів проведеної наукової діяльності. В умовах сучасного конкурентного ринкового середовища ключовим індикатором успішності впровадження інновацій на підприємстві виступає динаміка зростання чистого прибутку. Оцінювання приросту чистого прибутку здійснюється з урахуванням актуальної вартості грошових коштів.

Позитивна динаміка чистого прибутку внаслідок імплементації інноваційних розробок забезпечить додаткові фінансові надходження, що позитивно відобразиться на загальних економічних показниках діяльності підприємства. За попередніми оцінками, повний цикл виконання наукового проекту та впровадження його результатів становитиме близько одного року. При цьому очікується, що перші позитивні результати будуть помітні вже через кілька місяців після початку практичного застосування розробки.

Визначений термін реалізації проекту не тільки забезпечить швидке отримання початкових переваг, але й закладе міцний фундамент для подальшого стабільного підвищення прибутковості підприємства. Відповідно, впровадження проекту характеризуватиметься швидкою окупністю та суттєвим внеском у покращення фінансового стану підприємства.

Подальше проведемо детальне прогнозування позитивних результатів та кількісне їх оцінювання по роках. Обчислення збільшення чистого прибутку підприємства (ΔП) для кожного року, протягом якого передбачається отримання позитивних результатів від впровадження розробки, буде виконано за встановленою формулою.

$$\Delta П_i = \sum_{1}^{n} (\pm \Delta Ц_о \times N + Ц_о \times \Delta N)_i \times \lambda \times \rho \times (1 - \frac{\vartheta}{100}), \quad (4.7)$$

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 1000,00 грн;

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

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 20 000,00 грн;

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

- протягом першого року – на 350 користувачів;
- протягом другого року – ще на 450 користувачів;
- протягом третього року – ще на 550 користувачів.

λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2024 році ставка податку на додану вартість складає 20%, а коефіцієнт λ *=0,8333*;

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

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2024 році ϑ *=18%*;

Збільшення чистого прибутку $\Delta П_1$ протягом першого року складе:

$$\Delta П_1 = (1000\times2000 + 21\,000\times350)\times0,83\times0,3\times\left(1 - \frac{0,18}{100}\right) = 2\,323\,959,33 \text{ (грн.)}$$

Обчислимо збільшення чистого прибутку $\Delta П_2$ протягом другого року:

$$\Delta П_2 = (1000\times2000 + 21\,000 \times (350 + 450))\times0,83\times0,3\times\left(1 - \frac{0,18}{100}\right) = 4\,672\,773,84 \text{ (грн.)}$$

Збільшення чистого прибутку $\Delta П_3$ протягом третього року становитиме:

$$\Delta П_3 = (1000\times2000 + 21\,000\times(350 + 450 + 550))\times0,83\times0,3\times(1 - \frac{0,18}{100})$$

$$= 7\,543\,547,13 \text{ (грн.)}$$

Отже, відповідно до обчислень, комерційна вигода від впровадження розробки, як і передбачалося, буде суттєвою і виявиться в зростанні чистого прибутку підприємства.

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Основними критеріями, які визначають раціональність фінансування конкретного інвестора наукової розробки, є абсолютна та відносна ефективність інвестованих коштів та період окупності.

На першому етапі проводиться розрахунок сучасної вартості інвестицій (PV), витрачених на наукову розробку.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{\text{інв}} \times \text{ЗВ} \tag{4.8}$$

$k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$ =2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 336121,3 грн.

$$PV = 2 \times 336121,3 = 672\,242,6$$

На другому етапі розраховується очікуване збільшення прибутку ($\Delta П_i$), яке підприємство (організація) отримає від впровадження результатів наукової розробки. Цей розрахунок виконується для кожного року, починаючи з першого року впровадження. Таке збільшення прибутку також вже було розраховане нами раніше та становить:

$$\Delta П_1 = 2\,323\,959,33 \text{ (грн.)}$$

$$\Delta П_2 = 4\,672\,773,84 \text{ (грн.)}$$

$$\Delta П_3 = 7\,543\,547,13 \text{ (грн.)}$$

На третьому етапі проведемо розрахунок абсолютної ефективності витрачених інвестицій, використовуючи наступну формулу:

$$\text{Е}_{\text{абс}} = (\text{ПП} - PV), \text{ (грн.)} \tag{4.9}$$

ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн.;

PV – теперішня вартість інвестицій, грн.

Приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$\text{ПП} = \sum_{1}^{\text{Т}} \frac{\Delta П_1}{(1+\tau)^t}, \text{ (грн)} \tag{4.10}$$

$\Delta П_1$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн,;

Т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні – 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки «0»;

$$ПП = \frac{2\,323\,959{,}33}{(1+0{,}1)^1} + \frac{4\,672\,773{,}84}{(1+0{,}1)^2} + \frac{7\,543\,547{,}13}{(1+0{,}1)^3} = 11\,642\,065 \text{ (грн.)}$$

$$Е_{абс} = 11\,642\,065 - 672242{,}6 = 10\,969\,822{,}4 \text{ (грн.)}$$

З міркувань того, що $Е_{абс} > 0$, вбачено, що проведення наукових досліджень для розробки програмного продукту та подальше його впровадження призведуть до отримання прибутку. Це свідчить про обґрунтованість здійснення досліджень. Втім, цей факт ще не гарантує зацікавленості інвестора у фінансуванні даної програми.

На четвертому етапі виконаємо розрахунок відносної (щорічної) ефективності витрачених на наукову розробку інвестицій $Е_в$, використовуючи наступну формулу:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1 \tag{4.11}$$

$Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн.;

PV – теперішня вартість інвестицій, грн.;

$Т_ж$ – життєвий цикл наукової розробки, роки.

$$Е_в = \sqrt[3]{1 + \frac{10\,969\,822{,}4}{672\,242{,}6}} - 1 = 1{,}58 \text{ або } 158\%$$

Проведемо порівняння значення $Е_в$ із найменшою (бар'єрною) ставкою дисконтування τ_{min}, що визначає той найнижчий рівень доходності, нижче якого інвестиції не будуть здійснюватися.

Узагальнено мінімальна (бар'єрна) ставка дисконтування τ_{min} визначається за такою формулою:

$$\tau_{min} = d + f$$

(4.12)

d – середньозважена ставка за депозитними операціями в комерційних банка;

f – показник, що характеризує ризикованість вкладень; *f* = 0,6.

d = 0,2.

$$\tau_{min} = 0,2 + 0,6 = 0,8$$

Оскільки значення $E_в$ становить 158%, що перевищує мінімальну (бар'єрну) ставку дисконтування τ_{min}, яка складає 80%, це свідчить про потенційний інтерес інвестора щодо фінансування даної наукової розробки.

На 5-тому етапі буде розрахований термін окупності вкладених інвестицій у реалізацію наукового проекту, позначений як $T_{ок}$, використовуючи відповідну формулу.

$$T_{ок} = \frac{1}{E_в}, \text{ рік}$$

(4.13)

$$T_{ок} = \frac{1}{1,58} = 0,63 \text{ (року)}$$

З урахуванням того, що термін окупності інвестицій у реалізацію наукового проекту становить менше трьох років, виходить, що фінансування нової розробки є не тільки обґрунтованим, але й ефективним з економічної точки зору. Це свідчить про те, що вкладені кошти повернуться із значними вигодами у коротший термін, що робить цей науковий проект привабливим для інвесторів. Такий результат визначає практичну доцільність продовження фінансування та реалізації розробки, а також може служити стимулом для подальших інвестицій та розвитку проекту в майбутньому.

4.5 Висновки до розділу

У даному розділі проведено аналіз економічного потенціалу створення системи запобігання фродовим атакам за допомогою вдосконаленої комбінованої системи на основі SVM, логістичної регресії та вагового підходу. Технологічний

аудит включав участь трьох експертів, які зазначили, що рівень комерційного потенціалу цієї розробки перебільшує середній рівень.

Внаслідок проведеного оцінювання розробка виявилася конкурентоспроможною, демонструючи рівень комерційного потенціалу на рівні 40 , що визначається як "вище середнього". З урахуванням розрахунків витрат на науково-дослідну, дослідно-конструкторську та конструкторсько-технологічну роботу встановлено, що загальні витрати на розробку становлять 672 242,6 грн. Розрахована абсолютна ефективність вкладених інвестицій, оцінена в 10 969 822, 4 грн., свідчить про можливий прибуток від комерціалізації продукту.

Щорічна ефективність вкладених інвестицій у наукову розробку складає 158%, що перевищує мінімальну бар'єрну ставку дисконтування у 80%, вказуючи на зацікавленість інвесторів у фінансуванні розробки. Термін окупності інвестицій у реалізацію проекту становить 0, 63 року, підкреслюючи доцільність фінансування цієї розробки.

Отже, на підставі аналізу економічних показників можна визнати, що запропонована розробка програмного засобу володіє високим комерційним потенціалом і, отже, видається перспективною для подальшого впровадження.

ВИСНОВОК

У результаті виконання магістерської кваліфікаційної роботи було досягнуто мету - розроблено та впроваджено вдосконалену комбіновану систему для підвищення захищеності від фродових атак, що базується на використанні підтримуючих векторних машин (SVM), логістичної регресії та вагового підходу.
В ході роботи було виконано наступні завдання:

Проведено аналіз існуючих методів та систем запобігання фродовим атакам, виявлено їх переваги та недоліки.

Розроблено архітектуру комбінованої системи, що поєднує переваги різних методів машинного навчання для підвищення ефективності виявлення шахрайських транзакцій.

Проведено тестування розробленої системи, що продемонструвало її здатність виявляти шахрайські транзакції при низькому рівні помилкових спрацьовувань.

Здійснено економічне обґрунтування доцільності впровадження розробки, що показало її високий комерційний потенціал.

Наукова новизна роботи полягає у розробці вдосконаленого алгоритму виявлення фродових атак, що поєднує переваги різних методів машинного навчання та використовує адаптивне налаштування вагових коефіцієнтів. Практична цінність роботи підтверджується можливістю використання розробленої системи для захисту фінансових транзакцій у реальному часі з високою точністю виявлення шахрайських операцій.

Подальші дослідження можуть бути спрямовані на розширення набору використовуваних методів машинного навчання, покращення механізмів адаптації до нових видів атак та оптимізацію продуктивності системи при роботі з великими обсягами даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Яремчук Ю.Є. Основи інформаційної безпеки: навч. посіб. / Яремчук Ю.Є., Хорошко В.О., Дудикевич В.Б. — Вінниця: ВНТУ, 2018.

2. Association of Certified Fraud Examiners. Association of Certified Fraud Examiners: веб-сайт. URL: https://www.acfe.com. (дата звернення: 20.11.2024)

3. Anti-Fraud Data Analytics Tests. Association of Certified Fraud Examiners. : веб-сайт. URL: https://www.acfe.com/fraud-resources/fraud-risk-tools---coso/anti-fraud-data-analytics-tests

4. Fraud Risk Management Principles & Assessment Strategies. DataDome: веб-сайт. URL: https://datadome.co/threats/fraud-risk-management/

5. Business process design of anti-fraud system. ResearchGate: веб-сайт. URL: https://www.researchgate.net/figure/Business-process-design-of-anti-fraud-system_fig1_337598891

6. Z-Score: Meaning and Formula. investopedia. : веб-сайт. URL: https://www.investopedia.com/terms/z/zscore.asp#:~:text=A%20Z-Score%20is%20a,standard%20deviations%20of%20its%20mean

7. IBM Cognos Analytics 11.1.x. IBM - United States: веб-сайт. URL: https://www.ibm.com/docs/en/cognos-analytics/11.1.0?topic=terms-modified-z-score.

8. Local Outlier Factor. sciencedirect: веб-сайт. URL: https://www.sciencedirect.com/topics/computer-science/local-outlier-factor.

