НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

До захисту допущено:
Завідувач кафедри
_____ Оксана ТИМОЩУК
«___»_____2026 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системний аналіз і управління»

спеціальності 124 «Системний аналіз»

на тему: «Інтелектуальна система оцінки якості та автоматизованої актуалізації моделі для виявлення шахрайських платіжних транзакцій у реальному часі»

Виконав:
Студент IV курсу, групи КА-21
Гончарук Максим Ілліч _____

Керівник:
Асистент каф. ММСА
Древаль Максим Михайлович _____

Консультант з економічного розділу:
Доцент кафедри ЕК, к.е.н.
Рощина Надія Василівна _____

Консультант з нормоконтролю:
Асистент каф. ММСА, к.т.н.
Канцедал Георгій Олегович _____

Рецензент:
Доцент каф. СП, к.т.н.
Безносик Олександр Юрійович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2026 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 124 «Системний аналіз»
Освітня програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Оксана ТИМОЩУК
«___»_____2026 р.

ЗАВДАННЯ
на дипломну роботу студенту
Гончаруку Максиму Іллічу

1. Тема роботи «Інтелектуальна система оцінки якості та автоматизованої актуалізації моделі для виявлення шахрайських платіжних транзакцій у реальному часі», керівник роботи Древаль Максим Михайлович, асистент кафедри ММСА, затверджені наказом по університету від «27» травня 2026 р. № 1944-с.

2. Термін подання студентом роботи «10» червня 2026 р.

3. Вихідні дані до роботи: набір синтетичних платіжних транзакцій PaySim обсягом 6 362 620 записів за 743 часові кроки, наукові публікації з методів виявлення дрейфу даних і моніторингу моделей машинного навчання, експертні оцінки вагомості техніко-економічних параметрів програмного продукту.

4. Зміст роботи: 1. Аналіз предметної області та теоретичні основи. 2. Постановка задачі моніторингу та автоматизованої актуалізації моделі. 3. Практична реалізація системи. 4. Функціонально-вартісний аналіз програмного продукту.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): схема архітектури системи моніторингу та актуалізації моделі, графіки якості моделі на сценаріях штучного дрейфу, графіки бальної оцінки техніко-економічних параметрів, таблиці економічного розрахунку, презентація захисту.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Функціонально-вартісний аналіз	Рощина Н.В., доц., к.е.н.		

7. Дата видачі завдання:

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Аналіз предметної області та формулювання задачі	03.02.2026 – 28.02.2026	Виконано
2.	Огляд літератури та збір вхідних даних	01.03.2026 – 21.03.2026	Виконано
3.	Теоретичні основи моніторингу та постановка задачі актуалізації моделі	22.03.2026 – 12.04.2026	Виконано
4.	Розробка архітектури системи та реалізація сервісів прогнозування і моніторингу	13.04.2026 – 03.05.2026	Виконано
5.	Реалізація аналітичного модуля і сценаріїв експериментальної перевірки	04.05.2026 – 18.05.2026	Виконано
6.	Функціонально-вартісний аналіз програмного продукту	19.05.2026 – 28.05.2026	Виконано
7.	Оформлення пояснювальної записки та презентації	29.05.2026 – 09.06.2026	Виконано
8.	Попередній захист та захист дипломної роботи	10.06.2026 – 22.06.2026	Виконано

Студент _____ Максим ГОНЧАРУК

Керівник _____ Максим ДРЕВАЛЬ

РЕФЕРАТ

Дипломна робота: 98 с., 13 табл., 14 рис., 2 додатки, 22 джерела.

ВИЯВЛЕННЯ ШАХРАЙСТВА, ПОТОКОВІ ПЛАТЕЖІ, ДРЕЙФ ДАНИХ, КОНЦЕПЦІЙНИЙ ДРЕЙФ, МОНІТОРИНГ МОДЕЛІ, АВТОМАТИЗОВАНА АКТУАЛІЗАЦІЯ, PR-AUC, LIGHTGBM.

Об’єкт дослідження – процес автоматичного виявлення шахрайських платіжних транзакцій у режимі потоку даних.

Предмет дослідження – методи моніторингу якості та автоматичного оновлення моделей класифікації в умовах зміни характеристик вхідних даних і відкладеного отримання підтвердженої інформації про шахрайські транзакції.

Мета роботи – розробити інтелектуальну програмну систему, що класифікує платіжні транзакції в реальному часі, відстежує зміну якості моделі та запускає її перенавчання у разі стабільного погіршення показників.

Актуальність роботи зумовлена зростанням обсягу безготівкових платежів і ускладненням шахрайських схем, через що модель виявлення підозрілих транзакцій може втрачати точність на нових даних. Автоматичний моніторинг дає змогу швидше виявляти таке погіршення, ніж ручна перевірка.

Результати роботи – розроблено програмну систему з трьох основних компонентів: сервісу прогнозування на основі LightGBM, модуля моніторингу стану моделі та аналітичного модуля, який приймає рішення про перенавчання. На сценаріях штучної зміни даних PaySim система виявила погіршення якості, виконала перенавчання та порівняла нову модель з поточною за критерієм PR-AUC.

ABSTRACT

Diploma thesis: 98 p., 13 tables, 14 figures, 2 appendices, 22 references.

FRAUD DETECTION, STREAMING PAYMENTS, DATA DRIFT, CONCEPT DRIFT, MODEL MONITORING, AUTOMATED MODEL UPDATING, PR-AUC, LIGHTGBM.

Object of research – the process of automatic detection of fraudulent payment transactions in a data stream mode.

Subject of research – methods for monitoring the quality and automatically updating classification models under conditions of changing input data characteristics and delayed receipt of confirmed information about fraudulent transactions.

The aim of the work – to develop an intelligent software system that classifies payment transactions in real time, tracks changes in model quality, and triggers retraining in case of a stable deterioration in performance indicators.

The relevance of the work is determined by the growth in the volume of cashless payments and the increasing complexity of fraudulent schemes, which may cause the model for detecting suspicious transactions to lose accuracy on new data. Automated monitoring makes it possible to detect such deterioration faster than manual review.

Results of the work – a software system consisting of three main components was developed: a prediction service based on LightGBM, a model state monitoring module, and an analytical module that makes decisions on retraining. In scenarios with artificial changes in PaySim data, the system detected quality deterioration, performed retraining, and compared the new model with the current one using the PR-AUC criterion.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ADWIN – Adaptive Windowing.

DDM – Drift Detection Method, метод виявлення дрейфу за помилками моделі.

F1 – F1-міра, середнє гармонічне precision та recall.

JSON – JavaScript Object Notation, текстовий формат обміну даними.

KS – тест Колмогорова–Смірнова (Kolmogorov–Smirnov test).

LightGBM – Light Gradient Boosting Machine.

ML – Machine Learning, машинне навчання.

PaySim – синтетичний набір даних платіжних транзакцій, використаний у роботі.

PR-AUC – площа під precision-recall кривою (Area Under the Precision-Recall Curve).

PSI – Population Stability Index, індекс стабільності популяції.

ROC – Receiver Operating Characteristic.

ROC-AUC – площа під ROC-кривою (Area Under the ROC Curve).

ВСТУП

Сучасна платіжна інфраструктура є важливою складовою цифрової економіки, оскільки забезпечує оброблення великих обсягів безготівкових та онлайн-платежів у реальному часі. За даними Європейського центрального банку, у першому півріччі 2025 року в єврозоні було виконано 77,7 млрд безготівкових платежів на суму 116,0 трлн євро [1]. За таких масштабів своєчасне виявлення підозрілих операцій має істотне практичне значення.

Однією з найбільш суттєвих проблем у цій сфері є шахрайські платіжні транзакції, які спричиняють прямі фінансові втрати, додаткові операційні витрати та погіршення користувацького досвіду. Для зниження таких ризиків у платіжних системах застосовуються автоматизовані антифрод-рішення та моделі машинного навчання, що оцінюють ризик операції в реальному часі за сукупністю транзакційних, поведінкових і технічних ознак [2].

Разом з тим ефективність таких моделей не є сталою. У потоці транзакцій змінюються поведінка користувачів, сценарії шахрайства, статистичні властивості даних і структура ознак, а підтверджений результат операції часто надходить із затримкою. Тому виникає потреба не лише у побудові моделі класифікації, а й у моніторингу її якості, виявленні зміни розподілу даних і своєчасній актуалізації моделі.

Метою роботи є побудова підходу до створення інтелектуальної системи оцінки якості та автоматизованої актуалізації моделі для виявлення шахрайських платіжних транзакцій у реальному часі. Для досягнення цієї мети необхідно розглянути підходи до оцінювання якості моделей на незбалансованих даних, описати часову природу транзакційного потоку, визначити критерії погіршення якості та правила запуску оновлення моделі.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ТЕОРЕТИЧНІ ОСНОВИ

1.1 Аналіз предметної області онлайн-платежів і ризиків шахрайства

Предметна область онлайн-платежів охоплює послідовність операцій, у яких беруть участь клієнт, платіжний сервіс, банк-емітент, мерчант та внутрішня система ризик-контролю. У момент авторизації кожна транзакція супроводжується набором ознак, що характеризують суму, час, тип операції, канал оплати, поведінку користувача та технічний контекст виконання платежу. Саме на основі цих ознак система повинна оперативно оцінити ризик операції ще до отримання остаточного результату [2].

Платіжна інфраструктура має багаторівневу архітектуру, в якій кожна транзакція проходить через ланцюг учасників. Платіжний сервіс-провайдер (PSP) приймає запит від мерчанта і передає його банку-еквaєру, який маршрутизує транзакцію через карткову схему (Visa, Mastercard та інші) до банку-емітента, що ухвалює остаточне рішення про авторизацію. На більшості європейських платіжних потоків додатково застосовується протокол 3-D Secure 2, який передбачає сильну автентифікацію клієнта і поділяє транзакції на ті, що проходять frictionless flow, і ті, що потребують додаткового кроку підтвердження [3]. Така багатоланкова структура породжує характерні точки контролю та водночас створює технічні обмеження: фрод-аналіз має виконуватися в межах коротких таймаутів, заданих картковою схемою або емітентом, і часто паралельно іншим перевіркам, що ускладнює інтеграцію складних моделей.

Узагальнена схема ланцюга авторизації платіжної транзакції з виокремленням ролі сильної автентифікації клієнта наведена на рисунку 1.1.

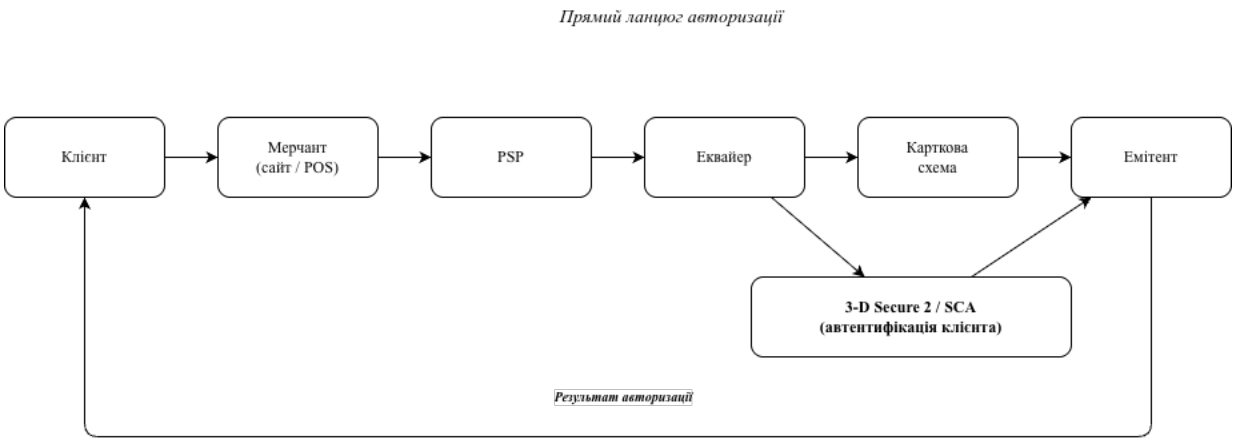


Рисунок 1.1 – Ланцюг авторизації платіжної транзакції з гілкою 3-D Secure 2

Шахрайські транзакції в такому середовищі можуть набувати різних форм, зокрема використання викрадених реквізитів, підміни поведінки звичайного клієнта, серійних дрібних операцій або нетипових сплесків активності. Через це задача захисту не зводиться до перевірки однієї ознаки чи одного правила. Антифрод-система має враховувати поєднання кількох характеристик транзакції та їхній зв'язок із попередньою поведінкою користувача й поточним станом потоку операцій.

Ключовою особливістю предметної області є поєднання жорстких часових обмежень і неповноти інформації. Рішення щодо пропуску, блокування або додаткової перевірки транзакції приймається одразу, тоді як підтверджений результат операції часто стає відомим пізніше. Крім того, шахрайські сценарії не є сталими: змінюються звички користувачів, канали оплати, шаблони поведінки зловмисників і статистичні властивості потоку. Саме це зумовлює потребу не лише у виявленні підозрілих операцій, а й у подальшому контролі актуальності моделі оцінювання ризику [4].

1.2 Задача класифікації транзакцій і підходи до її розв'язання

У формальному плані виявлення шахрайства розглядається як задача бінарної класифікації, у межах якої для кожної транзакції необхідно оцінити

ймовірність її належності до шахрайського класу. У науковій літературі для цього застосовують як контрольовані, так і неконтрольовані підходи. Супервізовані методи спираються на розмічені приклади шахрайських та легітимних операцій, тоді як несупервізовані підходи орієнтуються на пошук аномальних відхилень від звичайної поведінки [5].

До найбільш поширених супервізованих моделей належать логістична регресія, дерева рішень, випадкові ліси, градієнтний бустинг, нейромережеві та ансамблеві алгоритми. Вибір конкретного методу визначається не лише прогнозною якістю, а й вимогами прикладної системи, зокрема інтерпретованістю, швидкістю інференсу, стійкістю до шуму, можливістю працювати зі змішаними табличними ознаками та зручністю подальшого оновлення. Для транзакційних даних, де важливими є табличні характеристики та обмеження за латентністю, практично доцільними часто виявляються дерева рішень та ансамблі на їх основі.

