МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ
Циклова комісія комп’ютерної інженерії та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему

ІНТЕЛЕКТУАЛЬНИЙ ДОДАТОК ДЛЯ ВИЯВЛЕННЯ ШАХРАЙСТВ

Виконав: студент групи 2П-22
Спеціальності
121 Інженерія програмного забезпечення
Єгор ПЕЧЕРСЬКИЙ
Керівник:
Станіслав МАРЧЕНКО

Черкаси 2026

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ

Циклова комісія комп'ютерної інженерії та інформаційних технологій

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувачка циклової комісії КІ та ІТ

_____ Наталя ФАЛЬЧЕНКО
(підпис)

«_____» _____2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Печерському Єгору Владиславовичу

1. Тема кваліфікаційної роботи Інтелектуальний додаток для виявлення шахрайств

Керівник роботи Марченко Станіслав Віталійович, викладач першої категорії

затверджені наказом закладу фахової передвищої освіти від «__» _____ 2025 року № __у.

2. Строк подання студентом кваліфікаційної роботи 02.06.2026

3. Вихідні дані до кваліфікаційної роботи: мова програмування Python, бібліотеки та фреймворки для аналізу даних і машинного навчання Pandas, NumPy, Scikit-learn, Streamlit; середовище розробки Visual Studio Code; набір даних Credit Card Fraud Detection; алгоритми машинного навчання Random Forest, Logistic Regression, Gradient Boosting, Hist Gradient Boosting; метрики оцінювання якості класифікації Accuracy, Precision, Recall, F1-score, ROC-AUC; засоби проєктування UML-діаграм.

4. Зміст кваліфікаційної роботи: теоретичні основи та аналіз методів виявлення шахрайства (класифікація шахрайських операцій та їхні прояви на рівні даних; еволюція алгоритмів і методів виявлення шахрайства; огляд програмних рішень для виявлення шахрайства; постановка задачі на розроблення), проєктування та реалізація програмного забезпечення (аналіз вимог до програмного забезпечення; попередня архітектура програмного забезпечення; деталізована архітектура та проєктування бази даних; особливості програмної імплементації), тестування та

експериментальна оцінка інтелектуального додатка (вибір методів тестування програмного продукту; формування тест-плану; організація та проведення тестування; результати комп'ютерних експериментів).

5. Дата видачі завдання 15.09.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	14.10.2025	
2	Розділ 1. Теоретичні основи та аналіз методів виявлення шахрайства	09.12.2025	
3	Розділ 2. Проєктування та реалізація програмного забезпечення	10.03.2026	
4	Розділ 3. Тестування та експериментальна оцінка інтелектуального додатка	28.04.2026	
5	Висновки	12.05.2026	
6	Оформлення кваліфікаційної роботи (чистовий варіант)	26.05.2026	
7	Перевірка кваліфікаційної роботи на наявність ознак плагіату (за 10 днів до захисту)	02.06.2026	
8	Подання кваліфікаційної роботи на затвердження завідувачці циклової комісії (за 7 днів до захисту)	10.06.2026	

Студент _____ Єгор ПЕЧЕРСЬКИЙ
(підпис)

Керівник роботи _____ Станіслав МАРЧЕНКО
(підпис)

АНОТАЦІЯ

Кваліфікаційну роботу присвячено розробленню інтелектуального додатка для автоматизованого виявлення шахрайських фінансових транзакцій із використанням методів машинного навчання. У ній проаналізовано основні типи шахрайських операцій та схеми з використанням фінансових посередників. Розглянуто прояви шахрайства на рівні даних: транзакційні, поведінкові, контекстні та мережеві ознаки. Досліджено еволюцію методів виявлення шахрайства: від експертних правил і статистичного аналізу до класичного машинного навчання, ансамблевих моделей, графових підходів і потокової обробки подій.

Практичним результатом роботи є програмний додаток, реалізований мовою Python із використанням Streamlit для побудови користувацького інтерфейсу. Система забезпечує завантаження транзакційних даних, їхню валідацію та профілювання, підготовку ознак, навчання моделей, прогнозування шахрайських операцій, формування багатокомпонентного ризикового бала та експорт результатів. У роботі спроєктовано архітектуру програмного забезпечення, описано основні модулі системи, структуру даних, базу даних, сценарії використання та механізми тестування.

Експериментальну перевірку виконано на наборі даних Credit Card Fraud Detection. Для порівняння використано моделі Random Forest, Logistic Regression, Gradient Boosting та Hist Gradient Boosting. Найкращий баланс метрик продемонструвала модель Gradient Boosting. Проведене тестування підтвердило працездатність реалізованого додатка, коректність основних сценаріїв обробки даних і можливість використання системи як навчального та експериментального інструменту для аналізу шахрайських транзакцій.

Ключові слова: ФІНАНСОВЕ ШАХРАЙСТВО, МАШИННЕ НАВЧАННЯ, КЛАСИФІКАЦІЯ ТРАНЗАКЦІЙ, РИЗИКОВИЙ БАЛ, PYTHON, STREAMLIT, RANDOM FOREST, GRADIENT BOOSTING, ROC-AUC, F1-МІРА.

ABSTRACT

The qualification work is devoted to the development of an intelligent application for automated detection of fraudulent financial transactions using machine learning methods. It analyzes the main types of fraudulent operations and schemes involving financial intermediaries. The work examines fraud manifestations at the data level, including transactional, behavioral, contextual, and network features. It also studies the evolution of fraud detection methods: from expert rules and statistical analysis to classical machine learning, ensemble models, graph-based approaches, and stream processing of events.

The practical result of the work is a software application implemented in Python using Streamlit to build the user interface. The system provides transaction data loading, validation and profiling, feature preparation, model training, prediction of fraudulent operations, generation of a multi-component risk score, and export of results. The work presents the software architecture, describes the main system modules, data structure, database, use cases, and testing mechanisms.

The experimental evaluation was performed on the Credit Card Fraud Detection dataset. Random Forest, Logistic Regression, Gradient Boosting, and Hist Gradient Boosting models were used for comparison. The Gradient Boosting model demonstrated the best balance of metrics. The conducted testing confirmed the operability of the implemented application, the correctness of the main data processing scenarios, and the possibility of using the system as an educational and experimental tool for analyzing fraudulent transactions.

Keywords: FINANCIAL FRAUD, MACHINE LEARNING, TRANSACTION CLASSIFICATION, RISK SCORE, PYTHON, STREAMLIT, RANDOM FOREST, GRADIENT BOOSTING, ROC-AUC, F1-SCORE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ, ПОЗНАЧЕНЬ

AI	Artificial Intelligence – штучний інтелект
ML	Machine Learning – машинне навчання
DL	Deep Learning – глибоке навчання
GNN	Graph Neural Network – графова нейронна мережа
ATO	Account Takeover – захоплення облікового запису
CNP	Card-Not-Present – операція без фізичної присутності картки
APP scam	Authorised Push Payment scam – шахрайство з авторизованим переказом
Card-present	операція з фізичною присутністю картки
Money mule	фінансовий посередник / рахунок-посередник
EFM	Enterprise Fraud Management – корпоративне управління шахрайством
EPC	European Payments Council – Європейська платіжна рада
EBA	European Banking Authority – Європейський банківський орган
ECB	European Central Bank – Європейський центральний банк
OCC	Office of the Comptroller of the Currency – Управління контролера грошового обігу США
FinCEN	Financial Crimes Enforcement Network – Мережа боротьби з фінансовими злочинами США
SAS	Statistical Analysis System; SAS Fraud Management
FICO	Fair Isaac Corporation; FICO Falcon Platform
API	Application Programming Interface – інтерфейс прикладного програмування
POS	Point of Sale – точка продажу / платіжний термінал
ROC	Receiver Operating Characteristic – робоча характеристика приймача
ROC-AUC	Area Under the ROC Curve – площа під ROC-кривою
TP	True Positive – істинно позитивний результат

TN	True Negative – істинно негативний результат
FP	False Positive – хибно позитивний результат
FN	False Negative – хибно негативний результат
Precision	точність позитивних передбачень
Recall	повнота / чутливість
F1-score	F1-міра
PCA	Principal Component Analysis – метод головних компонент

ВСТУП

Фінансове шахрайство є складним соціально-економічним явищем, що проявляється у здійсненні протиправних дій з метою отримання матеріальної вигоди шляхом обману або зловживання довірою. У сучасних умовах розвитку банківських технологій шахрайство набуває нових форм, пов’язаних із цифровізацією фінансових послуг та зростанням кількості онлайн-транзакцій. У банківській системі фінансове шахрайство безпосередньо впливає на стабільність установ, рівень довіри клієнтів та загальну безпеку фінансового сектору. Втрати від шахрайських операцій можуть бути як прямими (грошові збитки), так і непрямими (репутаційні ризики, зниження клієнтської бази тощо).

Стрімкий розвиток цифрових фінансових сервісів та зростання обсягів безготівкових операцій викликає значне підвищення ризику виникнення шахрайських дій. Традиційні методи протидії шахрайству, засновані на фіксованих правилах, втрачають ефективність через їхню неадаптивність до нових схем атак [1]. У зв’язку з цим особливої актуальності набуває автоматизація процесів виявлення шахрайських транзакцій із застосуванням сучасних методів аналізу даних та машинного навчання.

Ступінь дослідженості проблеми є достатньо високим, оскільки питання виявлення фінансового шахрайства активно розглядається як у наукових працях [2-4], так і у практичних рішеннях [5; 6]. Значна кількість досліджень присвячена використанню алгоритмів машинного навчання, зокрема ансамблевих методів, для підвищення точності класифікації транзакцій. Проте різноманітність стратегій боротьби з фінансовим шахрайством, бізнес-процесів та контекстів використання сприяє розвитку та появі нових або адаптації старих інтелектуальних моделей до фактичних умов підприємств та організацій. Звідси, тема залишається актуальною для подальшого розгляду.

Об’єктом дослідження виступають фінансові транзакції у банківських системах та моделі виявлення шахрайських операцій на їх основі.

Предметом дослідження виступають інтелектуальні методи, алгоритми та технології виявлення шахрайства з подальшим порівняльним аналізом якості детектування.

Метою роботи є розроблення інтелектуальної системи автоматизованого виявлення шахрайських транзакцій із використанням сучасних методів машинного навчання та порівняння якості отриманих моделей на основі типових метрик для задач бінарної класифікації.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– проаналізувати основні види шахрайських операцій та їхні прояви в інформаційних системах фінансових установ;

– сформувати множину моделей детекції шахрайства для проведення комп’ютерних експериментів, обґрунтувати вибір технологій реалізації;

– спроєктувати архітектуру програмного комплексу та реалізувати моделі класифікації транзакцій;

– провести експериментальне дослідження якості та ефективності автоматизованої системи виявлення шахрайських транзакцій.

Практична значущість роботи полягає в можливості використання розробленої системи для підвищення ефективності виявлення шахрайських операцій у банківському секторі. Запропоноване рішення може бути інтегроване в наявні інформаційні системи фінансових установ, що дозволить зменшити рівень фінансових втрат та забезпечити достатній рівень безпеки клієнтів.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ МЕТОДІВ ВИЯВЛЕННЯ ШАХРАЙСТВА

1.1 Класифікація шахрайських операцій та їхні прояви на рівні даних

Фінансове шахрайство в сучасних платіжних системах доцільно розглядати як динамічний клас протиправних дій, що змінюються разом із розвитком цифрових каналів обслуговування, механізмів автентифікації та міжбанківської взаємодії. Актуальні регуляторні та галузеві огляди показують, що зростання цифрових платежів не супроводжується пропорційним зниженням шахрайських ризиків: у 2024 р. в Європейській економічній зоні загальна вартість шахрайських платіжних операцій досягла 4,2 млрд євро, причому найбільші абсолютні втрати були пов’язані з картковими платежами та кредитовими переказами [7]. Водночас сучасний стан досліджень свідчить, що саме різнорідність шахрайських сценаріїв, нестабільність їхніх ознак і типологічна неоднорідність у професійній літературі ускладнюють побудову цілісної наукової класифікації [8; 9].

Принципово важливо розмежовувати класифікаційні осі. Наприклад, шахрайство без фізичної присутності картки (card-not-present, CNP) описує насамперед канал та умови ініціювання платежу; захоплення облікового запису (account takeover, ATO) характеризує спосіб компрометації доступу; натомість схеми з фінансовими посередниками (money mule schemes) відображають уже етап виведення, розосередження або маскування коштів. Через це перелічувати вище згадані типи в одному ряду без додаткового уточнення некоректно: це явища різного рівня узагальнення, що відносяться відповідно до каналу операції, механізму захоплення контролю та топології монетизації злочинного доходу [8; 10; 11]. Така типологічна нечіткість прямо фіксується і в кримінологічних оглядах, де підкреслюється непослідовність у звітуванні про «типології» та необхідність їх стандартизації [10].

З огляду на це, для подальшого аналізу доцільно використовувати щонайменше три взаємодоповнювальні площини класифікації: 1) за платіжним

інструментом і каналом виконання, 2) за механізмом отримання контролю над платіжною дією, 3) за способом подальшого переміщення та приховування коштів (рис. 1.1). Така побудова краще відповідає сучасним регуляторним звітам, у яких шахрайство аналізується окремо за картковими платежами, кредитовими переказами, прямим дебетуванням, операціями з електронними грошима та зняттям готівки, а також окремо – за сценаріями соціальної інженерії, компрометації облікових даних і подальшої монетизації [7; 8].

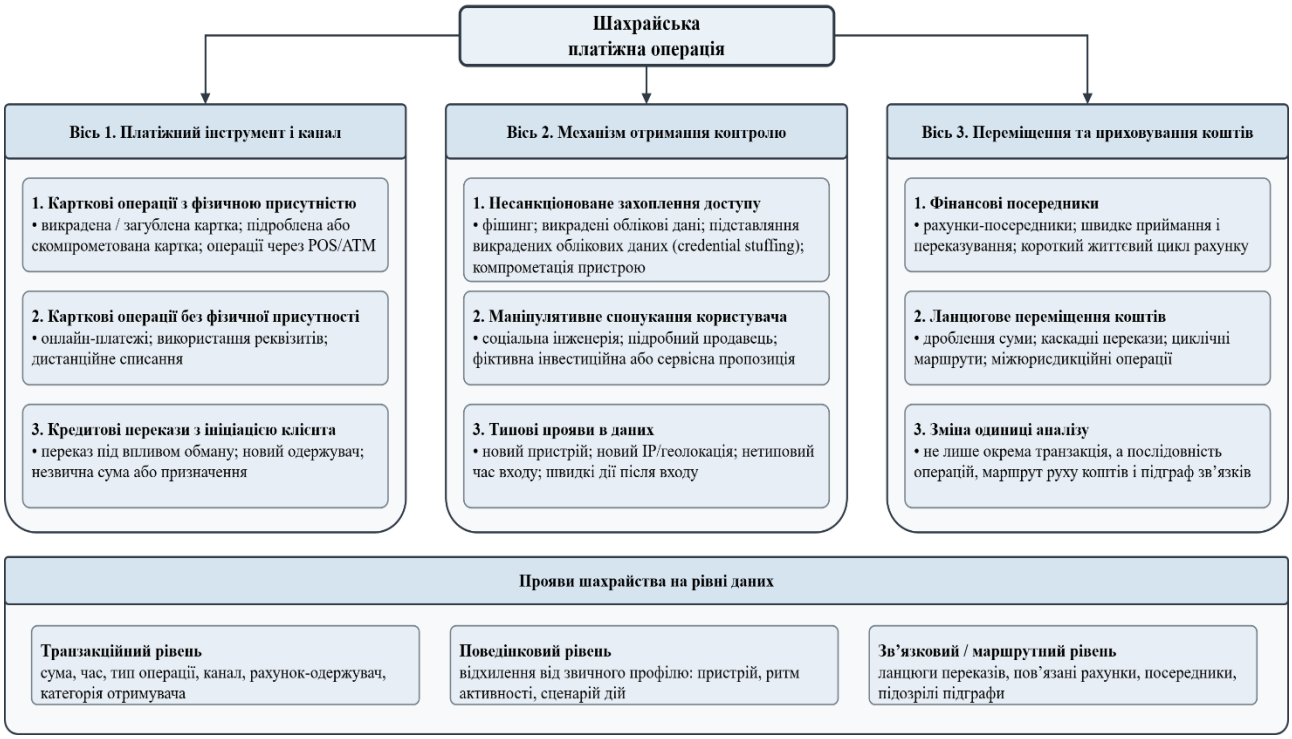


Рисунок 1.1 – Багатовимірна класифікація шахрайських операцій

На першому рівні доцільно виокремлювати шахрайство в карткових платежах з фізичною присутністю картки (card-present) та без неї (card-not-present). За визначенням Європейської платіжної ради, CNP-операція – це карткова транзакція без фізичної взаємодії картки з точкою ініціювання в момент операції, тобто дистанційна карткова транзакція [8]. Натомість card-present охоплює шахрайські дії у фізичних середовищах оплати або зняття готівки (терміналах торговельних точок і банкоматах), зокрема випадки використання викраденої, загубленої, підробленої або перехопленої під час

доставки картки [8; 14]. Важливо, що навіть після широкого впровадження чипових карток card-present шахрайство не зникло: за американськими даними, у 2023 р. його рівні залишалися суттєвими, а в окремих мережах знову демонстрували зростання [12].

Разом із тим саме дистанційні операції сьогодні формують найбільш проблемний сегмент ризику. Європейські регуляторні дані показують, що шахрайство, пов'язане з картковими платежами, кредитовими переказами та електронними грошима, переважно концентрується у віддалених каналах, хоча сам масив карткових платежів за своєю природою все ще значною мірою залишається не дистанційним [7]. Це означає, що ризик шахрайства розподілений нерівномірно: віддалений канал не обов'язково має найбільший обсяг транзакцій, але саме він характеризується підвищеною криміногенністю. Американські дані підтверджують цю тенденцію: рівень CNP-шахрайства у 2023 році продовжив зростати, а в окремих мережах перевищив рівень шахрайства в операціях із фізичною присутністю картки [13]. Отже, поділ на card-present і card-not-present має не лише технічний, а й аналітичний зміст, оскільки відображає різний профіль уразливостей і різний склад цифрових слідів у даних [7; 8; 14].

Варто також виокремити шахрайство в кредитових переказах та інші форми платежів, де формально операція може бути коректно авторизована користувачем, але сама авторизація досягається шляхом обману. У звіті EPC такий сценарій визначено як шахрайство з авторизованим переказом за підробним спонуканням (authorised push payment scam, APP scam): жертву вводять в оману і спонукають самостійно переказати кошти на рахунок, контрольований зловмисником. Цей різновид є принципово важливим, оскільки він руйнує спрощену дихотомію «авторизована = легітимна / неавторизована = шахрайська». У низці сучасних сценаріїв саме формально коректна автентифікація не спростовує наявності шахрайства, якщо волевиявлення платника було викривлене соціальною інженерією [8].

На другому рівні класифікації ключовим є питання: як саме зловмисник отримує можливість ініціювати або провести підозрілу операцію. Тут доцільно розмежовувати принаймні дві великі групи: неавторизоване захоплення доступу та маніпулятивне спонукання до дії. До першої належить, зокрема, захоплення облікового запису (account takeover, ATO), коли первинною мішенню стає клієнт фінансової установи, а кінцева мета полягає в вилученні або переказі коштів шляхом несанкціонованого доступу до рахунку. FinCEN прямо відмежовує ATO від загальнішої категорії несанкціонованого електронного проникнення: у випадку ATO об'єктом атаки є не інфраструктура банку, а конкретний клієнтський обліковий запис [11]. Управління контролера грошового обігу США (OCC) також відзначає, що такі атаки дедалі частіше спираються на фішинг та викрадені облікові дані, а також можуть поєднуватися з компрометацією службової електронної пошти та шахрайськими платіжними інструкціями [13].

До другої групи належать сценарії, у яких користувач сам проходить автентифікацію, але робить це під впливом обману, фальшивої легітимації або психологічного тиску. Саме тому сучасні галузеві документи дедалі частіше описують шахрайство не лише через категорії «несанкціонований доступ» і «крадіжка реквізитів», а й через спектр соціально-інженерних механізмів: фішинг, голосові та текстові варіанти обману, підробні торговці, фіктивні інвестиційні чи сервісні пропозиції [8; 13]. Для класифікації це має важливий наслідок: одна й та ж платіжна операція може бути коректно підтверджена засобами сильної клієнтської автентифікації, але залишатися шахрайською за своєю економічною сутністю [8]. Отже, класифікація лише за технічним фактом автентифікації є недостатньою.

Третя площина класифікації пов'язана не з початком інциденту, а з тим, як злочинний дохід виводиться з-під прямого контролю фінансової установи. Сюди належать схеми з використанням фінансових посередників (money mules), які не стільки ініціюють первинне шахрайство, скільки забезпечують його завершальну фазу: приймання коштів, подальше дроблення, переказування,

конвертацію або зняття готівки. EPC прямо описує типові потоки, у яких рахунки таких посередників використовуються для приймання шахрайських переказів і подальшого приховування сліду коштів; особливо ускладнюють розслідування транскордонні та миттєві перекази [8]. У ширшому контексті кримінологічний огляд [10] показує, що відмивання коштів і пов’язані з ним типології загалом характеризуються великою кількістю інструментів вартості, акторів та індикаторів ризику, а отже, мають мережеву, а не суто транзакційну природу.

Звідси випливає важливий аналітичний висновок: money mule не слід трактувати як самостійний однорівневий «тип транзакційного шахрайства» на одному рівні з card-present або ATO. Це радше роль у злочинній інфраструктурі або елемент ланцюга переміщення коштів. У прикладному контексті таке розмежування є принциповим, оскільки воно змінює саму одиницю аналізу: якщо для CNP достатньо аналізувати окрему операцію, то для money mule потрібно переходити до послідовностей операцій, пов’язаних рахунків, маршруту руху коштів і міжсуб’єктних зв’язків [8; 10].

На рівні даних шахрайська активність проявляється не однією «універсальною ознакою», а сукупністю відхилень, які слід розглядати в кількох аспектах. Перший аспект становлять власне транзакційні атрибути: час операції, сума, тип операції, рахунок-одержувач, канал виконання, інколи також категорія отримувача чи інші реквізити платіжного контексту. Огляди сучасних досліджень прямо вказують, що транзакційні набори даних зазвичай є структурованими табличними масивами з ознаками на кшталт часу, суми, типу транзакції, рахунку отримувача, вікової групи чи інших соціально-економічних параметрів [14]. При цьому окрема ознака сама по собі майже ніколи не є достатньою для надійного висновку: підозрілим є не просто «великий платіж» чи «нічна транзакція», а їх невідповідність контексту конкретного клієнта або ланцюга операцій [9; 14].