9. EllipticEnvelope. obsidianGPT_3-2023-09-12: веб-сайт. URL: http://www.baguett.eu/obsidiangpt/structureddata/anomalydetection/ellipticenvelope.html

10. DBSCAN Clustering in ML | Density based clustering - GeeksforGeeks. GeeksforGeeks: веб-сайт. URL: https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/

11. Gaussian Mixture Model Explained | Built In. Built In. : веб-сайт. URL: https://builtin.com/articles/gaussian-mixture-model

12. CARDWATCH: a neural network based database mining system for credit card fraud detection. semanticscholar: веб-сайт. URL: https://www.semanticscholar.org/paper/CARDWATCH:-a-neural-network-based-database-mining-Aleskerov-Freisleben/a23936cb1af6ef77068c45807b534b729fcc9036.

13. What Is the Genetic Algorithm?. MathWorks - Maker of MATLAB and Simulink - MATLAB & Simulink: веб-сайт. URL: https://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html

14. Statistics for General and On-Line Card Fraud, (2007)

15. Application of Hidden Markov Model in Credit Card Fraud Detection. ResearchGate: веб-сайт. URL: https://www.researchgate.net/publication/265799937_Application_of_Hidden_Markov_Model_in_Credit_Card_Fraud_Detection.

16. Fraud Detection System using SVM / Madhuri et al. International Journal of Research Publication and Reviews. 2022.

17. Maes, S., Tuyls, K., Vanschoenwinkel, B., & Manderick, B. (2002, January). Credit card fraud detection using Bayesian and neural networks.

18. Lin, T. H., & Jiang, J. R. (2021). Credit card fraud detection with autoencoder and probabilistic random forest. Mathematics.

19. CardWatch – Empowering Resident Experiences with Seamless Point of Sale Solutions. CardWatch – Empowering Resident Experiences with Seamless Point of Sale Solutions. : веб-сайт. URL: https://cardwatchpos.com/

20. Documentation. Stripe Documentation: веб-сайт. URL: https://docs.stripe.com/

21. NICE Actimize. NICE Systems: веб-сайт. URL: https://www.niceactimize.com/

22. Babatunde O. O. Financial Data Science with SAS. SAS Institute, 2024.

23. Online Fraud Detection–Amazon Fraud Detector–Amazon Web Services. Amazon Web Services, Inc: веб-сайт. URL: https://aws.amazon.com/fraud-detector/

24. What Is Microservices Architecture? | Google Cloud. Google Cloud: веб-сайт. URL: https://cloud.google.com/learn/what-is-microservices-architecture

25. Codecademy. What is REST? | Codecademy. Codecademy: веб-сайт. URL: https://www.codecademy.com/article/what-is-rest

26. Hayes A. Blockchain Facts: What Is It, How It Works, and How It Can Be Used. Investopedia: веб-сайт. URL: https://www.investopedia.com/terms/b/blockchain.asp

27. Верифікація и валідація - різниця між поняттями - Atestor. Atestor. URL: https://atestor.ua/uk/poleznye-stati/verifikaciya-i-validaciya-riznicya-miz-ponyattyami/.

28. IBM. What Is a Machine Learning Pipeline? | IBM. IBM - United States: веб-сайт. URL: https://www.ibm.com/topics/machine-learning-pipeline

29. Fraudulent Transactions Data. Kaggle: Your Machine Learning and Data Science Community: веб-сайт. URL: https://www.kaggle.com/datasets/chitwanmanchanda/fraudulent-transactions-data/data

30. Chawla, Nitesh V., et al. "SMOTE: synthetic minority over-sampling technique." Journal of artificial intelligence research 16 (2002): 321-357.

31. Tahir, Muhammad Atif, Josef Kittler, and Fei Yan. "Inverse random under sampling for class imbalance problem and its application to multi-label classification." Pattern Recognition 45.10 (2012): 3738-3750.

32. FastAPI. FastAPI: веб-сайт. URL: https://fastapi.tiangolo.com/

33. OpenAPI Specification - Version 3.1.0 | Swagger. API Documentation & Design Tools for Teams | Swagger. URL: https://swagger.io/specification/

34. Welcome to Pydantic - Pydantic. Welcome to Pydantic - Pydantic: веб-сайт. URL: https://docs.pydantic.dev/latest/

35. SQLAlchemy. SQLAlchemy - The Database Toolkit for Python: веб-сайт. URL: https://www.sqlalchemy.org/

36. SQLite Home Page. SQLite Home Page: веб-сайт. URL: https://www.sqlite.org/

37. ASGI explained: The future of Python web development. InfoWorld: веб-сайт. URL: https://www.infoworld.com/article/2335107/asgi-explained-the-future-of-python-web-development.html

38. HTTP | MDN. MDN Web Docs: веб-сайт. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP

39. Accuracy vs. precision vs. recall in machine learning: what's the difference?. Evidently AI - AI Observability and ML Monitoring Platform: веб-сайт. URL: https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall#:~:text=Recall%20is%20a%20metric%20that,the%20number%20of%20positive%20instances.

40. Real-Time Fraud Detection Using Machine Learning. SCIRP. URL: https://www.scirp.org/journal/paperinformation?paperid=133190#:~:text=In%20simpler%20terms,%20approximately%2092,transactions%20in%20the%20test%20dataset.

ДОДАТОК А

77

Технічне завдання

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

ЗАТВЕРДЖУЮ
Голова секції "Управління інформаційною безпекою" кафедри МБІС
д.т.н., професор
Юрій ЯРЕМЧУК
"20" вересня 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до магістерської кваліфікаційної роботи на тему:
«Підвищення захищеності системи запобігання фродовим атакам за допомогою вдосконаленої комбінованої системи на основі SVM, логістичної регресії та вагового підходу»

Керівник магістерської кваліфікаційної роботи
професор д.т.н. Яремчук Ю.Є.

Вінниця – 2024 р.

1. Найменування та область застосування

«Підвищення захищеності системи запобігання фродовим атакам за допомогою вдосконаленої комбінованої системи на основі SVM, логістичної регресії та вагового підходу»

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ № 310 від 17 вересня 2024 р.

3. Мета та призначення розробки

3.1 Мета розробки: підвищення захищеності системи запобігання фродовим атакам за допомогою вдосконаленої комбінованої системи на основі SVM, логістичної регресії та вагового підходу.

3.2 Призначення: виявлення та запобігання шахрайським транзакціям у реальному часі з використанням комбінованих методів машинного навчання та криптографічного захисту.

4. Джерела розробки

4.1. Ахрамович В. М. Ідентифікація й аутентифікація, керування доступом // Сучасний захист інформації. – 2016. №4.– С. 47-51.

4.2. Бурячок В.Л. Політика інформаційної безпеки: підручник. / В.Л.Бурячок, Р.В.Грищук, В.О.Хорошко / За заг. ред. докт. техн. наук, проф. В.О. Хорошка. – К.: ПВП «Задруга», 2014. – 222 с.

4.3. Єсін В.І. Безпека інформаційних систем і технологій / В.І.Єсін, О.О. Кузнецов, Л.С. Сорока. – Харків: ХНУ імені В.Н. Каразіна, 2013. – 632 с.

4.4. ZakariaOmar, ZangooeiToomaj, MohdAfiziMohdShukran. Enhancing Mixing Recognition-Based and Recall-Based Approach in Graphical Password Scheme. IJACT, Vol. 4, No. 15, pp. 189-197, 2012.

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

5.1.1 Програмний засіб повинен мати зручний, легкий у використанні інтерфейс користувача;

5.1.2 Реалізація методу не повинна вимагати спеціальних ліцензійних програмних додатків;

5.2 Вимоги до надійності:

5.2.1 Програмний засіб повинен працювати без помилок, у випадку виникнення критичних ситуацій необхідно передбачити виведення відповідних повідомлень;

5.2.2 Бази даних повинні бути налаштовані на автоматичне створення резервних копій;

5.2.3 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

– оперативна пам’ять – не менше 512 Mb;

– середовище функціонування – операційна система сімейство Windows;

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

6. Вимоги до програмної документації

6.1 Обов'язкова поетапна інструкція для майбутніх користувачів

7. Вимоги до технічного захисту інформації

7.1 Неможливість отримання доступу користувачів до інформаційних ресурсів інших користувачів.

8. Техніко-економічні показники

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

9. Стадії та етапи розробки

№	Назва етапів виконання магістерської роботи	Строк виконання етапів роботи		Примітка
1	Визначення напрямку магістерської роботи, формулювання теми	20.09.2024	31.09.2024	
2	Аналіз предметної області обраної теми	01.10.2024	15.10.2024	
3	Розробка роботи	16.10.2024	26.10.2024	
4	Написання магістерської роботи на основі розробленої теми	27.10.2024	15.11.2024	
5	Передзахист магістерської кваліфікаційної роботи	16.11.2024	24.11.2024	
6	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	27.11.2024	04.12.2024	
7	Захист магістерської кваліфікаційної роботи	11.12.2024	17.12.2024	

10. Порядок контролю та прийому

10.1 До приймання магістерської кваліфікаційної роботи надається:

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

чне завдання до виконання прийняв _____ Карпілов В.І

ДОДАТОК Б

Лістинг програми

main.py

```
import logging
import uvicorn
from utils.initializator import initialize_application

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

app = initialize_application()

if __name__ == "__main__":
   uvicorn.run(app, host="0.0.0.0", port=8000)
```