Серед супервізованих моделей особливе місце займає градієнтний бустинг над деревами рішень (Gradient Boosted Decision Trees, GBDT). Реалізації на кшталт XGBoost [6] та LightGBM [7] стали де-факто стандартом для табличних задач завдяки поєднанню високої прогнозної якості, ефективної роботи з пропусками й категоріальними ознаками, а також помірних вимог до обчислювальних ресурсів. У задачах антифроду додатковою перевагою є наявність вбудованих механізмів балансування класів, зокрема параметра scale_pos_weight, що дозволяє моделі коректно реагувати на сильну нерівномірність розподілу шахрайських транзакцій без штучного перевибірковування. Саме градієнтний бустинг обрано як базову модель у практичній частині цієї роботи.

У межах несупервізованих підходів широко застосовуються алгоритми виявлення аномалій. Isolation Forest будує ансамбль випадкових дерев, що ізолюють спостереження, і присвоює високу оцінку аномальності тим точкам, які потрапляють на короткі шляхи в дереві [8]. До інших поширених методів

належить One-Class SVM, що навчається на еталонному «нормальному» розподілі та позначає відхилення, а також автоенкодери, які використовують похибку реконструкції вхідного вектора як міру нетиповості. На практиці супервізовані та несупервізовані підходи нерідко поєднують у гібридних схемах, у яких несупервізована модель формує додаткові ознаки або фільтрує потік перед супервізованим класифікатором, що дозволяє реагувати і на відомі схеми шахрайства, і на нові, ще невідомі патерни.

Разом з тим у платіжному домені задача класифікації не може обмежуватися одноразовим навчанням моделі. Навіть за високих початкових показників якості модель з часом може втрачати придатність до роботи в реальному потоці. Тому класифікаційний підхід доцільно розглядати як частину ширшої системи, у якій поряд із прогнозуванням реалізовано моніторинг якості та механізм актуалізації моделі [4].

1.3 Оцінювання якості моделі: метрики та інтерпретованість

Характерною рисою задачі виявлення шахрайства є сильна незбалансованість класів: кількість шахрайських операцій зазвичай становить лише малу частку від усіх транзакцій. За таких умов традиційна метрика accuracy може створювати хибне уявлення про якість моделі, оскільки високий результат досягається навіть тоді, коли система майже завжди прогнозує лише легітимні операції [9].

Тому в задачах fraud detection більш інформативними є метрики, чутливі до позитивного класу, зокрема precision, recall, F1-міра, ROC-AUC та PR-AUC. Precision відображає частку дійсно шахрайських операцій серед усіх транзакцій, які система позначила як підозрілі, тоді як recall характеризує частку шахрайських операцій, які вдалося виявити. F1-міра дозволяє узгоджено враховувати ці два аспекти, якщо важливими є і точність спрацювань, і повнота виявлення.

Окрему увагу доцільно приділяти PR-AUC та ROC-AUC. Згідно з дослідженням T. Saito та M. Rehmsmeier, для сильно незбалансованих даних precision-recall представлення часто є більш інформативним, ніж ROC-простір, оскільки воно краще відображає якість виявлення рідкісного позитивного класу [9]. Водночас у прикладних антифрод-системах слід контролювати не лише класичні метрики бінарної класифікації, а й операційні показники, пов'язані з очікуваними втратами, часткою хибних спрацювань та навантаженням на ручну перевірку.

Окрім числових метрик якості, до моделей у платіжному ризик-контролі застосовуються вимоги щодо прозорості ухвалених рішень. У європейському регуляторному полі стаття 22 Загального регламенту захисту даних (GDPR) гарантує особі право не бути об'єктом виключно автоматизованого рішення, що має для неї істотні наслідки, без можливості отримати пояснення [10]. Подібні очікування зафіксовані й у PSD2 у частині, що стосується обґрунтованої відмови в авторизації платежу [3]. На операційному рівні пояснюваність потрібна аналітикам ризик-команди для перевірки складних випадків, а також відділам клієнтської підтримки для розгляду оскаржень з боку клієнтів. Тому навіть за технічно високих метрик якості непрозора модель може виявитися непридатною до експлуатації.

Підходи до забезпечення інтерпретованості умовно поділяються на два класи. Перший охоплює моделі, що є зрозумілими за побудовою, наприклад логістична регресія, неглибокі дерева рішень або градієнтний бустинг з монотонними обмеженнями на ознаки. Такі моделі дають внутрішнє пояснення у вигляді коефіцієнтів, правил або напрямів впливу окремих ознак, проте часто поступаються складнішим алгоритмам за точністю. Другий клас становлять post-hoc методи, що додаються над уже навченою моделлю довільної архітектури і пояснюють кожен її прогноз окремо, не змінюючи саму модель.

Серед post-hoc підходів найбільш поширені два сімейства. SHAP ґрунтується на теорії кооперативних ігор і розраховує внесок кожної ознаки в окремий прогноз з гарантією властивостей адитивності та узгодженості [11]. LIME локально апроксимує поведінку складної моделі простою інтерпретованою функцією в околі обраної точки [12]. У фінансовому секторі такі методи дозволяють поєднати високу прогнозну якість сучасних ансамблевих або нейронних моделей з вимогою регулятора пояснити окреме рішення, що підтверджується дослідженнями у задачах кредитного ризику та антифроду.

У межах цієї роботи вимога інтерпретованості значною мірою врахована на рівні модуля моніторингу. Інтегральний показник, який запускає актуалізацію моделі, побудований не як вихід ще однієї непрозорої моделі, а як зважена сума чотирьох названих сигналів – поточної якості, дрейфу ознак, поведінки скорів та достатності міток. Кожен сигнал має ясне змістове значення, кожна вага задана експертно з огляду на критичність відповідного аспекту, а ухвалене системою рішення можна розкласти на внески окремих сигналів. Це дозволяє і ризик-команді, і нормоконтролю простежити, чому саме систему ініційовано до перенавчання у конкретний момент часу.

1.4 Часова природа потоку і зміна розподілу даних

Платіжні транзакції надходять не як незалежна вибірка, а як часово впорядкований потік подій. На його властивості впливають сезонність, добові та тижневі цикли, поведінка окремих користувачів, маркетингові кампанії, зміни в роботі мерчантів і поява нових сценаріїв шахрайства. Через це одна й та сама модель може по-різному поводитися в різні моменти часу навіть без зміни внутрішньої архітектури.

Часова природа потоку має два важливі наслідки. По-перше, оцінювання якості моделі повинно враховувати хронологію спостережень, оскільки

випадкове перемішування транзакцій часто дає завищену оцінку узагальнювальної здатності. По-друге, істинні мітки для значної частини транзакцій надходять із затримкою, тому поточний стан моделі не завжди можна оцінити лише на основі вже підтверджених випадків [4].

Отже, в умовах потокового середовища необхідно поєднувати аналіз вхідних даних у реальному часі з відкладеним аналізом підтверджених результатів. Саме ця особливість зумовлює потребу у віконних процедурах оцінювання, відстеженні зміни розподілу даних та поступовому оновленні моделі.

Однією з головних причин деградації fraud-моделі є зміна розподілу даних у часі. У літературі це явище пов'язують із concept drift, тобто зміною статистичних закономірностей, на яких ґрунтувалося початкове навчання моделі [13]. У платіжному середовищі така зміна може бути зумовлена появою нових схем шахрайства, зміною поведінки клієнтів, сезонними ефектами або зміною структури доступних ознак.

У загальному випадку доцільно розрізняти принаймні два типи зрушень. Перший тип пов'язаний зі зміною розподілу вхідних ознак, коли нові транзакції статистично відрізняються від тих, на яких навчалася модель. Другий тип стосується зміни зв'язку між ознаками та цільовою змінною, тобто ситуації, коли навіть за схожих вхідних даних саме правило віднесення транзакції до шахрайських або легітимних змінюється [13].

За динамікою прояву concept drift також прийнято розрізняти кілька типів. Раптовий (sudden) дрейф виникає миттєво, коли поведінка системи різко змінюється, наприклад, після появи нової шахрайської кампанії або зміни правил у банку-емітенті. Поступовий (gradual) дрейф розвивається повільно, на тлі співіснування старої і нової моделей поведінки. Інкрементальний (incremental) дрейф є проміжним сценарієм, у якому зміна накопичується дрібними кроками без чітких меж між фазами. Окремо розглядають повторюваний (recurring) дрейф, типовий для сезонних явищ, коли попередні

режими поведінки повертаються через певні проміжки часу. Для практичних антифрод-систем найбільш чутливими є раптові та повторювані сценарії: перші вимагають швидкої реакції, а другі – пам'яті про попередні стани моделі та потоку [13].

Графічне представлення цих типів за динамікою зміни параметра розподілу даних у часі наведено на рисунку 1.2.

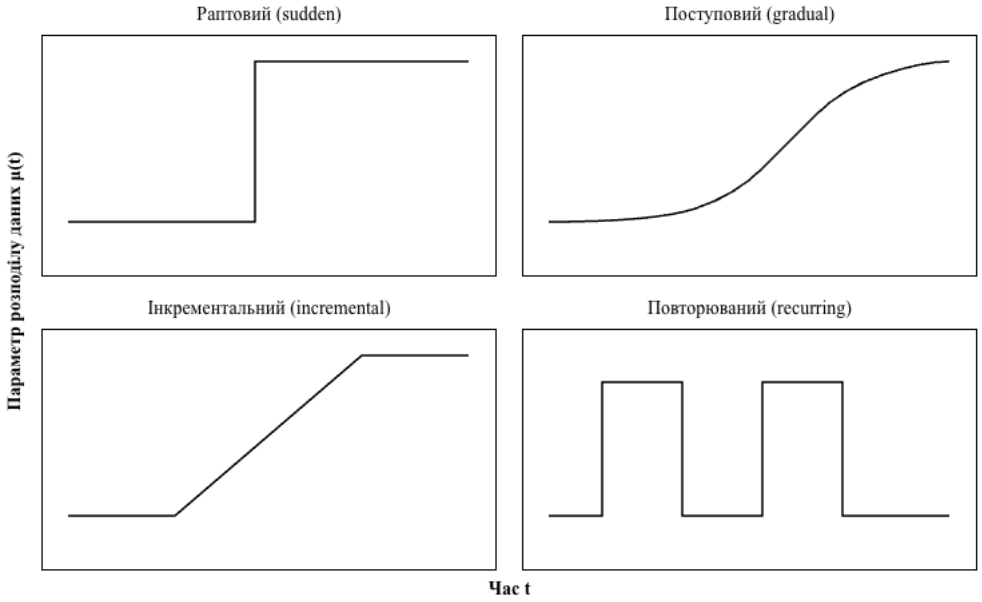


Рисунок 1.2 – Типи концептного дрейфу за динамікою (адаптовано з [13])

Для антифрод-систем concept drift є критичним, оскільки навіть невелике, але стійке зміщення даних з часом може призвести до накопичення помилок. Саме тому задача підтримання моделі в актуальному стані повинна включати не лише повторне навчання, а й раннє виявлення ознак втрати актуальності моделі [13].

1.5 Підходи до моніторингу якості моделі на потоці даних

Моніторинг якості моделі на потоці даних доцільно будувати на поєднанні двох груп сигналів. Перша група включає ознаки зміни самого потоку даних і може аналізуватися ще до появи істинних міток. До неї

належать зсув розподілу вхідних ознак, зміна статистик транзакційного потоку, а також зміна розподілу скорів, які повертає модель. Такий двоканальний підхід відповідає сучасній практиці моніторингу ML-систем [14].

Друга група сигналів базується на фактичних метриках якості, що обчислюються після надходження підтверджених міток. Такий підхід дозволяє контролювати реальну деградацію моделі, але через затримку у появі розмітки не забезпечує достатньо раннього реагування [4]. Тому в потокових fraud-системах доцільно поєднувати сигнали без міток і сигнали з мітками, розглядаючи їх як взаємодоповнювальні.

Практично це означає, що система повинна спостерігати не лише за значенням обраної метрики на останніх розмічених транзакціях, а й за поведінкою потоку в поточному часовому вікні. Саме така схема моніторингу створює підґрунтя для побудови багатосигнального механізму ухвалення рішення про актуалізацію моделі.

1.6 Огляд існуючих підходів та програмних рішень моніторингу моделей

Підтримання якості моделі в продуктивному середовищі поступово оформилося в окремий клас задач, відомий як ML observability або ML monitoring (моніторинг моделей машинного навчання у продуктивному середовищі). На відміну від класичного моніторингу інфраструктури, де предметом нагляду є стабільність сервісу, тут спостерігаються статистичні властивості вхідних даних, прогнозів та метрик якості. Робота Sculley та співавторів свого часу показала, що значна частина проблем у виробничих ML-системах виникає не в коді моделі, а в інфраструктурі її супроводу [15]. У сучасних оглядах MLOps моніторинг розглядається як невіддільна частина повного життєвого циклу моделі [14].

Серед архітектурних патернів супроводу моделі найбільшого поширення набув champion / challenger. У продуктивному режимі працює основна модель (champion), а паралельно існує одна або кілька кандидатних версій (challenger), яким подаються ті самі дані без впливу на остаточні рішення. Якщо кандидат демонструє стабільне покращення якості або кращу реакцію на дрейф, відбувається переключення активної версії. Близький до нього патерн shadow deployment передбачає роботу нової моделі «у тіні» поточної. Такий підхід особливо доречний у задачах виявлення шахрайства, де ціна помилкового рішення висока, а ризик зламати поведінку системи під час оновлення неприйнятний [14].

Серед відкритих бібліотек для ML observability важливе місце займає NannyML [16], яка дозволяє оцінювати якість моделі без негайного отримання справжніх міток. Це безпосередньо релевантно задачі моніторингу fraud-моделі, де підтверджений результат операції доступний із суттєвою затримкою. Інша поширена бібліотека Evidently AI пропонує базовий набір тестів на дрейф ознак та концептний дрейф, а також візуалізацію зміни метрик у часі [17].

З комерційних спеціалізованих платформ для платіжних потоків можна виділити Feedzai [18] та Sift [19] як приклади end-to-end рішень. Такі системи поєднують виявлення шахрайства з моніторингом якості моделі та автоматичним керуванням її версіями. Перевагою є закритий цикл і високий рівень готовності до експлуатації, обмеженням – закритість логіки прийняття рішень про оновлення, складність налаштування під конкретний платіжний контекст та вартість ліцензії, що часто робить такі платформи непридатними для прототипу.