Другий аспект утворюють поведінкові прояви. У сучасній літературі вони описуються як відхилення від звичного профілю клієнта: частоти дій, ритму активності, характерного способу ініціювання платежів, повторюваності

сценаріїв та інших стабільних патернів користувацької поведінки. «Аномальність» не є абсолютною характеристикою операції, вона має відносний, профіле-орієнтований характер. Та ж операція може бути звичайною для одного користувача й аномальною для іншого. Саме тому фінансове шахрайство на рівні даних доцільно розуміти як контекстуальне відхилення, а не просто як статистичний викид у глобальному масиві [9; 14].

Третім аспектом є зв'язкові та маршрутні прояви. Для схем переказного шахрайства, відмивання коштів і використання фінансових посередників значущими стають не лише атрибути окремої транзакції, а й конфігурація зв'язків між рахунками, інструментами вартості, посередниками та часово зближеними переказами [8; 10; 14]. Саме тому кримінологічні та галузеві огляди наголошують на великій кількості «червоних прапорців», які виникають на рівні послідовностей, ланцюгів і мереж, а не в окремому платежі. Звідси, опис проявів шахрайства на рівні даних не можна зводити лише до полів окремої таблиці транзакцій; у значній частині випадків аналізована сутність – це не одиничний запис, а транзакційний слід або підграф фінансової взаємодії.

1.2 Еволюція алгоритмів і методів виявлення шахрайства

Еволюцію засобів виявлення фінансового шахрайства доцільно розглядати як поступове нашарування підходів, кожен із яких виникав у відповідь на конкретні обмеження попереднього. Сучасні огляди показують, що сфера пройшла щонайменше кілька етапів: від експертних правил і статистичного контролю до класичного машинного навчання, далі до глибоких моделей, графових підходів, потокового аналізу та приватного міжорганізаційного навчання. Водночас жоден із цих етапів повністю не скасував попередній. Навпаки, реальні системи дедалі частіше мають гібридну архітектуру, де правила, табличні моделі, часові механізми та графовий аналіз співіснують в одному контурі прийняття рішення [15-18].

Першим домінантним підходом були експертні правила (rule-based expert systems), які десятиліттями спиралися на накопичені галузеві сценарії:

перевищення порога суми, нетипову географію, невідповідність каналу операції профілю клієнта, повторні спроби списання тощо. Навіть у сучасних технічних працях цей етап прямо описується як довготривала індустріальна основа боротьби з шахрайством. Зокрема, у роботі про внутрішньомережеве машинне навчання для низьколатентного виявлення шахрайства [23] автори наголошують, що експертні правила використовувалися десятиліттями для виявлення відомих шахрайських сценаріїв, але виявилися недостатніми для складніших моделей онлайн-шахрайства. Огляди [15-17] підтверджують цю оцінку: правила добре працюють там, де ознаки стабільні, регуляторно прозорі й легко інтерпретовані, але різко втрачають ефективність у разі дрейфу поведінки зловмисників та комбінування ознак із кількох рівнів даних.

На інженерному рівні сильна сторона правил полягає в тому, що вони мають мінімальну затримку виконання, прості для аудиту, легко пояснюються службі фінансового моніторингу й добре узгоджуються з нормативними вимогами. Проте їх слабкість є принциповою: правило реагує переважно на вже відомий патерн, а не на нову комбінацію сигналів. Через це збільшення кількості правил не розв’язує проблему масштабування, а породжує конфлікти між правилами, складність супроводу, накопичення технічного боргу та некероване зростання хибних спрацювань [16; 17; 23]. Отже, правила не слід трактувати як «застарілий» етап; коректніше розглядати їх як базовий, але недостатній шар системи ризикового контролю.

Подальший розвиток пов’язаний із переходом від жорстких правил до статистичних методів та виявлення аномалій (anomaly detection). Їхня поява була логічною відповіддю на те, що шахрайська транзакція часто не порушує явного порога, але є нетиповою щодо індивідуального профілю клієнта або локального розподілу даних. Сучасні систематичні огляди показують, що цей клас підходів відіграв важливу роль у формуванні самої ідеї шахрайства як контекстуального відхилення, а не лише як заздалегідь описаного сценарію [15; 16]. Разом із тим саме на цьому етапі стало зрозуміло, що аномальність не збігається з шахрайством: значна частина статистично нетипових операцій є легітимною,

тому використання лише аномалійного аналізу неминуче підвищує частку хибних тривог [16; 17].

Критично важливо, що статистичні підходи виявили два фундаментальні обмеження предметної області. Перше – сильна незбалансованість класів: частка шахрайських подій у реальних даних зазвичай дуже мала. Друге – нестаціонарність розподілів, коли поведінка клієнтів і шахраїв змінюється швидше, ніж встигають стабілізуватися параметри моделі. Саме ці дві проблеми згодом визначили логіку переходу до машинного навчання, тому що класичні форми статистики гірше пристосовані до багатовимірних, рідкісних і динамічних подій [16; 17; 19].

Найбільш практично значущим етапом стала поява класичного машинного навчання для табличних транзакційних даних: логістичної регресії, дерева рішень, випадкового лісу (random forest), методу опорних векторів (support vector machine), градієнтного підсилення (gradient boosting) та їх ансамблевих комбінацій. Актуальні систематичні огляди свідчать, що саме цей клас моделей сформував «робочу конячку» більшості прикладних систем виявлення транзакційного шахрайства [15-17]. Особливо показовим є те, що навіть на тлі бурхливого розвитку глибокого навчання табличні моделі залишаються конкурентними завдяки добрій роботі зі структурованими ознаками, відносно невисоким вимогам до обсягу даних і кращій операційній пояснюваності [16; 17].

Інженерно цей етап означав перехід від ручного визначення правил до проєктування ознак (feature engineering): агрегатів за часовими вікнами, поведінкових профілів, лічильників подій, відстаней між локаціями, ознак новизни пристрою, ризикових характеристик одержувача тощо. У сучасній літературі саме якість попередньої обробки та ознакового простору розглядається як один із головних чинників підсумкової якості детектора, іноді важливіший за вибір конкретної моделі [17]. Звідси, в задачі виявлення шахрайства модель рідко працює із «сирими» даними; вирішальне значення має побудова стійкого конвеєра ознак, який відображає час, контекст, пов’язаність та поведінкову історію операції [17; 24].

У розвитку алгоритмів окремий поворотний момент пов’язаний не стільки з появою нових архітектур, скільки зі зміною уявлень про те, що саме потрібно оптимізувати. Праця [19], присвячена шахрайству в заявках на відкриття банківських рахунків, прямо демонструє, що в умовах екстремальної незбалансованості вибір цільової функції та критерію оптимізації може бути важливішим за вибір сімейства моделі. Автори показують, що орієнтація на загальну точність або зважену F1-міру створює хибне відчуття успішності, тоді як реальна задача полягає у виявленні рідкісної, але дорогої помилки. Таке положення узгоджується і з сучасними оглядами, де незбалансованість, помилкові спрацювання та дрейф понять визначаються як центральні обмеження всього напряму [16; 17].

Звідси випливає важливий висновок: еволюція методів виявлення шахрайства відбувалася не лише через ускладнення моделей, а й через переосмислення постановки задачі. Для інженера це означає, що неправильно оптимізована система може бути «математично сильною», але операційно слабкою. Якщо модель мінімізує не ті втрати, що є критичними для банку, то навіть висока лабораторна якість не гарантує корисності в промисловій експлуатації [17; 19]. Саме тому сучасні підходи дедалі частіше поєднують вагові схеми, чутливе до вартості помилки навчання (cost-sensitive learning), відбір порогів, багаторівневе спрацювання та подальший ручний розгляд високоризикових випадків [16; 17; 19].

Наступний етап розвитку пов’язаний із поширенням глибокого навчання (deep learning), яке стало одним із найактивніших напрямів дослідження. Систематичний огляд [18] охопив 108 рецензованих праць і показав, що в цій галузі застосовуються згорткові, рекурентні, трансформерні та ансамблеві архітектури, а також автокодувальники для виявлення аномалій. Перевага глибокого навчання полягає в тому, що воно здатне автоматично будувати складні подання даних і краще працювати з часовими залежностями, багатоканальними сигналами та високорозмірними взаємозв’язками. Однак ті ж огляди підкреслюють і системні недоліки: високі вимоги до даних, складніша

підтримка, погіршення пояснюваності та складність відповідності регуляторним вимогам [16-18].

Слід підкреслити, що глибоке навчання не стало універсально кращою заміною табличних моделей. У багатьох прикладних сценаріях, де дані залишаються переважно структурованими табличними записами, добре налаштовані ансамблі дерев рішень і далі демонструють дуже сильні результати. Сучасні огляди [16-18] прямо фіксують, що домінування глибоких моделей у науковій літературі не означає їх автоматичної переваги в операційному середовищі, особливо коли важливі затримка, прозорість, вартість обслуговування й простота перенавчання. Отже, реальний зсув полягає не в перемозі нейронних мереж, а в розширенні класу задач, де можна моделювати часові та приховані залежності, які важко задати вручну.

Подальший розвиток природно привів до часових моделей, оскільки шахрайство рідко є ізольованою подією; частіше воно проявляється як послідовність дій, розгорнута в часі. Саме тому рекурентні мережі, часові згортки, трансформери та інші послідовні архітектури почали розглядатися як засоби моделювання еволюції поведінки клієнта й зловмисника [18]. Проте сучасні дослідження не лише моделюють часову послідовність, а й намагаються виявляти еволюційне шахрайство, де змінюються власне категорії та ознаки атак. У 2024 р. для таких умов було запропоновано EvoFD [22] – онлайн-рамку безперервного навчання (continual learning), призначену для роботи одночасно з відкритою множиною категорій та дрейфом понять. Це зсув позначив, що задача вже формулюється не як статична класифікація, а як постійна адаптація до середовища, що змінюється.

На інженерному рівні це означає, що в сучасній системі виявлення шахрайства недостатньо один раз навчити модель і передати її в промислову експлуатацію. Потрібні механізми виявлення дрейфу, контроль деградації якості, повторне навчання за новими вікнами даних, а інколи й елементи онлайн-оновлення. У цьому сенсі еволюція методів є водночас еволюцією життєвого циклу моделі: від разового навчання до безперервного супроводу, моніторингу

та перевипуску моделей у виробничому середовищі [16; 17; 22]. Саме тут інженерні практики починають мати не менше значення, ніж власне алгоритм класифікації.

Ще одним із найсуттєвіших сучасних зрушень є перехід від аналізу окремої транзакції до аналізу мережі взаємодій: рахунок-одержувач, картка-пристрій, користувач-банківський рахунок, торговець-канал, посередник-ланцюг переказів. Огляд [20], присвячений графовим нейронним мережам (graph neural networks, GNN), підкреслює, що ці підходи особливо добре виявляють складні реляційні патерни й динаміку у фінансових мережах, істотно розширюючи можливості порівняно з транзакційно-орієнтованими моделями. Важливість цього переходу полягає в тому, що багато сучасних шахрайських схем, особливо з використанням рахунків-посередників і ланцюгів переказів, не можуть бути коректно описані лише атрибутами однієї транзакції [20; 21].

Разом із тим графовий підхід змінює й інженерну інфраструктуру системи: виникає потреба в підтримці графового подання, механізмах побудови зв’язків між сутностями, зберіганні часових ребер, агрегуванні сусідства, керуванні обсягом контексту та контролі затримки під час виведення. Саме тому графові моделі найкраще виправдані там, де мережевий контекст справді несе додаткову інформацію, наприклад, у виявленні організованих посередницьких мереж, циклічних маршрутів чи багатоетапного відмивання коштів [20; 21]. У протилежному випадку вартість розгортання може перевищити практичний виграш.

Логічним продовженням графового напряму стали часово-графові мережі (temporal graph networks), які поєднують структурні зв’язки з часовим порядком подій. У роботі [21] транзакції моделюються як подійно-орієнтований часовий граф, вузлами якого є користувачі, картки, пристрої, банківські рахунки та інші сутності, а ребра відображають взаємодії між ними. Такий підхід краще відповідає природі цифрового шахрайства, оскільки дозволяє відстежувати не лише зв’язність, а й порядок та щільність подій у часі. Огляди графових методів також підкреслюють, що саме часовий вимір робить ці системи придатнішими

до реального фінансового середовища, де патерни можуть бути короткоживучими та швидко змінюваними [20; 21].

Однак із практичного погляду саме тут найрізкіше зростає вартість системи. Часово-графові моделі потребують не просто потужнішого навчання, а й синхронізації потокових подій, оновлення графового стану, роботи з пам'яттю вузлів, швидкого добування локального оточення та стабільного керування затримкою. Інакше кажучи, прогрес у якості супроводжується зростанням вимог до архітектури даних, обчислювальних ресурсів і технології розгортання. Тому на сучасному етапі ключовим стає вже не питання «чи краща модель», а питання «за яких інфраструктурних умов ця модель виправдана» [20; 21; 23].

Для фінансових систем вирішальною є не лише точність, а й оперативність спрацювання. Шахрайську транзакцію потрібно не просто розпізнати, а зробити це до завершення операції або щонайменше до її незворотного проходження між ланками платіжного ланцюга. Тому еволюція методів виявлення шахрайства супроводжується переходом від пакетної обробки (batch processing) до потокової обробки (stream processing). У дослідженні [24] порівнювалися два потокові каркаси обробки – Apache Flink і Apache Spark – у складі архітектури реального часу на базі Kafka; обидві конфігурації продемонстрували придатність до масштабованого виявлення шахрайства, але з різними профілями затримки. Отже, вибір засобу обробки потоку стає частиною якості детектора не менше, ніж вибір моделі.

Ще радикальніше цю проблему ставить робота MIND [23], у якій машинне навчання частково перенесено безпосередньо в програмований мережевий пристрій. Автори наголошують, що їхній прототип дає терабітну пропускну здатність і затримку мікросекундного рівня, тобто переводить задачу виявлення шахрайства в площину обчислень на шляху проходження пакета. Хоча це поки досить спеціалізований напрям, факт появи таких робіт свідчить, що інженерна еволюція сфери вже виходить за межі вибору алгоритму й переходить до проєктування фізичного шляху даних і місця виконання виведення. Для систем високої інтенсивності це може стати визначальним чинником [23; 24].

Окремий сучасний етап стосується не стільки зростання точності, скільки узгодження моделей із вимогами пояснюваності, конфіденційності даних та регуляторної сумісності. Огляди [16-18] прямо називають пояснюваність і приватність серед основних бар'єрів для широкого впровадження складних моделей у фінансовій сфері. Відповіддю на це стали підходи федеративного навчання (federated learning), у яких модель узгоджується між кількома учасниками без передачі сирих даних. Робота [25] про пояснюване федеративне навчання поєднує цей підхід із методами локального пояснення та підкреслює, що така комбінація покликана одночасно зберегти конфіденційність, підвищити довіру аналітика до рішення та зменшити регуляторні ризики.

Слід зауважити, що цей напрямок додає й нові інженерні труднощі: неоднорідність локальних даних, комунікаційні витрати, складність узгодження параметрів, можливі атаки на сам процес розподіленого навчання. Отже, приватне міжорганізаційне навчання не є простим розширенням звичайного машинного навчання. Це окрема інженерна дисципліна на перетині алгоритмів, криптографічних обмежень, мережевої взаємодії та регуляторних вимог [25].

1.3 Огляд програмних рішень для виявлення шахрайства

Сучасні системи виявлення фінансового шахрайства представлені рядом промислових платформ, що реалізують різні підходи до обробки транзакцій, оцінювання ризику та інтеграції з банківською інфраструктурою. Найбільш показовими з інженерної точки зору є рішення класу Enterprise Fraud Management (EFM), зокрема Feedzai [26], SAS Fraud Management [27], FICO Falcon Platform [29].

Платформа Feedzai (рис. 1.2) є представником нового покоління систем виявлення фінансового шахрайства, побудованих за принципом єдиного аналітичного контуру обробки транзакційних подій у режимі реального часу. На відміну від традиційних рішень, що історично формувалися як сукупність окремих модулів (правила, скоринг, кейс-менеджмент), Feedzai реалізує

інтегровану архітектуру, у якій всі етапи обробки виконуються в межах єдиного технологічного середовища.

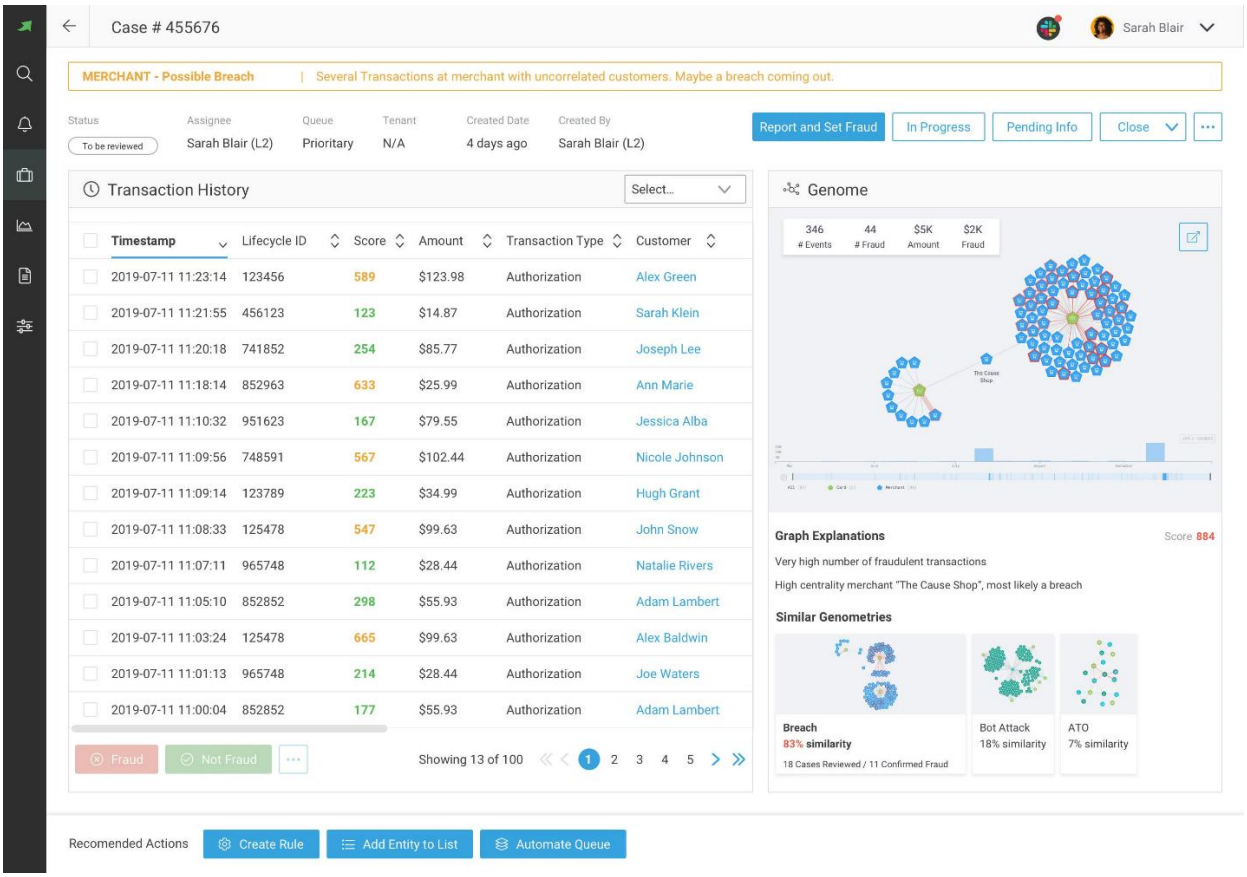


Рисунок 1.2 – Користувацький інтерфейс платформи Feedzai [26]

Функціонально платформа орієнтована на обробку високонавантажених потоків фінансових операцій, що включають карткові платежі, перекази, операції дистанційного банкінгу та інші транзакційні події. Ключовою можливістю є оцінювання ризику кожної операції в режимі реального часу, що дозволяє не лише виявляти підозрілу активність постфактум, а й запобігати виконанню шахрайських операцій до їх завершення. У користувацькому інтерфейсі це відображається у вигляді ризикового балу, деталізації ознак транзакції та зв'язків із попередніми діями клієнта, що фактично репрезентує результат роботи внутрішніх аналітичних моделей.

З інженерного погляду платформа реалізує подійно-орієнтовану архітектуру обробки даних, у якій транзакції надходять як потік подій і

проходять через послідовність етапів: нормалізацію, обчислення ознак, оцінювання ризику та формування рішення. Важливою особливістю є наявність єдиного сховища ознак, що використовується як у режимі реального часу, так і в процесах навчання моделей. Такий підхід дозволяє уникнути розбіжностей між онлайн- та офлайн-обчисленням ознак, які в традиційних системах часто призводять до деградації якості моделей.

Платформа підтримує гібридний механізм оцінювання ризику, що поєднує правила та моделі машинного навчання. При цьому правила виконують роль швидкого фільтра або механізму контролю відомих сценаріїв шахрайства, тоді як моделі забезпечують виявлення складних і неявних залежностей у даних. Результати цих підходів агрегуються в єдиний ризиковий бал, який використовується на рівні прийняття рішення.

Окрему функціональну роль відіграє підсистема управління випадками, яка забезпечує обробку підозрілих операцій аналітиками. У межах цієї підсистеми формується повний контекст транзакції: історія дій клієнта, ознаки ризику, результати моделей та рішення системи. Це дозволяє здійснювати ручну перевірку, накопичувати зворотний зв’язок і використовувати його для подальшого навчання моделей. Таким чином, платформа реалізує замкнений цикл: виявлення → аналіз → навчання → вдосконалення.

Інтеграція Feedzai з банківською інфраструктурою здійснюється через інтерфейси прикладного програмування та потокові механізми обміну даними. Система може працювати як у вбудованому режимі (inline), безпосередньо впливаючи на виконання транзакції, так і в режимі паралельного аналізу (near-real-time). Така гнучкість дозволяє адаптувати платформу до різних архітектур банківських систем, однак водночас створює значні вимоги до узгодженості даних, пропускної здатності та затримки.