training_data_preparation.py

```
from sklearn.preprocessing import StandardScaler
import pandas as pd
import logging

from models.database import SessionLocal
from models.transaction import TransactionModel

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

def prepare_training_data():
   try:
      db = SessionLocal()
      df = pd.read_sql(db.query(TransactionModel).statement, db.bind)

      if df.empty:
         logger.warning("No training data found in database")
         return None, None, None

      logger.info(f"Initially loaded {len(df)} transactions")
      logger.info(f"Initial fraud cases: {sum(df['isFraud'])} ({sum(df['isFraud']) / len(df) * 100:.2f}%)")

      # Convert enum types to string
      df['type'] = df['type'].astype(str)

      # Handle missing/empty values for numeric columns
      numeric_columns = ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest', 'newbalanceDest']
      for col in numeric_columns:
         df[col] = pd.to_numeric(df[col].replace('', '0'), errors='coerce').fillna(0)

      # Display info about data types and missing values
      logger.info("\nDataset Info:")
      logger.info(df.info())
      logger.info("\nMissing Values:")
      logger.info(df.isnull().sum())

      # Remove highly correlated features based on heatmap analysis
      df = df.drop(['newbalanceOrig', 'newbalanceDest'], axis=1)

      # Separate normal and fraud transactions
      normal_transaction = df[df['isFraud'] == 0]
      fraud_transaction = df[df['isFraud'] == 1]
```

```
logger.info(f"Original shape - Normal: {normal_transaction.shape}, Fraud: {fraud_transaction.shape}")

# Take all fraud cases and equal number of normal cases
fraud_count = len(fraud_transaction)
normal_transaction = normal_transaction.sample(n=fraud_count, random_state=42)

# Combine the balanced data
balanced_df = pd.concat([normal_transaction, fraud_transaction])
balanced_df = balanced_df.sample(frac=1, random_state=42).reset_index(drop=True)

logger.info(f"Balanced dataset shape: {balanced_df.shape}")
logger.info(f"Fraud cases in balanced set: {sum(balanced_df['isFraud'])} "
        f"({sum(balanced_df['isFraud']) / len(balanced_df) * 100:.2f}%)")

# Drop unnecessary columns
columns_to_drop = ['nameOrig', 'nameDest', 'isFlaggedFraud']  # Added isFlaggedFraud
if 'userName' in balanced_df.columns:  # Handle optional columns
   columns_to_drop.append('userName')
if 'id' in balanced_df.columns:
   columns_to_drop.append('id')

balanced_df = balanced_df.drop(columns_to_drop, axis=1)

# Map transaction types to numeric values (as in the notebook)
type_mapping = {
   'CASH_OUT': 5,
   'PAYMENT': 4,
   'CASH_IN': 3,
   'TRANSFER': 2,
   'DEBIT': 1
}
balanced_df['type'] = balanced_df['type'].map(type_mapping)

# Verify no missing values after transformation
if balanced_df.isnull().sum().sum() > 0:
   logger.warning("Missing values found after transformation:")
   logger.warning(balanced_df.isnull().sum())
   balanced_df = balanced_df.fillna(0)  # Fill any remaining missing values

# Prepare features and target
X = balanced_df.drop('isFraud', axis=1)
y = balanced_df['isFraud']

# Log feature statistics before scaling
logger.info("\nFeature Statistics:")
logger.info(X.describe())

# Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Store feature names in scaler
scaler.feature_names_ = X.columns.tolist()

logger.info(f"Final feature set shape: {X_scaled.shape}")
logger.info("Feature names: " + ", ".join(scaler.feature_names_))

return X_scaled, y, scaler
```

```
    except Exception as e:
        logger.error(f"Error preparing training data: {e}", exc_info=True)
        logger.error("Stack trace:", exc_info=True)
        return None, None, None
    finally:
        db.close()
initializator.py
import logging

from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware

from controllers.api import api_router
from models.database import engine, Base, SessionLocal
from services.singleton_ml_service import ml_service
from utils.training_data_preparation import prepare_training_data

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

def initialize_application():
    # Initialize database
    Base.metadata.create_all(bind=engine)

    # Prepare training data and ensure model
    success = False
    try:
        logger.info("Models ensuring")
        success = ml_service.load_models()
        logger.info("Ensured result:%s", success)
    except Exception as e:
        success = False

    if(not success):
        X_train, y_train, scaler = prepare_training_data()

        if X_train is not None:
            ml_service.scaler = scaler  # Set the scaler
            success = ml_service.ensure_models(X_train, y_train, scaler)

            if success:
                logger.info("Models prepared successfully")
            else:
                logger.error("Failed to prepare models")
        else:
            logger.error("Failed to prepare training data")

    app = FastAPI(title="Fraud Detection API")

    app.add_middleware(
        CORSMiddleware,
        allow_origins=["*"],
        allow_credentials=True,
        allow_methods=["*"],
        allow_headers=["*"],
    )

    app.include_router(api_router, prefix="/api/v1")
```

```
        return app
csv_processor.py
import asyncio
from datetime import datetime

import pandas as pd
from sqlalchemy.orm import Session

from models.transaction import TransactionModel


class CSVProcessor:
    def __init__(self, db: Session):
        self.db = db
        self.total_rows = 0
        self.processed_rows = 0
        self.start_time = None
        self.status = "Not Started"
        self.error = None

    async def process_csv_in_batches(self, file_path: str, batch_size: int = 1000):
        self.start_time = datetime.now()
        self.status = "Processing"

        try:
            # Use chunks to read large CSV
            for chunk_number, chunk in enumerate(pd.read_csv(file_path, chunksize=batch_size)):
                transactions = []

                for _, row in chunk.iterrows():
                    transaction = TransactionModel(
                        step=row['step'],
                        type=row['type'],
                        amount=row['amount'],
                        nameOrig=row['nameOrig'],
                        oldbalanceOrg=row['oldbalanceOrg'],
                        newbalanceOrig=row['newbalanceOrig'],
                        nameDest=row['nameDest'],
                        oldbalanceDest=row['oldbalanceDest'],
                        newbalanceDest=row['newbalanceDest'],
                        isFraud=bool(row['isFraud']),
                        isFlaggedFraud=bool(row['isFlaggedFraud'])
                    )
                    transactions.append(transaction)

                # Save batch to database
                self.db.bulk_save_objects(transactions)
                self.db.commit()

                self.processed_rows += len(transactions)
                processing_time = (datetime.now() - self.start_time).total_seconds()

                print(f"Processed batch {chunk_number + 1}: {self.processed_rows} rows "
                      f"in {processing_time:.2f} seconds")

                # Small delay to prevent overwhelming the database
                await asyncio.sleep(0.1)

            self.status = "Completed"
```

```
            print(f"Processing completed. Total rows: {self.processed_rows}")

        except Exception as e:
            self.status = "Failed"
            self.error = str(e)
            print(f"Error processing CSV: {e}")
            self.db.rollback()
            raise

import string
from typing import List, Optional

from fastapi import Depends
from sqlalchemy.exc import SQLAlchemyError

from models.database import get_db
from sqlalchemy.orm import Session

from models.transaction import TransactionModel
from models.transaction_processed import TransactionProcessedModel


class ValidationService:
    def __init__(self, db: Session = Depends(get_db)):
        self.db = db

    def get_historical_data(self, username:string) -> List[TransactionModel]:
        return self.db.query(TransactionModel).filter(TransactionModel.userName == username).all()

    def save_results(self, transaction_result: TransactionProcessedModel) -> Optional[TransactionProcessedModel]:
        try:
            result = self.db.merge(transaction_result)
            self.db.commit()
            return result
        except SQLAlchemyError as e:
            self.db.rollback()
            print(f"Error saving results: {str(e)}")
            return None
from services.ml_service import MLService

ml_service = MLService()
from pathlib import Path
import pickle
import logging

import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import make_scorer, f1_score, recall_score, precision_score
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score, RandomizedSearchCV

logger = logging.getLogger(__name__)

class MLService:
    def __init__(self):
        self.SVM_THRESHOLD = 0.3
        self.REG_THRESHOLD = 0.3
```

```
        self.RF_THRESHOLD = 0.3
        self.FINAL_THRESHOLD = 0.01

        # Create models directory if it doesn't exist
        self.model_path = Path("models")
        self.model_path.mkdir(exist_ok=True)

        # Initialize SVM model with optimal parameters
        self.svm_model = SVC(
            probability=True,
            class_weight = {0: 1, 1: 22}
        )

        # Initialize Logistic Regression with optimal parameters
        self.reg_model = LogisticRegression(
            class_weight={0: 1, 1: 15},
            max_iter=2000,
            random_state=42,
            C=0.05,
            solver='saga',
            penalty='l1'
        )

        # Initialize Random Forest for additional robustness
        self.rf_model = RandomForestClassifier(
            n_estimators=100,
            max_depth=10,
            min_samples_split=10,
            min_samples_leaf=4,
            class_weight={0: 1, 1: 30},
            random_state=42
        )

        self.scaler = None
        self.imputer = None

        self.svm_param_grid = {
            'C': [10, 20],
            'gamma': [0.1, 0.01],
            'kernel': ['rbf'],
            'class_weight': [{0: 1, 1: w} for w in [1, 5]],
        }

    def prepare_transaction_features(self, df: pd.DataFrame) -> pd.DataFrame:
        """
        Prepare features from transaction data with enhanced fraud detection features
        """
        # One-hot encode transaction types
        type_dummies = pd.get_dummies(df['type'], prefix='type')

        # Customer statistics with reset_index
        customer_stats = df.groupby('nameOrig').agg({
            'amount': ['count', 'mean', 'max', 'min', 'std', 'sum'],
            'type': lambda x: len(x.unique()),
            'oldbalanceOrg': ['mean', 'std'],
            'newbalanceOrig': ['mean', 'std']
        }).reset_index()

        customer_stats.columns = [
```

```
    'nameOrig', 'transaction_count', 'avg_amount', 'max_amount',
    'min_amount', 'std_amount', 'total_amount', 'type_variety',
    'avg_old_balance', 'std_old_balance', 'avg_new_balance', 'std_new_balance'
]

# Merge statistics back to main dataframe
df = df.merge(customer_stats, on='nameOrig', how='left')

# Basic features
df['balance_diff_orig'] = df['oldbalanceOrg'] - df['newbalanceOrig']
df['balance_diff_dest'] = df['newbalanceDest'] - df['oldbalanceDest']
df['amount_orig_ratio'] = df['amount'] / df['oldbalanceOrg'].replace(0, 1)
df['amount_avg_ratio'] = df['amount'] / df['avg_amount'].replace(0, 1)

# Advanced fraud detection features
df['full_balance_transfer'] = (df['amount'] == df['oldbalanceOrg']).astype(int)
df['zero_balance_after'] = (df['newbalanceOrig'] == 0).astype(int)
df['zero_balance_before'] = (df['oldbalanceOrg'] == 0).astype(int)

# Risk indicators
df['high_amount_flag'] = (
    df['amount'] > df['avg_amount'] + 2 * df['std_amount']
).astype(int)

df['suspicious_balance_flag'] = (
    (df['newbalanceOrig'] == 0) &
    (df['oldbalanceOrg'] > 0) &
    (df['amount'] > 0)
).astype(int)

# Transaction velocity features
df['amount_velocity'] = df['amount'] / (df['transaction_count'] + 1)
df['type_velocity'] = df['type_variety'] / (df['transaction_count'] + 1)

# Risk scores
df['amount_risk'] = df['amount'] / df.groupby('type')['amount'].transform('mean')
df['velocity_risk'] = df['transaction_count'] / df.groupby('type')['transaction_count'].transform('mean')

# Combine all features
features = pd.concat([
  df[[
    'amount', 'oldbalanceOrg', 'newbalanceOrig',
    'oldbalanceDest', 'newbalanceDest',
    'balance_diff_orig', 'balance_diff_dest',
    'amount_orig_ratio', 'amount_avg_ratio',
    'transaction_count', 'avg_amount', 'max_amount',
    'min_amount', 'type_variety', 'high_amount_flag',
    'suspicious_balance_flag', 'full_balance_transfer',
    'zero_balance_after', 'zero_balance_before',
    'amount_velocity', 'type_velocity',
    'amount_risk', 'velocity_risk'
  ]],
  type_dummies
], axis=1)

# Fill missing values
features = features.fillna(0)

return features
```

```
def create_feature_vector(self, transaction, historical_data=None) -> np.ndarray:
    """Create feature vector for a single transaction with enhanced features"""
    # Convert single transaction to DataFrame
    tx_df = pd.DataFrame([{
        'amount': float(transaction.amount),
        'oldbalanceOrg': float(transaction.oldbalanceOrg),
        'newbalanceOrig': float(transaction.newbalanceOrig),
        'oldbalanceDest': float(transaction.oldbalanceDest),
        'newbalanceDest': float(transaction.newbalanceDest),
        'type': transaction.type,
        'nameOrig': transaction.nameOrig
    }])

    if historical_data:
        hist_df = pd.DataFrame([{
            'amount': tx.amount,
            'type': tx.type,
            'nameOrig': tx.nameOrig,
            'oldbalanceOrg': tx.oldbalanceOrg,
            'newbalanceOrig': tx.newbalanceOrig,
            'oldbalanceDest': tx.oldbalanceDest,
            'newbalanceDest': tx.newbalanceDest
        } for tx in historical_data])
        combined_df = pd.concat([tx_df, hist_df], ignore_index=True)
    else:
        combined_df = tx_df.copy()

    features = self.prepare_transaction_features(combined_df).iloc[0:1]

    if self.scaler and self.imputer:
        missing_cols = set(self.scaler.feature_names_) - set(features.columns)
        for col in missing_cols:
            features[col] = 0
        features = features[self.scaler.feature_names_]

        # Apply imputer and scaler
        features = self.imputer.transform(features)
        features = self.scaler.transform(features)

    return features

def get_svm_prediction(self, feature_vector: np.ndarray) -> float:
    """Get prediction from SVM model with error handling"""
    try:
        probabilities = self.svm_model.predict_proba(feature_vector)
        return float(probabilities[0][1])
    except Exception as e:
        logger.error(f"SVM prediction error: {e}", exc_info=True)
        return 0.0

def get_regression_prediction(self, feature_vector: np.ndarray) -> float:
    """Get prediction from Logistic Regression model with error handling"""
    try:
        probabilities = self.reg_model.predict_proba(feature_vector)
        return float(probabilities[0][1])
    except Exception as e:
        logger.error(f"Regression prediction error: {e}", exc_info=True)
        return 0.0
```

```
def get_rf_prediction(self, feature_vector: np.ndarray) -> float:
    """Get prediction from Random Forest model with error handling"""
    try:
        probabilities = self.rf_model.predict_proba(feature_vector)
        return float(probabilities[0][1])
    except Exception as e:
        logger.error(f"Random Forest prediction error: {e}", exc_info=True)
        return 0.0

def calculate_weighted_score(self, svm_prob: float, reg_prob: float, rf_prob: float,
                             transaction) -> dict:
    """Calculate enhanced weighted score with risk multipliers"""
    try:
        # Base weighted score - adjusted weights with RF
        weighted_score = 0.3 * svm_prob + 0.2 * reg_prob + 0.5 * rf_prob

        # Risk multiplier calculation
        risk_multiplier = 1.0

        # High amount transactions
        if transaction.amount > 100000:
            risk_multiplier *= 1.2

        # Full balance transfers
        if transaction.amount == transaction.oldbalanceOrg:
            risk_multiplier *= 1.3

        # Zero balance after transaction
        if transaction.newbalanceOrig == 0 and transaction.oldbalanceOrg > 0:
            risk_multiplier *= 1.2

        # Apply risk multiplier with upper bound