Узагальнене порівняння розглянутих рішень за основними критеріями наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння підходів та програмних рішень для моніторингу моделей

Рішення	**Тип**	**Методи виявлення дрейфу**	**Авто-актуалізація**	**Спеціалізація**
NannyML	open-source	оцінка якості без міток (CBPE/DLE); KS, χ^2 для ознак	немає	загальна
Evidently AI	open-source	KS, PSI, Wasserstein; колонкові тести	немає	загальна
Feedzai / Sift	комерційна	вбудовано, деталі закриті	так, закритий цикл	платіжне шахрайство

Загалом готові платформи закривають базові потреби моніторингу, проте рішення про оновлення моделі приймається в них або за фіксованим пороговим правилом за однією метрикою, або як закрита логіка з невідомою комбінацією сигналів. У межах цієї роботи обрано легкий кастомний підхід, що поєднує сигнали різного типу – поточну якість, дрейф ознак, поведінку скорів та накопичення нових міток – у явному інтегральному показнику. Така архітектура зберігає прозорість критерію тригера і дозволяє експертне калібрування ваг під специфіку платіжного потоку, як буде формалізовано в розділі 2.

1.7 Критерії виявлення погіршення якості та правила запуску оновлення моделі

Рішення про оновлення fraud-моделі доцільно формалізувати, оскільки реакція лише на окремий локальний симптом, наприклад короткочасне падіння однієї метрики, може бути нестійкою. У загальному випадку критерії актуалізації можуть враховувати погіршення якості на останніх розмічених даних, силу зсуву розподілу ознак, зміну розподілу скорів моделі та накопичення достатнього обсягу нових спостережень для перенавчання.

Такий підхід дозволяє перейти від жорсткого порогового правила до більш змістовного механізму оцінювання стану системи. Інтелектуальність у цьому випадку полягає не в побудові окремої другої моделі, а в агрегуванні кількох сигналів стану, кожен із яких відображає окремий аспект придатності основної fraud-моделі до подальшої роботи.

Автоматизація таких правил є важливою частиною системи супроводу моделі, оскільки забезпечує відтворюваність процесу оновлення, зменшує залежність від ручного контролю та створює основу для подальшої реалізації програмного ядра системи моніторингу й актуалізації.

1.8 Висновки до розділу 1

У першому розділі розглянуто специфіку платіжного середовища, ризики шахрайських транзакцій та особливості побудови антифрод-систем у реальному часі. Показано, що задача виявлення шахрайства пов'язана з незбалансованістю класів, часовою природою потоку даних, затримкою в отриманні міток і зміною розподілу даних у часі. Це обґрунтовує необхідність переходу від статичної моделі класифікації до системи, яка поєднує прогнозування, моніторинг якості та формалізовані правила актуалізації моделі.

РОЗДІЛ 2 ПОСТАНОВКА ЗАДАЧІ МОНІТОРИНГУ ТА АКТУАЛІЗАЦІЇ МОДЕЛІ

2.1 Формалізація транзакційного потоку

У межах цієї роботи транзакційний потік розглядається як часово впорядкована послідовність спостережень, у якій кожна транзакція описується вектором ознак та результатом її подальшої перевірки. Така постановка відповідає реальній логіці функціонування платіжної системи, де нові операції надходять послідовно, а рішення щодо них необхідно приймати до моменту отримання остаточного підтвердження [4].

$$(x_t, y_t),\ t = 1, ..., T \tag{2.1}$$

де x_t — вектор ознак транзакції в момент часу t;

y_t — результат її перевірки після надходження підтвердженої мітки.

Основна fraud-модель для кожної транзакції формує оцінку ризику належності до шахрайського класу.

$$\hat{p}_t = P(y_t = 1 \mid x_t) \tag{2.2}$$

У такій постановці модель не лише прогнозує ризик окремої операції, а й генерує послідовність скорів, поведінка яких у часі надалі використовується в системі моніторингу. Це дозволяє поєднати задачу класифікації транзакцій із задачею контролю актуальності моделі в потоковому середовищі.

2.2 Зміна розподілу даних і затримка в отриманні міток

У потоковому fraud detection зміна поведінки даних і погіршення якості моделі не завжди спостерігаються одночасно. Частину ознак деградації можна зафіксувати ще до надходження підтверджених міток, якщо поточний потік транзакцій починає статистично відрізнятися від даних, на яких модель навчалася [13, 20].

$$P_{\text{train}}(\text{X}, \text{Y}) \neq P_{\text{t}}(\text{X}, \text{Y}) \quad (2.3)$$

$$P_{\text{train}}(\text{X}) \neq P_{\text{t}}(\text{X}) \quad (2.4)$$

Перше співвідношення відображає зміну спільного розподілу ознак і міток, а друге описує зміну розподілу самих вхідних ознак. У практичній системі другий випадок особливо важливий, оскільки його можна виявляти ще до накопичення достатнього обсягу нових розмічених транзакцій для прямої оцінки якості моделі.

Отже, система моніторингу працює в умовах часткової інформації. На частині кроків доступні лише ознаки транзакцій та скори моделі, а на частині кроків з'являється також підтверджений результат операції. Саме ця особливість зумовлює потребу у використанні кількох сигналів стану замість одного локального критерію.

2.3 Порівняльний аналіз методів виявлення дрейфу даних

Завдання виявлення дрейфу формулюється як визначення моменту, коли поточний розподіл даних або залежність між ознаками та цільовою змінною суттєво відхиляється від тих, на яких модель навчалася. У сучасній літературі описано низку методів, що різняться за тим, який тип зрушення вони фіксують, чи потребують вони підтверджених міток, і чи можуть працювати в потоковому режимі [17]. У межах цієї роботи доцільно поглянути на чотири характерні підходи, що репрезентують різні класи методів і дають змогу обґрунтувати вибір базового алгоритму для подальшої реалізації.

Перший клас становлять статистичні тести подібності розподілів. До нього належить тест Колмогорова–Смірнова (KS), який порівнює емпіричні функції розподілу еталонного та поточного вікон і не вимагає міток [20, 21]. На близьку логіку спирається індекс стабільності популяції (Population Stability Index, PSI), що обчислюється як зважена сума логарифмічних відмінностей часток у дискретизованих інтервалах. PSI набув особливого

поширення у фінансовому ризик-моделюванні, зокрема в кредитному скорингу та антифроді, де є усталеним правилом інтерпретації значень [22].

Другий клас становлять методи, орієнтовані на стрімінгове відстеження змін. Найвідоміший серед них – ADWIN (Adaptive Windowing), який підтримує адаптивне вікно змінного розміру. Воно автоматично скорочується, коли в його межах виявляється статистично значуща зміна середнього значення обраної ознаки або помилки моделі [23]. На відміну від KS і PSI, ADWIN не потребує окремого еталонного вікна, бо порівняння відбувається всередині самого ковзного вікна.

Третій клас об'єднує error-based методи, що фіксують погіршення моделі через зростання частоти помилок. Класичним представником є DDM (Drift Detection Method), який моделює кількість помилок як біноміальний процес і сигналізує про дрейф, коли поточний рівень помилок суттєво перевищує стандартне відхилення від базової оцінки [24]. Такі методи реагують на найбільш практично значущий тип зміни – власне концептний дрейф, проте вимагають своєчасних підтверджених міток, що в задачах антифроду рідко виконується [4].

Підсумкове порівняння розглянутих методів за основними характеристиками наведено в таблиці 2.1.

Таблиця 2.1 – Порівняння методів виявлення дрейфу

Метод	**Тип дрейфу**	**Потрібні мітки**	**Режим роботи**
KS (наш)	covariate	ні	пакетно / на вікні
PSI	covariate	ні	пакетно / на вікні
ADWIN	covariate / concept	опційно	стрімінг
DDM	concept	так	стрімінг

У межах цієї роботи як базовий метод обрано KS. Його перевагами для моніторингу fraud-моделі є відсутність вимоги до своєчасних міток, низька обчислювальна складність та простота інтеграції з ковзним вікном. PSI розглядається як близький альтернативний варіант для категоріальних

розрізів, а методи на кшталт DDM відкладено як компонент, що залежить від підтверджених результатів операції, які надходять із суттєвою затримкою.

2.4 Контроль якості моделі на ковзному вікні

Для оперативного аналізу стану fraud-моделі доцільно використовувати ковзне вікно останніх розмічених спостережень. Такий підхід дає можливість оцінювати поточну якість без змішування її з далеким історичним контекстом і відповідає часовій природі транзакційного потоку [4].

$$Q_t = Q((x_i, y_i), i = t - W_L + 1, ..., t) \quad (2.5)$$

де WL — розмір останнього розміченого вікна;

Qt — значення вибраної метрики якості на цьому вікні.

Як така метрика може використовуватися PR-AUC, recall для шахрайського класу або інший показник, що коректно відображає якість моделі в умовах сильного дисбалансу класів [9]. Еталонне значення Qref визначається на окремому базовому вікні, на якому модель вважається актуальною після початкового навчання. Після прийняття нової версії моделі еталонне вікно та відповідні reference-значення можуть бути переобчислені для нового стану системи.

Щоб зафіксувати саме погіршення якості, доцільно порівнювати поточне значення метрики з еталонним рівнем, отриманим на базовому вікні або на валідаційній вибірці.

$$D_{quality}(t) = \max(0, Q_{ref} - Q_t) / Q_{ref} \quad (2.6)$$

Чим більше поточна якість відхиляється від еталонної, тим більшим є внесок цього сигналу в рішення про актуалізацію моделі.

2.5 Інтегральний показник потреби в актуалізації

Інтелектуальний компонент системи полягає не лише у використанні моделі машинного навчання для оцінювання ризику шахрайства окремої

транзакції, а й в адаптивному оцінюванні стану моделі за кількома каналами: зміною якості прогнозування, зміною розподілу вхідних даних і накопиченням нових підтверджених міток. На основі цих сигналів система автоматизовано визначає потребу в актуалізації моделі, що відрізняє її від статичного підходу, за якого модель навчається один раз і надалі використовується без контролю її актуальності.

Рішення про оновлення fraud-моделі недоцільно приймати лише за одним локальним критерієм. Погіршення окремої метрики може мати тимчасовий характер, а зміна вхідних даних не завжди одразу призводить до відчутного зниження якості. Тому потребу в актуалізації доцільно визначати на основі сукупності кількох сигналів, що характеризують стан моделі та потоку даних.

Суттєві для прийняття такого рішення сигнали можна поділити на чотири групи. До першої групи належать сигнали якості моделі, які відображають зміну значення метрики на останньому розміченому вікні. До другої групи належать сигнали зміни вхідних даних, пов'язані зі зсувом розподілу ознак у поточному вікні. Третю групу становлять сигнали поведінки самої моделі, зокрема зміна розподілу скорів або частки невпевнених прогнозів. Окремо враховуються сигнали готовності до оновлення, які показують, чи накопичено достатній обсяг нових розмічених даних для змістовного перенавчання.

Для кількісного оцінювання зміни розподілу ознак одним із можливих показників може бути статистика Колмогорова–Смірнова, обчислена для кожної контрольованої ознаки на еталонному та поточному вікнах [20, 21].

$$K_j(t) = \sup_x |F_{ref,j}(x) - F_{t,j}(x)| \quad (2.7)$$

$$D_{drift}(t) = (1 / m) \Sigma_{j=1}^{m} K_j(t) \quad (2.8)$$

де m — кількість ознак, для яких контролюється статистична стійкість розподілу.

Такий сигнал відображає, наскільки поточний потік даних відрізняється від еталонного стану, на якому модель вважалася актуальною.

Для оцінювання зміни поведінки самої моделі доцільно використовувати частку невпевнених прогнозів на поточному вікні скорів. Якщо p_i є оцінкою ризику для і-ї транзакції, а інтервал [a; b] відповідає зоні невизначеності, то можна ввести показник

$$U_t = (1 / W_S) \Sigma_i I(a \le p_i \le b) \quad (2.9)$$

$$D_{conf}(t) = |U_t - U_{ref}| \quad (2.10)$$

де W_S — розмір вікна скорів;

U_{ref} — еталонна частка невпевнених прогнозів.

У практичній реалізації межі інтервалу [a; b] задаються експертно навколо зони невизначеності моделі, тобто в області значень ризику, для яких прогноз не можна вважати достатньо впевненим. Зростання цього відхилення може бути ознакою того, що модель працює в умовах, які відрізняються від тих, на яких вона була налаштована.

$$D_{data}(t) = \min(1, N_{new}(t) / N_{min}) \quad (2.11)$$

де $N_{new}(t)$ — кількість нових розмічених спостережень, доступних у момент t;

N_{min} — мінімальний обсяг, достатній для перенавчання моделі.

З огляду на це потребу в актуалізації моделі можна описати узагальненим показником

$$S_t = f(D_{quality}(t), D_{drift}(t), D_{conf}(t), D_{data}(t)) \quad (2.12)$$

У практичній реалізації така функція може бути подана у вигляді зваженої суми окремих сигналів.

$$S_t = w_1 D_{quality}(t) + w_2 D_{drift}(t) + w_3 D_{conf}(t) + w_4 D_{data}(t) \quad (2.13)$$

У цій роботі вагові коефіцієнти задаються експертно з урахуванням відносної критичності окремих сигналів для задачі підтримання актуальності fraud-моделі. При цьому вважається, що $w_i >= 0$, а сума всіх коефіцієнтів дорівнює одиниці, тобто $w_1 + w_2 + w_3 + w_4 = 1$. Найбільшу вагу доцільно надати

сигналу погіршення якості моделі, оскільки він безпосередньо відображає втрату її практичної ефективності. Дещо меншу, але також суттєву вагу має сигнал зміни розподілу вхідних даних, який може виступати ранньою ознакою майбутньої деградації. Сигнал зміни поведінки скорів і сигнал готовності нових даних до перенавчання мають допоміжний характер і враховуються з меншими вагами.

Для цієї роботи експертно прийнято такі значення: $w_1 = 0{,}40$, $w_2 = 0{,}30$, $w_3 = 0{,}15$ та $w_4 = 0{,}15$.

$$\text{Trigger}(t) = 1, \text{ якщо } S_t \geq \tau_S \tag{2.14}$$

Якщо ця умова виконується, поточний стан моделі вважається таким, що потребує актуалізації. Таким чином, рішення про оновлення приймається не за окремою локальною ознакою, а за узагальненою оцінкою стану моделі та потоку даних.

Узагальнена структура обчислення інтегрального показника та умова спрацювання тригера перенавчання зображені на рисунку 2.1.