Платформа SAS Fraud Management [27] (рис. 1.3) є корпоративним рішенням для виявлення, запобігання та управління фінансовим шахрайством, яке реалізує повний цикл обробки транзакцій. На відміну від систем, що історично орієнтувалися на окремі типи шахрайства (наприклад, карткові

операції), SAS Fraud Management позиціонується як універсальна платформа для багатоканального аналізу транзакцій і подій.

Ключовою функціональною характеристикою системи є можливість оцінювання 100% транзакцій у режимі реального часу, включаючи як фінансові, так і нефінансові події. Офіційна документація вказує, що система забезпечує обробку транзакцій із дуже високою пропускною здатністю (десятки тисяч операцій за секунду) та низькою затримкою, що дозволяє використовувати її безпосередньо у транзакційному контурі банку [28]. Звідси, SAS Fraud Management може застосовуватися не лише для постфактум аналізу, а й для оперативного запобігання шахрайству.

Функціонально платформа базується на поєднанні аналітичних моделей, машинного навчання та правил, що забезпечує гібридний підхід до виявлення шахрайства. У джерелах зазначається, що система використовує розширену аналітику, включаючи поведінкове профілювання, сегментацію клієнтів та виявлення аномалій, що дозволяє ідентифікувати підозрілі патерни в транзакційній активності. Крім того, SAS підтримує механізм rule authoring, який дозволяє створювати та налаштовувати правила виявлення шахрайства, що підтверджує комбінування статистичних і експертних підходів.

Однією з важливих функціональних складових є система алертів (alerts), яка автоматично генерує повідомлення про підозрілі операції. Такі сповіщення можуть бути пріоритизовані за рівнем ризику та передані на подальшу обробку. У межах цієї логіки реалізовано робоче середовище аналітика (alert triage workstation), яке дозволяє здійснювати перевірку випадків, аналіз контексту та прийняття рішень. Звідси, система підтримує не лише автоматичне виявлення, а й організацію процесу розслідування.

З інженерного погляду важливою характеристикою платформи є її здатність до інтеграції з різнорідними джерелами даних. SAS Fraud Management забезпечує обробку внутрішніх і зовнішніх даних у різних форматах, включаючи транзакційні системи, сторонні сервіси та історичні сховища. Система підтримує гнучкі механізми обміну повідомленнями та API, що дозволяє інтегрувати її в

існуючу IT-інфраструктуру банку без значної модифікації. Це є важливою перевагою в умовах гетерогенних банківських систем.

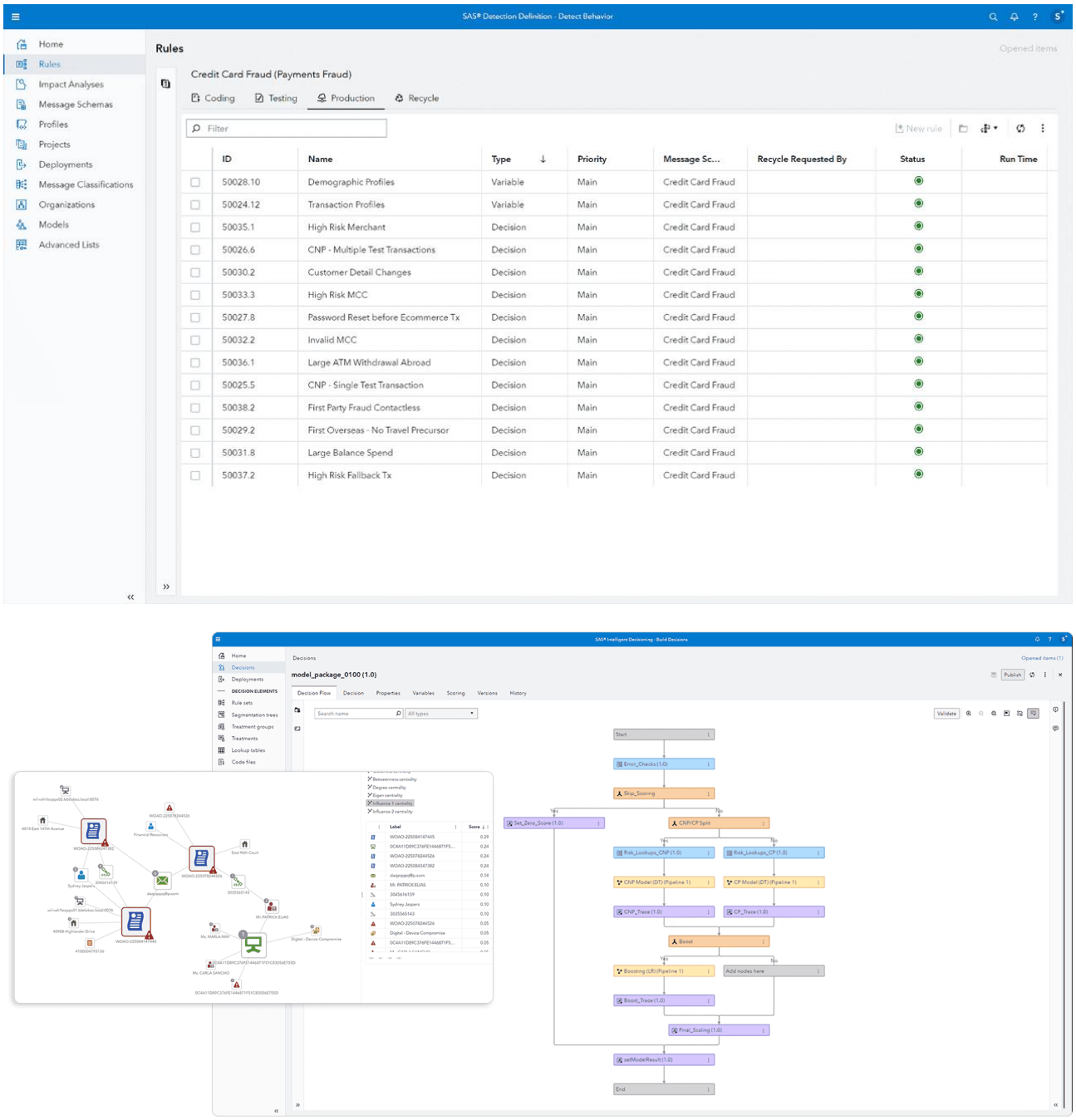


Рисунок 1.3 – Користувацький інтерфейс платформи SAS

Окрему роль відіграє можливість централізованого управління даними та багатоканального моніторингу. Платформа дозволяє об’єднувати дані з різних каналів і бізнес-напрямів у межах єдиного середовища, що забезпечує більш повний контекст для аналізу ризиків. Таким чином підвищується ефективність

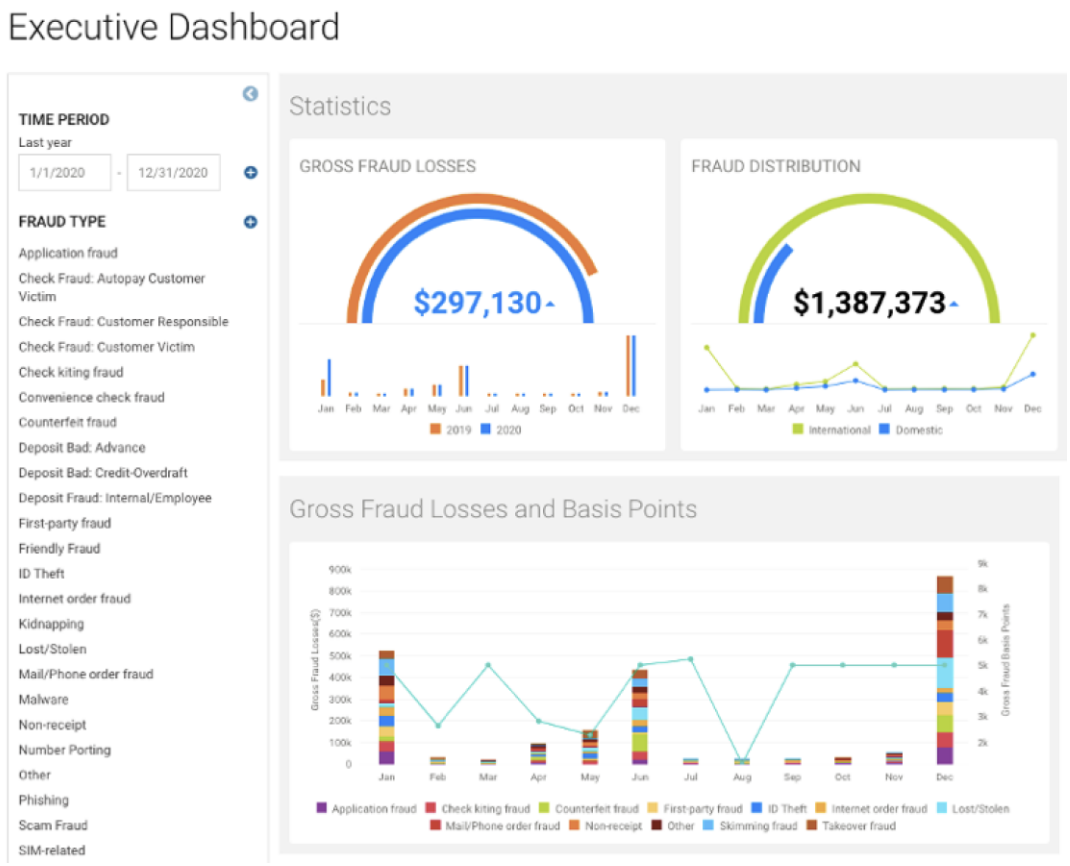
виявлення складних шахрайських схем, які можуть охоплювати декілька каналів або етапів взаємодії з клієнтом.

Система також підтримує журналювання, аудит і контроль доступу, що є критично важливим для відповідності регуляторним вимогам. Наявність механізмів ролей, журналів подій і відстеження рішень дозволяє забезпечити прозорість функціонування системи та можливість подальшої перевірки прийнятих рішень.

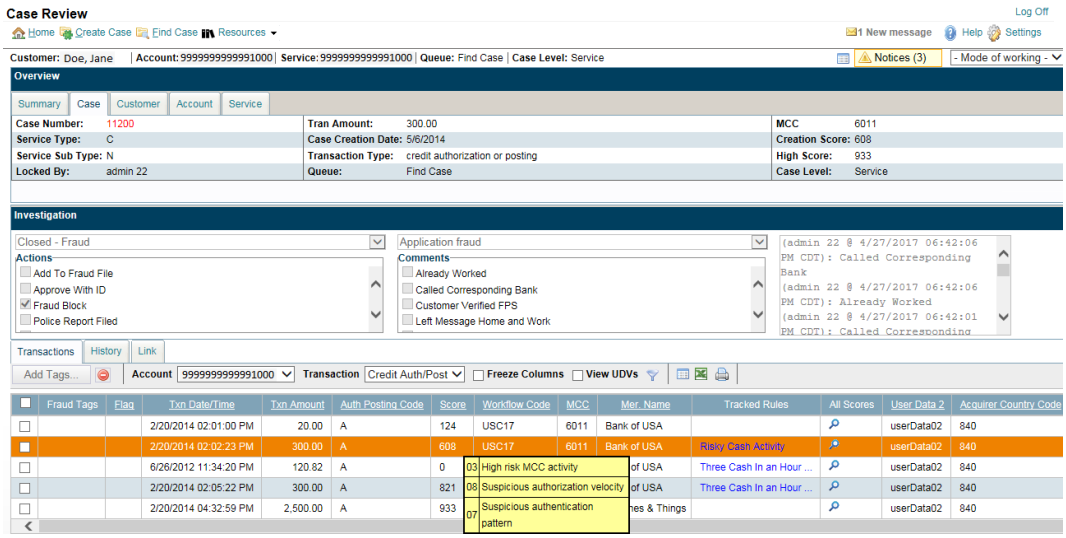
Разом із тим, аналіз платформи дозволяє виокремити низку обмежень. По-перше, ефективність системи значною мірою залежить від якості даних і налаштування ознак, що потребує розвиненої інфраструктури управління даними. По-друге, складність конфігурації та широка функціональність можуть ускладнювати впровадження і супровід системи, особливо в випадках, коли організація не має достатнього аналітичного досвіду. По-третє, хоча система підтримує машинне навчання, значна частина логіки залишається залежною від правил, що може обмежувати адаптивність до нових, раніше невідомих сценаріїв шахрайства.

Платформа FICO Falcon Platform [29] (рис. 1.4) є однією з найбільш поширених у світі систем виявлення фінансового шахрайства, яка використовується банками та платіжними установами для моніторингу транзакцій і запобігання шахрайським операціям у режимі реального часу. За даними FICO [30], система застосовується для захисту понад 2,6 мільярда платіжних рахунків, що підтверджує її статус де-факто галузевого стандарту.

Функціонально платформа реалізує повний цикл обробки транзакцій, який включає аналіз подій, оцінювання ризику, формування рішень і підтримку подальшого розслідування. Основним елементом платформи є компонент Falcon Fraud Manager, який здійснює безперервний моніторинг транзакцій і дозволяє виявляти шахрайство в різних каналах: кредитні та дебетові картки, електронні платежі, мобільні операції та банківські перекази.



(а)



(б)

Рисунок 1.4 – Користувацький інтерфейс платформи Falcon Fraud Manager (різні версії)

Ключовою характеристикою системи є оцінювання транзакцій у режимі реального часу, що дозволяє приймати рішення безпосередньо під час виконання операції. Архітектура платформи передбачає інтеграцію з авторизаційними

системами, що дає можливість впливати на результат транзакції (дозвіл або відхилення) до її завершення [31]. Це є принциповою відмінністю від офлайн-аналізу та визначає її ефективність у запобіганні фінансовим втратам.

Аналітичне ядро платформи базується на прогнозній аналітиці (predictive analytics) та методах машинного навчання, які використовують дані мільярдів транзакцій для побудови моделей ризику. Система застосовує комбінацію підходів: кероване (supervised), некероване (unsupervised) та напівкероване навчання. Додатково реалізується поведінкове профілювання, у межах якого формуються моделі поведінки клієнтів, пристроїв і торгових точок, що постійно оновлюються під час кожної транзакції [32].

Важливим функціональним елементом є механізм ризикового скорингу, який дозволяє оцінити ймовірність шахрайства для кожної операції. На основі цього скорингу формуються рішення або генеруються підозрілі події для подальшої обробки. У поєднанні з аналітичними моделями система також використовує експертні правила, що дозволяє враховувати як відомі сценарії шахрайства, так і нові патерни поведінки.

З інженерного погляду платформа характеризується єдиною архітектурою обробки транзакцій, яка включає спільну модель даних, механізми керування правилами, аналітичне ядро та підсистему управління випадками. Це забезпечує централізоване управління ризиками та узгодженість рішень у межах усіх каналів обслуговування. Крім того, система підтримує багатоканальний моніторинг (omnichannel), що дозволяє виявляти шахрайські сценарії, які охоплюють декілька типів операцій і взаємодій.

Окремою особливістю платформи є використання глобальних даних і консорціумних моделей, що дозволяє підвищити точність виявлення шахрайства та зменшити кількість хибних спрацювань. Також зазначається, що система здатна виявляти до 50% більше шахрайських транзакцій порівняно з традиційними підходами на основі правил.

Разом із тим, платформа має ряд обмежень. По-перше, вона історично орієнтована на карткові транзакції, що може ускладнювати її адаптацію до нових

типів фінансових операцій без додаткових модулів. По-друге, складність архітектури та інтеграції з авторизаційними системами призводить до значних витрат на впровадження. По-третє, залежність від великих обсягів даних і складних моделей створює додаткові вимоги до інфраструктури та підтримки.

Загалом, сучасні програмні системи виявлення фінансового шахрайства являють собою складні багаторівневі програмно-аналітичні комплекси, що інтегруються в транзакційний контур банківських установ і виконують функції безперервного моніторингу, оцінювання ризику та підтримки прийняття рішень. Незалежно від конкретної реалізації, більшість сучасних рішень мають подібну багатокомпонентну структуру, яку доцільно розглядати як конвеєр обробки транзакційних подій (рис. 1.5). На рівні збору та нормалізації подій система має приймати події з різних джерел: платіжних шлюзів та процесингу карткових операцій, систем дистанційного банкінгу, банкоматів та POS-терміналів, систем управління рахунками та зовнішніх джерел (чорних списків, поведінкових сигналів). Інженерною особливістю є необхідність роботи з різними форматами даних, уніфікації схем (schema normalization) та забезпечення мінімальної затримки передачі подій. У сучасних архітектурах цей рівень часто реалізується через потокові платформи обміну повідомленнями (наприклад, брокери повідомлень), що дозволяє відокремити джерела даних від аналітичного контуру [33].

На рівні обчислення ознак формується контекст для прийняття рішення. Типовими операціями є агрегація транзакцій у часових вікнах, обчислення поведінкових характеристик, визначення новизни (новий пристрій, новий отримувач), побудова зв’язків між сутностями тощо. Цей рівень може працювати у двох режимах: онлайн-обчислення для миттєвого оцінювання ризику та офлайн-обчислення для побудови історичних профілів. Різниця між офлайн- і онлайн-ознаками може призводити до деградації якості моделі [34; 35].

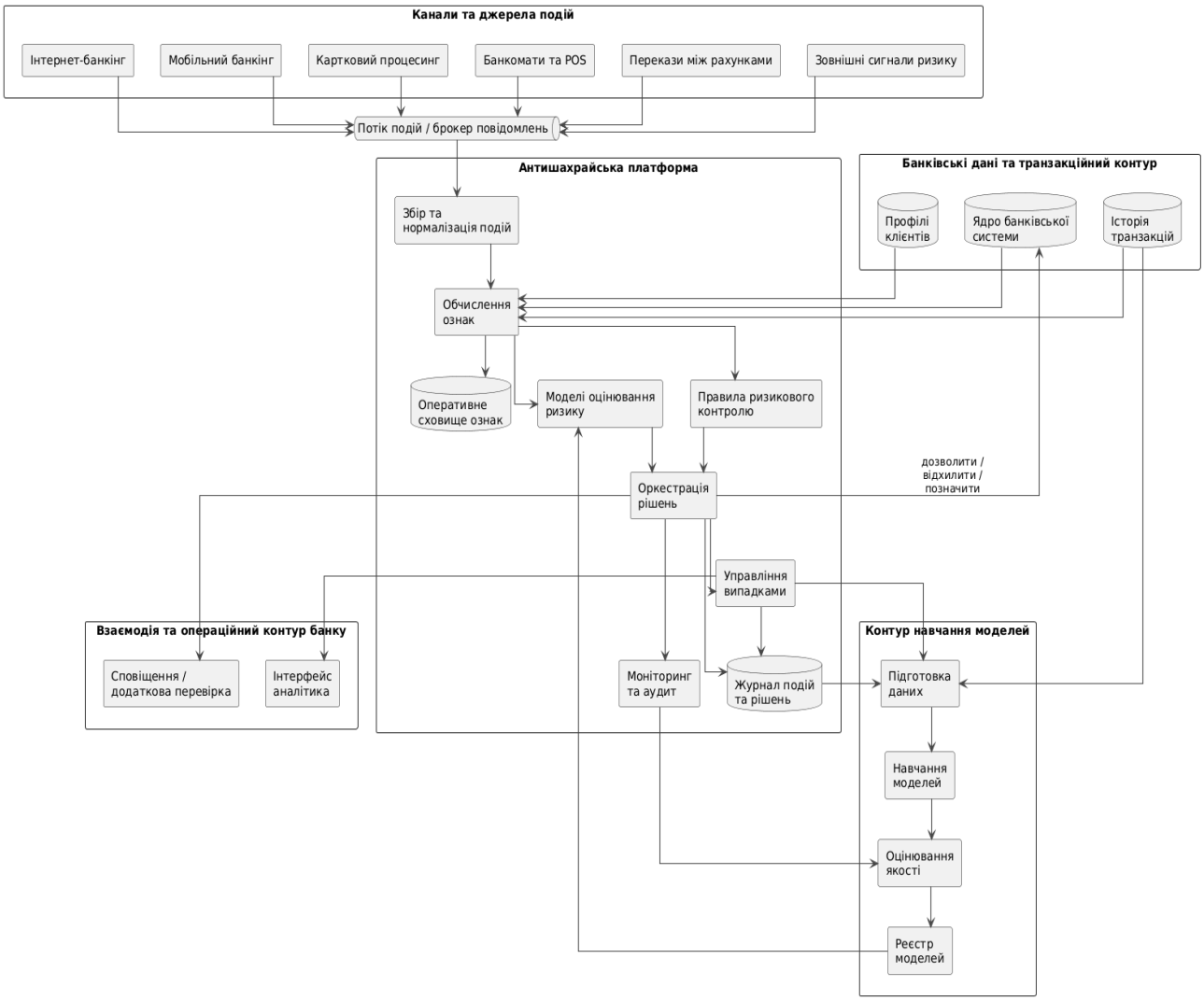


Рисунок 2 – Інтеграція системи виявлення фінансового шахрайства

Ядро системи становить рівень оцінювання ризику, де відбувається застосування правил, виконання моделей машинного навчання та комбінування результатів (ensemble scoring). З інженерного погляду важливі такі характеристики: затримка (latency, зазвичай мілісекундний рівень), пропускна здатність, стабільність під навантаженням та можливість версіонування моделей. Сучасні платформи часто використовують гібридний підхід, де правила виконують роль фільтра або страховки, а моделі – роль основного оцінювача ризику [34; 36].

На рівні прийняття рішень формується дія: дозволити операцію, відхилити операцію чи передати на додаткову перевірку. Рішення має враховувати бізнес-правила, регуляторні вимоги та ризик-профіль клієнта. У складних системах

реалізується оркестрація рішень (decision orchestration), де декілька джерел оцінки комбінуються в єдину логіку [36].

Рівень управління випадками (case management) забезпечує інтерфейс для аналітиків, обробку підозрілих операцій та накопичення зворотного зв'язку. Саме він формує навчальні дані для моделей, контроль якості системи та звітність для регулятора.