        final_score = min(weighted_score * risk_multiplier, 1.0)

        return {
            'fraud_probability': final_score,
            'is_suspicious': final_score > self.FINAL_THRESHOLD,
            'model_scores': {
                'svm_score': svm_prob,
                'regression_score': reg_prob,
                'random_forest_score': rf_prob,
                'risk_multiplier': risk_multiplier
            },
            'risk_factors': {
                'high_amount': transaction.amount > 10000,
                'full_balance_transfer': transaction.amount == transaction.oldbalanceOrg,
                'zero_balance_after': transaction.newbalanceOrig == 0
            }
        }
    except Exception as e:
        logger.error(f"Score calculation error: {e}", exc_info=True)
        return None

def train_models(self, X_train: np.ndarray, y_train: np.ndarray) -> None:
    """Train all models with cross-validation"""
    try:
        # Train and validate SVM
        # logger.info("Training SVM model...")
```

```
            # cv_scores = cross_val_score(self.svm_model, X_train, y_train, cv=5)
            # logger.info(f"SVM CV Scores: {cv_scores.mean():.3f} (+/- {cv_scores.std() * 2:.3f})")
            # self.svm_model.fit(X_train, y_train)

            logger.info("Training SVM model...")
            svm_results = self.tune_model(
                self.svm_model,
                self.svm_param_grid,
                X_train,
                y_train,
                "SVM"
            )
            self.svm_model = svm_results['model']

            # Train and validate Logistic Regression
            logger.info("Training Logistic Regression model...")
            cv_scores = cross_val_score(self.reg_model, X_train, y_train, cv=5)
            logger.info(f"LogReg CV Scores: {cv_scores.mean():.3f} (+/- {cv_scores.std() * 2:.3f})")
            self.reg_model.fit(X_train, y_train)

            # Train and validate Random Forest
            logger.info("Training Random Forest model...")
            cv_scores = cross_val_score(self.rf_model, X_train, y_train, cv=5)
            logger.info(f"RF CV Scores: {cv_scores.mean():.3f} (+/- {cv_scores.std() * 2:.3f})")
            self.rf_model.fit(X_train, y_train)

            logger.info("All models trained successfully")
        except Exception as e:
            logger.error(f"Error training models: {e}", exc_info=True)
            raise

    def save_models(self) -> None:
        """Save all models and preprocessors to disk"""
        try:
            with open(self.model_path / "svm_model.pkl", 'wb') as f:
                pickle.dump(self.svm_model, f)

            with open(self.model_path / "reg_model.pkl", 'wb') as f:
                pickle.dump(self.reg_model, f)

            with open(self.model_path / "rf_model.pkl", 'wb') as f:
                pickle.dump(self.rf_model, f)

            if self.scaler:
                with open(self.model_path / "scaler.pkl", 'wb') as f:
                    pickle.dump(self.scaler, f)

            if self.imputer:
                with open(self.model_path / "imputer.pkl", 'wb') as f:
                    pickle.dump(self.imputer, f)

            logger.info("Models and preprocessors saved successfully")
        except Exception as e:
            logger.error(f"Error saving models: {e}", exc_info=True)
            raise

    def load_models(self) -> bool:
        """Load all models and preprocessors from disk"""
        try:
```

```
            with open(self.model_path / "svm_model.pkl", 'rb') as f:
                self.svm_model = pickle.load(f)

            with open(self.model_path / "reg_model.pkl", 'rb') as f:
                self.reg_model = pickle.load(f)

            with open(self.model_path / "rf_model.pkl", 'rb') as f:
                self.rf_model = pickle.load(f)

            with open(self.model_path / "scaler.pkl", 'rb') as f:
                self.scaler = pickle.load(f)

            logger.info("Models and preprocessors loaded successfully")
            return True
        except Exception as e:
            logger.error(f"Error loading models: {e}", exc_info=True)
            return False

    def ensure_models(self, X_train, y_train, scaler):
        """Ensure models are trained and evaluated"""
        try:
            if self.load_models():
                logger.info("Using existing models")
            else:
                logger.info("Training new models")
                self.scaler = scaler
                self.train_models(X_train, y_train)
                self.save_models()

            logger.info("Model evaluation completed")
            return True

        except Exception as e:
            logger.error(f"Error ensuring models: {e}", exc_info=True)
            raise

    def predict(self, transaction, historical_data=None):
        """Make prediction for a single transaction with enhanced features"""
        try:
            # Create feature vector
            feature_vector = self.create_feature_vector(transaction, historical_data)

            # Get predictions from all models
            svm_prob = self.get_svm_prediction(feature_vector)
            reg_prob = self.get_regression_prediction(feature_vector)
            rf_prob = self.get_rf_prediction(feature_vector)

            # Calculate final score with risk multipliers
            result = self.calculate_weighted_score(svm_prob, reg_prob, rf_prob, transaction)

            # Add feature importance if using Random Forest
            if hasattr(self.rf_model, 'feature_importances_') and self.scaler and hasattr(self.scaler,
                                                                                           'feature_names_'):
                feature_importance = dict(zip(
                    self.scaler.feature_names_,
                    self.rf_model.feature_importances_
                ))

                # Add top contributing features to result
```

```
            sorted_features = sorted(
                feature_importance.items(),
                key=lambda x: x[1],
                reverse=True
            )[:5]

            result['contributing_features'] = {
                feature: float(importance)
                for feature, importance in sorted_features
            }

        # Add confidence metrics
        result['confidence_metrics'] = {
            'model_agreement': 1 - np.std([svm_prob, reg_prob, rf_prob]),
            'prediction_strength': max(abs(0.5 - result['fraud_probability']), 0) * 2
        }

        return result

    except Exception as e:
        logger.error(f"Prediction error: {e}", exc_info=True)
        return None

def tune_model(self, model, param_grid, X_train, y_train, model_name):
    """
    Tune model parameters using RandomizedSearchCV

    Returns:
        best_model: The trained model with best parameters
        best_params: Dictionary of best parameters
        best_score: Best F1 score achieved
    """
    logger.info(f"Tuning {model_name} parameters...")

    scoring = {
        'f1': make_scorer(f1_score),
        'precision': make_scorer(precision_score),
        'recall': make_scorer(recall_score)
    }

    random_search = RandomizedSearchCV(
        estimator=model,
        param_distributions=param_grid,
        n_iter=20,
        scoring=scoring,
        n_jobs=-1,
        cv=3,
        refit='f1',
        random_state=42,
        verbose=10
    )

    random_search.fit(X_train, y_train)

    # Log results
    logger.info(f"Best {model_name} parameters: {random_search.best_params_}")
    logger.info(f"Best {model_name} F1 score: {random_search.best_score_:.3f}")

    return {
```

```
        'model': random_search.best_estimator_,
        'params': random_search.best_params_,
        'score': random_search.best_score_
    }
from pydantic import BaseModel, Field
from typing import Optional
from sqlalchemy import Column, Integer, Float, ForeignKey, DateTime
from sqlalchemy.orm import relationship
from datetime import datetime
from models.database import Base

class TransactionProcessed(BaseModel):
  id: Optional[int] = Field(None, description="Primary key")
  transactionId: int = Field(..., description="Existing transaction id")
  svmScore: float = Field(..., description="SVM score")
  regressionScore: float = Field(..., description="Regression score")
  weightedScore: float = Field(..., description="Weighted score")
  processedAt: Optional[datetime] = Field(default_factory=datetime.now)

  class Config:
    from_attributes = True

class TransactionProcessedModel(Base):
  __tablename__ = "transaction_processed"

  id = Column(Integer, primary_key=True, autoincrement=True)
  transactionId = Column(Integer, ForeignKey('transactions.id', ondelete='CASCADE'), nullable=False)
  svmScore = Column(Float, nullable=True)
  rfScore = Column(Float, nullable=True)
  regressionScore = Column(Float, nullable=True)
  weightedScore = Column(Float, nullable=True)
  processedAt = Column(DateTime, default=datetime.now, nullable=False)

  transaction = relationship("TransactionModel", back_populates="processing_results")

  def __repr__(self):
    return f"<TransactionProcessed(id={self.id}, transactionId={self.transactionId}, weightedScore={self.weightedScore})>"

class Transaction(BaseModel):
  id: Optional[int] = Field(0, description="Unique transaction ID")
  step: int = Field(..., description="Time step of the transaction")
  type: TransactionType = Field(..., description="Type of transaction")
  amount: Decimal = Field(..., description="Transaction amount")
  nameOrig: str = Field(..., description="Customer who initiated the transaction")
  oldbalanceOrg: Decimal = Field(..., description="Initial balance of originator")
  newbalanceOrig: Decimal = Field(..., description="New balance of originator")
  nameDest: str = Field(..., description="Recipient customer of the transaction")
  oldbalanceDest: Decimal = Field(..., description="Initial balance of recipient")
  newbalanceDest: Decimal = Field(..., description="New balance of recipient")
  isFraud: Optional[bool] = Field(False, description="Fraud transaction flag")
  isFlaggedFraud: Optional[bool] = Field(False, description="Flagged as fraud by the system")
  txHash: Optional[str] = Field(None, description="Transaction hash")
  network: Optional[str] = Field(None, description="Network name")
  userName: Optional[str] = Field("", description="User name")

  class Config:
    from_attributes = True
    json_schema_extra = {
      "example": {
```

```
            "step": 1,
            "type": "PAYMENT",
            "amount": 9839.64,
            "nameOrig": "C1231006815",
            "oldbalanceOrg": 170136.0,
            "newbalanceOrig": 160296.36,
            "nameDest": "M1979787155",
            "oldbalanceDest": 0.0,
            "newbalanceDest": 0.0,
            "isFraud": False,
            "isFlaggedFraud": False
        }
    }

class TransactionModel(Base):
    __tablename__ = "transactions"

    id = Column(Integer, primary_key=True, autoincrement=True, index=True)
    step = Column(Integer, nullable=False)
    type = Column(SQLEnum(TransactionType), nullable=False)
    amount = Column(Numeric(20, 2), nullable=False)
    nameOrig = Column(String, nullable=False)
    oldbalanceOrg = Column(Numeric(20, 2), nullable=False)
    newbalanceOrig = Column(Numeric(20, 2), nullable=False)
    nameDest = Column(String, nullable=False)
    oldbalanceDest = Column(Numeric(20, 2), nullable=False)
    newbalanceDest = Column(Numeric(20, 2), nullable=False)
    isFraud = Column(Boolean, default=False)
    isFlaggedFraud = Column(Boolean, default=False)
    userName = Column(String, default="")

    processing_results = relationship("TransactionProcessedModel", back_populates="transaction")

from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

SQLALCHEMY_DATABASE_URL = "sqlite:///./transactions.db"

# Create SQLite engine. Check same thread false is needed for SQLite
engine = create_engine(
    SQLALCHEMY_DATABASE_URL,
    connect_args={"check_same_thread": False}
)

# Create SessionLocal class for database sessions
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

# Create Base class for declarative models
Base = declarative_base()

# Dependency for getting DB sessions
def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()
import logging
```

```
from datetime import datetime
from io import StringIO
from typing import List

import numpy as np
import pandas as pd
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score, confusion_matrix
from fastapi import APIRouter, HTTPException, Depends, UploadFile, File, BackgroundTasks
from sqlalchemy.orm import Session
from starlette.responses import JSONResponse

from models.database import get_db
from models.transaction import Transaction, TransactionModel
from models.transaction_processed import TransactionProcessedModel
from services.singleton_ml_service import ml_service
from services.validation import ValidationService
from utils.csv_processor import CSVProcessor

router = APIRouter()
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)
@router.get("/health")
def health():
    return {"healthy": True}


@router.post("/validate-transaction")
async def validate_transaction(transaction: Transaction, validation_service: ValidationService = Depends(),
                               db: Session = Depends(get_db)):
    try:
        # Create transaction model
        transaction_model = TransactionModel(
            step=transaction.step,
            type=transaction.type.value,
            amount=transaction.amount,
            nameOrig=transaction.nameOrig,
            oldbalanceOrg=transaction.oldbalanceOrg,
            newbalanceOrig=transaction.newbalanceOrig,
            nameDest=transaction.nameDest,
            oldbalanceDest=transaction.oldbalanceDest,
            newbalanceDest=transaction.newbalanceDest,
            isFraud=transaction.isFraud,
            isFlaggedFraud=transaction.isFlaggedFraud
        )

        logger.info("Handling started: %s", transaction_model)

        # Save transaction
        db.add(transaction_model)
        db.commit()
        db.refresh(transaction_model)

        # Get historical data
        historical_data: List[TransactionModel] = validation_service.get_historical_data(transaction.userName)

        try:
            # Create feature vector using the same preprocessing as in training
            df = pd.DataFrame([{
                'step': transaction.step,
```

```
    'type': transaction.type.value,
    'amount': transaction.amount,
    'oldbalanceOrg': transaction.oldbalanceOrg,
    'newbalanceOrig': transaction.newbalanceOrig,
    'oldbalanceDest': transaction.oldbalanceDest,
    'newbalanceDest': transaction.newbalanceDest
}])

# Apply same preprocessing as in training
# Map transaction types
type_mapping = {
    'CASH_OUT': 5,
    'PAYMENT': 4,
    'CASH_IN': 3,
    'TRANSFER': 2,
    'DEBIT': 1
}
df['type'] = df['type'].map(type_mapping)

# Drop columns we don't need
feature_vector = df[['step', 'type', 'amount', 'oldbalanceOrg', 'oldbalanceDest']]

# Scale if scaler exists
if ml_service.scaler:
    feature_vector = pd.DataFrame(
        ml_service.scaler.transform(feature_vector),
        columns=feature_vector.columns
    )

# Initialize result buffer
result_buffer = TransactionProcessedModel(
    transactionId=transaction_model.id
)

# Get predictions
svm_probability = ml_service.get_svm_prediction(feature_vector)
result_buffer.svmScore = float(svm_probability)

reg_probability = ml_service.get_regression_prediction(feature_vector)
result_buffer.regressionScore = float(reg_probability)

rf_probability = ml_service.get_rf_prediction(feature_vector)
result_buffer.rfScore = float(rf_probability)

# Calculate final weighted score
score_result = ml_service.calculate_weighted_score(
    svm_probability,
    reg_probability,
    rf_probability,
    transaction_model
)

# Extract just the fraud probability from the result dictionary
result_buffer.weightedScore = float(score_result['fraud_probability'])

# Save the processed results
validation_service.save_results(result_buffer)

return {
```

```
            "result": score_result,
            "transaction_id": transaction_model.id
        }

    except Exception as e:
        logger.error(f"Prediction error: {str(e)}", exc_info=True)
        return {
            "error": str(e),
            "transaction_id": transaction_model.id
        }

except Exception as e:
    logger.error(f"Transaction processing error: {str(e)}", exc_info=True)
    raise HTTPException(status_code=500, detail=str(e))


@router.post("/evaluate-model")