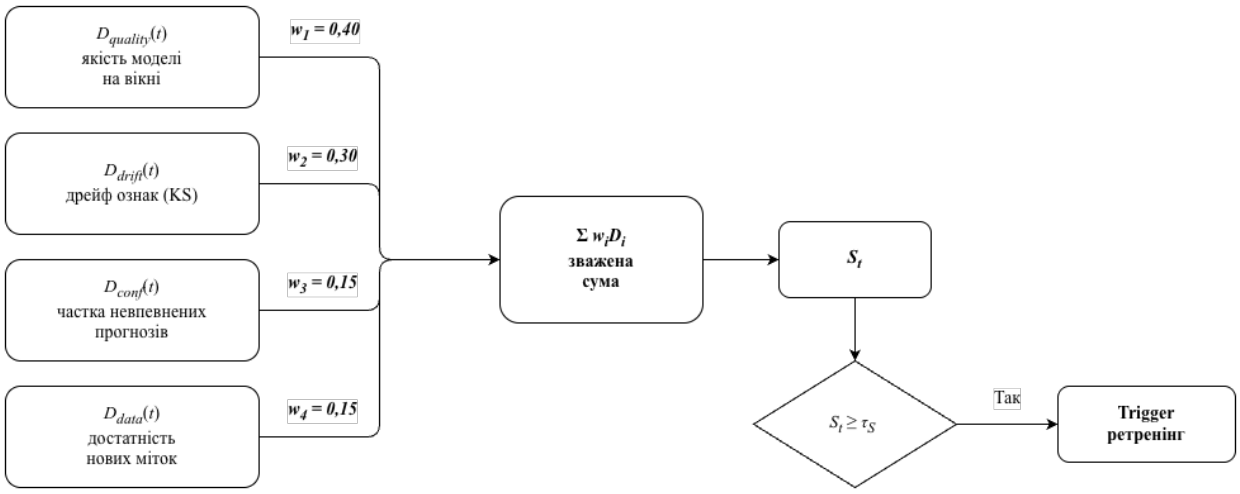


Рисунок 2.1 – Структура обчислення інтегрального показника S_t

2.6 Перенавчання моделі на актуальному часовому вікні

Якщо інтегральний показник перевищує допустимий рівень, система ініціює актуалізацію моделі. Перенавчання доцільно виконувати не на всій

історії даних, а на актуальному часовому вікні, яке краще відображає поточний стан транзакційного середовища.

$$D_t = \{(x_i, y_i) : t - W_R \leq i \leq t\} \quad (2.15)$$

$$M_t = \text{Train}(D_t) \quad (2.16)$$

де W_R — ширина вікна перенавчання;

D_t — навчальний набір, сформований із останнього вікна доступних розмічених спостережень;

M_t — оновлена версія моделі.

Після перенавчання нову версію моделі доцільно оцінити на окремому актуальному контрольному вікні, яке не використовується для самого перенавчання і формується з більш пізніх розмічених спостережень. Це дає змогу порівняти нову і попередню версії моделі в однакових умовах та уникнути змішування етапів навчання й перевірки. Якщо оновлена модель не демонструє покращення або виявляється менш стабільною, її впровадження в робочий контур не виконується.

Отже, формалізація задачі зводиться до безперервного циклу, у якому модель прогнозує ризик транзакцій, система моніторингу оцінює її стан за кількома сигналами, а механізм актуалізації запускає оновлення в разі виявлення стійкої деградації. Саме така логіка буде покладена в основу практичної реалізації системи.

2.7 Висновки до розділу 2

У другому розділі сформульовано постановку задачі моніторингу та актуалізації fraud-моделі в умовах потокового надходження транзакцій. Описано часову природу даних, затримку в отриманні міток, зміну розподілу ознак і принцип контролю якості на ковзному вікні. Запропоновано інтегральний показник потреби в актуалізації, який поєднує сигнали якості, drift, поведінки скорів та готовності нових даних до перенавчання.

РОЗДІЛ 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Характеристика набору даних PaySim

Для експериментальної перевірки запропонованого підходу в роботі використано набір даних PaySim, описаний у праці Lopez-Rojas, Elmir та Axelsson [25]. У цій праці PaySim подано як симулятор транзакцій mobile money service, побудований на основі характеристик реальних фінансових операцій, що робить його придатним відкритим середовищем для досліджень у задачах fraud detection.

Набір містить 6 362 620 транзакцій, впорядкованих за 743 часовими кроками поля step. Кожний запис містить тип транзакції, її суму, баланси відправника та отримувача до і після виконання операції, а також цільову ознаку isFraud. За структурою набору переважають операції CASH-OUT 35,17 % та PAYMENT 33,81 %, далі йдуть CASH-IN 21,99 %, TRANSFER 8,38 % і DEBIT 0,65 %.

Кількість шахрайських транзакцій становить 8 213, тобто близько 0,129 % від загального обсягу даних. Отже, набір є сильно незбалансованим за класами. Шахрайські події фактично зосереджені в операціях CASH-OUT і TRANSFER: для CASH-OUT частка fraud становить близько 0,184 %, а для TRANSFER – близько 0,769 %. Це підтверджує, що тип операції є важливою інформативною ознакою для моделі.

У межах цієї роботи PaySim розглядається не як повна модель усіх сценаріїв електронних платежів, а як контрольоване експериментальне середовище для перевірки базової моделі класифікації, модуля моніторингу якості та механізму актуалізації.

3.2 Підготовка даних і первинний аналіз

На етапі підготовки даних було відібрано поля, які безпосередньо характеризують транзакцію та можуть бути використані в моделі й модулі моніторингу: step, type, amount, oldbalanceOrg, newbalanceOrig, oldbalanceDest, newbalanceDest та цільову ознаку isFraud. Ідентифікатори nameOrig і nameDest не використовувалися, оскільки вони не несуть узагальнюваного змісту для побудови моделі. Ознака isFlaggedFraud також не включалася до набору ознак, бо в наявних даних вона активується лише у 16 випадках.

Первинний аналіз показав, що розподіл сум транзакцій є виражено асиметричним: середнє значення amount становить 179 861,92, тоді як медіана дорівнює 74 871,94. Для верхніх квантилів отримано такі значення: 90-й перцентиль – 365 423,30, 95-й – 518 634,19, 99-й – 1 615 979,50. Це свідчить про наявність довгого правого хвоста та істотної неоднорідності числових ознак.

Часова структура потоку також є нерівномірною. Для 743 значень step середня кількість транзакцій на один часовий крок становить 8 563,42, тоді як медіанне значення дорівнює лише 529. Максимальна кількість транзакцій спостерігається на кроці 19 і становить 51 352, а найбільша кількість fraud-подій зафіксована на кроці 212 і дорівнює 40. Отже, дані природно інтерпретуються як потік із мінливою інтенсивністю.

Для практичної реалізації категоріальна ознака type перетворюється в набір бінарних індикаторів, а на основі балансів додатково формуються похідні характеристики зміни коштів відправника та отримувача. Поле step використовується як часовий індекс для хронологічного поділу даних на початковий навчальний інтервал і подальший потоковий сегмент, що дозволяє перейти від статичної класифікації до моделювання реального сценарію послідовного надходження транзакцій.

3.3 Базова модель виявлення шахрайських транзакцій

Як базову модель для виявлення шахрайських транзакцій у практичній частині використано LightGBM – реалізацію градієнтного бустингу над деревами рішень, орієнтовану на ефективну роботу з табличними даними [7]. Такий вибір є доречним для задачі fraud detection, оскільки модель працює з неоднорідними числовими ознаками, дозволяє описувати нелінійні залежності між характеристиками транзакцій і при цьому залишається достатньо швидкою для багаторазового перенавчання в межах експериментального сценарію.

У поточній реалізації до моделі подаються числові характеристики транзакції amount, oldbalanceOrg, newbalanceOrig, oldbalanceDest та newbalanceDest, а також дві похідні ознаки, що описують зміну балансу відправника і отримувача. Категоріальна ознака type кодується набором бінарних індикаторів для п'яти можливих типів операцій. У підсумку базова модель працює з компактним набором табличних ознак, який зберігає як фінансові характеристики транзакції, так і інформацію про її операційний тип.

Початкове навчання моделі виконується хронологічно на історичній частині даних, виділеній за полем step. У реалізації кроки step $\leq$ 200 відносяться до початкового навчального інтервалу, після чого останні 20 % цього інтервалу використовуються як еталонне вікно для оцінки початкової якості та формування reference-характеристик моделі. Для компенсації сильного дисбалансу класів використовується параметр scale_pos_weight, який задається як відношення кількості легітимних транзакцій до кількості шахрайських у навчальному вікні.

У практичній реалізації використано помірні параметри моделі: n_estimators = 200, learning_rate = 0,05, num_leaves = 31, subsample = 0,8 та colsample_bytree = 0,8. Така конфігурація не претендує на повну оптимальність, однак забезпечує достатньо сильний і водночас

обчислювально помірний baseline для перевірки модуля моніторингу та актуалізації. Натренована модель зберігається як артефакт model_v0.joblib і завантажується сервісом прогнозування при старті системи.

3.4 Реалізація модуля моніторингу та актуалізації

У підрозділі розглянуто основні компоненти реалізованої системи, які забезпечують обробку транзакційного потоку, формування прогнозів, накопичення результатів і прийняття рішення про актуалізацію моделі. Спочатку описано загальну архітектуру системи, після чого окремо розглянуто сервіс прогнозування, симулятор потоку транзакцій, сервіс моніторингу та аналітичний модуль. Разом ці компоненти утворюють програмний контур, у межах якого модель не лише виконує класифікацію транзакцій, а й контролюється з погляду подальшої придатності до використання.

3.4.1 Загальна архітектура системи

Практична реалізація системи побудована як набір незалежних мікросервісів, що взаємодіють через спільний том файлової системи. Такий підхід дозволяє запускати та масштабувати компоненти окремо, спрощує заміну окремих модулів і робить взаємодію між ними прозорою. Система складається з чотирьох сервісів: сервісу прогнозування, симулятора потоку транзакцій, сервісу моніторингу та аналітичного модуля. Усі вони запускаються у середовищі Docker одним викликом docker compose up і обмінюються даними через спільну директорію /shared. Узагальнена архітектура системи з потоками даних між сервісами наведена на рисунку 3.1.

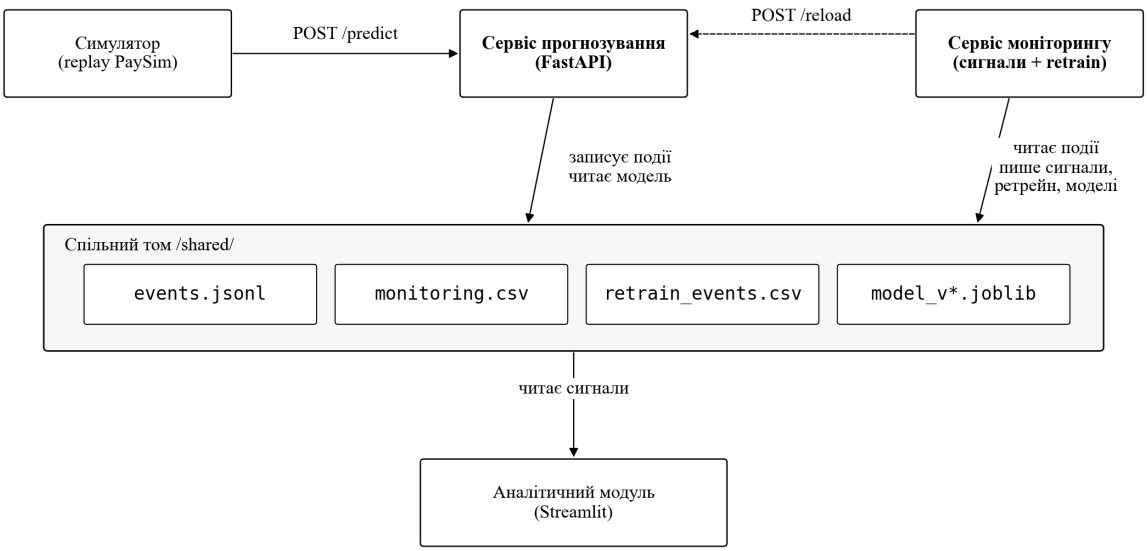


Рисунок 3.1 – Архітектура мікросервісної реалізації системи моніторингу

3.4.2 Сервіс прогнозування

Сервіс прогнозування реалізований як HTTP API на основі бібліотеки FastAPI. При запуску він завантажує останню версію навченої моделі з директорії /shared/models. Єдиний робочий ендпоінт POST /predict приймає JSON з ознаками транзакції, повертає оцінку ризику шахрайства та номер активної версії моделі. Кожен виклик додається до журналу /shared/logs/events.jsonl у вигляді одного JSON-рядка з міткою часу, ознаками транзакції, скором і номером моделі. Окремий ендпоінт POST /reload використовується сервісом моніторингу після успішного перенавчання для миттєвого перемикання активної версії моделі.

3.4.3 Симулятор потоку транзакцій

Симулятор замінює реальний потік платежів і дозволяє відтворювати сценарії з контрольованими параметрами. Він читає підвибірку PaySim, накладає на неї один з трьох сценаріїв з модуля demo_scenario (natural, drift_only, degradation) і послідовно надсилає кожну транзакцію на ендпоінт

прогнозування з заданою швидкістю (за замовчуванням 500 транзакцій за секунду). Конфігурація сценарію та швидкість керуються змінними оточення, що дозволяє швидко перемикати тип експерименту без перезбірки контейнера.

3.4.4 Сервіс моніторингу та актуалізації

Сервіс моніторингу є центральним компонентом системи. Він періодично читає журнал подій /shared/logs/events.jsonl, групує транзакції за номером часового вікна (за замовчуванням 8 значень step у швидкому режимі демонстрації) та обчислює чотири індивідуальні сигнали D_{quality}, D_{drift}, D_{conf}, D_{data} і зважений інтегральний показник S_t. Логіка обчислення сигналів повторно використовує функції з модуля monitoring, описаного у розділі 2. Кожне завершене вікно записується у /shared/logs/monitoring.csv. У разі перевищення інтегральним показником порогу $\tau = 0{,}4$ і накопичення достатнього обсягу нових міток сервіс ініціює перенавчання: навчає кандидатну модель на оновленому історичному вікні, валідує її на останньому розміченому вікні та, якщо PR-AUC кандидата не нижчий за 0,98 від поточної моделі, зберігає нову версію в /shared/models і викликає ендпоінт /reload сервісу прогнозування. Усі рішення фіксуються у /shared/logs/retrain_events.csv. Узагальнена логічна схема цього механізму прийняття рішень наведена на рисунку 3.2.