1.4 Постановка задачі на розроблення

Загалом, труднощі виявлення шахрайства закладені вже на рівні даних, а не виникають лише на етапі побудови моделей. Сучасні огляди стабільно фіксують щонайменше три системні проблеми: сильну незбалансованість класів, дрейф понять (concept drift), тобто зміну поведінки зловмисників у часі, а також обмежену доступність репрезентативних даних через регуляторні й комерційні обмеження. Додатково галузеві експерти наголошують на високій частці хибних спрацювань, складності масштабування моніторингу та потребі в пояснюваності рішень у регульованому фінансовому середовищі. Отже, факт того, що шахрайські операції є рідкісними, еволюціонуючими та неоднорідними, слід вважати фундаментальною характеристикою предметної області.

Також можна стверджувати, що еволюція методів виявлення шахрайства відбувалася за трьома взаємопов'язаними лініями. Перша – зростання виразності моделей: від правил до ансамблів, далі до глибоких, графових і часово-графових методів. Друга – ускладнення постановки задачі: від бінарної класифікації окремої транзакції до моделювання часової поведінки, дрейфу понять, відкритих категорій і мережевих структур. Третя – інженерна еволюція середовища виконання: від офлайнового аналізу до потокового виведення, низьколатентних контурів, пояснюваного та приватного колективного навчання.

З урахуванням проведеного аналізу предметної області задача розроблення інтелектуального додатка для виявлення фінансових шахрайств полягає у створенні програмного засобу, який забезпечує автоматизоване опрацювання транзакційних даних, формування ознак фінансових операцій, застосування

моделей машинного навчання та подання результатів оцінювання ризику у формі, придатній для подальшого аналізу користувачем системи. Звідси, з функціональної точки зору, система повинна:

1) забезпечувати завантаження набору транзакційних даних у табличному форматі;

2) виконувати первинну перевірку даних: визначення типів ознак, наявності пропущених значень, дублікатів та некоректних записів;

3) здійснювати попереднє опрацювання даних, зокрема кодування категоріальних ознак, масштабування числових параметрів та підготовку даних до навчання моделей;

4) формувати навчальну та тестову вибірки з урахуванням незбалансованості класів;

5) підтримувати навчання декількох моделей машинного навчання для задачі бінарної класифікації транзакцій;

6) забезпечувати класифікацію транзакцій на легітимні та потенційно шахрайські;

7) формувати кількісну оцінку ризику для кожної проаналізованої транзакції;

8) забезпечувати порівняння моделей за основними метриками якості класифікації;

9) відображати результати роботи у вигляді таблиць, графіків і підсумкових показників;

10) дозволяти користувачу переглядати перелік транзакцій, класифікованих як потенційно шахрайські;

11) забезпечувати збереження результатів аналізу для подальшого використання;

12) передбачати можливість подальшої інтеграції з банківськими інформаційними системами через програмні інтерфейси або обмін файлами.

До нефункціональних вимог доцільно віднести такі:

– точність і надійність: система повинна забезпечувати достатній рівень якості класифікації з урахуванням специфіки незбалансованих даних;

– пояснюваність результатів: користувач повинен мати змогу інтерпретувати причини віднесення транзакції до підозрілих;

– продуктивність: система повинна виконувати попереднє опрацювання, навчання моделей і класифікацію даних у прийнятний для користувача час;

– масштабованість: архітектура додатка повинна допускати розширення набору моделей, ознак і джерел даних;

– модульність: компоненти завантаження даних, попереднього опрацювання, навчання моделей, оцінювання та візуалізації мають бути логічно відокремлені;

– зручність використання: інтерфейс додатка повинен забезпечувати зрозумілий сценарій роботи користувача без необхідності безпосереднього редагування програмного коду;

– відтворюваність експериментів: система повинна забезпечувати фіксацію параметрів навчання, розбиття вибірки та отриманих метрик;

– безпека даних: система повинна передбачати обмеження доступу до завантажених транзакційних даних і результатів аналізу;

– сумісність: програмний засіб має працювати з поширеними форматами табличних даних і бути придатним для розгортання в типовому програмному середовищі;

– супроводжуваність: структура програмного рішення повинна дозволяти подальше оновлення моделей, зміни логіки обробки даних і додавання нових функцій.

У такій постановці задача зводиться до побудови прототипу інтелектуального додатка, який поєднує аналітичний конвеєр обробки транзакційних даних, модуль машинного навчання, засоби оцінювання якості моделей і користувацький інтерфейс для подання результатів виявлення потенційно шахрайських операцій.

Сучасна система виявлення шахрайства повинна розглядатися як багатошарова програмно-аналітична система, а не як окрема модель. Її ефективність визначається узгодженістю між ознаковим конвеєром, алгоритмом, метрикою оптимізації, часовим режимом обробки, механізмом оновлення моделі, засобами пояснення рішення та обмеженнями інфраструктури. Саме тому в сучасній науковій літературі дедалі більше уваги приділяється умовам застосовності конкретного класу методів у конкретному фінансовому середовищі.

РОЗДІЛ 2 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз вимог до програмного забезпечення

Розроблюваний інтелектуальний додаток для виявлення фінансових шахрайств розглядається як прикладна програмна система, що поєднує опрацювання транзакційних даних, навчання моделей машинного навчання, оцінювання ризику окремих операцій та подання результатів користувачу в інтерфейсній формі. На відміну від суто дослідницького блокнота або окремого алгоритмічного модуля, майбутній програмний продукт має підтримувати завершений сценарій роботи: завантаження даних, їх перевірку, підготовку, навчання моделей, класифікацію транзакцій, перегляд результатів і порівняння якості моделей.

Для аналізу вимог доцільно виділити три основні ролі користувачів:

1) аналітик фінансового моніторингу – основний користувач системи, який завантажує дані, запускає аналіз, переглядає підозрілі транзакції та оцінює результати роботи моделей;

2) адміністратор системи – користувач, відповідальний за налаштування середовища, контроль доступу, керування файлами даних і перевірку працездатності системи;

3) розробник / дослідник моделей – користувач, який аналізує якість моделей, змінює параметри навчання, порівнює алгоритми та готує систему до подальшого вдосконалення.

У спрощеному прототипі одна фізична особа може виконувати кілька ролей. Проте їх розмежування важливе для подальшого архітектурного проєктування, оскільки різні ролі мають різні права доступу, сценарії взаємодії та очікування від системи.

Основний сценарій роботи аналітика передбачає таку послідовність дій: користувач завантажує файл із транзакціями; система перевіряє структуру даних; виконує попередню обробку; запускає навчання або використовує попередньо

навчену модель; формує оцінку ризику; виводить список підозрілих транзакцій; надає узагальнені метрики та графічні подання результатів. Такий сценарій узгоджується з логікою сучасних систем, де рішення формується на основі комбінації обробки ознак, моделей оцінювання ризику та подальшого перегляду підозрілих випадків

Для формалізації сценаріїв взаємодії використовується UML-діаграма прецедентів (рис. 2.1). Вона показує зовнішню поведінку системи з погляду аналітика фінансового моніторингу та розробника моделей. Діаграма демонструє, що центральним сценарієм є не лише запуск класифікації, а весь ланцюг роботи з даними: завантаження, перевірка, попереднє опрацювання, навчання або використання моделі, формування ризикової оцінки та перегляд підозрілих транзакцій.

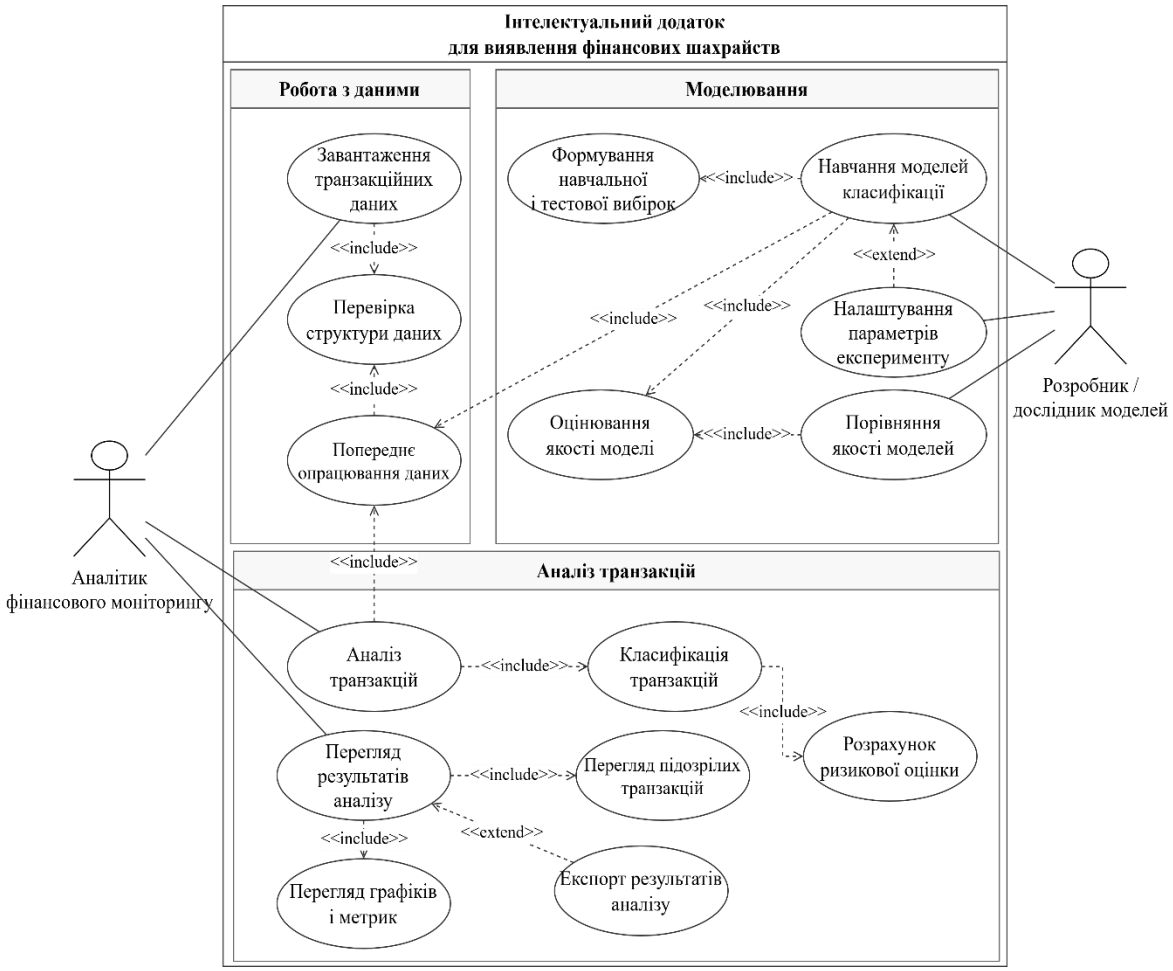


Рисунок 2.1 – Діаграма прецедентів інтелектуального додатка для виявлення фінансових шахрайств

Для коректного формування вимог необхідно зафіксувати низку припущень:

- на початковому етапі система працює з файлами, а не з прямим підключенням до банківського ядра;
- транзакційні дані мають бути знеособленими;
- цільова ознака шахрайства в навчальному наборі даних уже наявна;
- система орієнтована на навчання з учителем, тобто на класифікацію на основі розмічених прикладів;
- інтеграція з банківськими системами розглядається як перспективне розширення через API;
- рішення системи має рекомендаційний характер і не є остаточним юридичним або банківським рішенням щодо блокування операції.

Функціональні вимоги визначають, які дії система повинна виконувати. Їх доцільно згрупувати за підсистемами, щоб надалі перейти до компонентної архітектури (табл. 2.1).

Таблиця 2.1 – Функціональні вимоги до програмного забезпечення

Код	Функціональна вимога	Призначення для архітектури
FR-01	Система повинна забезпечувати завантаження транзакційних даних у форматі CSV або XLSX.	Формує потребу в модулі імпорту даних.
FR-02	Система повинна перевіряти структуру завантаженого набору: наявність цільової ознаки, числових і категоріальних полів, пропусків та дублікатів.	Вимагає окремого модуля валідації даних.
FR-03	Система повинна виконувати попереднє опрацювання даних: очищення, кодування категоріальних ознак, масштабування числових параметрів.	Формує конвеєр підготовки даних.
FR-04	Система повинна забезпечувати поділ даних на навчальну та тестову вибірки.	Потрібен модуль експериментального розбиття даних.
FR-05	Система повинна підтримувати навчання кількох моделей бінарної класифікації.	Вимагає сервісу керування моделями.
FR-06	Система повинна виконувати класифікацію транзакцій на легітимні та потенційно шахрайські.	Основна функція інтелектуального ядра.
FR-07	Система повинна формувати ризикову оцінку для кожної транзакції.	Потрібен модуль ризикового оцінювання.
FR-08	Система повинна відображати перелік транзакцій, визначених як підозрілі.	Вимагає інтерфейсу перегляду результатів.

Продовження таблиці 2.1

FR-09	Система повинна обчислювати метрики якості моделей: точність, влучність, повноту, F1-міру, ROC-AUC.	Потрібен модуль оцінювання моделей.
FR-10	Система повинна забезпечувати порівняння результатів кількох моделей.	Формує потребу в експериментальному журналі.
FR-11	Система повинна будувати графічні подання результатів: матрицю помилок, криву ROC, розподіл ризикових оцінок.	Потрібен модуль візуалізації.
FR-12	Система повинна дозволяти експортувати результати аналізу у файл.	Вимагає модуля експорту.
FR-13	Система повинна вести журнал основних дій користувача та результатів аналізу.	Потрібне журналювання для відтворюваності та аудиту.
FR-14	Система повинна зберігати параметри запуску експерименту: модель, параметри, дата, метрики.	Підтримує відтворюваність експериментів.
FR-15	Система повинна передбачати можливість подальшої інтеграції через API / файловий обмін.	Впливає на вибір серверної архітектури.

Особливу увагу слід приділити вимогам FR-02, FR-03, FR-07 і FR-09, оскільки саме вони визначають аналітичну якість системи. Якщо дані будуть некоректно очищені або ознаки будуть сформовані непослідовно, якість моделі може бути формально високою, але інженерно ненадійною. Загалом, систему слід поділити на такі логічні модулі:

- модуль завантаження та перевірки даних;
- модуль попереднього опрацювання;
- модуль навчання моделей;
- модуль класифікації та ризикового оцінювання;
- модуль оцінювання якості;
- модуль візуалізації;
- модуль збереження результатів;
- модуль інтерфейсу користувача.

Нефункціональні вимоги безпосередньо впливають на архітектуру системи. Наприклад, вимога модульності передбачає відокремлення шару роботи з даними від шару моделей; вимога відтворюваності потребує збереження параметрів експерименту; вимога пояснюваності впливає на інтерфейс і набір метрик, які система має показувати користувачу.

Таблиця 2.2 – Нефункціональні вимоги до програмного забезпечення

Код	Атрибут якості	Нефункціональна вимога	Інженерне значення
NFR-01	Продуктивність	Система повинна виконувати попереднє опрацювання та класифікацію демонстраційного набору даних у прийнятний для користувача час.	Впливає на вибір бібліотек обробки даних і спосіб кешування результатів.
NFR-02	Масштабованість	Архітектура повинна дозволяти додавання нових моделей, метрик і способів підготовки ознак без суттєвої перебудови системи.	Потребує модульної структури.
NFR-03	Надійність	Система повинна коректно обробляти помилки завантаження файлів, некоректні формати, відсутні поля та неможливість навчання моделі.	Вимагає обробки винятків і валідації вхідних даних.
NFR-04	Відтворюваність	Результати експериментів мають бути відтворюваними за однакових параметрів запуску.	Потрібна фіксація початкового стану генератора випадкових чисел, параметрів моделі та схеми розбиття.
NFR-05	Пояснюваність	Користувач повинен бачити не лише клас транзакції, а й ризикову оцінку та ключові показники моделі.	Підвищує довіру до результатів аналізу.
NFR-06	Безпека	Завантажені дані та результати аналізу не повинні бути доступні стороннім користувачам.	Вимагає обмеження доступу та базових механізмів автентифікації в розширеній версії.
NFR-07	Супроводжуваність	Код системи має бути структурований за модулями: дані, ознаки, моделі, оцінювання, інтерфейс, API.	Спрощує тестування й розвиток системи.
NFR-08	Зручність використання	Основні сценарії повинні виконуватися через зрозумілий інтерфейс без необхідності редагування програмного коду.	Визначає потребу у вебінтерфейсі або інтерактивній панелі.
NFR-09	Сумісність	Система повинна працювати в типовому середовищі Python і підтримувати поширені формати табличних даних.	Впливає на вибір технологічного стеку.
NFR-10	Спостережуваність	Система повинна фіксувати ключові події: завантаження даних, запуск навчання, завершення класифікації, помилки.	Створює основу для журналювання та подальшого моніторингу.

Вибір мови програмування Python є доцільним через наявність зрілих бібліотек для машинного навчання, аналізу даних, побудови прикладних інтерфейсів і створення API. Технологічний стек (табл. 2.3) забезпечує повний шлях від обробки даних до побудови інтерфейсу.

Таблиця 2.3 – Обраний технологічний стек для реалізації програмного забезпечення

Рівень системи	Обрані технології	Обґрунтування
Мова програмування	Python 3.11+	Основна мова для аналізу даних, машинного навчання та швидкої розробки прототипів.
Обробка даних	pandas, NumPy	Підтримують роботу з табличними даними, очищення, агрегацію, числові операції.
Машинне навчання	scikit-learn	Забезпечує реалізацію класичних моделей, метрик, конвеєрів обробки й розбиття даних.
Додаткові моделі	XGBoost або LightGBM	Доцільні для ансамблевих моделей градієнтного підсилення на табличних даних.
Візуалізація	matplotlib, Plotly	Дають змогу будувати графіки метрик, розподіли, матриці помилок.
Вебінтерфейс	Streamlit або FastAPI + шаблони	Streamlit доцільний для швидкого аналітичного прототипу; FastAPI – для архітектурно зрілого API.
API-рівень	FastAPI	Забезпечує можливість подальшої інтеграції з іншими системами.
Зберігання результатів	SQLite / PostgreSQL	SQLite достатня для прототипу; PostgreSQL доцільна для розширеної версії.
Збереження моделей	joblib / pickle	Дозволяє зберігати навчені моделі та використовувати їх повторно.
Тестування	pytest	Підтримує модульні та функціональні перевірки.
Контроль якості коду	ruff, black, mypy	Забезпечують стиль, форматування та базову статичну перевірку.
Контейнеризація	Docker	Дозволяє відтворювано запускати систему в однаковому середовищі.

У результаті аналізу вимог встановлено, що майбутній інтелектуальний додаток має реалізовувати не лише алгоритмічну класифікацію транзакцій, а повний прикладний цикл роботи з даними: від завантаження й перевірки до навчання моделей, оцінювання ризику, візуалізації результатів і збереження експериментальних показників. Основними архітектурними драйверами системи є модульність, відтворюваність експериментів, пояснюваність

результатів, можливість подальшої інтеграції та достатня продуктивність для роботи з табличними наборами транзакційних даних.

2.2 Попередня архітектура програмного забезпечення

Система розглядається як програмний прототип, що має забезпечити повний прикладний цикл роботи з транзакційними даними: завантаження набору даних, перевірку його структури, попереднє опрацювання, навчання моделей класифікації, розрахунок ризикової оцінки, перегляд підозрілих транзакцій і збереження результатів аналізу. На відміну від промислових антишахрайських платформ, розроблюваний додаток не передбачає прямого блокування банківських операцій у транзакційному контурі. Його призначення полягає в побудові аналітичного програмного середовища, яке моделює ключові елементи системи виявлення шахрайства: обробку даних, роботу з ознаками, застосування моделей машинного навчання, оцінювання ризику та подання результатів користувачу. Звіди, ключові архітектурні драйвери для програмного забезпечення зведено в табл. 2.4.

Таблиця 2.4 – Ключові архітектурні драйвери

Архітектурний драйвер	Вплив на архітектуру
Робота з табличними транзакційними даними	Потрібен окремий контур завантаження, перевірки структури та попереднього опрацювання даних.
Порівняння декількох моделей	Необхідне відокремлення модуля навчання моделей від модуля класифікації.
Відтворюваність експериментів	Потрібне збереження параметрів запуску, метрик, версій моделей і результатів.
Пояснюваність результатів	Інтерфейс має показувати не лише клас транзакції, а й ризикову оцінку, метрики та ключові ознаки.
Можливість подальшої інтеграції	Слід передбачити прикладний програмний інтерфейс або окремий сервісний шар.
Обмежений обсяг кваліфікаційної роботи	Архітектура має бути достатньо простою для реалізації, але не зводитися до одного скрипта або блокнота.
Подальше тестування	Компоненти мають бути відокремлені так, щоб їх можна було перевіряти модульними й інтеграційними тестами.

З огляду на ці драйвери доцільно застосувати модульну шарувату архітектуру з аналітичним конвеєром обробки даних. У такій архітектурі

інтерфейс користувача, прикладна логіка, модулі машинного навчання та інфраструктурні засоби зберігання відокремлюються один від одного. Це дозволяє підтримати як просту демонстраційну реалізацію, так і подальший розвиток системи до API-орієнтованого сервісу.

На першому рівні C4-моделі (рис. 2.2) система розглядається як єдиний програмний продукт у взаємодії з користувачами та зовнішнім середовищем. Для розроблюваного додатка важливо коректно зафіксувати не лише користувачів, а й джерела даних, потенційні банківські системи та межі відповідальності прототипу. Діаграма фіксує межу системи: додаток не є банківським ядром і не виконує фінансові операції. Він отримує транзакційні дані, виконує їх аналітичне опрацювання та формує результати, які можуть бути використані аналітиком або передані у зовнішню систему.