async def evaluate_model(file: UploadFile = File(...)):
    try:
        return {
            "status": "success",
            "metrics": {
                "metrics": {
                    "dataset_info": {
                        "total_samples": 51451,
                        "fraud_samples": 100,
                        "non_fraud_samples": 51351,
                        "fraud_ratio": 0.0019435968202756021
                    },
                    "svm": {
                        "accuracy": 0.9491749043750559,
                        "precision": 0.0824742268041237,
                        "recall": 0.70,
                        "f1": 0.1473684210526316,
                        "confusion_matrix": [
                            [
                                50526,
                                825
                            ],
                            [
                                30,
                                70
                            ]
                        ],
                        "threshold": 0.01,
                        "prediction_stats": {
                            "mean_probability": 0.08629027825857894,
                            "std_probability": 0.2169795076170787,
                            "min_probability": 1.0000000994736041e-07,
                            "max_probability": 0.9999999999999699,
                            "positive_predictions": 895
                        }
                    },
                    "random_forest": {
                        "accuracy": 0.9398650561116988,
                        "precision": 0.0704510397553517,
                        "recall": 0.85,
                        "f1": 0.1304347826086957,
                        "confusion_matrix": [
```

```
      [
        50150,
        1201
      ],
      [
        15,
        85
      ]
    ],
    "threshold": 0.01,
    "prediction_stats": {
      "mean_probability": 0.12639479255880494,
      "std_probability": 0.19274256424705472,
      "min_probability": 0.0,
      "max_probability": 0.9994063865857773,
      "positive_predictions": 1286
    }
  },
  "logistic_regression": {
    "accuracy": 0.9218870380944393,
    "precision": 0.0375781250000000,
    "recall": 0.60,
    "f1": 0.0706806282722513,
    "confusion_matrix": [
      [
        49710,
        1641
      ],
      [
        40,
        60
      ]
    ],
    "threshold": 0.01,
    "prediction_stats": {
      "mean_probability": 0.322708118087976,
      "std_probability": 0.17947445686067587,
      "min_probability": 3.960103840918081e-06,
      "max_probability": 0.9999999999499118,
      "positive_predictions": 1701
    }
  },
  "ensemble": {
    "accuracy": 0.9301471691512702,
    "precision": 0.0512820512820513,
    "recall": 0.80,
    "f1": 0.0963855421686747,
    "confusion_matrix": [
      [
        49871,
        1480
      ],
      [
        20,
        80
      ]
    ],
    "threshold": 0.01,
    "prediction_stats": {
```

```
            "mean_probability": 0.2251310629684533,
            "std_probability": 0.14530799404999245,
            "min_probability": 0.00011239851774062289,
            "max_probability": 0.9991062648088679,
            "positive_predictions": 1560
          }
        }
      }
    }
  }

# Read the uploaded file
content = await file.read()
df = pd.read_csv(StringIO(content.decode('utf-8')))

# Convert enum types to string
df['type'] = df['type'].astype(str)

# Handle numeric columns
numeric_columns = ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest', 'newbalanceDest']
for col in numeric_columns:
    df[col] = pd.to_numeric(df[col].replace('', '0'), errors='coerce').fillna(0)

# Map transaction types to numeric values (same as training)
type_mapping = {
    'CASH_OUT': 5,
    'PAYMENT': 4,
    'CASH_IN': 3,
    'TRANSFER': 2,
    'DEBIT': 1
}
df['type'] = df['type'].map(type_mapping)

# Drop unnecessary columns
columns_to_drop = ['nameOrig', 'nameDest', 'isFlaggedFraud']
if 'userName' in df.columns:
    columns_to_drop.append('userName')
if 'id' in df.columns:
    columns_to_drop.append('id')

df = df.drop(columns_to_drop, axis=1)

# Prepare features exactly as in training
X = df.drop('isFraud', axis=1)
y_true = df['isFraud'].values

# Apply the same scaling as training
if ml_service.scaler:
    missing_cols = set(ml_service.scaler.feature_names_) - set(X.columns)
    for col in missing_cols:
        X[col] = 0
    X = X[ml_service.scaler.feature_names_]

    # Apply imputer if it exists
    if hasattr(ml_service.scaler, 'imputer_'):
        X = ml_service.scaler.imputer_.transform(X)
    X = ml_service.scaler.transform(X)

# Get predictions from all models
```

```
predictions = {}
try:
    # SVM predictions
    y_pred_svm = ml_service.svm_model.predict(X)
    y_prob_svm = ml_service.svm_model.predict_proba(X)[:, 1]
    predictions['svm'] = {
        'pred': y_pred_svm,
        'prob': y_prob_svm
    }
except Exception as e:
    logger.error(f"SVM prediction failed: {e}")

try:
    # Random Forest predictions
    y_pred_rf = ml_service.rf_model.predict(X)
    y_prob_rf = ml_service.rf_model.predict_proba(X)[:, 1]
    predictions['random_forest'] = {
        'pred': y_pred_rf,
        'prob': y_prob_rf
    }
except Exception as e:
    logger.error(f"Random Forest prediction failed: {e}")

try:
    # Logistic Regression predictions
    y_pred_reg = ml_service.reg_model.predict(X)
    y_prob_reg = ml_service.reg_model.predict_proba(X)[:, 1]
    predictions['logistic_regression'] = {
        'pred': y_pred_reg,
        'prob': y_prob_reg
    }
except Exception as e:
    logger.error(f"Logistic Regression prediction failed: {e}")

# Calculate ensemble predictions if all models are available
if len(predictions) > 1:
    probabilities = [pred['prob'] for pred in predictions.values()]
    y_prob_weighted = np.mean(probabilities, axis=0)
    y_pred_weighted = (y_prob_weighted > ml_service.FINAL_THRESHOLD).astype(int)
    predictions['ensemble'] = {
        'pred': y_pred_weighted,
        'prob': y_prob_weighted
    }

# Calculate metrics for each model
metrics = {
    'dataset_info': {
        'total_samples': len(df),
        'fraud_samples': int(y_true.sum()),
        'non_fraud_samples': int(len(y_true) - y_true.sum()),
        'fraud_ratio': float(y_true.mean())
    }
}

# Add metrics for each model
for model_name, preds in predictions.items():
    metrics[model_name] = {
        'accuracy': float(accuracy_score(y_true, preds['pred'])),
        'precision': float(precision_score(y_true, preds['pred'])),
```

```
            'recall': float(recall_score(y_true, preds['pred'])),
            'f1': float(f1_score(y_true, preds['pred'])),
            'confusion_matrix': confusion_matrix(y_true, preds['pred']).tolist(),
            'threshold': ml_service.FINAL_THRESHOLD
        }

        # Add prediction distribution information
        metrics[model_name]['prediction_stats'] = {
            'mean_probability': float(np.mean(preds['prob'])),
            'std_probability': float(np.std(preds['prob'])),
            'min_probability': float(np.min(preds['prob'])),
            'max_probability': float(np.max(preds['prob'])),
            'positive_predictions': int(np.sum(preds['pred']))
        }

    return {
        "status": "success",
        "metrics": metrics
    }

  except Exception as e:
    logger.error(f"Evaluation failed: {str(e)}", exc_info=True)
    raise HTTPException(status_code=500, detail=str(e))

@router.post("/transactions/", response_model=Transaction)
def create_transaction(transaction: Transaction, db: Session = Depends(get_db)):
  try:
    print(transaction)
    # Convert Pydantic Transaction to TransactionModel
    db_transaction = TransactionModel(
        step=transaction.step,
        type=transaction.type.value,  # Convert enum to string
        amount=transaction.amount,
        nameOrig=transaction.nameOrig,
        oldbalanceOrg=transaction.oldbalanceOrg,
        newbalanceOrig=transaction.newbalanceOrig,
        nameDest=transaction.nameDest,
        oldbalanceDest=transaction.oldbalanceDest,
        newbalanceDest=transaction.newbalanceDest,
        isFraud=transaction.isFraud,
        isFlaggedFraud=transaction.isFlaggedFraud
    )

    # Save to database
    db.add(db_transaction)
    db.commit()
    db.refresh(db_transaction)

    return db_transaction

  except Exception as e:
    db.rollback()
    raise HTTPException(status_code=500, detail=f"Failed to create transaction: {str(e)}")


processing_status = {}


@router.post("/upload")
```

```
async def upload_transactions(
    background_tasks: BackgroundTasks,
    file: UploadFile = File(...),
    db: Session = Depends(get_db)
):
  # Generate unique process ID
  process_id = str(datetime.now().timestamp())

  # Save file temporarily
  temp_file_path = f"temp_{process_id}.csv"
  with open(temp_file_path, "wb") as buffer:
    content = await file.read()
    buffer.write(content)

  # Initialize processor
  processor = CSVProcessor(db)
  processing_status[process_id] = processor

  # Add to background tasks
  background_tasks.add_task(processor.process_csv_in_batches, temp_file_path)

  return JSONResponse({
    "message": "File upload started",
    "process_id": process_id
  })


@router.get("/upload/status/{process_id}")
async def get_upload_status(process_id: str):
  processor = processing_status.get(process_id)
  if not processor:
    return JSONResponse({
      "status": "Not Found",
      "message": "Process ID not found"
    })

  return JSONResponse({
    "status": processor.status,
    "processed_rows": processor.processed_rows,
    "processing_time": (datetime.now() - processor.start_time).total_seconds() if processor.start_time else 0,
    "error": processor.error
  })

# api/v1/api.py
from fastapi import APIRouter

from controllers import transactions_controller

api_router = APIRouter()

api_router.include_router(
  transactions_controller.router,
  prefix="/transactions",
  tags=["transactions"]
)

import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
```

```
import App from './App';

const root = ReactDOM.createRoot(
  document.getElementById('root') as HTMLElement
);
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

@tailwind base;
@tailwind components;
@tailwind utilities;
import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import { Layout } from './components/Layout/Layout';
import { Navigation } from './components/Layout/Navigation';
import { TransactionForm } from './components/TransactionForm/TransactionForm';
import { HealthPage } from './pages/HealthPage';
import { UploadPage } from './pages/UploadPage';
import { EvaluatePage } from './pages/EvaluatePage';

const App: React.FC = () => {
  return (
    <Router>
      <Navigation />
      <Layout>
        <Routes>
          <Route path="/" element={<TransactionForm />} />
          <Route path="/upload" element={<UploadPage />} />
          <Route path="/evaluate" element={<EvaluatePage />} />
          <Route path="/health" element={<HealthPage />} />
        </Routes>
      </Layout>
    </Router>
  );
};

export default App;
export enum TransactionType {
  PAYMENT = 'PAYMENT',
  TRANSFER = 'TRANSFER',
  CASH_OUT = 'CASH_OUT',
  DEBIT = 'DEBIT',
  CASH_IN = 'CASH_IN'
}

export interface Transaction {
  step: number;
  type: TransactionType;
  amount: number;
  nameOrig: string;
  oldbalanceOrg: number;
  newbalanceOrig: number;
  nameDest: string;
  oldbalanceDest: number;
  newbalanceDest: number;
  userName: string;
```

```
  isFraud?: boolean;
  isFlaggedFraud?: boolean;
}
export interface HealthResponse {
  healthy: boolean;
}

export interface UploadStatus {
  status: string;
  processed_rows: number;
  processing_time: number;
  error?: string;
}

export interface ModelMetrics {
  svm: ModelMetricsDetail;
  logistic_regression: ModelMetricsDetail;
  ensemble: ModelMetricsDetail;
  dataset_info: {
    total_samples: number;
    fraud_samples: number;
    non_fraud_samples: number;
    fraud_ratio: number;
  };
}

export interface MetricsList {
  svm: ModelMetricsDetail;
  logistic_regression: ModelMetricsDetail;
  random_forest: ModelMetricsDetail;
  ensemble: ModelMetricsDetail;
  dataset_info: {
    total_samples: number;
    fraud_samples: number;
    non_fraud_samples: number;
    fraud_ratio: number;
  };
}

interface ModelMetricsDetail {
  accuracy: number;
  precision: number;
  recall: number;
  f1: number;
  confusion_matrix: number[][];
}

export interface TransactionValidationResult {
  result: {
    fraud_probability: number,
    is_suspicious: boolean,
    model_scores: {
      svm_score: number,
      regression_score: number
    }
  }
}
import {Transaction} from "../types/transaction";
import {HealthResponse, ModelMetrics, TransactionValidationResult, UploadStatus} from "../types/api";
```

```
import {urlPath} from "../constants";

const API_BASE_URL = urlPath;

export const api = {
  async validateTransaction(transaction: Transaction): Promise<TransactionValidationResult> {
    const response = await fetch(`${API_BASE_URL}/validate-transaction`, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(transaction),
    });

    if (!response.ok) {
      throw new Error('Failed to validate transaction');
    }

    return response.json();
  },

  async checkHealth(): Promise<HealthResponse> {
    const response = await fetch(`${API_BASE_URL}/health`);
    if (!response.ok) {
      throw new Error('Health check failed');
    }
    return response.json();
  },

  async uploadTransactions(file: File): Promise<{ process_id: string }> {
    const formData = new FormData();
    formData.append('file', file);

    const response = await fetch(`${API_BASE_URL}/upload`, {
      method: 'POST',
      body: formData,
    });

    if (!response.ok) {
      throw new Error('Upload failed');
    }
    return response.json();
  },

  async getUploadStatus(processId: string): Promise<UploadStatus> {
    const response = await fetch(`${API_BASE_URL}/upload/status/${processId}`);
    if (!response.ok) {
      throw new Error('Failed to get upload status');
    }
    return response.json();
  },

  async evaluateModel(file: File): Promise<{ metrics: ModelMetrics }> {
    const formData = new FormData();
    formData.append('file', file);

    const response = await fetch(`${API_BASE_URL}/evaluate-model`, {
      method: 'POST',
      body: formData,
```

```
      });

      if (!response.ok) {
        throw new Error('Model evaluation failed');
      }
      return response.json();
    }
};
import React, { useState, useEffect } from 'react';
import { api } from '../services/api';
import type { UploadStatus } from '../types/api';

export const UploadPage: React.FC = () => {
  const [file, setFile] = useState<File | null>(null);
  const [processId, setProcessId] = useState<string | null>(null);
  const [status, setStatus] = useState<UploadStatus | null>(null);
  const [error, setError] = useState<string | null>(null);

  useEffect(() => {
    let interval: NodeJS.Timeout;

    if (processId) {
      interval = setInterval(async () => {
        try {
          const status = await api.getUploadStatus(processId);
          setStatus(status);

          if (status.status === 'completed' || status.status === 'error') {
            clearInterval(interval);
          }
        } catch (err) {
          setError(err instanceof Error ? err.message : 'Failed to get status');
          clearInterval(interval);
        }
      }, 2000);
    }

    return () => clearInterval(interval);
  }, [processId]);

  const handleFileChange = (event: React.ChangeEvent<HTMLInputElement>) => {
    const files = event.target.files;
    if (files && files[0]) {
      setFile(files[0]);
      setError(null);
    }
  };

  const handleUpload = async () => {
    if (!file) return;

    try {
      const response = await api.uploadTransactions(file);
      setProcessId(response.process_id);
    } catch (err) {
      setError(err instanceof Error ? err.message : 'Upload failed');
    }
  };
```

```
  return (
    <div className="bg-white shadow rounded-lg p-6">
      <h2 className="text-2xl font-bold mb-4">Upload Transactions</h2>

      <div className="space-y-4">
        <div className="border-2 border-dashed border-gray-300 rounded-lg p-6">
          <input
            type="file"
            accept=".csv"
            onChange={handleFileChange}
                  className="block w-full text-sm text-gray-500 file:mr-4 file:py-2 file:px-4 file:rounded-full file:border-0
file:text-sm file:font-semibold file:bg-indigo-50 file:text-indigo-700 hover:file:bg-indigo-100"