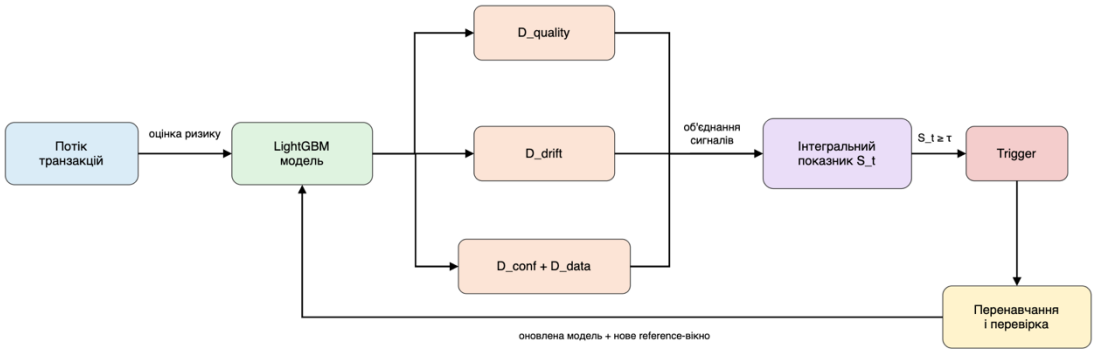


Рисунок 3.2 – Узагальнена логічна схема обчислення інтегрального показника та прийняття рішення про оновлення моделі

3.4.5 Аналітичний модуль

Аналітичний модуль реалізований як вебдодаток на основі бібліотеки Streamlit. Він читає журнали моніторингу та подій з директорії /shared/logs у режимі автоматичного оновлення кожні три секунди та відображає в одному інтерфейсі: значення інтегрального показника S_t у часі з позначенням порогу і подій перенавчання, поведінку чотирьох сигналів моніторингу окремо, динаміку PR-AUC за розміченими вікнами, а також статус системи та кількість оброблених вікон. Загальний вигляд інтерфейсу під час прогону зі сценарієм degradation наведено на рисунку 3.3.

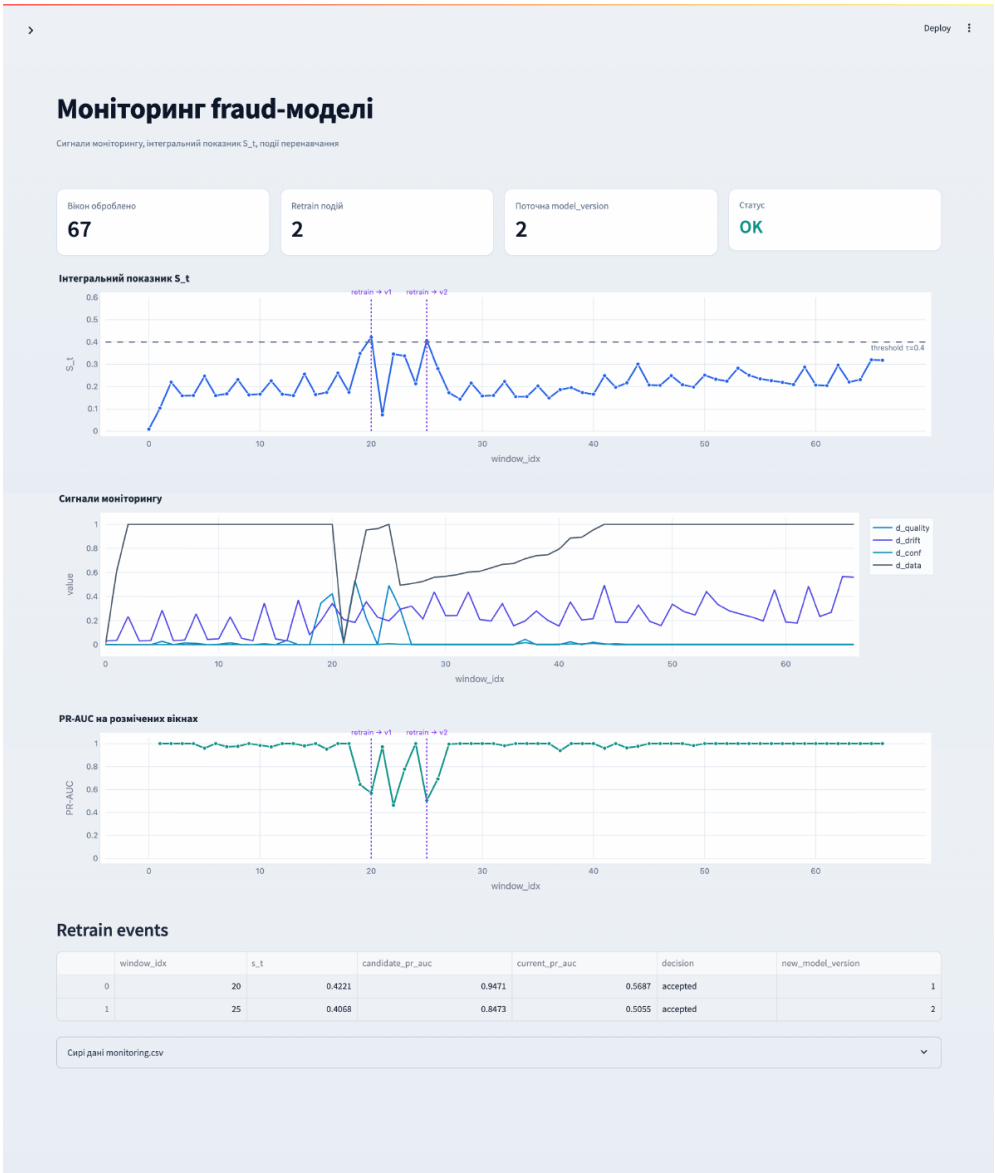


Рисунок 3.3 – Загальний вигляд аналітичного модуля під час прогону зі сценарієм degradation

3.5 Демонстраційний сценарій експериментальної перевірки

Базові параметри прогону, які залишалися спільними для всіх сценаріїв, наведено в таблиці 3.1.

Таблиця 3.1 – Параметри конфігурації демонстраційного прогону

Параметр	**Значення**
Ширина моніторингового вікна (WINDOW_STEPS)	8 значень step
Затримка появи міток (LABEL_DELAY)	1 вікно
Мінімум нових міток для тригера (MIN_LABELS_FOR_RETRAIN)	100
Поріг інтегрального показника (τ_s)	0,40
Ваги сигналів (w_1,$w_{2,}$,w_3,w_4)	0,40 / 0,30 / 0,15 / 0,15
Швидкість симулятора (RATE_TPS)	500 запитів/с
Частка підвибірки за step (QUICK_FRACTION)	10 %
Множник грошових ознак (degradation)	4
Множник грошових ознак (drift_only)	8
Частка PAYMENT-fraud (degradation)	3 %

Для наочної перевірки розробленого механізму в практичній частині використано не лише природний потік PaySim, а й окремий демонстраційний сценарій деградації. Його мета полягає в тому, щоб контрольовано створити умови, за яких початкова модель втрачає актуальність, а система моніторингу фіксує цей стан і ініціює оновлення.

Сценарій складається з трьох фаз. На фазі normal дані залишаються без змін, і модель працює у близьких до еталонних умовах. На фазі drift всі грошові ознаки множаться на коефіцієнт 4,0, а частина транзакцій типу PAYMENT у найбільшій сумовій зоні додатково позначається як шахрайська з частотою 3 %. Таким чином одночасно моделюються і зміна розподілу ознак, і поява нового шахрайського патерну, якого початкова модель не бачила під час навчання. На фазі recovery цей змінений режим зберігається, що дозволяє оцінити, чи здатна оновлена модель адаптуватися до нового стану потоку. Послідовність фаз сценарію схематично показано на рисунку 3.4.

Загальна тривалість одного демонстраційного прогону при швидкості симуляції 500 запитів за секунду становить близько 30 секунд. У кожному прогоні формується 67 моніторингових вікон, які покривають step від 201 до 736. Параметри прогону, описані в таблиці 3.1, вибрані для швидкого але змістовного відтворення поведінки системи у трьох сценаріях.

Рисунок 3.4 – Логіка демонстраційного сценарію normal -> drift -> recovery

Експериментальна перевірка виконана у вигляді трьох незалежних запусків мікросервісної системи з різними сценаріями: natural, drift_only та degradation. Усі три прогони використовують той самий початковий стан моделі та один і той самий набір даних, відмінність лише у типі застосованого збурення. Журнали моніторингу та події перенавчання, сформовані під час прогонів, є основою для подальшого аналізу результатів у підрозділі 3.6.

3.6 Аналіз результатів експериментальної перевірки

Експериментальна перевірка системи виконувалася на трьох сценаріях, що відрізнялися лише типом застосованого збурення вхідних даних. У кожному прогоні сформовано 67 моніторингових вікон, що покривають інтервал від step 201 до step 736 з кроком вісім. Узагальнене порівняння трьох сценаріїв за основними метриками наведено в таблиці 3.2.

Таблиця 3.2 – Порівняння результатів трьох демонстраційних сценаріїв

Сценарій	**Вікон**	**Retrain**	**Mean S_t (drift)**	**Max S_t**	**Mean PR-AUC (drift)**	**PR-AUC після retrain**
natural	67	0	0,200	0,306	0,992	—
drift_only	67	1	0,273	0,419	0,936	0,963
degradation	67	2	0,222	0,422	0,899	0,947 → 0,847

У сценарії natural дані залишалися без змін, інтегральний показник стабільно тримався в межах 0,17–0,30 і жодного разу не перевищив поріг τ =

0,4. PR-AUC моделі залишався вище 0,98 у всіх фазах. Це підтверджує, що за відсутності збурень система не ініціює помилкових перенавчань.

У сценарії drift_only зсув грошових ознак у вісім разів спричинив швидке зростання D_{quality} і відчутне погіршення якості прогнозу. На вікні 19 поточний PR-AUC опустився до 0,532, що в поєднанні з накопиченням нових міток вивело S_t до 0,419 і запустило перенавчання. Кандидатна модель досягла PR-AUC 0,963 на тому ж вікні і була прийнята як версія v1. Після цього PR-AUC утримувався вище 0,93 до кінця прогону.

У сценарії degradation поєднання зсуву грошових ознак у чотири рази з появою нового патерну шахрайства серед транзакцій типу PAYMENT спричинило два спрацювання тригера. Перше відбулося на вікні 20 (S_t = 0,422, PR-AUC поточної моделі 0,569, кандидата 0,947, прийнято як v1), друге – на вікні 25 (S_t = 0,407, PR-AUC поточної 0,505, кандидата 0,847, прийнято як v2). Двоступеневе перенавчання відображає логіку адаптації системи до тривалого збурення: перший кандидат стабілізував якість частково, а другий повністю освоїв новий патерн.

Зміна інтегрального показника у часі для трьох сценаріїв наведена на рисунку 3.5. Чітко видно, що в сценарії natural крива тримається нижче порогу, тоді як у двох інших сценаріях вона перетинає поріг у момент тригера, що позначено вертикальними штриховими лініями.

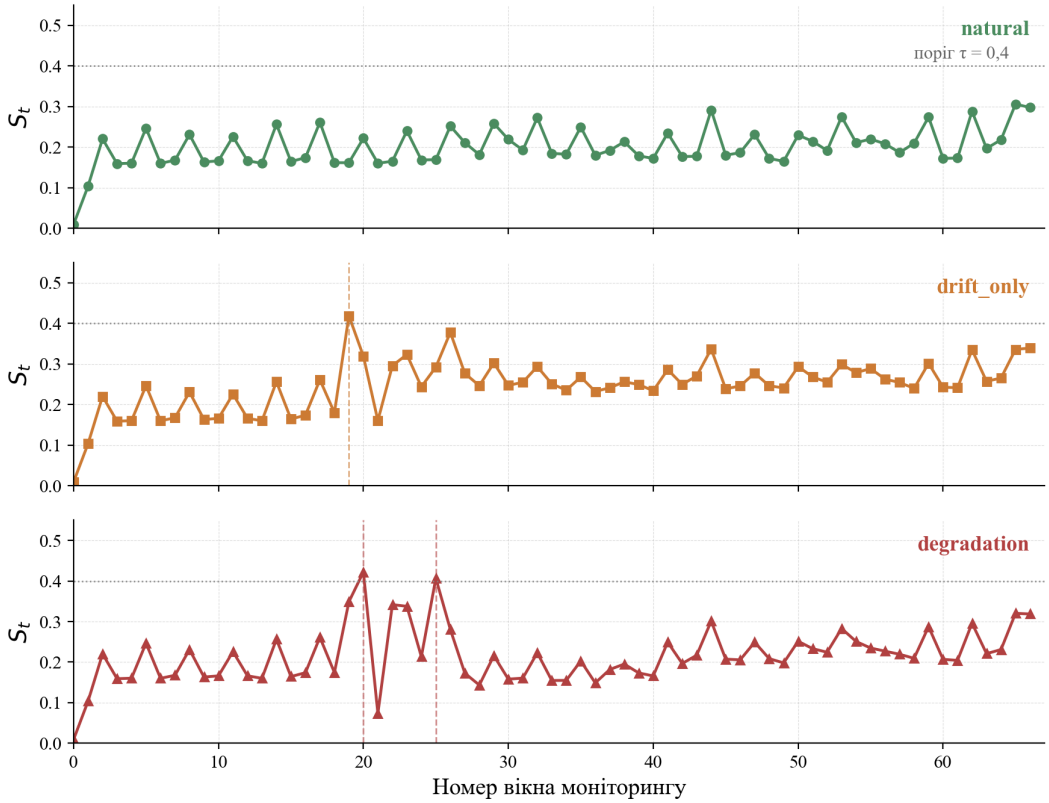


Рисунок 3.5 – Інтегральний показник S_t у часі для трьох сценаріїв

Відновлення якості моделі після прийняття кандидатних версій видно на рисунку 3.6: у сценаріях drift_only та degradation PR-AUC падає у фазі drift до значень 0,47–0,69, але після перенавчання повертається до рівня вище 0,93. У сценарії natural жодного провалу PR-AUC не спостерігається.