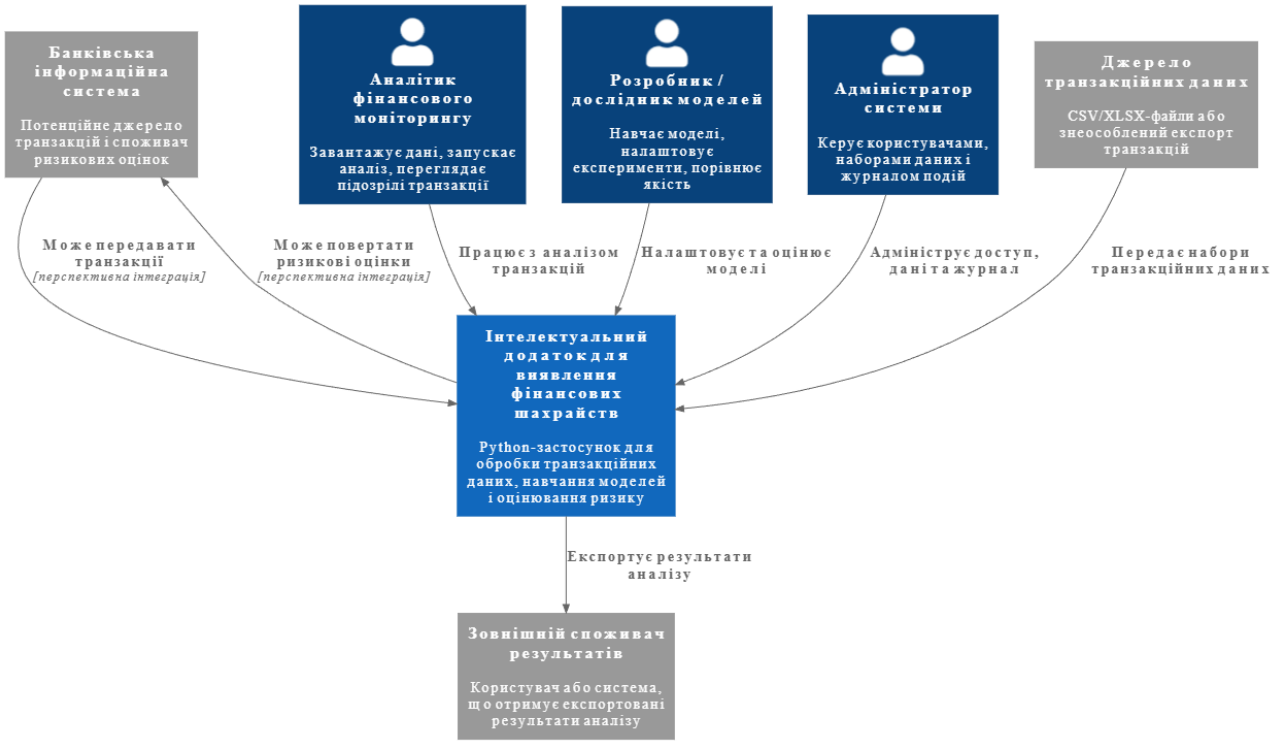


Рисунок 2.2 – C4-модель, системний контекст інтелектуального додатка (рівень 1)

Другий рівень C4-моделі (рис. 2.3) деталізує внутрішню структуру системи як набір логічних контейнерів. У C4-моделі контейнер не обов’язково означає

Docker-контейнер; це може бути вебзастосунок, сервіс, база даних, сховище файлів або окремий виконуваний модуль.

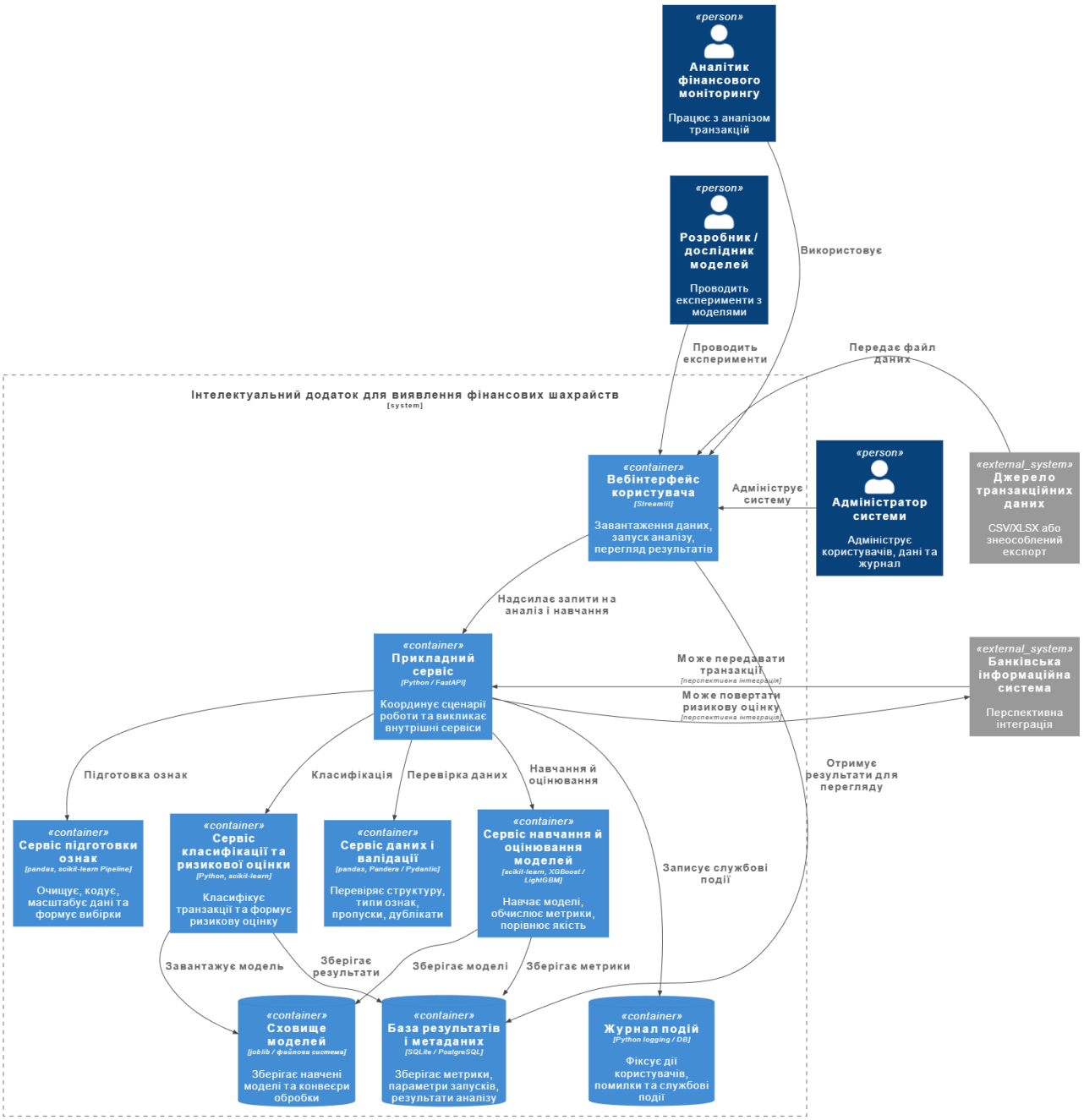


Рисунок 2.3 – C4-модель, контейнерна архітектура додатка (рівень 2)

Контейнерна архітектура відокремлює інтерфейс, прикладну логіку, обробку даних, машинне навчання, класифікацію, збереження моделей і журналювання. Така організація відповідає окремій групі вимог з попереднього підрозділу, а також спрощує тестування, оскільки модулі валідації даних,

підготовки ознак, навчання та класифікації можуть перевірятися окремо. Крім того, архітектура дає можливість надалі перейти від локального прототипу до сервісної інтеграції з банківськими системами без повного переписування аналітичного ядра.

Для уточнення поведінки системи доцільно описати основний сценарій: користувач завантажує набір транзакцій, система перевіряє дані, виконує попереднє опрацювання, навчає модель або використовує збережену модель, класифікує транзакції, розраховує ризикову оцінку та відображає результати. Діаграма послідовності (рис. 2.4) показує, що сценарій аналізу транзакцій має дві критичні точки контролю. Перша – перевірка структури даних: якщо набір не містить потрібних ознак або має некоректний формат, система не повинна передавати його на навчання моделі. Друга – вибір режиму роботи: навчання нової моделі або використання вже збереженої. У результаті слід підтримати два практичні сценарії: експериментальний, коли дослідник порівнює моделі, і прикладний, коли аналітик використовує вже підготовлену модель для класифікації транзакцій.

Загалом, обраний архітектурний стиль є компромісом між інженерною зрілістю та реалістичністю реалізації. Він не є надмірно складним, як мікросервісна або потокова архітектура, але водночас не спрощує систему до неструктурованого експериментального коду. Попередня архітектура інтелектуального додатка передбачає побудову системи як сукупності взаємопов’язаних модулів: вебінтерфейсу, прикладного сервісу, модуля валідації даних, модуля підготовки ознак, модуля навчання моделей, модуля класифікації та ризикового оцінювання, сховища моделей, бази результатів і журналу подій. Такий поділ відповідає функціональним вимогам, підтримує основні нефункціональні вимоги та створює основу для подальшої програмної реалізації в екосистемі Python.

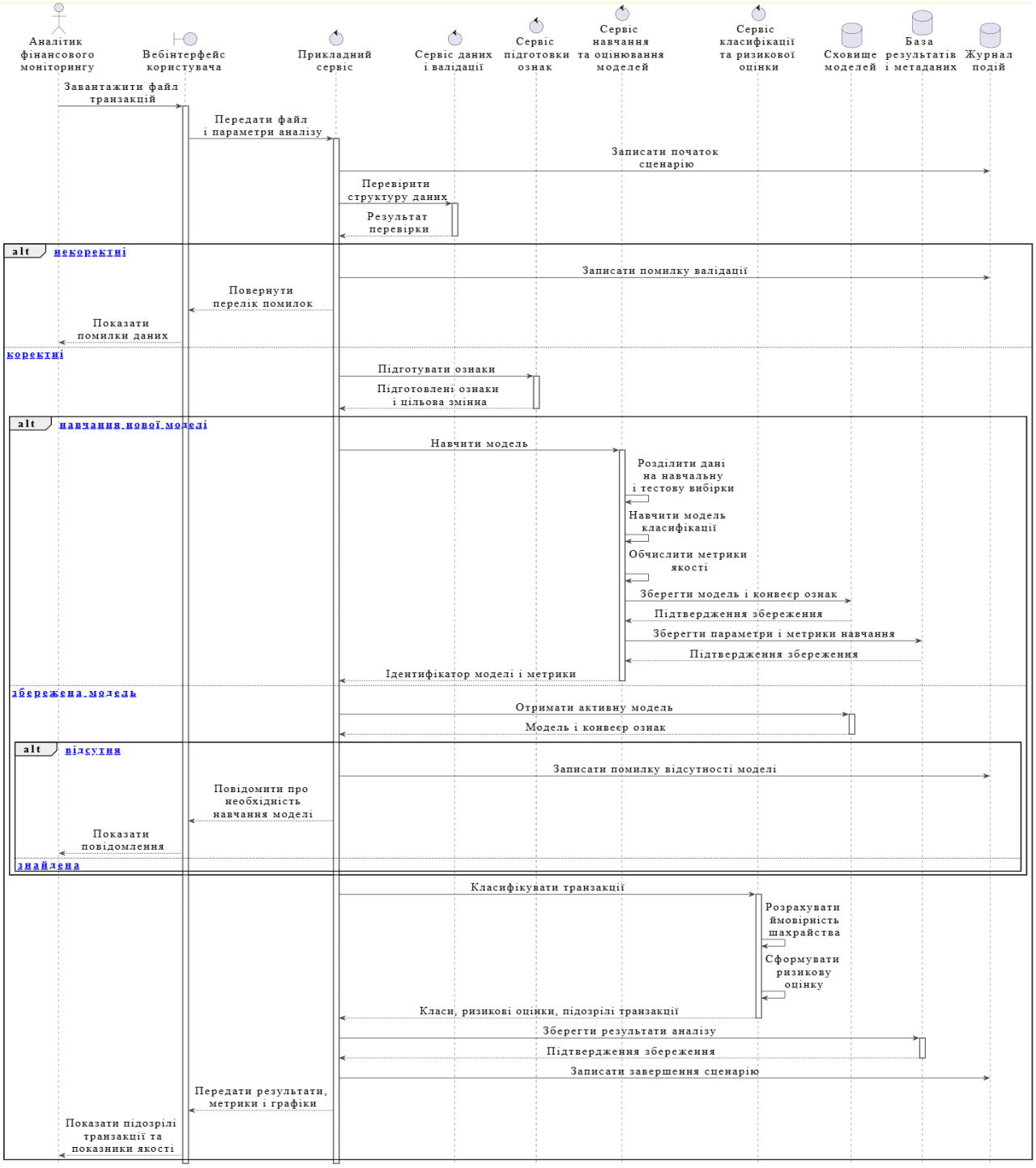


Рисунок 2.4 – Діаграма послідовності для основного сценарію

Системний контекст C4-моделі показує межі відповідальності додатка й зовнішні взаємодії, контейнерна діаграма деталізує внутрішню структуру, а діаграма послідовності уточнює основний сценарій аналізу транзакцій. Разом ці артефакти формують перехід від вимог до реалізаційної архітектури, що буде деталізована в наступних підрозділах.

2.3 Деталізована архітектура та проєктування бази даних

Оскільки система реалізується в екосистемі Python, структура проєкту має відповідати принципу розділення відповідальностей. Кожен пакет повинен мати чітке призначення й не дублювати функції інших частин системи. Користувацький інтерфейс не повинен напряму працювати з моделями машинного навчання або базою даних. Він має звертатися до сервісного шару, який координує сценарії використання. Сервісний прошарок, у свою чергу, взаємодіє з пакетами data, features, ml, scoring і persistence. Завдяки цьому архітектура залишається керованою: зміна алгоритму класифікації не потребує зміни інтерфейсу, а зміна способу збереження результатів не впливає на логіку навчання моделей. На рисунку 2.5 подано деталізовану структуру фронтенд-частини програмного продукту у вигляді UML-діаграми класів, згрупованих за пакетами.

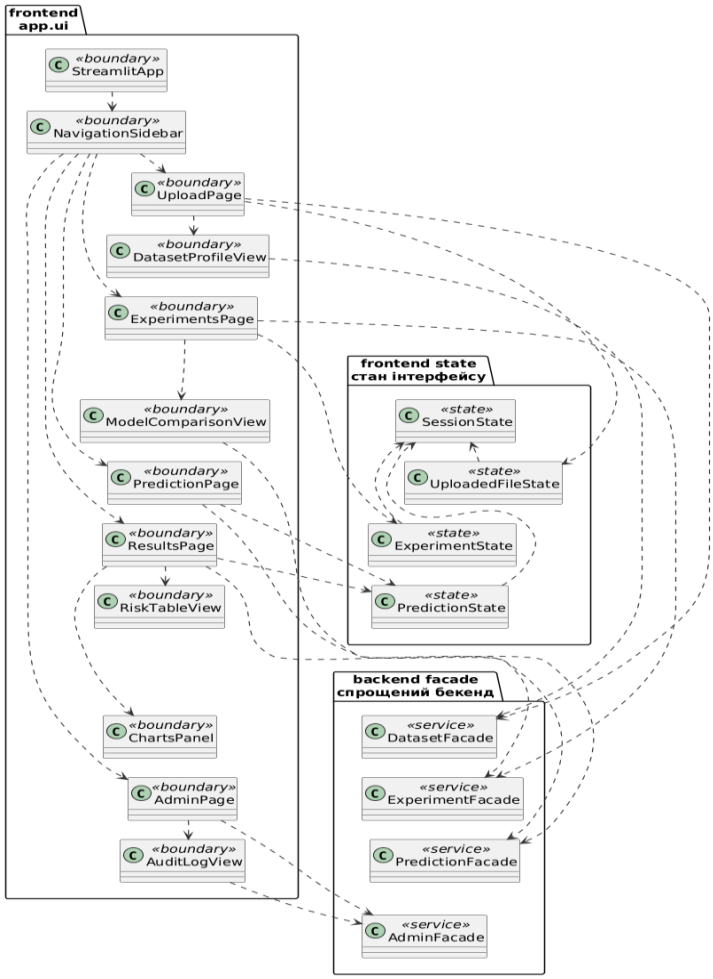


Рисунок 2.5 – UML-діаграма фронтенд-частини зі спрощеним бекендом

На рисунку 2.6 подано деталізовану структуру бекенд-частини додатка. Фронтенд згорнуто до класу StreamlitClient, який ініціює прикладні сценарії через сервісний шар. Основну роль у бекенді виконують сервіси DatasetService, ExperimentService, PredictionService та AdminService. Вони координують завантаження й перевірку даних, підготовку ознак, навчання моделей, класифікацію транзакцій, збереження результатів, роботу з користувачами та журналювання.

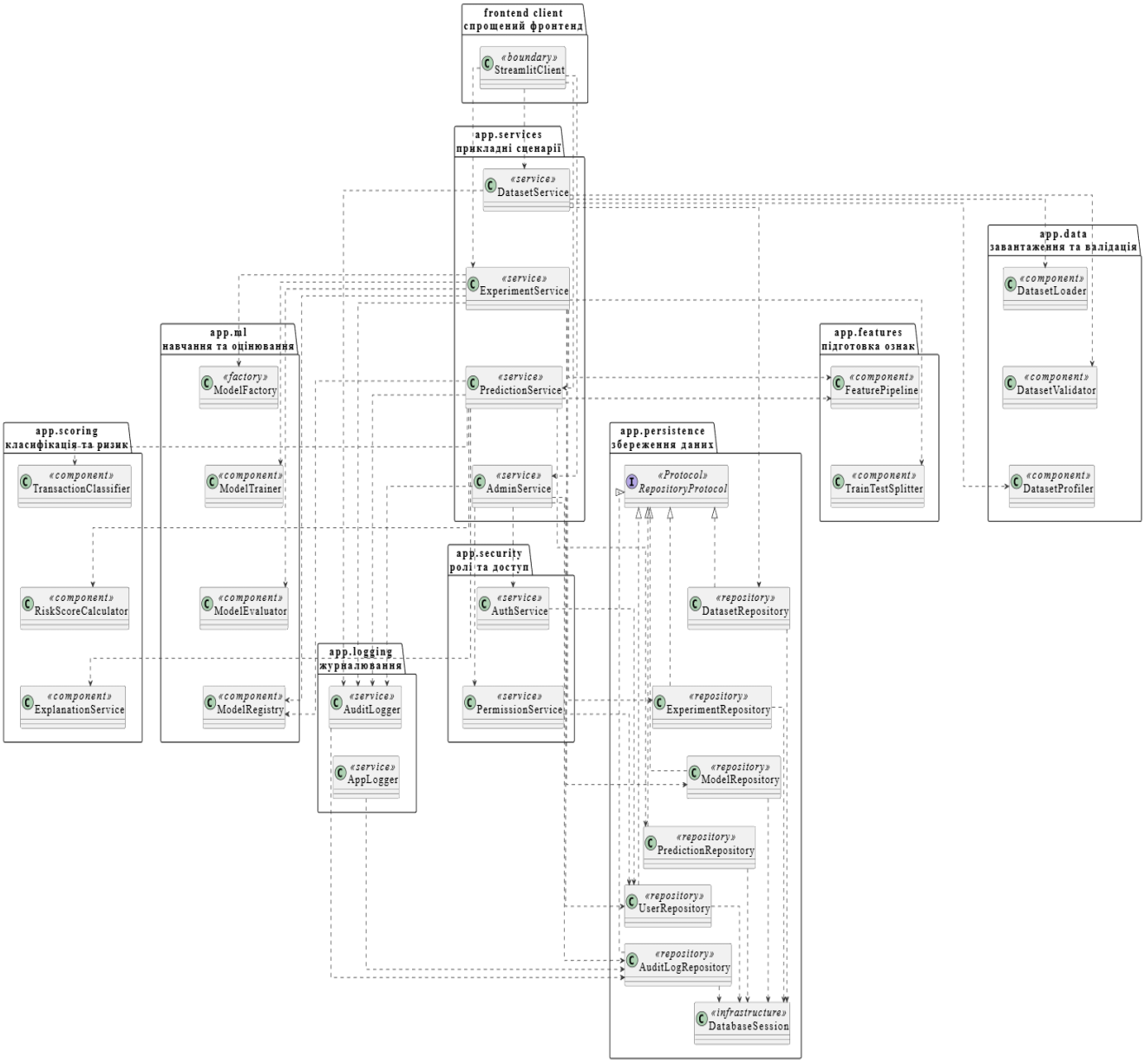


Рисунок 2.6 – UML-діаграма бекенд-частини інтелектуального додатка зі спрощеним поданням фронтенду

Аналітичний конвеєр реалізовано через послідовність пакетів data, features, ml і scoring. Пакет persistence ізолює роботу з базою даних за допомогою репозиторіїв і RepositoryProtocol, що відповідає Python-підходу до типізації через протоколи. Пакети security і logging забезпечують контроль доступу та фіксацію подій системи.

База даних системи має зберігати не власне банківські операції у повному промисловому обсязі, а метадані наборів даних, параметри експериментів, відомості про навчені моделі, метрики якості, результати класифікації, користувачів, ролі та журнал подій. Такий підхід є доцільним для прототипу, оскільки повні транзакційні дані можуть бути великими, різнорідними й чутливими. Тому завантажені файли доцільно зберігати у файловому сховищі, а в базі даних фіксувати їх опис, контрольну суму, кількість рядків, перелік колонок і результати аналізу.

Проєктування бази даних виконується з урахуванням таких принципів:

- дані користувачів, ролей і прав доступу зберігаються окремо;
- метадані наборів даних відокремлюються від самих файлів;
- результати валідації не змішуються з описом набору даних;
- експеримент, модель і метрики зберігаються як різні сутності;
- результати прогнозування прив’язуються до конкретного запуску прогнозування;
- журнал подій не дублює бізнес-таблиці, а лише фіксує факт дії, її тип, час і користувача;
- навчені моделі зберігаються як файлові артефакти, а база даних містить лише шлях, параметри й службові метадані.

У запропонованій схемі бази даних (рис. 2.7) більшість таблиць також відповідають нормальній формі Бойса-Кодда, оскільки кожний функціональний визначник є потенційним ключем або має унікальне обмеження. Наприклад, users.email і roles.code мають бути унікальними. Для таблиці datasets доцільно встановити унікальність file_hash, щоб один і той самий файл не завантажувався повторно як новий набір даних без потреби.