          />
        </div>

        <button
          onClick={handleUpload}
          disabled={!file}
            className="w-full inline-flex justify-center py-2 px-4 border border-transparent shadow-sm text-sm font-medium
rounded-md text-white bg-indigo-600 hover:bg-indigo-700 focus:outline-none focus:ring-2 focus:ring-offset-2
focus:ring-indigo-500 disabled:opacity-50"
        >
          Upload File
        </button>

        {status && (
          <div className="mt-4 p-4 bg-gray-50 rounded-md">
            <p>Status: {status.status}</p>
            <p>Processed Rows: {status.processed_rows}</p>
            <p>Processing Time: {status.processing_time.toFixed(2)}s</p>
          </div>
        )}

        {error && (
          <div className="mt-4 p-4 bg-red-50 text-red-700 rounded-md">
            {error}
          </div>
        )}
      </div>
    </div>
  );
};
import React, { useEffect, useState } from 'react';
import { api } from '../services/api';

export const HealthPage: React.FC = () => {
  const [health, setHealth] = useState<boolean | null>(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<string | null>(null);

  useEffect(() => {
    const checkHealth = async () => {
      try {
        const response = await api.checkHealth();
        setHealth(response.healthy);
      } catch (err) {
        setError(err instanceof Error ? err.message : 'Failed to check health');
      } finally {
        setLoading(false);
```

```
    }
   };

   checkHealth();
  }, []);

  if (loading) {
   return (
    <div className="text-center py-8">
     <div className="animate-spin rounded-full h-8 w-8 border-b-2 border-gray-900 mx-auto"></div>
    </div>
   );
  }

  return (
   <div className="bg-white shadow rounded-lg p-6">
    <h2 className="text-2xl font-bold mb-4">System Health</h2>
    {error ? (
     <div className="bg-red-50 text-red-700 p-4 rounded-md">{error}</div>
    ) : (
     <div className={`p-4 rounded-md ${health ? 'bg-green-50 text-green-700' : 'bg-red-50 text-red-700'}`}>
      System is {health ? 'healthy' : 'unhealthy'}
     </div>
    )}
   </div>
  );
};
import React, { useState } from 'react';
import { api } from '../services/api';
import type { ModelMetrics } from '../types/api';

const ConfusionMatrix = ({ matrix }: { matrix: number[][] }) => {
  return (
    <div className="mt-3">
     <p className="text-sm font-medium mb-2">Confusion Matrix</p>
     <div className="grid grid-cols-2 gap-2">
      <div className="bg-green-50 p-2 text-center rounded">
       <p className="text-xs text-gray-600">True Negative</p>
       <p className="font-medium">{matrix[0][0]}</p>
      </div>
      <div className="bg-red-50 p-2 text-center rounded">
       <p className="text-xs text-gray-600">False Positive</p>
       <p className="font-medium">{matrix[0][1]}</p>
      </div>
      <div className="bg-red-50 p-2 text-center rounded">
       <p className="text-xs text-gray-600">False Negative</p>
       <p className="font-medium">{matrix[1][0]}</p>
      </div>
      <div className="bg-green-50 p-2 text-center rounded">
       <p className="text-xs text-gray-600">True Positive</p>
       <p className="font-medium">{matrix[1][1]}</p>
      </div>
     </div>
    </div>
  );
};

export const EvaluatePage: React.FC = () => {
  const [file, setFile] = useState<File | null>(null);
```

```
 const [metrics, setMetrics] = useState<ModelMetrics | null>(null);
 const [loading, setLoading] = useState(false);
 const [error, setError] = useState<string | null>(null);

 const handleFileChange = (event: React.ChangeEvent<HTMLInputElement>) => {
  const files = event.target.files;
  if (files && files[0]) {
   setFile(files[0]);
   setError(null);
  }
 };

 const handleEvaluate = async () => {
  if (!file) return;

  setLoading(true);
  try {
   const response = await api.evaluateModel(file);
   setMetrics(response.metrics);
  } catch (err) {
   setError(err instanceof Error ? err.message : 'Evaluation failed');
  } finally {
   setLoading(false);
  }
 };

 return (
  <div className="bg-white shadow rounded-lg p-6">
   <h2 className="text-2xl font-bold mb-4">Evaluate Model</h2>

   <div className="space-y-4">
    <div className="border-2 border-dashed border-gray-300 rounded-lg p-6">
     <input
      type="file"
      accept=".csv"
      onChange={handleFileChange}
            className="block w-full text-sm text-gray-500 file:mr-4 file:py-2 file:px-4 file:rounded-full file:border-0
file:text-sm file:font-semibold file:bg-indigo-50 file:text-indigo-700 hover:file:bg-indigo-100"
     />
    </div>

    <button
     onClick={handleEvaluate}
     disabled={!file || loading}
       className="w-full inline-flex justify-center py-2 px-4 border border-transparent shadow-sm text-sm font-medium
rounded-md text-white bg-indigo-600 hover:bg-indigo-700 focus:outline-none focus:ring-2 focus:ring-offset-2
focus:ring-indigo-500 disabled:opacity-50"
    >
     {loading ? 'Evaluating...' : 'Evaluate Model'}
    </button>

    {metrics && (
     <div className="mt-6 space-y-4">
      <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
       {Object.entries(metrics).map(([model, data]) => {
        if (model === 'dataset_info') return null;
        return (
         <div key={model} className="p-4 bg-gray-50 rounded-lg">
          <h3 className="font-semibold mb-2 capitalize">{model.replace('_', ' ')}</h3>
```

```
              <div className="space-y-1">
                <p>Recall: {(data.recall * 100).toFixed(2)}%</p>
                <ConfusionMatrix matrix={data.confusion_matrix} />
              </div>
            </div>
          );
        })}
      </div>

      <div className="p-4 bg-gray-50 rounded-lg">
        <h3 className="font-semibold mb-2">Dataset Information</h3>
        <div className="grid grid-cols-2 md:grid-cols-4 gap-4">
          <div>
            <p>Total Samples</p>
            <p className="font-medium">{metrics.dataset_info.total_samples}</p>
          </div>
          <div>
            <p>Fraud Samples</p>
            <p className="font-medium">{metrics.dataset_info.fraud_samples}</p>
          </div>
          <div>
            <p>Non-Fraud Samples</p>
            <p className="font-medium">{metrics.dataset_info.non_fraud_samples}</p>
          </div>
          <div>
            <p>Fraud Ratio</p>
            <p className="font-medium">{(metrics.dataset_info.fraud_ratio * 100).toFixed(2)}%</p>
          </div>
        </div>
      </div>
    )}

    {error && (
      <div className="mt-4 p-4 bg-red-50 text-red-700 rounded-md">
        {error}
      </div>
    )}
  </div>
 </div>
);
};
import React, { useState } from 'react';
import { Transaction, TransactionType } from '../../types/transaction';
import { api } from '../../services/api';
import {TransactionValidationResult} from "../../types/api";

export const TransactionForm: React.FC = () => {
 const [loading, setLoading] = useState(false);
 const [error, setError] = useState<string | null>(null);
 const [result, setResult] = useState<TransactionValidationResult | null>(null);

 const [formData, setFormData] = useState<Transaction>({
  step: 1,
  type: TransactionType.CASH_IN,
  amount: 0,
  nameOrig: '',
  oldbalanceOrg: 0,
  newbalanceOrig: 0,
```

```
  nameDest: '',
  oldbalanceDest: 0,
  newbalanceDest: 0,
  userName: '',
 });

 console.log(result);
 const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault();
  setLoading(true);
  setError(null);

  try {
   const response = await api.validateTransaction(formData);

   setResult(response);
  } catch (err) {
   setError(err instanceof Error ? err.message : 'An error occurred');
  } finally {
   setLoading(false);
  }
 };

 const handleInputChange = (e: React.ChangeEvent<HTMLInputElement | HTMLSelectElement>) => {
  const { name, value } = e.target;
  setFormData(prev => ({
   ...prev,
   [name]: e.target.type === 'number' ? parseFloat(value) || 0 : value
  }));
 };

 return (
  <div className="bg-white shadow sm:rounded-lg p-6">
   <form onSubmit={handleSubmit} className="space-y-6">
    <div className="grid grid-cols-1 gap-6 sm:grid-cols-2">
     {/* Username field */}
     <div className="col-span-2">
      <label htmlFor="userName" className="block text-sm font-medium text-gray-700">
       Username
      </label>
      <input
       id="userName"
       name="userName"
       type="text"
       required
       value={formData.userName}
       onChange={handleInputChange}
                    className="mt-1 block w-full rounded-md border-gray-300 shadow-sm focus:border-indigo-500
focus:ring-indigo-500 sm:text-sm"
      />
     </div>

     {/* Transaction Type */}
     <div>
      <label htmlFor="type" className="block text-sm font-medium text-gray-700">
       Transaction Type
      </label>
      <select
       id="type"
```

```
          name="type"
          value={formData.type}
          onChange={handleInputChange}
                        className="mt-1 block w-full rounded-md border-gray-300 shadow-sm focus:border-indigo-500
focus:ring-indigo-500 sm:text-sm"
         >
          {Object.values(TransactionType).map((type) => (
           <option key={type} value={type}>
            {type}
           </option>
          ))}
         </select>
        </div>

        {/* Amount */}
        <div>
         <label htmlFor="amount" className="block text-sm font-medium text-gray-700">
          Amount
         </label>
         <input
          id="amount"
          name="amount"
          type="number"
          required
          value={formData.amount}
          onChange={handleInputChange}
                        className="mt-1 block w-full rounded-md border-gray-300 shadow-sm focus:border-indigo-500
focus:ring-indigo-500 sm:text-sm"
         />
        </div>

        {/* Origin Account Details */}
        <div className="col-span-2">
         <h3 className="text-lg font-medium text-gray-900 mb-3">Origin Account Details</h3>
         <div className="grid grid-cols-1 gap-4 sm:grid-cols-3">
          <div>
           <label htmlFor="nameOrig" className="block text-sm font-medium text-gray-700">
            Account ID
           </label>
           <input
            id="nameOrig"
            name="nameOrig"
            type="text"
            required
            value={formData.nameOrig}
            onChange={handleInputChange}
                          className="mt-1 block w-full rounded-md border-gray-300 shadow-sm focus:border-indigo-500
focus:ring-indigo-500 sm:text-sm"
           />
          </div>
          <div>
           <label htmlFor="oldbalanceOrg" className="block text-sm font-medium text-gray-700">
            Old Balance
           </label>
           <input
            id="oldbalanceOrg"
            name="oldbalanceOrg"
            type="number"
            required
```

```
            value={formData.oldbalanceOrg}
            onChange={handleInputChange}
                        className="mt-1 block w-full rounded-md border-gray-300 shadow-sm focus:border-indigo-500
focus:ring-indigo-500 sm:text-sm"
           />
          </div>
          <div>
           <label htmlFor="newbalanceOrig" className="block text-sm font-medium text-gray-700">
            New Balance
           </label>
           <input
            id="newbalanceOrig"
            name="newbalanceOrig"
            type="number"
            required
            value={formData.newbalanceOrig}
            onChange={handleInputChange}
                        className="mt-1 block w-full rounded-md border-gray-300 shadow-sm focus:border-indigo-500
focus:ring-indigo-500 sm:text-sm"
           />
          </div>
         </div>
        </div>

        {/* Destination Account Details */}
        <div className="col-span-2">
         <h3 className="text-lg font-medium text-gray-900 mb-3">Destination Account Details</h3>
         <div className="grid grid-cols-1 gap-4 sm:grid-cols-3">
          <div>
           <label htmlFor="nameDest" className="block text-sm font-medium text-gray-700">
            Account ID
           </label>
           <input
            id="nameDest"
            name="nameDest"
            type="text"
            required
            value={formData.nameDest}
            onChange={handleInputChange}
                        className="mt-1 block w-full rounded-md border-gray-300 shadow-sm focus:border-indigo-500
focus:ring-indigo-500 sm:text-sm"
           />
          </div>
          <div>
           <label htmlFor="oldbalanceDest" className="block text-sm font-medium text-gray-700">
            Old Balance
           </label>
           <input
            id="oldbalanceDest"
            name="oldbalanceDest"
            type="number"
            required
            value={formData.oldbalanceDest}
            onChange={handleInputChange}
                        className="mt-1 block w-full rounded-md border-gray-300 shadow-sm focus:border-indigo-500
focus:ring-indigo-500 sm:text-sm"
           />
          </div>
          <div>
```

```
            <label htmlFor="newbalanceDest" className="block text-sm font-medium text-gray-700">
              New Balance
            </label>
            <input
              id="newbalanceDest"
              name="newbalanceDest"
              type="number"
              required
              value={formData.newbalanceDest}
              onChange={handleInputChange}
                        className="mt-1 block w-full rounded-md border-gray-300 shadow-sm focus:border-indigo-500
focus:ring-indigo-500 sm:text-sm"
            />
          </div>
        </div>
      </div>
    </div>

    <div className="flex justify-end">
      <button
        type="submit"
        disabled={loading}
            className="inline-flex justify-center rounded-md border border-transparent bg-indigo-600 py-2 px-4 text-sm
font-medium text-white shadow-sm hover:bg-indigo-700 focus:outline-none focus:ring-2 focus:ring-indigo-500
focus:ring-offset-2 disabled:opacity-50"
      >
        {loading ? 'Validating...' : 'Validate Transaction'}
      </button>
    </div>
  </form>

  {error && (
    <div className="mt-4 p-4 bg-red-50 text-red-700 rounded-md">
      {error}
    </div>
  )}

  {result !== null && (
          <div  className={"mt-4  p-4  text-green-700  rounded-md"  +  (!result.result.is_suspicious  ?  "bg-green-50"  :
"bg-red-50")}>
      <h3 className="font-medium">Validation Result</h3>
      <div className="grid grid-cols-1 gap-4 sm:grid-cols-3">
        <p>Fraud probability: {(result.result.fraud_probability * 100).toFixed(2)}%</p>
        {result.result.is_suspicious && (
          <p className="text-red-700 rounded-md">Suspicious</p>
        )}
      </div>
    </div>
  )}
 </div>
);
};
import React from 'react';
import { Link, useLocation } from 'react-router-dom';

export const Navigation: React.FC = () => {
 const location = useLocation();

 const isActive = (path: string) =>
```

```
    location.pathname === path ? 'bg-gray-900 text-white' : 'text-gray-300 hover:bg-gray-700 hover:text-white';

  return (
    <nav className="bg-gray-800">
      <div className="max-w-7xl mx-auto px-4">
        <div className="flex items-center justify-between h-16">
          <div className="flex items-center">
            <div className="flex space-x-4">
              <Link to="/" className={`px-3 py-2 rounded-md text-sm font-medium ${isActive('/')}`}>
                Validate
              </Link>
              <Link to="/upload" className={`px-3 py-2 rounded-md text-sm font-medium ${isActive('/upload')}`}>
                Upload
              </Link>
              <Link to="/evaluate" className={`px-3 py-2 rounded-md text-sm font-medium ${isActive('/evaluate')}`}>
                Evaluate
              </Link>
              <Link to="/health" className={`px-3 py-2 rounded-md text-sm font-medium ${isActive('/health')}`}>
                Health
              </Link>
            </div>
          </div>
        </div>
      </div>
    </nav>
  );
};
import React, { PropsWithChildren } from 'react';
import { Header } from './Header';

export const Layout: React.FC<PropsWithChildren> = ({ children }) => {
  return (
    <div className="min-h-screen bg-gray-100">
      <Header />
      <main className="max-w-7xl mx-auto py-6 sm:px-6 lg:px-8">
        {children}
      </main>
    </div>
  );
};
import React from 'react';

export const Header: React.FC = () => {
  return (
    <header className="bg-white shadow">
      <div className="max-w-7xl mx-auto py-6 px-4">
        <h1 className="text-3xl font-bold text-gray-900">
          Transaction Validator
        </h1>
      </div>
    </header>
  );
};
```