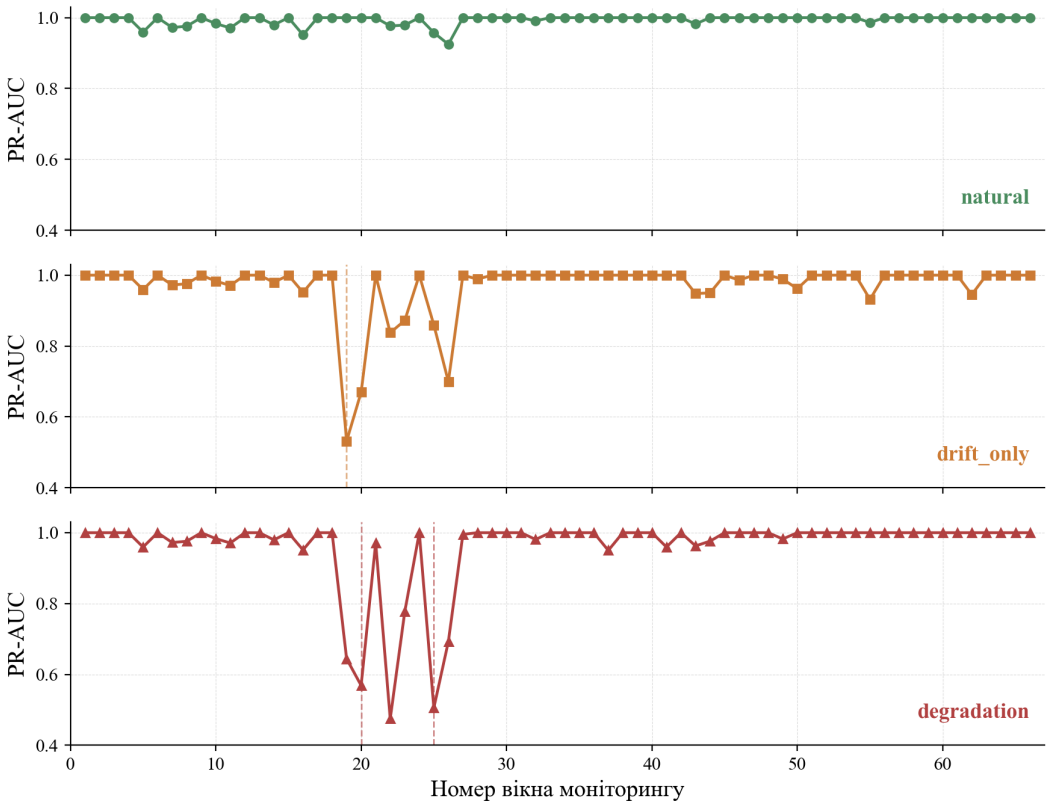


Рисунок 3.6 – PR-AUC у часі для трьох сценаріїв

Результати трьох сценаріїв ілюструють два різні механізми спрацювання тригера. У сценарії drift_only рішення про перенавчання приймається переважно через комбінацію зростаючого D_{quality} і поступового падіння якості, спричиненого зсувом ознак. У сценарії degradation тригер активується насамперед через стрибок D_{quality} – модель не розпізнає новий тип шахрайства, поки на ньому не навчиться. Інтегральний показник зводить ці різні джерела деградації до єдиного критерію, що дозволяє ухвалювати рішення про оновлення в обох випадках за тим самим правилом.

3.7 Висновки до розділу 3

У третьому розділі подано практичну реалізацію системи оцінки якості та актуалізації fraud-моделі на потокових транзакційних даних. Описано набір даних PaySim, побудовано базову модель на основі LightGBM, реалізовано

модуль моніторингу та механізм оновлення моделі. Результати експериментальної перевірки показали, що запропонований підхід дає змогу виявляти деградацію моделі та відновлювати якість її роботи в змінених умовах потоку. Реалізація виконана як мікросервісна система з чотирьох компонентів – сервісу прогнозування, симулятора потоку, сервісу моніторингу та аналітичного модуля. Експериментальна перевірка на трьох сценаріях підтвердила, що інтегральний показник S_t коректно реагує як на зсуви розподілу вхідних ознак, так і на появу нових патернів шахрайства, і дозволяє автоматично запускати перенавчання моделі в обох випадках.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

4.1 Постановка задачі проєктування

У цьому розділі виконано техніко-економічне обґрунтування програмного продукту — інтелектуальної системи оцінки якості та автоматизованої актуалізації моделі для виявлення шахрайських платіжних транзакцій у реальному часі. Для порівняння можливих варіантів реалізації застосовано метод функціонально-вартісного аналізу (ФВА), який дозволяє оцінити доцільність кожної підфункції майбутньої системи через зіставлення її технічних характеристик і витрат на реалізацію.

Аналіз будується послідовно. Спочатку визначаються головна функція продукту та підфункції, які забезпечують її виконання. Далі для кожної підфункції розглядаються альтернативні шляхи реалізації, з яких формуються два повноцінні варіанти. Для кожного варіанта обчислюється коефіцієнт технічного рівня та повна вартість розробки. Підсумкове рішення приймається за коефіцієнтом техніко-економічного рівня.

Як вхідні дані використано результати практичної реалізації з розділу 3: характеристики базової моделі на основі LightGBM, обсяги потокових даних PaySim, склад модуля моніторингу. Це забезпечує узгодженість оцінок з фактичним станом програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція програмного продукту F0 полягає у безперервній оцінці якості моделі виявлення шахрайства на потоці транзакцій та

автоматичному запуску процедури перенавчання у разі деградації. Реалізація F0 декомпонується на чотири підфункції:

1) F1 — базова модель класифікації транзакцій;
2) F2 — метод виявлення дрейфу розподілу вхідних даних;
3) F3 — спосіб агрегації сигналів моніторингу та формування рішення про потребу перенавчання;
4) F4 — критерій прийняття кандидатної моделі після перенавчання.

Для кожної підфункції розглянуто по два варіанти реалізації, що формують морфологічну карту програмного продукту:

1) F1.а — LightGBM (градієнтний бустинг над деревами рішень);
2) F1.б — логістична регресія;
3) F2.а — двовибірковий тест Колмогорова–Смирнова;
4) F2.б — індекс стабільності популяції (PSI);
5) F3.а — інтегральний показник S_t як зважена сума сигналів погіршення якості, дрейфу даних, падіння впевненості та зміни обсягу потоку;
6) F3.б — прості порогові правила за принципом «спрацьовує будь-який окремий сигнал»;
7) F4.а — відносний критерій: PR-AUC кандидата не нижче 0,98 від PR-AUC поточної моделі на тому ж розміченому вікні;
8) F4.б — абсолютний критерій: PR-AUC кандидата не нижче фіксованого порогового значення 0,70.

Варіанти реалізації основних функцій F1–F4 наведені у морфологічній карті системи (рис. 4.1).

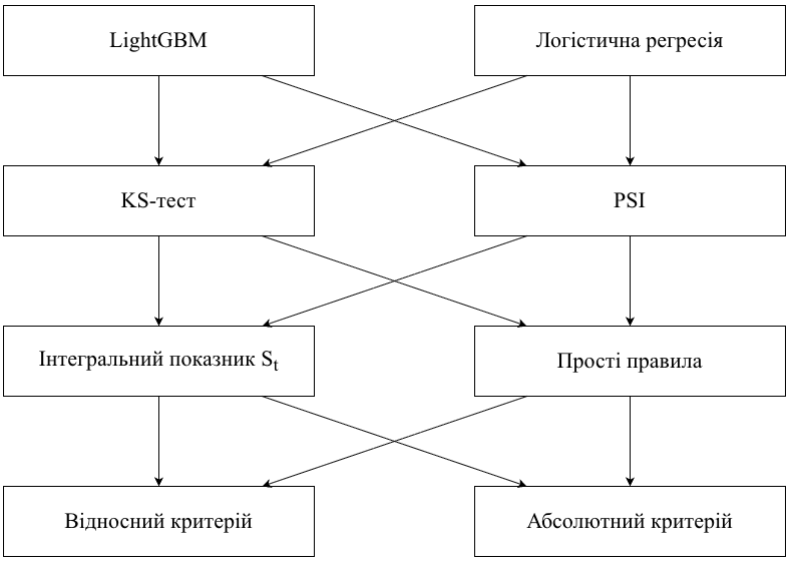


Рисунок 4.1 – Морфологічна карта програмного продукту

Переваги і недоліки кожного варіанта реалізації узагальнено в позитивно-негативній матриці (таблиця 4.1).

Таблиця 4.1 – Позитивно-негативна матриця варіантів реалізації функцій

Функція	**Варіант**	**Переваги**	**Недоліки**
F1	а — LightGBM	Висока якість на табличних даних, стійкість до незбалансованості	Більший час навчання, складніша інтерпретація
	б — логістична регресія	Швидке навчання, проста інтерпретація коефіцієнтів	Обмежена якість на нелінійних залежностях
F2	а — KS-тест	Чутливість до зміщення розподілу неперервних ознак, відома статистика	Менша стабільність на малих вибірках
	б — PSI	Стійкість до варіації обсягу даних, проста інтерпретація	Менша чутливість до невеликих змін розподілу

Продовження таблиці 4.1

Функція	Варіант	Переваги	Недоліки
F3	а — інтегральний показник S_t	Враховує одночасно якість, дрейф, впевненість і обсяг даних	Потребує задання ваг компонент
	б — прості правила	Простота реалізації, прозорість рішення	Часті хибні спрацювання при незалежному моніторингу сигналів
F4	а — відносний критерій	Враховує поточну якість моделі, адаптивний	Потребує надійної оцінки PR-AUC поточної моделі
	б — абсолютний критерій	Простота, прозоре правило	Не враховує поточний стан моделі

На основі морфологічної карти і позитивно-негативної матриці сформовано два варіанти реалізації програмного продукту:

Варіант 1: F1.а + F2.а + F3.а + F4.а — LightGBM як базова модель, KS-тест для виявлення дрейфу, інтегральний показник S_t для агрегації сигналів, відносний критерій прийняття кандидата. Цей варіант відповідає реалізації, описаній у розділі 3.

Варіант 2: F1.б + F2.б + F3.б + F4.б — логістична регресія, PSI для виявлення дрейфу, прості порогові правила як стратегія тригерування перенавчання, абсолютний критерій прийняття кандидата. Це класична спрощена альтернатива з прозорою інтерпретацією, але обмеженішими можливостями адаптації.

Подальші розрахунки виконуються для цих двох варіантів.

4.3 Обґрунтування системи параметрів програмного продукту

Для кількісного порівняння варіантів сформовано систему техніко-економічних параметрів програмного продукту. Параметри обрано так, щоб

охопити основні характеристики системи: якість моделі, швидкість роботи на потоці, тривалість циклу актуалізації та трудомісткість розробки.

X1 — якість базової моделі, виміряна на тестовому вікні після стабілізації (PR-AUC, %). Більше — краще.

X2 — час інференсу однієї транзакції на сервісі прогнозування (мс). Менше — краще.

X3 — час повного циклу перенавчання моделі від тригера до публікації нової версії (хв). Менше — краще.

X4 — трудомісткість реалізації програмного продукту (люд-год). Менше — краще.

Діапазон допустимих значень кожного параметра наведено в таблиці 4.2. Гірше значення відповідає нижній межі прийнятної якості системи виявлення шахрайства, краще — рівню, який досягається у промислових ML-системах моніторингу моделей.

Таблиця 4.2 – Основні параметри програмного продукту

Параметр	**Позначення**	**Одиниці**	**Гірше**	**Середнє**	**Краще**
Якість базової моделі (PR-AUC)	X1	%	70	85	95
Час інференсу транзакції	X2	мс	200	80	20
Час циклу перенавчання	X3	хв	60	20	5
Трудомісткість реалізації	X4	люд-год	1200	800	400

Бальна оцінка параметра Bi змінюється від 0 до 100 і обчислюється лінійною інтерполяцією між значеннями "гірше" та "краще". Для параметрів X2, X3, X4 шкала інверсна: менше значення параметра відповідає вищому балу. Бали, отримані для двох варіантів реалізації, подано далі у таблиці 4.6 разом із розрахунком рівня якості.

Графічне представлення шкал бальної оцінки за параметрами X1–X4 наведено на рисунках 4.2–4.5. На кожному рисунку відмічено фактичні

значення для Варіанта 1 і Варіанта 2 разом із відповідними бальними оцінками.