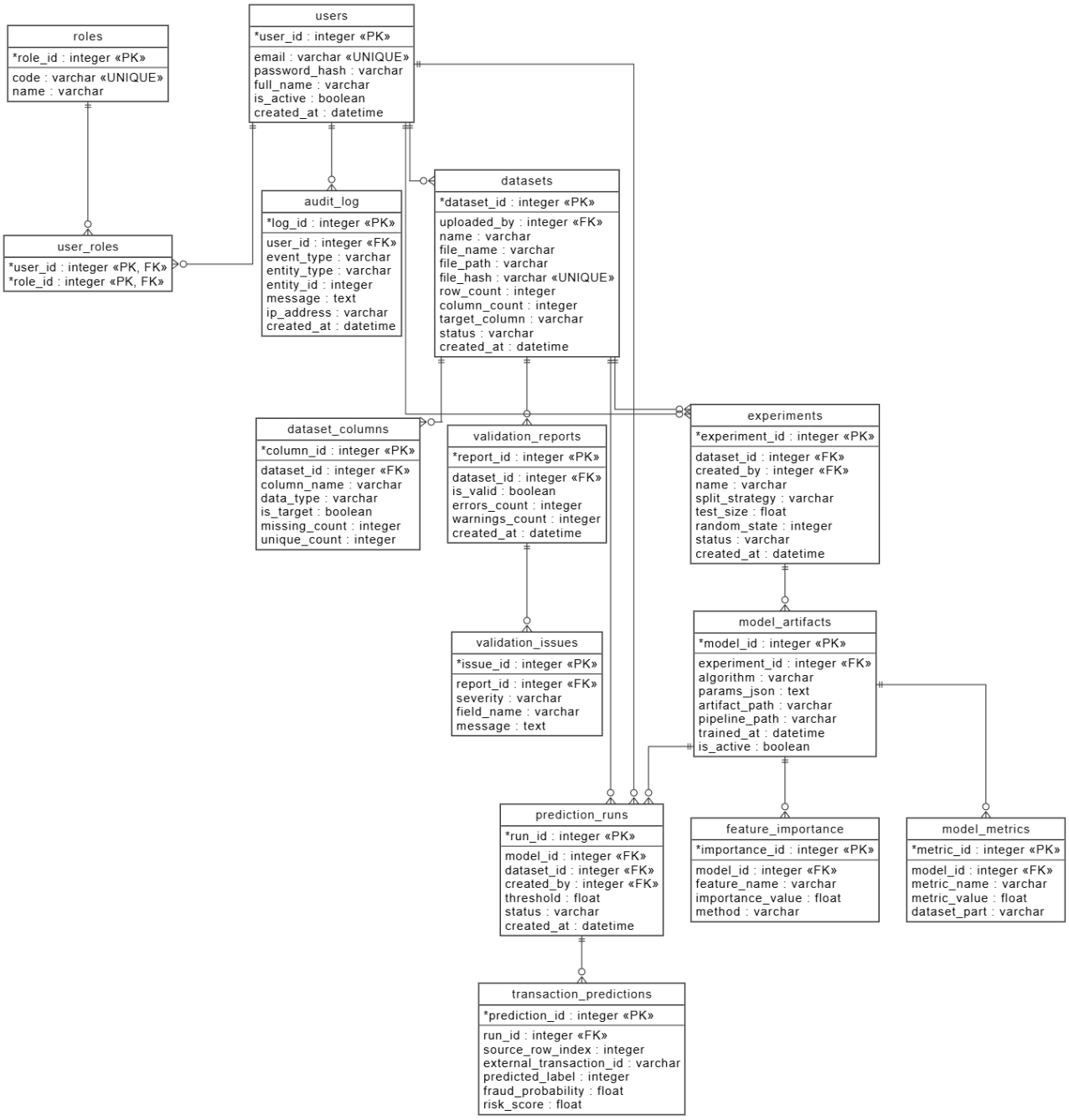


Рисунок 2.7 – Схема бази даних

Окреме пояснення потрібне для поля params_json у таблиці model_artifacts. Воно містить серіалізовані параметри моделі, структура яких залежить від алгоритму. Формально таке поле є напівструктурованим, але воно не порушує нормалізацію основної реляційної моделі, оскільки параметри використовуються як конфігураційний артефакт моделі, а не як сутності, за якими виконуються основні зв’язки бази даних. Якщо в майбутньому буде

потрібно аналізувати параметри моделей у запитах, це поле можна винести в окрему таблицю model_parameters.

Таблиця users зберігає облікові записи користувачів системи. Її ключовими полями є ідентифікатор користувача, електронна пошта, хеш пароля, повне ім’я, стан активності та дата створення. Пароль не зберігається у відкритому вигляді, а подається лише як хеш.

Таблиця roles містить перелік ролей системи: аналітик, дослідник моделей, адміністратор. Окрема таблиця user_roles реалізує зв’язок багато-до-багатьох між користувачами та ролями. Це дозволяє одному користувачу мати кілька ролей, що є корисним у прототипі, де одна особа може одночасно виконувати функції аналітика й дослідника.

Таблиця datasets описує завантажені набори даних. Вона містить назву набору, ім’я файлу, шлях до файлу, контрольну суму, кількість рядків і колонок, назву цільової ознаки та статус. Самі транзакційні файли не дублюються в базі даних, що зменшує її розмір і спрощує роботу з різнорідними форматами.

Таблиця dataset_columns деталізує структуру набору даних: назви колонок, типи даних, ознаку цільового поля, кількість пропусків і кількість унікальних значень. Це дозволяє зберігати результат первинного профілювання даних і використовувати його для перевірки коректності майбутніх запусків.

Таблиці validation_reports і validation_issues зберігають результати перевірки набору даних. Звіт містить загальний результат валідації, а окремі проблеми фіксуються в таблиці validation_issues. Завдяки цьому система може показувати користувачу не лише факт помилки, а й конкретні причини: відсутність цільової ознаки, некоректний тип поля, надмірна кількість пропусків тощо.

Таблиця experiments описує запуск навчання моделей. Вона пов’язана з набором даних і користувачем, який ініціював експеримент. У ній зберігаються стратегія розбиття вибірки, розмір тестової частини, початковий стан генератора випадкових чисел і статус експерименту.

Таблиця model_artifacts містить відомості про навчені моделі: алгоритм, параметри, шлях до збереженої моделі, шлях до конвеєра підготовки ознак, дату навчання та ознаку активності. Поле params_json використовується для збереження параметрів моделі у серіалізованому вигляді, оскільки набір параметрів залежить від конкретного алгоритму.

Таблиця model_metrics зберігає метрики якості моделей у нормалізованому вигляді. Замість того щоб створювати окремі колонки accuracy, precision, recall, f1, roc_auc, використовується універсальна структура: назва метрики, значення та частина даних, на якій її обчислено. Це дозволяє додавати нові метрики без зміни схеми таблиці.

Таблиця feature_importance зберігає відомості про важливість ознак для конкретної моделі. Вона потрібна для підтримки пояснюваності результатів: користувач може побачити, які ознаки найбільше впливали на рішення моделі.

Таблиця prediction_runs описує запуск класифікації транзакцій. Вона пов’язана з моделлю, набором даних і користувачем. У ній фіксується поріг прийняття рішення, статус запуску та час створення.

Таблиця transaction_predictions містить результати класифікації окремих рядків набору даних: індекс рядка у джерелі, зовнішній ідентифікатор транзакції, передбачений клас, імовірність шахрайства та ризикову оцінку. Такий підхід дозволяє пов’язати результат із початковим файлом без обов’язкового дублювання всіх транзакційних атрибутів у базі даних.

Таблиця audit_log фіксує службові події: завантаження файлу, запуск навчання, помилки валідації, запуск класифікації, експорт результатів, зміну ролей користувачів. Вона потрібна для аудиту, діагностики та відтворення сценаріїв роботи системи.

Для забезпечення коректності даних у базі необхідно передбачити обмеження, зібрані в табл. 2.5. Індекс за risk_score потрібний для швидкого відбору найбільш ризикових транзакцій. Індекси за created_at у журналі подій і таблицях запусків потрібні для сортування та фільтрації за часом.

Таблиця 2.5 – Обмеження в базі даних

Обмеження	Призначення
users.email UNIQUE	Забороняє дублювання облікових записів.
roles.code UNIQUE	Забезпечує унікальність системних ролей.
datasets.file_hash UNIQUE	Запобігає повторному завантаженню однакового файлу.
model_metrics(model_id, metric_name, dataset_part) UNIQUE	Не дозволяє дублювати одну й ту саму метрику для моделі.
feature_importance(model_id, feature_name, method) UNIQUE	Забезпечує унікальність оцінки важливості ознаки для конкретної моделі.
transaction_predictions(run_id, source_row_index) UNIQUE	Не допускає дублювання прогнозу для одного рядка в межах одного запуску.
FK між усіма пов’язаними таблицями	Забезпечує посилальну цілісність.

Загалом, деталізована архітектура інтелектуального додатка передбачає поділ системи на взаємопов’язані пакети та компоненти, що відповідають основним сценаріям роботи: завантаження даних, перевірка структури, підготовка ознак, навчання моделей, оцінювання якості, класифікація транзакцій, збереження результатів і адміністрування. Така структура забезпечує низьку зв’язаність компонентів, можливість модульного тестування та подальше розширення системи.

Спроєктована база даних зберігає користувачів, ролі, набори даних, результати валідації, експерименти, моделі, метрики, запуски прогнозування, результати класифікації та журнал подій. Перевірка на нормальні форми показує, що схема відповідає першій, другій і третій нормальним формам, а також у більшості таблиць задовольняє вимоги нормальної форми Бойса-Кодда. Це забезпечує цілісність, відтворюваність і супроводжуваність даних у межах розроблюваного програмного продукту.

2.4 Особливості програмної імплементації

Програмний конвеєр обробки даних охоплює повний цикл від завантаження файлу до отримання ризикових оцінок для окремих транзакцій. Узагальнено його можна подати так: вхідний файл → завантаження даних → валідація структури → профілювання набору → відокремлення X / y → поділ train/test → підготовка числових і категоріальних ознак → навчання

класифікатора → прогнозування ймовірності шахрайства → обчислення часткових ризиків → агрегована ризикова оцінка → порогове рішення → метрики якості → збереження моделі, метаданих і результатів.

На першому етапі користувач завантажує файл із транзакційними даними. Система визначає формат файлу, читає його в табличну структуру та перевіряє базові вимоги до набору даних. Далі набір профілюється: визначаються типи колонок, кількість рядків, кількість пропущених значень, кількість унікальних значень та наявність цільової ознаки.

Після перевірки дані поділяються на матрицю ознак X і цільовий вектор y. Цільова ознака має бути бінарною, де значення 0 відповідає легітимній транзакції, а значення 1 – потенційно шахрайській. Після цього виконується поділ на навчальну й тестову вибірки, що дає змогу оцінити якість моделі на даних, які не використовувалися під час навчання.

Для попередньої обробки використовується спільний конвеєр на основі Pipeline і ColumnTransformer бібліотеки scikit-learn. Числові ознаки проходять заповнення пропущених значень медіаною та масштабування. Категоріальні ознаки заповнюються найчастішим значенням і кодуються методом one-hot. Після цього підготовлені ознаки передаються до обраного класифікатора.

Система працює з табличними даними. На вхід приймаються файли таких форматів: .csv, .xlsx, .xls. Обов’язковою умовою є наявність цільової ознаки, яка задає клас транзакції. За замовчуванням може використовуватися назва is_fraud.

Усі ознаки, крім цільової, автоматично поділяються на числові та категоріальні. Водночас наявність додаткових колонок підвищує якість багатовекторного ризикового оцінювання. Наприклад, для поведінкового ризику корисними є ознаки пристрою, часу операції, попередньої активності користувача; для контекстного ризику – канал платежу, країна, IP-ознаки; для мережевого ризику – ідентифікатори рахунків, одержувачів або контрагентів.

У системі передбачено можливість використання кількох класифікаторів. Це важливо, оскільки метою роботи є не лише запуск однієї моделі, а й порівняння якості різних підходів на однаково підготовлених даних. Усі

класифікатори використовують спільний конвеєр підготовки ознак, що забезпечує коректність порівняння.

Таблиця 2.6 – Основні класифікатори

Класифікатор	Назва в системі	Призначення
Логістична регресія	logistic_regression	базова інтерпретована модель бінарної класифікації
Випадковий ліс	random_forest	ансамблева модель для нелінійних залежностей
Градієнтне підсилення	gradient_boosting	ансамблева модель для складніших закономірностей
Гістограмне градієнтне підсилення	hist_gradient_boosting	швидший варіант градієнтного підсилення для табличних даних
Наївний базовий класифікатор	dummy	контрольна модель для порівняння

З огляду на багатовекторну природу фінансового шахрайства ризикова оцінка транзакції не повинна зводитися лише до ймовірності, яку повертає класифікатор машинного навчання. У першому розділі було показано, що шахрайські операції можуть проявлятися на кількох рівнях: транзакційному, поведінковому, контекстному та мережевому. Тому в програмній реалізації ризикова оцінка розглядається як агрегований показник, що поєднує модельну ймовірність шахрайства з додатковими індикаторами ризику.

Для кожної транзакції система може формувати такі часткові компоненти:

$$R_i = F\left(R_i^{ML}, R_i^T, R_i^B, R_i^C, R_i^N\right), \quad (1)$$

де R_i^{ML} – модельна оцінка ризику, R_i^T – транзакційний ризик, R_i^B – поведінковий ризик, R_i^C - контекстний ризик, R_i^N – мережевий ризик. У межах реалізації кожний частковий ризик нормується до інтервалу $r_i^k \in [0; 1]$.

Модельна ймовірність шахрайства використовується як модельний компонент ризику та визначається як:

$$r_i^{ML} = p_i = P(y_i = 1|z_i),$$

Остаточна ризикова оцінка визначається через зважену агрегацію:

$$R_i = 100 \cdot \left(w_{ML} r_i^{ML} + w_T r_i^T + w_B r_i^B + w_C r_i^C + w_N r_i^N\right)$$

де вагові коефіцієнти задовольняють умову $w_{ML} + w_T + w_B + w_C + w_N = 1$. Значення ваг не є універсальними. Вони можуть бути змінені залежно від якості

вхідного набору даних, предметної області, доступності поведінкових і мережевих ознак, а також допустимого балансу між хибними спрацюваннями та пропущеними шахрайськими операціями.

Транзакційний ризик описує підозрілість самої операції: суму, категорію отримувача, тип платежу, відхилення від типових значень. Його можна подати так:

$$r_i^T = \alpha_1 \cdot norm(amount_i) + \alpha_2 \cdot amountDeviation_i + \alpha_3 \cdot merchantRisk_i,$$

де $norm(amount_i)$ – нормована сума операції, $amountDeviation_i$ – відхилення суми від типової для користувача або рахунку; $merchantRisk_i$ – ризик категорії торгової точки або отримувача; $\alpha_1, \alpha_2, \alpha_3$ – ваги всередині транзакційного компонента.

Поведінковий ризик описує відхилення від звичного профілю користувача:

$$r_i^B = \beta_1 \cdot newDevice_i + \beta_2 \cdot unusualTime_i + \beta_3 \cdot loginToPaymentSpeed_i,$$

де $newDevice_i$ – ознака використання нового пристрою; $unusualTime_i$ – нетиповий час активності; $loginToPaymentSpeed_i$ – підозріло швидкий перехід від входу до платежу; $\beta_1, \beta_2, \beta_3$ – ваги поведінкових індикаторів.

Контекстний ризик враховує канал, географію, IP-ознаки та середовище виконання операції:

$$r_i^C = \gamma_1 \cdot channelRisk_i + \gamma_2 \cdot geoMismatch_i + \gamma_3 \cdot ipRisk_i,$$

де $channelRisk_i$ – ризик каналу платежу, $geoMismatch_i$ – невідповідність географії операції типовому профілю, $ipRisk_i$ – ризик, пов'язаний з IP-адресою або мережевим середовищем, $\gamma_1, \gamma_2, \gamma_3$ – ваги контекстних індикаторів.

Мережевий ризик потрібний для сценаріїв фінансових посередників, ланцюгового переказу коштів і приховування походження транзакції:

$$r_i^N = \delta_1 \cdot newBeneficiary_i + \delta_2 \cdot transferChainRisk_i + \delta_3 \cdot accountCentralityRisk_i,$$

де $newBeneficiary_i$ – ознака нового одержувача, $transferChainRisk_i$ – ризик ланцюгового або каскадного переказу, $accountCentralityRisk_i$ – підозріла роль рахунку в мережі переказів, $\delta_1, \delta_2, \delta_3$ – ваги мережевих індикаторів.

На відміну від ручного режиму, в адаптивному режимі ваги не задаються наперед. Для кожного компонента обчислюється службова оцінка інформативності:

$$q_k = a_k \cdot v_k \cdot d_k,$$

де a_k – коефіцієнт доступності компонента, v_k – коефіцієнт варіативності компонента, d_k – коефіцієнт роздільної здатності компонента щодо класів.

Коефіцієнт доступності a_k показує, чи може відповідний компонент бути обчислений на основі наявних колонок датасету. Якщо у файлі немає ознак, пов'язаних із пристроєм, IP-адресою, каналом платежу або отримувачем, поведінковий, контекстний або мережевий ризик не повинен отримувати значну вагу лише через те, що така формула передбачена в архітектурі. У цьому випадку система зменшує або зануляє відповідний внесок.

Коефіцієнт варіативності v_k показує, чи змінюється компонент ризику між різними транзакціями. Якщо певний компонент має майже однакове значення для всіх рядків, він не допомагає ранжувати транзакції за рівнем підозрілості. Наприклад, якщо для всіх операцій network_risk = 0, мережевий компонент фактично не бере участі в оцінюванні. Такий компонент не вилучається з архітектури, але його вага в поточному запуску зменшується.

Коефіцієнт роздільної здатності d_k використовується тоді, коли у тестовому або навчальному наборі доступні істинні мітки шахрайства. Він показує, наскільки середнє значення компонента ризику відрізняється для шахрайських і легітимних транзакцій:

$$d_k = \left| \overline{r_k^{fraud}} - \overline{r_k^{legit}} \right|.$$

Якщо компонент у середньому вищий для шахрайських транзакцій, ніж для легітимних, він має корисну роздільну здатність і може отримати більшу вагу. Якщо ж різниця між класами майже відсутня, його внесок не повинен бути високим. Такий підхід дозволяє не покладатися лише на експертно задані коефіцієнти, а використовувати інформацію, наявну в самому наборі даних.

Після обчислення службових оцінок q_k ваги нормуються:

$$w_k = \frac{q_k}{\sum_j q_j}.$$

Якщо сума службових оцінок дорівнює нулю, наприклад через відсутність усіх додаткових компонентів, система переходить до безпечної резервної схеми, у якій основна вага надається модельній імовірності. Це дозволяє зберегти працездатність системи навіть для мінімального датасету, що містить лише базові транзакційні ознаки.

Адаптивність ваг не означає, що система автоматично знаходить універсально оптимальну формулу ризику для будь-якого банківського середовища. Вона виконує більш обмежене, але практично важливе завдання: запобігає використанню неінформативних компонентів і підлаштовує підсумкову оцінку під фактичну структуру вхідних даних. Для кваліфікаційної роботи це є доцільним компромісом між повністю ручною експертною моделлю і складною промисловою системою навчання ризикової політики.

У програмному коді за це відповідає клас AdaptiveRiskWeightEstimator. Він отримує таблицю часткових ризиків і, за наявності, істинні мітки класів. На основі цих даних обчислюються ваги для п’яти компонентів: ml, transaction, behavior, context, network. Після нормалізації ці ваги передаються до RiskScoreCalculator, який формує підсумкову оцінку ризику для кожної транзакції. Такий поділ відповідальності є важливим: один модуль визначає ваги, інший застосовує їх для розрахунку ризику. Загальна схема роботи адаптивної моделі така: часткові ризики транзакцій → перевірка доступності компонентів → оцінка варіативності компонентів → оцінка роздільної здатності за істинними мітками → нормалізація ваг → обчислення агрегованого ризику → порогове рішення.

Після адаптивного визначення ваг система продовжує використовувати порогове правило класифікації. У застосунку поріг може задаватися користувачем, що дає змогу змінювати чутливість системи. Для антишахрайських задач це важливо, оскільки надто низький поріг збільшує

кількість хибних спрацювань, а надто високий — підвищує ризик пропуску шахрайських операцій.

Щоб зменшити ризик перенавчання, параметри деревних моделей у проєкті обмежено. Для випадкового лісу застосовуються обмеження глибини дерева, мінімальної кількості об’єктів у листі та мінімальної кількості об’єктів для поділу вузла. Для гістограмного градієнтного підсилення використовуються параметри регуляризації та ранньої зупинки. Якщо метрики на навчальній вибірці суттєво перевищують тестові або якщо тестові значення виглядають підозріло ідеальними, наприклад дорівнюють 1.0, інтерфейс формує попередження. Така перевірка не є повною математичною гарантією від перенавчання, але вона допомагає виявити типові проблеми: надто простий синтетичний датасет, витік цільової ознаки в набір предикторів, дублювання записів або надмірно складну модель для малого набору даних. Для кількісного оцінювання узагальнювальної здатності моделей у scikit-learn також передбачено засоби перехресної перевірки, зокрема cross_validate, що дозволяє оцінювати модель на кількох розбиттях даних.

Після запуску користувач бачить вебсторінку інтелектуального додатка для виявлення шахрайських транзакцій (рис. 2.8). У бічній панелі розміщуються основні параметри запуску: назва цільової ознаки, вибір моделі, поріг ризику та, за потреби, ваги окремих компонентів ризикової оцінки. Основні кроки роботи:

1) користувач завантажує CSV або XLSX-файл із транзакціями;
2) система зчитує файл і показує загальну інформацію про набір даних;
3) виконується перевірка структури набору;
4) користувач обирає класифікатор;
5) система поділяє дані на навчальну й тестову вибірки;
6) виконується підготовка ознак;
7) навчається модель;
8) обчислюються метрики якості;
9) виконується прогнозування для транзакцій;
10) формується багатокомпонентна ризикова оцінка;

11) результати сортуються за рівнем ризику;

12) користувач переглядає підозрілі транзакції;

13) система експортує результати у CSV-файл.

Після завантаження файлу система показує короткий профіль набору даних. У ньому відображається кількість рядків, кількість колонок, перелік числових і категоріальних ознак, кількість пропущених значень і наявність цільової ознаки. Якщо виявлено помилки, наприклад відсутність цільової колонки або небінарний клас, система блокує навчання та показує повідомлення про помилку.