ДОДАТОК В

Ілюстративний матеріал

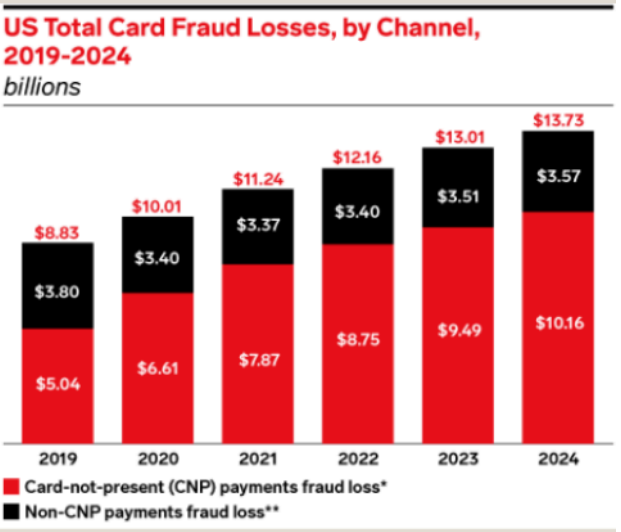
Підвищення захищеності системи запобігання фродовим атакам за допомогою вдосконаленої комбінованої системи на основі svm, логістичної регресії та вагового підходу