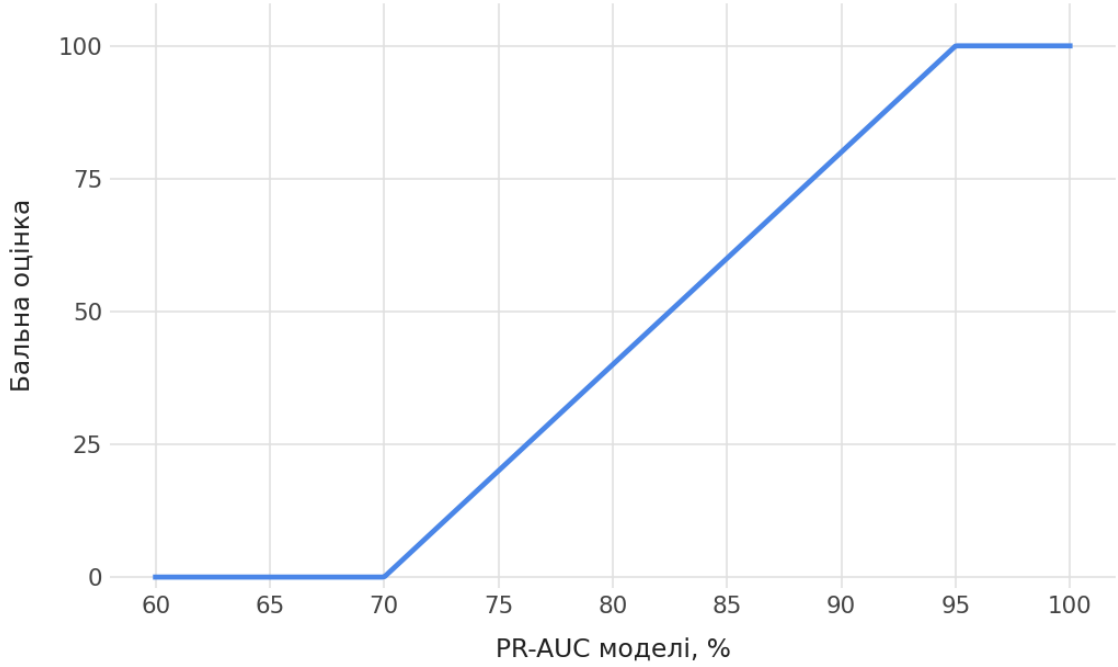


Рисунок 4.2 – Шкала бальної оцінки параметра X1 (PR-AUC моделі)

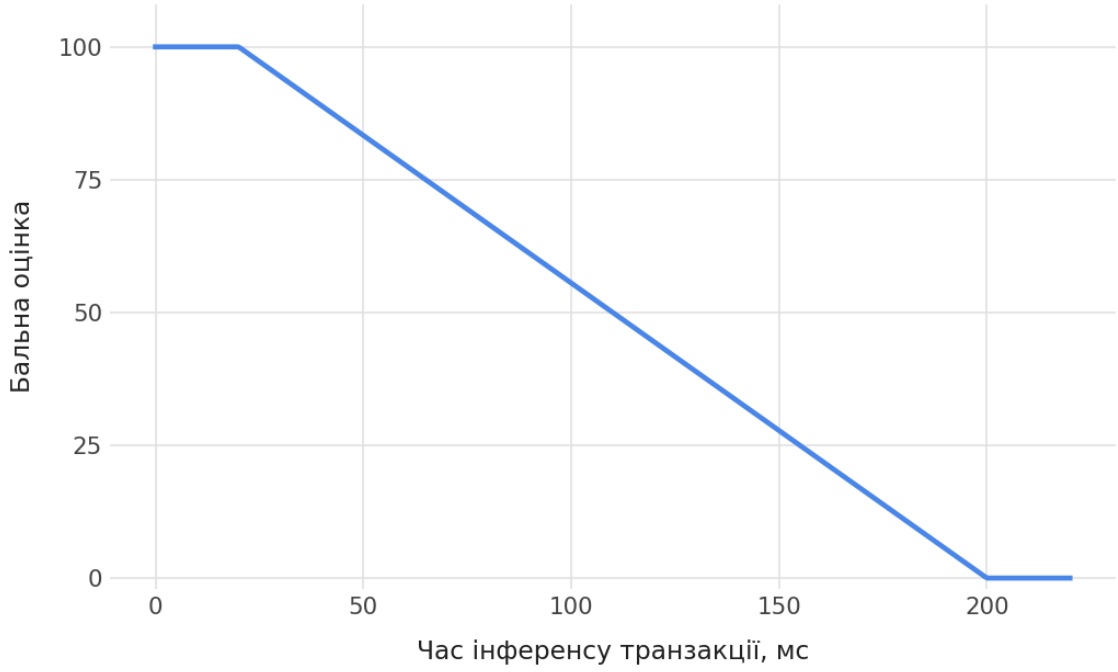


Рисунок 4.3 – Шкала бальної оцінки параметра X2 (час інференсу транзакції)

Шкалу бальної оцінки за параметром X3 наведено на рисунку 4.4.

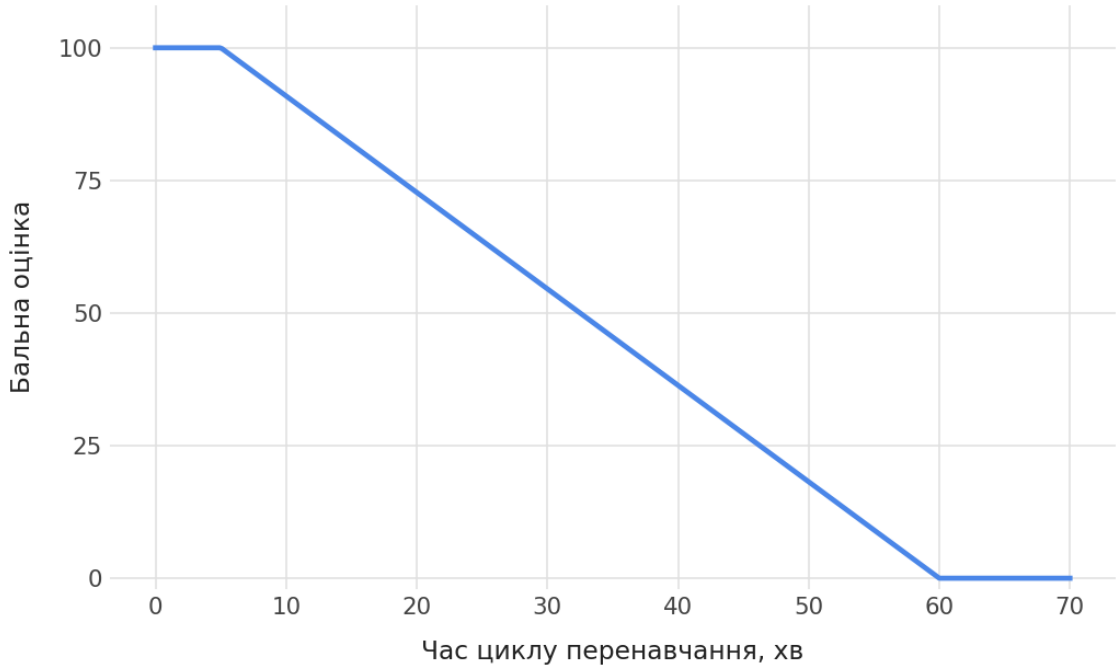


Рисунок 4.4 – Шкала бальної оцінки параметра X3 (час циклу перенавчання)

Аналогічну шкалу для параметра X4 показано на рисунку 4.5.

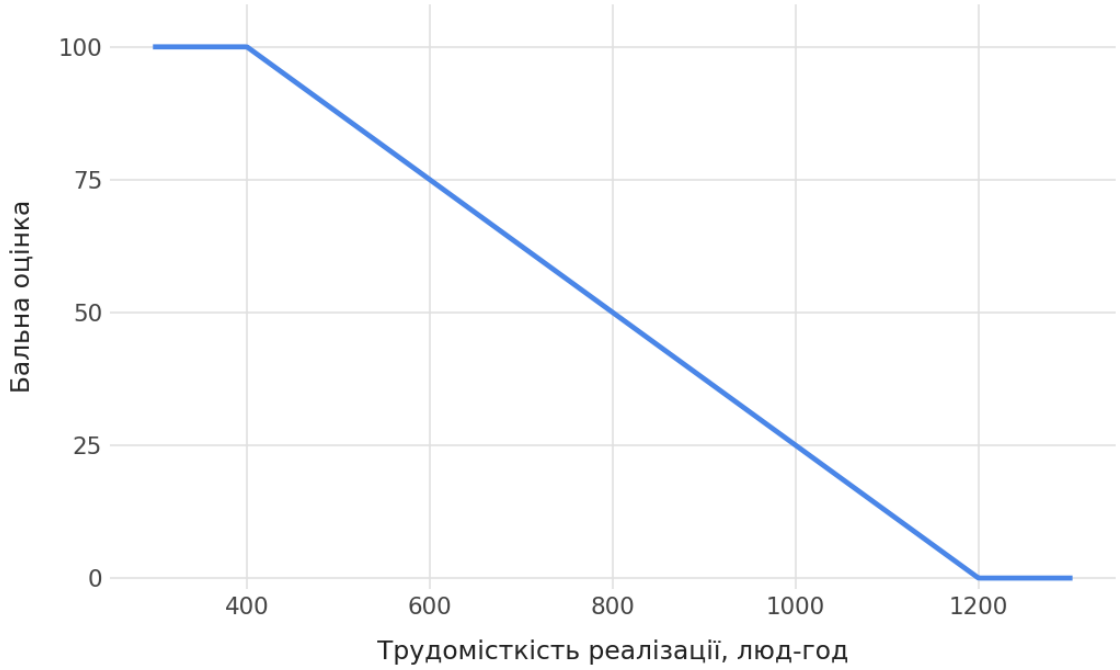


Рисунок 4.5 – Шкала бальної оцінки параметра X4 (трудомісткість реалізації)

Як видно з рисунків 4.2–4.5, Варіант 1 має значну перевагу за параметром якості моделі X1 та помітно поступається Варіанту 2 лише за

параметром часу перенавчання X3. За двома іншими параметрами різниця незначна.

4.4 Аналіз експертного оцінювання параметрів

Вагомість параметрів встановлюється шляхом експертного оцінювання. До експертної групи увійшло сім фахівців: два науково-педагогічні працівники кафедри ММСА, три ML-інженери з досвідом виробничих систем моніторингу моделей, два спеціалісти з протидії платіжному шахрайству. Кожен експерт упорядкував параметри X1–X4 від найважливішого (ранг 1) до найменш важливого (ранг 4). Результати ранжування зведено в таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

Позна-чення	**Параметр**	**Од.**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**	**Σ Ri**	**Δi²**
X1	PR-AUC моделі	%	1	1	1	2	1	1	1	8	90,25
X2	Час інференсу	мс	2	3	2	1	2	2	3	15	6,25
X3	Час перенавчання	хв	4	4	4	4	4	4	4	28	110,25
X4	Трудомісткість	люд-год	3	2	3	3	3	3	2	19	2,25

Сумарна сума рангів по всіх експертах становить 70, що відповідає теоретичному значенню $N \cdot n \cdot (n+1)/2 = 7 \cdot 4 \cdot 5/2 = 70$. Це підтверджує відсутність пропусків або порушень порядку ранжування.

Середня сума рангів T обчислюється за формулою (4.1):

$$T = \frac{1}{2} \cdot N \cdot (n + 1) = 0{,}5 \cdot 7 \cdot 5 = 17{,}5. \quad (4.1)$$

де N = 7 — кількість експертів;

n = 4 — кількість параметрів.

Сума квадратів відхилень суми рангів від середнього (формула 4.2):

$$S = \sum_{i=1}^{n} \Delta_i^2 = 90{,}25 + 6{,}25 + 110{,}25 + 2{,}25 = 209{,}00. \quad (4.2)$$

Коефіцієнт узгодженості (конкордації) Кендалла обчислюється за формулою (4.3):

$$W = \frac{12 \cdot S}{N^2 \cdot (n^3 - n)} = \frac{12 \cdot 209}{49 \cdot 60} = 0{,}853. \quad (4.3)$$

Отримане значення W = 0,853 перевищує нормативний поріг 0,67, тому результати ранжування вважаються узгодженими і придатними для подальших розрахунків.

Для уточнення вагомості параметрів використано метод попарного порівняння. Ступінь переваги параметра Xi над Xj визначається числовим значенням aij за правилом: aij = 1,5, якщо Xi важливіший за Xj; aij = 1,0, якщо параметри рівнозначні; aij = 0,5, якщо Xi менш важливий. Результати порівняння зведено в таблиці 4.4.

Таблиця 4.4 – Попарне порівняння параметрів

Пара	**Підсумкова оцінка експертів**	**aij**	**aji**
X1 і X2	X1 > X2	1,5	0,5
X1 і X3	X1 > X3	1,5	0,5
X1 і X4	X1 > X4	1,5	0,5
X2 і X3	X2 > X3	1,5	0,5
X2 і X4	X2 > X4	1,5	0,5
X3 і X4	X3 < X4	0,5	1,5

На основі попарного порівняння побудовано матрицю A = ‖aij‖ і обчислено вагомість параметрів. Спочатку обчислюється сума елементів кожного рядка bi = Σj aij, після чого виконується нормування Кві = bi / Σbi. Розрахунок виконується ітеративно: на кожному наступному кроці bi' = Σj aij · bj. Процес зупиняється, коли максимальна різниця між послідовними значеннями Кві стає меншою за 2%. Результати трьох ітерацій наведено в таблиці 4.5.

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметр	**X1**	**X2**	**X3**	**X4**	**Ітер. 1 (bi / Кві)**	**Ітер. 2 (bi' / Кві')**	**Ітер. 3 (bi'' / Кві'')**
X1	1,0	1,5	1,5	1,5	5,5 / 0,3438	21,25 / 0,3602	77,875 / 0,3605
X2	0,5	1,0	1,5	1,5	4,5 / 0,2813	16,25 / 0,2754	59,125 / 0,2737
X3	0,5	0,5	1,0	0,5	2,5 / 0,1563	9,25 / 0,1568	34,125 / 0,1580
X4	0,5	0,5	1,5	1,0	3,5 / 0,2188	12,25 / 0,2076	44,875 / 0,2078
Σ	—	—	—	—	16,0 / 1,0000	59,0 / 1,0000	216,0 / 1,0000

Максимальна різниця між значеннями вагомості на другій і третій ітераціях не перевищує 1%, тому подальші ітерації не виконуються. Підсумкові значення вагомості параметрів: Кві1 = 0,3605, Кві2 = 0,2737, Кві3 = 0,1580, Кві4 = 0,2078. Найвагомішим параметром закономірно виявилася якість базової моделі — ключова характеристика системи виявлення шахрайства. Другим за важливістю — час інференсу транзакції, що відповідає вимозі роботи у режимі реального часу.

4.5 Аналіз рівня якості варіантів реалізації функцій

Для кожного з двох варіантів реалізації визначено фактичні значення параметрів X1–X4. Значення X1 для Варіанта 1 узгоджене з результатами експериментальної перевірки з розділу 3 (LightGBM на стабільному вікні набору PaySim); для Варіанта 2 — з оцінкою якості логістичної регресії на тих самих ознаках. Значення X2 і X3 для обох варіантів отримано натурним вимірюванням на серверній конфігурації, описаній у розділі 3. За цими значеннями обчислено бальні оцінки Bi методом лінійної інтерполяції між граничними значеннями таблиці 4.2.

Коефіцієнт технічного рівня варіанта розраховується за формулою (4.4):

$$К_к = \sum_{i=1}^{n} B_i \cdot Кв_i. \quad (4.4)$$

Результати розрахунку рівня якості наведено в таблиці 4.6.

Таблиця 4.6 – Розрахунок рівня якості варіантів реалізації

Параметр	**Кві**	**Варіант 1: значення**	**Варіант 1: Bі · Кві**	**Варіант 2: значення**	**Варіант 2: Bі · Кві**
X1, %	0,3605	92 (Bі = 88,0)	31,724	72 (Bі = 8,0)	2,884
X2, мс	0,2737	25 (Bі = 97,2)	26,604	5 (Bі = 100,0)	27,370
X3, хв	0,1580	12 (Bі = 87,3)	13,793	3 (Bі = 100,0)	15,800
X4, люд-год	0,2078	800 (Bі = 50,0)	10,390	750 (Bі = 56,3)	11,699
Кк	—	—	82,51	—	57,75

Коефіцієнт технічного рівня Варіанта 1 становить 82,51, Варіанта 2 — 57,75. Розрив у 24,76 пункта формується переважно за рахунок параметра X1 (якості базової моделі): логістична регресія на сильно незбалансованих даних з фактичною часткою шахрайських транзакцій близько 0,13% поступається LightGBM приблизно на 20 пунктів PR-AUC. Варіант 2 виграє у швидкості інференсу та тривалості перенавчання, але відносний внесок цих параметрів менший за внесок якості моделі.