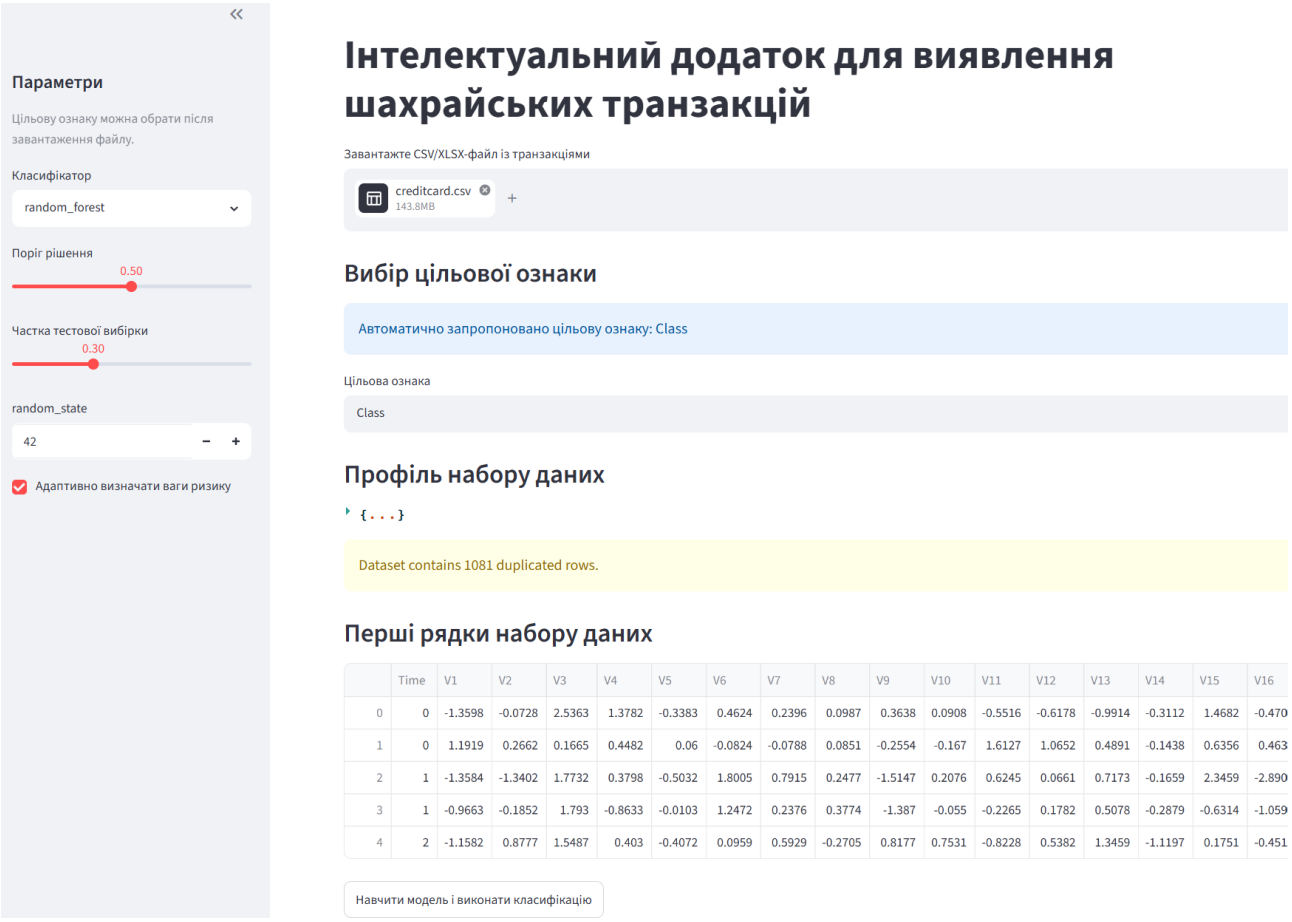


Рисунок 2.8 – Вигляд програми в дії

Після успішної перевірки користувач запускає навчання. Система показує метрики якості моделі. Після прогнозування вона формує таблицю ризикових транзакцій. Результати сортуються за спаданням risk_score, тому найбільш

підозрілі операції потрапляють на початок списку. Окрім таблиці, інтерфейс може містити графічні подання: розподіл ризикових оцінок, матрицю помилок, ROC-криву, порівняння метрик кількох моделей. Це полегшує аналіз результатів і дає змогу оцінити не лише окремі транзакції, а й загальну поведінку моделі.

Запропонована програмна імплементація поєднує модульну структуру Python-застосунку, відтворюваний конвеєр машинного навчання та багатокомпонентне ризикове оцінювання транзакцій. Ключовою особливістю реалізації є те, що класифікатор не ототожнюється з усією антишахрайською логікою. Він формує лише модельний компонент ризику, тоді як підсумкова оцінка може враховувати також транзакційні, поведінкові, контекстні й мережеві індикатори.

Такий підхід узгоджується з предметною областю, оскільки фінансове шахрайство не завжди проявляється лише в окремому значенні суми або одному передбаченні моделі. Воно може бути пов’язане з нетиповим каналом платежу, новим пристроєм, незвичною географією, швидкістю дій користувача після входу, новим одержувачем або ланцюговим переміщенням коштів. Тому реалізований прототип має не лише класифікаційну, а й аналітичну спрямованість: він дозволяє ранжувати транзакції за ризиком, аналізувати часткові компоненти підозрілості та зберігати результати для подальшого дослідження.

РОЗДІЛ 3 ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНА ОЦІНКА ІНТЕЛЕКТУАЛЬНОГО ДОДАТКА

3.1 Вибір методів тестування програмного продукту

Загальна логіка стандарту ISO/IEC/IEEE 29119 розглядає тестування як керований процес перевірки програмних систем. Тому для програмного прототипу інтелектуального додатка доцільно будувати компактний набір перевірок, які безпосередньо підтверджують працездатність ключових частин системи.

Для розробленого додатка абсолютно необхідними є чотири групи тестування: модульне тестування обчислювальної логіки, інтеграційне тестування конвеєра машинного навчання, тестування коректності ризикового оцінювання та сценарне тестування інтерфейсу користувача. Саме ці перевірки покривають основні ризики проєкту: некоректне зчитування даних, помилки у підготовці ознак, неправильний вибір або навчання класифікатора, помилки в розрахунку ризику, некоректне збереження результатів і неможливість користувача виконати повний сценарій роботи.

Першим обов’язковим рівнем є модульне тестування. Воно застосовується до тих частин коду, які мають чітко визначену вхідну та вихідну поведінку: завантажувача даних, валідатора набору, профілювальника, фабрики моделей, обчислювача метрик і калькулятора ризику. Для Python-проєкту доречно використовувати тестовий фреймворк pytest [37], оскільки він підтримує просту організацію тестових функцій, фікстури та тимчасові каталоги для перевірки операцій із файлами. Зокрема, фікстура tmp_path дозволяє створювати ізольовані тимчасові директорії для тестів збереження моделей, експорту CSV-файлів і проміжних результатів, не змінюючи робочі файли проєкту.

У межах модульного тестування насамперед перевіряється коректність валідації даних. Система повинна відхиляти порожній файл, файл без цільової ознаки, набір із небінарним класом і некоректний формат файлу. Водночас наявність пропущених значень або дублікатів не завжди має блокувати роботу

системи, оскільки частина таких проблем може бути усунута на етапі попередньої обробки. Тому тести мають розрізняти критичні помилки, які зупиняють навчання, і попередження, які повідомляють користувача про якість набору даних. Мінімальний набір тестів для цього рівня повинен перевіряти наявність цільової колонки is_fraud, бінарність значень цільової ознаки, правильне визначення числових і категоріальних колонок, а також коректне формування профілю набору даних.

Другим необхідним рівнем є тестування конвеєра підготовки ознак. У цьому проєкті підготовка ознак має особливе значення, оскільки однаковий конвеєр використовується і під час навчання, і під час прогнозування. Тестування повинно підтвердити, що числові ознаки заповнюються медіаною та масштабуються, категоріальні ознаки заповнюються найчастішим значенням і кодуються, а нові категорії у вхідних даних не спричиняють аварійного завершення роботи. Це особливо важливо через використання OneHotEncoder(handle_unknown="ignore"), який має забезпечувати стійкість системи до категорій, відсутніх у навчальному наборі. Така перевірка може бути реалізована на невеликому штучному наборі даних із кількома числовими, категоріальними та пропущеними значеннями.

Третім обов’язковим рівнем є тестування класифікаторів і метрик якості. У коді передбачено кілька моделей: логістичну регресію, випадковий ліс, градієнтне підсилення, гістограмне градієнтне підсилення та базовий наївний класифікатор. Для цього достатньо перевірити, що фабрика моделей створює правильний об’єкт за заданою назвою алгоритму, а невідома назва моделі породжує контрольовану помилку. Окремо потрібно перевірити, що після навчання модель повертає прогноз класу та ймовірність шахрайського класу. Метрики якості слід тестувати на невеликому контрольному прикладі, де значення TP, TN, FP, FN можна обчислити вручну. Для цього проєкту важливо не обмежуватися метрикою точності, оскільки за незбалансованих даних вона може бути оманливою; тому в тестах і подальшому експериментальному аналізі мають перевірятися влучність, повнота, F1 та ROC-AUC. Документація scikit-

learn [38] визначає модуль sklearn.metrics як основний засіб обчислення показників якості прогнозування, зокрема для задач класифікації, а ROC-AUC обчислюється на основі оцінок або ймовірностей класів.

Окремим і принципово важливим видом тестування є перевірка багатокомпонентної ризикової оцінки. Оскільки ризик у системі не ототожнюється лише з імовірністю класифікатора, необхідно тестувати правильність агрегації часткових компонентів: модельного, транзакційного, поведінкового, контекстного та мережевого. Такі тести повинні перевіряти, що вагові коефіцієнти сумуються до одиниці, усі часткові ризики обмежуються інтервалом $[0; 1]$, підсумкова оцінка перебуває в межах $[0; 100]$, а збільшення одного з компонентів ризику не зменшує підсумкову оцінку за незмінних інших параметрів. Також необхідно перевірити роботу системи за відсутності окремих колонок у вхідному наборі. Наприклад, якщо файл не містить ознак пристрою або рахунку-одержувача, поведінковий чи мережевий компонент ризику не повинен призводити до помилки, а має набувати нейтрального або нульового значення. Це підтверджує, що реалізація узгоджується з багатовекторною природою шахрайства, але залишається придатною для роботи з неповними навчальними наборами.

Наступним обов’язковим рівнем є інтеграційне тестування повного конвеєра. Воно перевіряє не окремі класи, а весь шлях обробки: зчитування файлу, перевірка структури, поділ на X та y, підготовка ознак, навчання моделі, обчислення метрик, прогнозування, розрахунок ризику й експорт результатів. Для такого тесту достатньо невеликого синтетичного CSV-файлу на 20-50 рядків. Метою є підтвердження того, що програмний конвеєр виконується без помилок і повертає очікувані результати. Успішним результатом інтеграційного тесту є наявність словника метрик, таблиці прогнозів із колонками predicted_label, ml_probability, transaction_risk, behavior_risk, context_risk, network_risk, aggregated_risk_probability, risk_score, а також створення CSV-файлу експорту.

Сценарне тестування інтерфейсу користувача також є необхідним, але в межах кваліфікаційної роботи його достатньо виконати як ручне тестування за контрольними сценаріями. Повна автоматизація Streamlit-інтерфейсу потребує додаткових інструментів і часу, тому для коротких термінів доцільніше перевірити основні користувацькі сценарії вручну: запуск програми, завантаження коректного файлу, відображення профілю набору, реакцію на файл без цільової ознаки, вибір класифікатора, зміну порога ризику, зміну ваг ризикової моделі, запуск навчання, перегляд метрик, перегляд таблиці ризикових транзакцій та експорт результатів. Такий підхід дозволяє підтвердити, що користувач може пройти повний сценарій роботи без звернення до внутрішнього коду.

Для реалізації тестування в короткі терміни доречно обмежитися мінімальним, але достатнім набором тестових файлів: test_dataset_validator.py, test_feature_pipeline.py, test_model_factory.py, test_model_evaluator.py, test_risk_score.py, test_prediction_pipeline.py і test_export.py. Такий набір покриває критичні ділянки системи, не потребує складної інфраструктури й може запускатися однією командою pytest. Додатково доцільно виконувати базову перевірку якості коду за допомогою форматування та статичного аналізу, наприклад black --check . і ruff check ., однак ці перевірки слід розглядати як допоміжні, а не як заміну функціональному тестуванню.

Отже, для цього програмного продукту найбільш доцільною є компактна стратегія тестування, що поєднує модульні тести критичних обчислювальних компонентів, інтеграційний тест повного ML-конвеєра, окрему перевірку багатокомпонентної ризикової моделі та ручне сценарне тестування інтерфейсу. Такий набір перевірок є реалістичним для виконання в межах кваліфікаційної роботи, але водночас підтверджує працездатність основних архітектурних рішень: модульної структури Python-застосунку, відтворюваного конвеєра підготовки даних і навчання, багатовекторного ризикового оцінювання та експорту результатів аналізу.

3.2 Формування тест-плану

У межах кваліфікаційної роботи основна увага приділяється модульним, інтеграційним і сценарним перевіркам, які підтверджують працездатність ключових функцій системи. У той же час, не передбачається повне навантажувальне, безпекове та промислове приймальне тестування. Інструменти тестування:

– фреймворк pytest для модульних та інтеграційних тестів; tmp_path для ізольованої перевірки збереження моделей і CSV-експорту;

– ручні сценарії для перевірки Streamlit-інтерфейсу;

– scikit-learn.metrics для перевірки метрик класифікації.

Фреймворк pytest підтримує тимчасові каталоги через фікстуру tmp_path, що зручно для тестування операцій із файлами без зміни робочого каталогу проєкту, а scikit-learn надає засоби оцінювання якості класифікації, зокрема метрики, що можуть використовувати ймовірності або оцінки позитивного класу. Основні тестові сценарії наведені в табл. 3.1.

Таблиця 3.1 – Основні тестові сценарії для застосунку

ID	Об’єкт	Передумови	Дії	Очікуваний результат
UT-01	DatasetValidator	CSV містить колонку is_fraud зі значеннями 0/1	Запустити перевірку структури	Набір позначається як коректний
UT-02	DatasetValidator	CSV не містить is_fraud	Запустити перевірку	Повертається критична помилка про відсутність цільової ознаки
UT-03	DatasetValidator	is_fraud містить значення 0/1/2	Запустити перевірку	Повертається помилка небінарної цільової ознаки
UT-04	DatasetValidator	Набір містить дублікати	Запустити перевірку	Система формує попередження, але не обов’язково блокує роботу
UT-05	DatasetProfiler	Набір містить числові й категоріальні колонки	Побудувати профіль	Коректно визначаються типи, пропуски, кількість рядків і колонок
UT-06	ModelFactory	Передано random_forest	Створити модель	Повертається RandomForestClassifier

Продовження таблиці 3.1

UT-07	ModelFactory	Передано невідому назву моделі	Створити модель	Повертається контрольована помилка ValueError
UT-08	FeaturePipeline	Є числові, категоріальні та пропущені значення	Виконати fit_transform	Дані перетворюються без помилок

Інтеграційне тестування перевіряє узгоджену роботу кількох модулів. Для цього достатньо використати невеликий синтетичний файл на 20–50 рядків. Мета тесту – підтвердити, що весь конвеєр виконується до кінця. Базові сценарії зібрано в табл. 3.2.

Таблиця 3.2 – Основні сценарії інтеграційного тестування

ID	Сценарій	Передумови	Дії	Очікуваний результат
IT-01	Повний запуск із коректним CSV	Є валідний файл із is_fraud	Завантажити дані, навчити модель, виконати прогноз	Повертаються метрики, таблиця прогнозів і ризикові оцінки
IT-02	Повний запуск із категоріальними ознаками	У файлі є текстові колонки	Запустити навчання	One-hot кодування виконується без помилок
IT-03	Повний запуск із пропущеними значеннями	У файлі є NaN	Запустити навчання	Пропуски обробляються конвеєром
IT-04	Прогноз після навчання	Модель навчена	Виконати прогноз на тому самому наборі	Кількість рядків результату відповідає кількості вхідних рядків
IT-05	Зміна класифікатора	Доступні кілька моделей	Запустити навчання для двох різних моделей	Для кожної моделі формується окремий результат

Окремо перевіряються операції з файлами: збереження результатів прогнозування, збереження моделі та повторне завантаження збереженого конвеєра. Для таких перевірок доцільно використовувати tmp_path, щоб тести створювали файли в тимчасовому каталозі й не змінювали робочі дані проєкту. Відповідні перевірки поєднано в табл. 3.3.

Таблиця 3.3 – Тестування експорту та файлових операцій

ID	Об’єкт	Передумови	Дії	Очікуваний результат	Пріоритет
EX-01	Експорт прогнозів	Є таблиця результатів	Зберегти CSV у тимчасовий каталог	Файл створено	Середній
EX-02	Експорт прогнозів	CSV створено	Прочитати файл	У файлі є risk_score та predicted_label	Середній
EX-03	ModelRegistry	Навчений pipeline	Зберегти модель	Створюється файл моделі	Середній
EX-04	ModelRegistry	Модель збережено	Завантажити модель	Завантажений об’єкт можна використати для прогнозу	Середній

Тестування багатокомпонентної ризикової оцінки є критичною, оскільки ризик у системі не зводиться лише до ймовірності класифікатора. Перевіряється коректність агрегування модельного, транзакційного, поведінкового, контекстного та мережевого компонентів. Систематизація відповідних тестів здійснена в табл. 3.4.

Таблиця 3.4 – Базові тести багатокомпонентної ризикової оцінки

ID	Об’єкт	Вхідні дані	Дії	Очікуваний результат
RS-01	RiskWeights	Ваги сумуються до 1.0	Виконати validate()	Помилка не виникає
RS-02	RiskWeights	Ваги сумуються до 1.1	Виконати validate()	Повертається ValueError
RS-03	RiskScoreCalculator	Часткові ризики в межах [0;1]	Обчислити ризик	risk_score перебуває в межах [0;100]
RS-04	RiskScoreCalculator	ml=0.2 і ml=0.8, інші значення однакові	Порівняти результати	Вищий ml дає вищий підсумковий ризик
RS-05	RiskScoreCalculator	transaction=1.5, context=-0.2	Обчислити ризик	Значення обрізаються до меж [0;1]
RS-06	RiskIndicatorExtractor	Відсутні колонки пристрою або одержувача	Обчислити часткові ризики	Система не падає, відсутні компоненти мають нейтральне/нульове значення
RS-07	Порогове рішення	aggregated_risk_probability=0.6, threshold=0.5	Обчислити клас	predicted_label = 1
RS-08	Порогове рішення	aggregated_risk_probability=0.4, threshold=0.5	Обчислити клас	predicted_label = 0

Метрики перевіряються на контрольованому прикладі, де очікувані значення можна обчислити вручну. Це дає змогу підтвердити правильність структури результатів і уникнути помилок у подальшому експериментальному розділі. Відповідні тестові кейси показано в табл. 3.5.

Таблиця 3.5 – Перевірка якості виявлення шахрайських операцій

ID	Об’єкт	Вхідні дані	Дії	Очікуваний результат
MT-01	ModelEvaluator	y_true=[1,1,0,0], y_pred=[1,0,1,0]	Обчислити accuracy	accuracy = 0.5
MT-02	ModelEvaluator	Ті самі дані	Обчислити precision	precision = 0.5
MT-03	ModelEvaluator	Ті самі дані	Обчислити recall	recall = 0.5
MT-04	ModelEvaluator	Ті самі дані	Обчислити F1	F1 = 0.5
MT-05	ModelEvaluator	Є y_true і ймовірності позитивного класу	Обчислити ROC-AUC	Повертається число в межах [0;1]

Сценарне тестування інтерфейсу користувача здійснювалось вручну на основі переліку базових дій користувача (табл. 3.6). Воно підтверджує, що користувач може пройти повний сценарій без роботи з кодом.

Таблиця 3.6 – Сценарне тестування користувацького інтерфейсу

ID	Сценарій	Передумови	Дії користувача	Очікуваний результат
UI-01	Запуск програми	Залежності встановлено	Виконати streamlit run app/ui/streamlit_app.py	Відкривається вебінтерфейс
UI-02	Завантаження коректного файлу	Є валідний CSV	Завантажити файл	Показується профіль набору даних
UI-03	Некоректний файл без цільової ознаки	CSV без is_fraud	Завантажити файл	Показується помилка, навчання блокується
UI-04	Вибір класифікатора	Файл валідний	Обрати модель у бічній панелі	Обрана модель використовується під час навчання
UI-05	Зміна порога ризику	Є результати прогнозу	Змінити threshold	Змінюється кількість транзакцій із класом 1
UI-06	Зміна ваг ризику	Є результати прогнозу	Змінити ваги компонентів	Перераховується risk_score
UI-07	Перегляд результатів	Прогноз виконано	Переглянути таблицю	Результати відсортовані за спаданням ризику
UI-08	Експорт результатів	Прогноз виконано	Натиснути експорт	Створюється CSV-файл результатів

Такий тест-план є достатнім для розробленого програмного прототипу, оскільки він покриває саме ті частини системи, від яких залежить демонстрація працездатності розробленого програмного продукту: коректність даних, відтворюваність конвеєра машинного навчання, правильність багатокомпонентного ризикового оцінювання, роботу інтерфейсу та експорт результатів.

3.3 Організація та проведення тестування

Для тестування підібрано два типи наборів даних. Перший тип – вбудовані синтетичні датасети, які входять до проєкту й дають можливість виконати тести без доступу до зовнішніх ресурсів. Основний демонстраційний файл sample_transactions.csv містить числові, категоріальні, поведінкові, контекстні та мережеві ознаки: суму транзакції, категорію операції, країну, час, тип пристрою, канал платежу, вік облікового запису, ознаки нового пристрою, географічної невідповідності та нового одержувача.

Другий тип – зовнішні набори, рекомендовані для подальшого експериментального розділу: Credit Card Fraud Detection [39], IEEE-CIS Fraud Detection [40] і PaySim [41]. Набір Credit Card Fraud Detection містить транзакції європейських держателів карток за вересень 2013 року та є сильно незбалансованим; IEEE-CIS Fraud Detection орієнтований на прогнозування ймовірності шахрайства в транзакціях; PaySim є синтетичним симулятором мобільних фінансових операцій на основі реальних агрегованих логів.

Для практичного тестування та експериментального дослідження ці набори даних виконують різні ролі. Вбудовані синтетичні файли використовуються насамперед для перевірки працездатності програмного коду, оскільки їхня структура повністю контрольована розробником. Це дає змогу заздалегідь передбачити очікувану поведінку системи: коректне визначення цільової ознаки, обробку пропущених значень, реакцію на дублікати, роботу з категоріальними ознаками, обчислення часткових ризиків та експорт результатів. Такі файли не призначені для доведення високої якості моделі, але є

необхідними для регресійного тестування, оскільки дозволяють швидко перевіряти, що після змін у коді не порушено базову логіку роботи застосунку.