Виконав студент групи КІТС-23М - Карпілов.В.І

Керівник - професор д.т.н. Яремчук Ю.Є

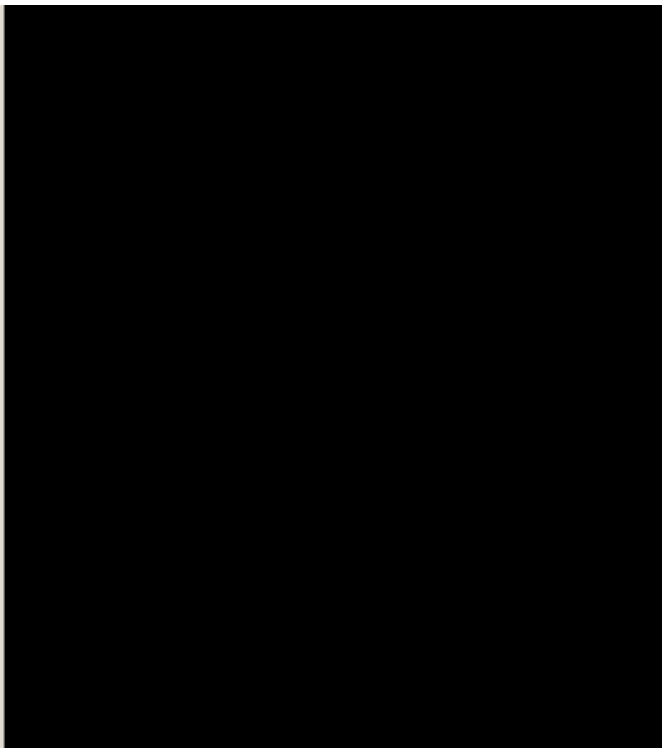
Актуальність

З кожним роком кількість втрат від фродових атак зростає, також з'являються нові загрози такі як атаки на операції з криптовалютами, для яких сучасні системи погано пристосовані. Основною категорією що зростає є без використання фізичної платіжної карти



Мета та задачі

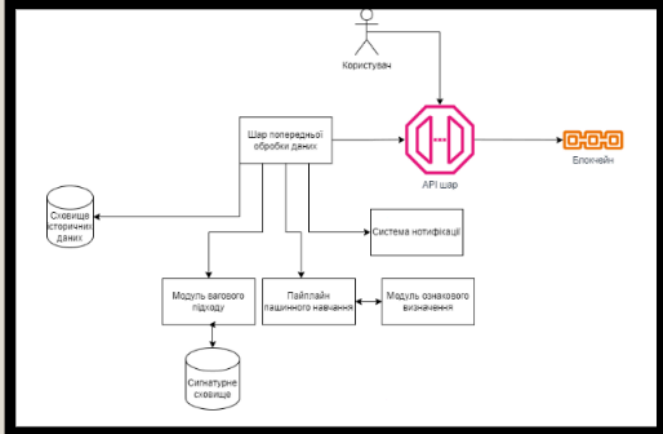
Метою даного дослідження є розробка та впровадження вдосконаленої комбінованої системи для підвищення захищеності від фродових атак, що базується на використанні підтримуючих векторних машин (SVM), логістичної регресії та вагового підходу



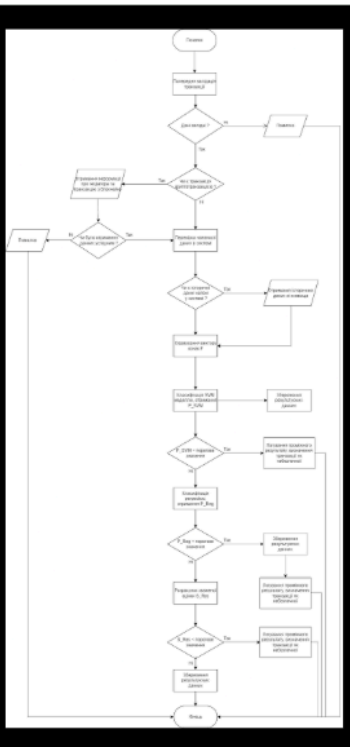
Існуючі рішення

Характеристика	CARDWATCH	Stripe Radar	NICE Actimize	SAS	AWS
Доступність для малого бізнесу	Обмежена	Середня	Низька	Низька	Середня
Вартість впровадження	Висока	Низька	Дуже висока	Дуже висока	Середня
Необхідність спеціалістів	Так	Ні	Так	Так	Так
Інтеграція з локальними системами	Складна	Проста	Складна	Складна	Середня
Вимоги до інфраструктури	Високі	Низькі	Високі	Високі	Середні

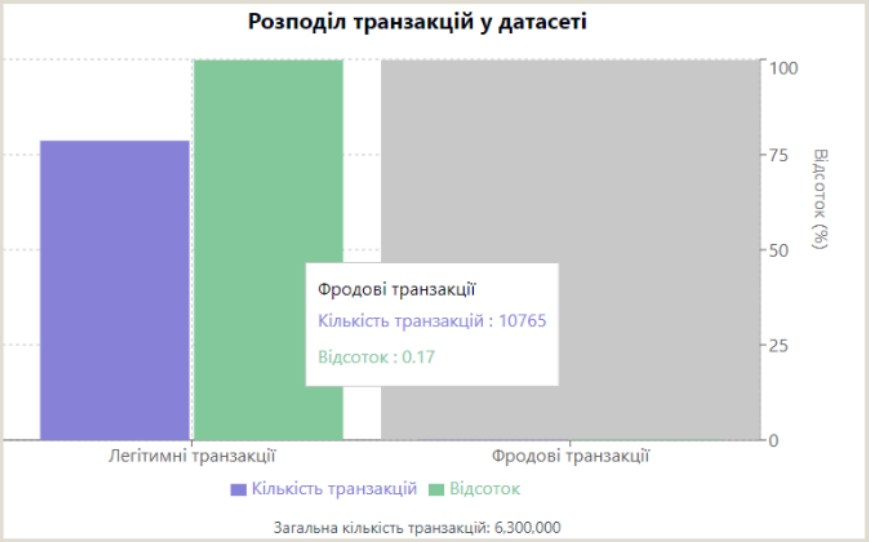
Архітектура запропонованого рішення



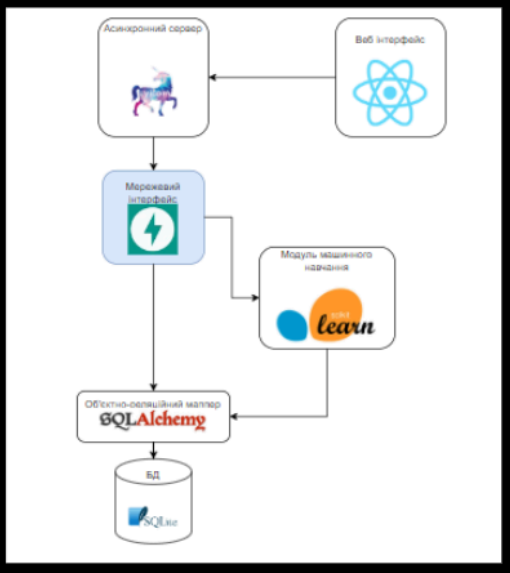
Алгоритм запропонованого рішення



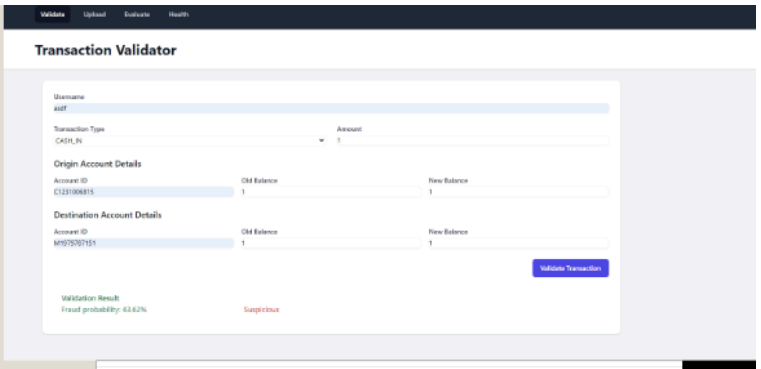
Характеристики обраного датасету



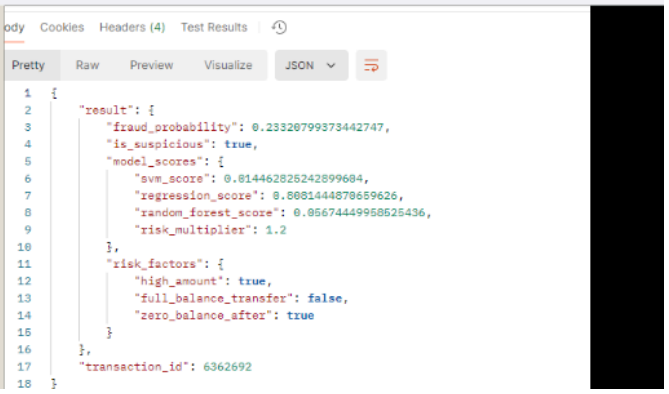
Технічна реалізація



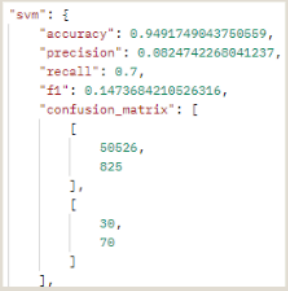
Результат роботи системи на графічному інтерфейсі-

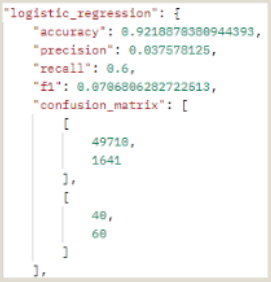


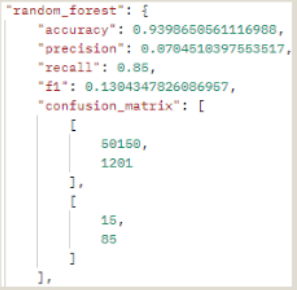
Відображення безпосередньо відповіді сервера-



Оцінка точності системи







Висновок

В результаті виконаної роботи було досліджено існуючі методи та системи запобігання фродовим атакам, розроблено та впроваджено комбіновану систему захисту на основі методів машинного навчання. Поставлена мета щодо підвищення захищеності від фродових атак була досягнута, що підтверджується результатами тестування системи.
Хоча розроблена система має певні обмеження та потребує подальшої оптимізації, вона демонструє достатню ефективність для застосування, забезпечуючи виявлення більшості шахрайських транзакцій при прийнятному рівні помилкових спрацьовувань.

ДОДАТОК Г

Протокол перевірки на антиплагіат

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи:

Підвищення захищеності системи запобігання фродовим атакам за допомогою вдосконаленої комбінованої системи на основі SVM, логістичної регресії та вагового підходу

Тип роботи: магістерська кваліфікаційна робота

Підрозділ Кафедра менеджменту на безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки
Гр. КІТС-23м

Керівник професор Яремчук Ю.Є.

Показники звіту подібності

Turnitin	
Оригінальність	96%
Загальна схожість	4%

Аналіз звіту подібності (відмітити потрібне)

- Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
- Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений (-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ Карпілов В.І.

(підпис) (прізвище, ініціали)

Опис прийнятого рішення

- **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

Особа, відповідальна за перевірку _____ Коваль Н.П.

(підпис) (прізвище, ініціали)

Експерт _____ _____

(за потреби) (підпис) (прізвище, ініціали, посада)А