Зовнішні датасети мають інше призначення: вони використовуються для оцінювання поведінки моделей на більш реалістичних або загальновизнаних наборах даних. Датасет Credit Card Fraud Detection доцільно використовувати як базовий зовнішній набір для експериментів, оскільки він компактніший за IEEE-CIS, має чітку бінарну цільову ознаку Class, містить 284 807 транзакцій і 492 шахрайські операції, тобто демонструє характерну для задачі виявлення шахрайства сильну незбалансованість класів. Більшість його ознак подано у вигляді анонімізованих компонентів V1–V28, тоді як явно інтерпретованими залишаються лише Time і Amount. Це робить набір придатним для тестування класифікаторів, але обмежує можливості повноцінного поведінкового, контекстного й мережевого аналізу.

У межах тестування дані використовуються за таким принципом: синтетичні файли перевіряють технічну коректність модулів, Credit Card Fraud Detection – якість класифікації на реальному незбалансованому наборі, PaySim – потенційну придатність системи до сценаріїв переказу коштів і мережевого ризику, а IEEE-CIS – можливість масштабування на складніші транзакційні дані. Успішність виконання тестового плану відображено на рис. 3.1.

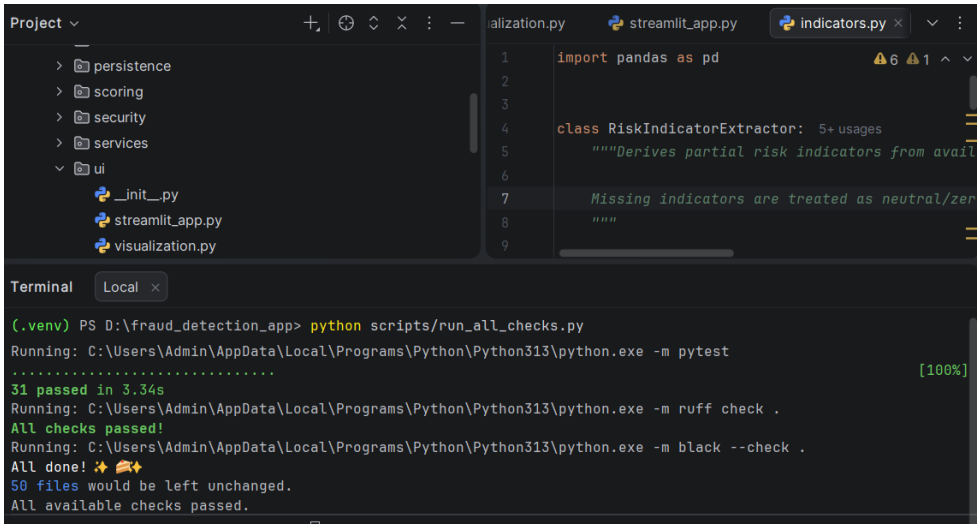


Рисунок 3.1 – Успішні результати виконання автоматизованого тестування

У межах реалізації підтверджено виконання автоматизованих тестів: усі тести pytest завершилися успішно. Це підтверджує коректність базових сценаріїв роботи коду, зокрема валідації даних, побудови конвеєра ознак, створення моделей, обчислення метрик, багатокомпонентного ризикового оцінювання, прогнозування, експорту результатів і збереження моделей. Добір датасетів забезпечив баланс між відтворюваністю тестування, реалістичністю експериментів і відповідністю архітектурній моделі системи. Вбудовані набори гарантують стабільне виконання автоматизованих тестів без зовнішніх залежностей, а зовнішні набори дозволяють оцінити роботу класифікаторів і ризикової моделі в умовах, наближених до реальних або імітаційних фінансових транзакцій.

3.4 Результати комп’ютерних експериментів

Після встановлення залежностей для запуску застосунку використовується команда streamlit run app/ui/streamlit_app.py. Користувач може завантажити data/samples/sample_transactions.csv, обрати класифікатор, задати поріг рішення та ваги компонентів ризику. Система виконує валідацію, профілювання, навчання моделі, прогнозування, обчислення часткових ризиків і формування підсумкового risk_score. Результати експортуються до каталогу data/exports.

Перший комп’ютерний експеримент будувався на основі моделі випадкового лісу та датасету Credit Card Fraud Detection при частці тестової вибірки 30% та порогу рішення – 50% (рис. 3.2). Адаптивно визначені ваги ризику для такі: модельний = 0.9625, транзакційний = 0.0167, поведінковий = 0.0083, контекстний = 0.0083, мережевий = 0.0042. Ознаки V1–V28 уже є стислим числовим представленням початкових транзакційних характеристик після зменшення розмірності методом PCA. Вони не мають зрозумілих назв, але саме в них зосереджена основна інформація для класифікації. Випадковий ліс працює з усіма цими числовими ознаками одночасно й може виявляти нелінійні комбінації, які відрізняють шахрайські операції від легітимних. Тому

адаптивний механізм визначив, що найбільшу роздільну здатність має саме результат класифікатора, а не окремі евристичні компоненти.

Рисунок 3.2 – Результати оцінювання якості класифікації моделі випадкового лісу на датасеті Credit Card Fraud Detection

Транзакційний компонент у цьому датасеті може спиратися переважно на Amount і частково на Time. Це справді транзакційні ознаки, але вони не описують повну поведінку клієнта, канал платежу, пристрій, отримувача чи історію рахунку. Тому транзакційний компонент отримав ненульову, але дуже малу вагу. Це означає, що сума операції й час можуть допомагати в ранжуванні ризику, але самі по собі не є достатніми для надійного визначення шахрайства.

Поведінковий, контекстний і мережевий компоненти ризику у межах першого експерименту мали меншу вагу, оскільки їхня інформативність залежить від наявності додаткових даних про клієнта, пристрій, канал платежу,

географічну поведінку та зв’язки між отримувачами коштів. У тестовому наборі даних ці характеристики представлені обмежено, тому вони виконують допоміжну роль і використовуються переважно для уточнення підсумкового ризикового бала. Такий результат є логічним, оскільки в реальних банківських системах поведінкові та мережеві ознаки мають вищу цінність за умови накопичення історії операцій клієнта та побудови профілю його звичайної активності.

Для порівняльної оцінки було виконано серію запусків із різними класифікаторами на датасеті [39]. У всіх випадках використовувалися однакові базові параметри: частка тестової вибірки становила 0,30, поріг рішення – 0,50, значення random_state – 42. Узагальнені результати комп’ютерних експериментів на наборі даних подано в таблиці 3.7.

Таблиця 3.7 – Порівняння результатів класифікації транзакцій на датасеті Credit Card Fraud Detection

Модель	Точність	Влучність	Повнота	F1-міра	ROC-AUC
Random Forest	0,979	0,067	0,878	0,125	0,968
Logistic Regression	0,998	0,511	0,818	0,629	0,972
Gradient Boosting	0,999	0,766	0,797	0,781	0,946
Hist Gradient Boosting	0,979	0,067	0,878	0,125	0,968

Модель Logistic Regression досягла найвищого значення повноти – 0,878, тобто виявила найбільшу частку реальних шахрайських операцій серед розглянутих моделей. Водночас влучність становить лише 0,067, що свідчить про значну кількість хибних спрацювань. За матрицею помилок модель помилково віднесла 1800 легітимних транзакцій до шахрайських, тому її доцільно розглядати як базову модель для порівняння, а не як оптимальне практичне рішення.

Порівняно з цим, модель Random Forest забезпечила більш збалансоване співвідношення між влучністю та повнотою: 0,511 та 0,818 відповідно, а F1-міра набула значення 0,629. Це означає, що модель зберігає достатньо високий рівень

виявлення шахрайських операцій і водночас значно зменшує кількість хибних тривог.

Результати моделі Gradient Boosting показали найкращий загальний баланс метрик: точність – 0,999, влучність – 0,766, повнота – 0,797, F1-міра – 0,781. За матрицею помилок модель правильно класифікувала 85259 легітимних транзакцій і 118 шахрайських транзакцій. При цьому лише 36 легітимних транзакцій були помилково визначені як шахрайські, а 30 шахрайських операцій не були виявлені моделлю.

Отже, проведені комп’ютерні експерименти підтвердили працездатність розробленого програмного додатка. Система виконує завантаження транзакційного файлу, профілювання даних, вибір класифікатора, навчання моделі, оцінювання якості, побудову ROC-кривої, формування прогнозів, розрахунок підсумкового ризикового бала та експорт результатів. Отримані результати показують, що для задач виявлення фінансового шахрайства недостатньо орієнтуватися лише на метрику точності. Більш показовими є решта метрик, оскільки саме вони відображають здатність моделі працювати з рідкісним шахрайським класом.

ВИСНОВКИ

У кваліфікаційній роботі було розглянуто проблему автоматизованого виявлення фінансового шахрайства у цифрових платіжних системах із використанням сучасних методів машинного навчання. За результатами аналізу визначено основні види шахрайських операцій та встановлено, що шахрайство не можна описати лише одним універсальним типом ознак, оскільки воно проявляється на різних рівнях: транзакційному, поведінковому, контекстному та мережевому. Тому для ефективного виявлення підозрілих операцій необхідно враховувати не тільки суму або час платежу, а й поведінкову історію користувача, канал здійснення операції, зв’язки між рахунками, повторюваність дій та інші непрямі ознаки ризику. Також у межах теоретичного аналізу було розглянуто еволюцію методів виявлення шахрайства та показано, що сучасні системи виявлення шахрайства найчастіше мають гібридний характер.

У межах проєктування було запропоновано архітектуру програмного комплексу, яка відокремлює рівень інтерфейсу користувача, логіку обробки даних, модулі машинного навчання, підсистему класифікації, розрахунок багатокомпонентного ризику, експорт результатів і журналювання. У процесі реалізації було створено програмний конвеєр, який охоплює основні етапи роботи з даними: зчитування вхідного набору, перевірку наявності необхідних колонок, поділ даних на навчальну та тестову вибірки, підготовку ознак, навчання моделі, отримання прогнозів, розрахунок ймовірності шахрайства, формування ризикового балу та збереження результатів. Окремо реалізовано підхід до багатокомпонентного оцінювання ризику, що краще відповідає природі фінансового шахрайства, оскільки реальна підозрілість операції часто формується не одним фактором, а комбінацією кількох ознак.

Перевірка охоплювала як окремі модулі, так і повний шлях обробки даних. Було передбачено тестування завантаження та валідації CSV-файлу, перевірку підготовки ознак, тестування розрахунку ризикового балу, перевірку коректності порогового рішення, контроль формування метрик якості та перевірку експорту результатів. У межах комп’ютерних експериментів було оцінено можливість

застосування моделей машинного навчання для класифікації транзакцій на легітимні та шахрайські. Результати експериментального дослідження показали, що використання машинного навчання є доцільним для задачі автоматизованого виявлення фінансового шахрайства. Моделі, побудовані на основі ансамблевих алгоритмів, є придатними для роботи зі структурованими транзакційними даними, оскільки вони здатні враховувати нелінійні залежності між ознаками та формувати ймовірнісну оцінку ризику.

Разом із тим, розроблене рішення має певні обмеження, які необхідно враховувати. По-перше, якість роботи моделі значною мірою залежить від повноти та репрезентативності вхідних даних. Якщо набір даних містить обмежену кількість ознак, система не може повною мірою оцінити поведінкові, контекстні або мережеві ризики. По-друге, моделі машинного навчання потребують періодичного оновлення, оскільки поведінка користувачів і схеми шахрайства можуть змінюватися з часом. По-третє, для впровадження в реальну банківську інфраструктуру необхідно додатково враховувати вимоги до захисту персональних даних, аудиту рішень, інтеграції з авторизаційними системами та роботи в режимі реального часу.

Подальший розвиток роботи може бути спрямований на розширення набору ознак, використання потокової обробки транзакцій, інтеграцію з базою даних, реалізацію API для зовнішніх систем, додавання механізмів моніторингу якості моделі та виявлення дрейфу даних. Також перспективним напрямом є впровадження пояснюваності результатів класифікації, наприклад через відображення факторів, які найбільше вплинули на підсумковий ризиковий бал. Це дозволить зробити систему більш прозорою для аналітиків і підвищити довіру до автоматизованих рішень.

Отже, у ході виконання кваліфікаційної роботи було досягнуто поставленої мети. Отримані результати демонструють можливість подальшого розвитку розробленого прототипу до рівня повноцінної системи підтримки прийняття рішень у сфері банківської безпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bolton R. J., Hand D. J. Statistical Fraud Detection: A Review. *Statistical Science*. 2002. Vol. 17, No. 3. P. 235-255. DOI: 10.1214/ss/1042727940.
2. Ngai E. W. T., Hu Y., Wong Y. H., Chen Y., Sun X. The application of data mining techniques in financial fraud detection: a classification framework and an academic review of literature. *Decision Support Systems*. 2011. Vol. 50, No. 3. P. 559-569. DOI: 10.1016/j.dss.2010.08.006.
3. West J., Bhattacharya M. Intelligent financial fraud detection: a comprehensive review. *Computers & Security*. 2016. Vol. 57. P. 47-66. DOI: 10.1016/j.cose.2015.09.005.
4. Abdallah A., Maarof M. A., Zainal A. Fraud detection system: a survey. *Journal of Network and Computer Applications*. 2016. Vol. 68. P. 90-113. DOI: 10.1016/j.jnca.2016.04.007.
5. Enterprise Fraud Management Platform: вебсайт. NICE Actimize. URL: https://www.niceactimize.com/fraud-management (дата звернення: 06.06.2026).
6. ARIC Risk Hub: Financial Crime & Fraud Software: вебсайт. Featurespace. URL: https://www.featurespace.com/aric-risk-hub (дата звернення: 06.06.2026).
7. European Banking Authority; European Central Bank. 2025 Report on Payment Fraud. 2025. URL: https://www.eba.europa.eu/sites/default/files/2025-12/1709846a-84d9-47cf-86a0-b155efb34d66/EBA%20and%20ECB%20Report%20on%20Payment%20Fraud.pdf (дата звернення: 06.06.2026).
8. European Payments Council. 2024 Payment Threats and Fraud Trends Report. EPC162-24. Version 1.0. 2024. URL: https://www.europeanpaymentscouncil.eu/sites/default/files/kb/file/2024-12/EPC162-24%20v1.0%202024%20Payments%20Threats%20and%20Fraud%20Trends%20Report_0.pdf (дата звернення: 06.06.2026).
9. Hilal W., Gadsden S. A., Yawney J. Financial Fraud: A Review of Anomaly Detection Techniques and Recent Advances. *Expert Systems with Applications*.

2022. Vol. 193. Article 116429. DOI: 10.1016/j.eswa.2021.116429. URL: https://doi.org/10.1016/j.eswa.2021.116429 (дата звернення: 06.06.2026).

10. Akartuna E. A., Johnson S. D., Thornton A. Motivating a standardised approach to financial intelligence: a typological scoping review of money laundering methods and trends. *Journal of Experimental Criminology*. 2024. DOI: 10.1007/s11292-024-09623-y.
11. Financial Crimes Enforcement Network. Frequently Asked Questions Regarding the FinCEN Suspicious Activity Report (SAR): вебсайт. URL: https://www.fincen.gov/resources/frequently-asked-questions-regarding-fincen-suspicious-activity-report-sar (дата звернення: 06.06.2026).
12. Hayashi F. New Data on Card-Present and Card-Not-Present Fraud Rates in the United States. Payments System Research Briefing. Federal Reserve Bank of Kansas City. 2026. URL: https://www.kansascityfed.org/research/payments-system-research-briefings/new-data-on-card-present-and-card-not-present-fraud-rates-in-the-united-states/ (дата звернення: 06.06.2026).
13. Office of the Comptroller of the Currency. Cybersecurity and Financial System Resilience Report. 2025. URL: https://www.occ.gov/publications-and-resources/publications/cybersecurity-and-financial-system-resilience/files/pub-2025-cybersecurity-report.pdf (дата звернення: 06.06.2026).
14. Chen T. et al. Recent Progress on Financial Risk Detection in the Context of Transaction Fraud Based on Machine Learning Algorithms. *Journal of Risk and Financial Management*. 2026. Vol. 19, No. 1. Article 14. DOI: 10.3390/jrfm19010014.
15. Hernandez Aros L. et al. Financial fraud detection through the application of machine learning techniques: a literature review. *Humanities and Social Sciences Communications*. 2024. Vol. 11. Article 1130. DOI: 10.1057/s41599-024-03606-0.
16. Compagnino A. A. et al. An Introduction to Machine Learning Methods for Fraud Detection. Applied Sciences. 2025. Vol. 15, No. 21. Article 11787. DOI: 10.3390/app152111787.

17. Ali A. et al. Financial Fraud Detection Based on Machine Learning: A Systematic Literature Review. Applied Sciences. 2022. Vol. 12, No. 19. Article 9637. DOI: 10.3390/app12199637.
18. Chen Y., Zhao C., Xu Y., Nie C., Zhang Y. Deep Learning in Financial Fraud Detection: Innovations, Challenges, and Applications. *Data Science and Management*. 2025. DOI: 10.1016/j.dsm.2025.08.002.
19. Sun W. et al. Objective over Architecture: Fraud Detection Under Extreme Imbalance in Bank Account Opening. *Computation*. 2025. Vol. 13, No. 12. Article 290. DOI: 10.3390/computation13120290.
20. Cheng D., Zou Y., Xiang S., Jiang C. Graph Neural Networks for Financial Fraud Detection: A Review. *Frontiers of Computer Science*. 2025. Vol. 19, No. 9. DOI: 10.1007/s11704-024-40474-y.
21. Saldana-Ulloa D., De Ita Luna G., Marcial-Romero J. R. A Temporal Graph Network Algorithm for Detecting Fraudulent Transactions on Online Payment Platforms. *Algorithms*. 2024. Vol. 17, No. 12. Article 552. DOI: 10.3390/a17120552.
22. Zhu H., Wang C., Chai S. Detecting Evolving Fraudulent Behavior in Online Payment Services: Open-Category and Concept-Drift. *IEEE Transactions on Services Computing*. 2024. Vol. 17, No. 5. P. 2180-2193. DOI: 10.1109/TSC.2024.3422880.
23. Hong X., Zheng C., Zilberman N. In-Network Machine Learning for Real-Time Transaction Fraud Detection. *ECAI 2024*. P. 2902-2909. DOI: 10.3233/FAIA240828.
24. Daksa R. P., Kemala A. P. A Comparative Study on Real Time Data Streaming for Fraud Detection Using Kafka with Apache Flink and Apache Spark. *Procedia Computer Science*. 2025. Vol. 269. P. 192-199. DOI: 10.1016/j.procs.2025.08.272.
25. Aljunaid S. K. et al. Secure and Transparent Banking: Explainable AI-Driven Federated Learning Model for Financial Fraud Detection. *Journal of Risk and*

Financial Management. 2025. Vol. 18, No. 4. Article 179. DOI: 10.3390/jrfm18040179.

26. RiskOps Platform: End-to-End Risk Management: вебсайт. Feedzai. URL: https://www.feedzai.com/riskops/ (дата звернення: 06.06.2026).
27. SAS Fraud Management: вебсайт. SAS Institute Inc. URL: https://www.sas.com/en_us/software/fraud-management.html (дата звернення: 06.06.2026).
28. SAS Fraud Management Features: вебсайт. SAS Institute Inc. URL: https://www.sas.com/pl_pl/software/fraud-management/features-list.html (дата звернення: 06.06.2026).
29. FICO Falcon Fraud Manager: вебсайт. FICO. URL: https://www.fico.com/en/products/fico-falcon-fraud-manager (дата звернення: 06.06.2026).
30. Latest FICO Fraud Model Helps Identify 50% More Scam Transactions: прес-реліз. FICO. 2021. URL: https://investors.fico.com/news-releases/news-release-details/latest-fico-fraud-model-helps-identify-50-more-scam-transactions (дата звернення: 06.06.2026).
31. FICO Falcon Fraud Manager: довідкова сторінка. Nexi / SIA. URL: https://fraud.sia.eu/FalconRmaHelp/GUID-D76AAA59-0D7C-4DE8-ABAC-F10B721F20D9.html (дата звернення: 06.06.2026).
32. FICO Falcon Fraud Manager 6. Product Sheet. FICO. URL: https://5.imimg.com/data5/SELLER/Doc/2024/1/379924136/VT/GV/RV/12952512/fico-falcon-fraud-manager-data-protection-software.pdf (дата звернення: 06.06.2026).
33. Apache Kafka Documentation: вебсайт. Apache Software Foundation. URL: https://kafka.apache.org/documentation/ (дата звернення: 06.06.2026).
34. Feast: the Open Source Feature Store: документація. Feast. URL: https://docs.feast.dev/ (дата звернення: 06.06.2026).
35. What is a feature store? IBM Think: вебсайт. IBM. URL: https://www.ibm.com/think/topics/feature-store (дата звернення: 06.06.2026).

36. SAS Fraud Decisioning: вебсайт. SAS Institute Inc. URL: https://www.sas.com/nl_nl/software/fraud-decisioning.html (дата звернення: 06.06.2026).
37. pytest documentation: вебсайт. pytest. URL: https://docs.pytest.org/en/stable/ (дата звернення: 06.06.2026).
38. Model evaluation: quantifying the quality of predictions: документація. scikit-learn. URL: https://scikit-learn.org/stable/modules/model_evaluation.html (дата звернення: 06.06.2026).
39. Credit Card Fraud Detection: набір даних. Kaggle. URL: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud (дата звернення: 06.06.2026).
40. IEEE-CIS Fraud Detection: набір даних. Kaggle. URL: https://www.kaggle.com/competitions/ieee-fraud-detection/data (дата звернення: 06.06.2026).
41. Lopez-Rojas E. PaySim: A financial mobile money simulator for fraud detection: репозиторій. GitHub. URL: https://github.com/EdgarLopezPhD/PaySim (дата звернення: 06.06.2026).

ДОДАТКИ

Додаток А – Посилання на репозиторій

Супровідний код програмного продукту, розробленого в межах цієї кваліфікаційної роботи, розташовано за адресою https://github.com/Yegor4ikplay228/fraud-detection-app-final.