4.6 Економічний аналіз варіантів розробки ПП

Витрати на розробку програмного продукту складаються з заробітної плати виконавців, відрахувань на єдиний соціальний внесок, вартості машинного часу та накладних витрат. Для розрахунку прийнято бригаду з двох виконавців: розробник з окладом 30 000 грн і керівник проєкту з окладом 55 000 грн на місяць. Обидва оклади перевищують встановлений законодавством мінімальний розмір заробітної плати, а в межах бригади ставки різні, що відповідає вимогам нормативного регулювання оплати праці.

Трудомісткість розробки розраховується за формулою (4.5):

$$T = T_Р \cdot К_П \cdot К_{СК} \cdot К_М \cdot К_{СТ} \cdot К_{СТ.М}, \qquad (4.5)$$

де ТР — базова трудомісткість за групами новизни і складності алгоритму;

КП — поправковий коефіцієнт характеру вхідної інформації;

КСК — складність контролю інформації;

КМ — рівень використовуваної мови програмування;

КСТ — використання стандартних модулів;

КСТ.М — складність математичного забезпечення.

Базова трудомісткість для обох варіантів прийнята однаковою за ТР = 64 людино-дні (нова інженерна задача — група новизни А, складність алгоритму — 3). Коефіцієнти КП = 1,40 (обробка великого обсягу потокових транзакційних даних) і КСК = 1,00 спільні для варіантів. КМ для мови Python дорівнює 0,85. КСТ і КСТ.М розрізняються для варіантів: у Варіанта 1 використовується більше стандартних модулів наукових обчислень (LightGBM, scipy.stats, pandas) — КСТ = 0,80; у Варіанта 2 значна частина статистики PSI і правил агрегації пишеться вручну — КСТ = 0,85. Значення КСТ.М відповідно 1,20 і 1,15. Підсумкові значення трудомісткості наведено у таблиці 4.7.

Таблиця 4.7 – Розрахунок трудомісткості розробки

Варіант	**ТР, днів**	**КП**	**КСК**	**КМ**	**КСТ**	**КСТ.М**	**Т, люд-год (Т · 8)**
Варіант 1	64	1,40	1,00	0,85	0,80	1,20	732 (округлено 800)
Варіант 2	64	1,40	1,00	0,85	0,85	1,15	744 (округлено 750)

Округлення до 800 і 750 людино-годин враховує час на тестування, документування й налаштування експерименту. У Варіанті 1 цей надлишок більший за рахунок калібрування ваг інтегрального показника S_t і перевірки відносного критерію прийняття кандидата на історичних вікнах.

Усереднена погодинна ставка виконавців обчислюється за формулою (4.6):

$$С_Ч = \frac{М_{розр} + М_{кер}}{2 \cdot N_{рд} \cdot t_д} = \frac{30000 + 55000}{2 \cdot 21 \cdot 8} = 252{,}98 \text{ грн/год.} \quad (4.6)$$

Заробітна плата виконавців з урахуванням додаткової (КД = 0,2) обчислюється за формулою (4.7):

$$С_{ЗП} = С_Ч \cdot Т \cdot (1 + К_Д). \quad (4.7)$$

Для Варіанта 1: СЗП1 = 252,98 · 800 · 1,2 = 242 860,80 грн. Для Варіанта 2: СЗП2 = 252,98 · 750 · 1,2 = 227 682,00 грн.

Відрахування на єдиний соціальний внесок (ЄСВ) становлять 22% від нарахованої заробітної плати (формула 4.8):

$$С_{від} = С_{ЗП} \cdot 0{,}22. \quad (4.8)$$

Для Варіанта 1: Свід1 = 242 860,80 · 0,22 = 53 429,38 грн. Для Варіанта 2: Свід2 = 227 682,00 · 0,22 = 50 090,04 грн.

Витрати на машинний час обчислюються через річні експлуатаційні витрати обладнання та собівартість однієї машино-години. Прийнято ціну ноутбука розробника ЦПР = 40 000 грн, коефіцієнт транспортно-монтажних робіт КТМ = 1,15, норму амортизації КА = 0,25, норму поточного ремонту КР = 0,05.

Амортизаційні відрахування (формула 4.9):

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1{,}15 \cdot 0{,}25 \cdot 40000 = 11\,500{,}00 \text{ грн.} \quad (4.9)$$

Витрати на поточний ремонт (формула 4.10):

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1{,}15 \cdot 40000 \cdot 0{,}05 = 2\,300{,}00 \text{ грн.} \quad (4.10)$$

Ефективний фонд часу роботи обладнання за рік обчислюється з урахуванням кількості робочих днів, тривалості зміни та коефіцієнта використання обладнання у часі КВ = 0,5 (формула 4.11):

$$Т_{ЕФ} = (Д_К - Д_В - Д_С - Д_Р) \cdot t_З \cdot К_В =$$
$$= (365 - 104 - 12 - 16) \cdot 8 \cdot 0{,}5 = 932 \text{ год.} \quad (4.11)$$

Витрати на електроенергію обчислюються через тариф для другого класу напруги ЦЕН = 13,64 грн/(кВт·год), споживану потужність NC = 0,2 кВт і коефіцієнт зайнятості обладнання КЗ = 0,2 (формула 4.12):

$$\text{С}_{\text{ЕЛ}} = \text{Т}_{\text{ЕФ}} \cdot N_{\text{С}} \cdot \text{К}_{\text{З}} \cdot \text{Ц}_{\text{ЕН}} = \\ = 932 \cdot 0{,}2 \cdot 0{,}2 \cdot 13{,}64 = 508{,}57 \text{ грн.} \quad (4.12)$$

Заробітна плата оператора ЕОМ розраховується від річного фонду з коефіцієнтом зайнятості КЗ = 0,2 при окладі М_оп = 22 000 грн:

$$\text{С}_{\text{Г}} = 12 \cdot \text{М}_{\text{оп}} \cdot \text{К}_{\text{З}} = 12 \cdot 22000 \cdot 0{,}2 = 52\,800{,}00 \text{ грн}; \quad (4.13)$$

$$\text{С}_{\text{ЗП}}(\text{ЕОМ}) = \text{С}_{\text{Г}} \cdot \left(1 + \text{К}_{\text{Д}}\right) = 52\,800 \cdot 1{,}2 = 63\,360{,}00 \text{ грн}; \quad (4.14)$$

$$\text{С}_{\text{від}}(\text{ЕОМ}) = \text{С}_{\text{ЗП}}(\text{ЕОМ}) \cdot 0{,}22 = 63\,360 \cdot 0{,}22 = 13\,939{,}20 \text{ грн}. \quad (4.15)$$

Накладні витрати на утримання обладнання обчислюються як 67% від його вартості (формула 4.16):

$$\text{С}_{\text{Н}}(\text{ЕОМ}) = \text{Ц}_{\text{ПР}} \cdot 0{,}67 = 40\,000 \cdot 0{,}67 = 26\,800{,}00 \text{ грн}. \quad (4.16)$$

Сумарні річні експлуатаційні витрати (формула 4.17):

$$\text{С}_{\text{ЕКС}} = \text{С}_{\text{ЗП}}(\text{ЕОМ}) + \text{С}_{\text{від}}(\text{ЕОМ}) + \text{С}_{\text{А}} + \text{С}_{\text{Р}} + \text{С}_{\text{ЕЛ}} + \text{С}_{\text{Н}}(\text{ЕОМ}) = \\ = 118\,407{,}77 \text{ грн}. \quad (4.17)$$

Собівартість однієї машино-години (формула 4.18):

$$\text{С}_{\text{М}-\text{Г}} = \frac{\text{С}_{\text{ЕКС}}}{\text{Т}_{\text{ЕФ}}} = \frac{118\,407{,}77}{932} = 127{,}05 \text{ грн/год}. \quad (4.18)$$

Витрати на машинний час для кожного варіанта (формула 4.19):

$$\text{С}_{\text{М}} = \text{С}_{\text{М}-\text{Г}} \cdot \text{Т}. \quad (4.19)$$

Для Варіанта 1: СМ1 = 127,05 · 800 = 101 640,00 грн. Для Варіанта 2: СМ2 = 127,05 · 750 = 95 287,50 грн.

Накладні витрати розробки приймаються як 67% від заробітної плати виконавців (формула 4.20):

$$\text{С}_{\text{Н}} = \text{С}_{\text{ЗП}} \cdot 0{,}67. \quad (4.20)$$

Для Варіанта 1: СН1 = 242 860,80 · 0,67 = 162 716,74 грн. Для Варіанта 2: СН2 = 227 682,00 · 0,67 = 152 546,94 грн.

Повна вартість розробки програмного продукту обчислюється як сума складових (формула 4.21):

$$С_{ПП} = С_{ЗП} + С_{від} + С_{М} + С_{Н}. \quad (4.21)$$

Підсумкові розрахунки за двома варіантами зведено в таблиці 4.8.

Таблиця 4.8 – Зведений економічний розрахунок варіантів реалізації

Стаття витрат, грн	**Варіант 1**	**Варіант 2**
Заробітна плата виконавців (СЗП)	242 860,80	227 682,00
Єдиний соціальний внесок (Свід)	53 429,38	50 090,04
Витрати на машинний час (СМ)	101 640,00	95 287,50
Накладні витрати розробки (СН)	162 716,74	152 546,94
Повна вартість розробки (СПП)	560 646,92	525 606,48

Повна вартість розробки Варіанта 1 становить 560 646,92 грн, Варіанта 2 — 525 606,48 грн. Різниця у 35 040,44 грн (приблизно 6,7%) формується за рахунок дещо більшої трудомісткості Варіанта 1, обумовленої калібруванням ваг інтегрального показника та експериментальним підбором порогу спрацювання.

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Остаточний вибір кращого варіанта реалізації виконується за коефіцієнтом техніко-економічного рівня (формула 4.22):

$$К_{ТЕР} = \frac{К_{к}}{С_{ПП}}. \quad (4.22)$$

Для Варіанта 1: КТЕР1 = 82,51 / 560 646,92 = 1,472 · 10^{-4}. Для Варіанта 2: КТЕР2 = 57,75 / 525 606,48 = 1,099 · 10^{-4}. Підсумкові техніко-економічні показники обох варіантів зведено в таблиці 4.9.

Таблиця 4.9 – Підсумкові показники варіантів реалізації

Показник	**Варіант 1**	**Варіант 2**
Коефіцієнт технічного рівня Кк	82,51	57,75
Повна вартість розробки СПП, грн	560 646,92	525 606,48
КТЕР	$1{,}472 \cdot 10^{-4}$	$1{,}099 \cdot 10^{-4}$

Коефіцієнт техніко-економічного рівня Варіанта 1 більший за відповідний коефіцієнт Варіанта 2 у 1,34 раза. При помірному зростанні витрат на 6,7% Варіант 1 забезпечує суттєво вищу якість виявлення шахрайства та точнішу логіку прийняття рішення про потребу актуалізації моделі. Тому як оптимальний варіант реалізації обрано Варіант 1: LightGBM як базова модель, KS-тест для виявлення дрейфу, інтегральний показник S_t для агрегації сигналів моніторингу та відносний критерій прийняття кандидата. Цей варіант повністю відповідає реалізації, описаній у розділі 3.

4.8 Висновки до розділу 4

У межах четвертого розділу виконано функціонально-вартісний аналіз інтелектуальної системи оцінки якості та автоматизованої актуалізації моделі для виявлення шахрайських платіжних транзакцій.

На основі морфологічної карти та позитивно-негативної матриці сформовано два варіанти реалізації: побудований на LightGBM, KS-тесті, інтегральному показнику S_t та відносному критерії прийняття кандидата (Варіант 1) і спрощений на логістичній регресії, PSI, правилах за окремими порогами та абсолютному критерії (Варіант 2).

Експертне оцінювання сімома фахівцями виявило найбільш значущим параметром якість базової моделі (Кві = 0,3605). Коефіцієнт узгодженості експертів W = 0,853 перевищує нормативний поріг 0,67, тому ранжування визнане достовірним. Підсумкові ваги отримано після трьох ітерацій попарного порівняння.

За результатами розрахунку коефіцієнт технічного рівня Варіанта 1 становить 82,51, Варіанта 2 — 57,75. Повна вартість розробки Варіанта 1 — 560 646,92 грн, Варіанта 2 — 525 606,48 грн. Коефіцієнт техніко-економічного рівня Варіанта 1 дорівнює $1{,}472 \cdot 10^{-4}$, що в 1,34 раза перевищує відповідний показник Варіанта 2.

Оптимальним за технічними та економічними показниками визнано Варіант 1. Цей варіант обрано як цільову конфігурацію програмного продукту і реалізовано у практичній частині роботи. Отриманий результат підтверджує доцільність прийнятих архітектурних рішень: переваги в якості моделі та узгодженості логіки актуалізації виправдовують помірне збільшення трудомісткості розробки.

ВИСНОВКИ

У дипломній роботі розроблено інтелектуальну систему оцінки якості та автоматизованої актуалізації моделі для виявлення шахрайських платіжних транзакцій у реальному часі. Основна ідея роботи полягала не лише у побудові моделі класифікації транзакцій, а й у створенні механізму, який відстежує зміну її якості на потоці даних і запускає перенавчання у разі виявлення стійкої деградації.

У теоретичній частині розглянуто специфіку задачі виявлення шахрайських транзакцій, проблему незбалансованості класів, часову природу платіжного потоку, затримку в отриманні підтвердженої інформації про результат операції та явище дрейфу даних. Обґрунтовано доцільність використання PR-AUC як однієї з основних метрик якості для задачі з рідкісним позитивним класом, а також необхідність поєднання кількох сигналів моніторингу для прийняття рішення про оновлення моделі.

Для експериментальної перевірки використано набір даних PaySim, який містить 6 362 620 транзакцій за 743 часові кроки. Кількість шахрайських транзакцій становить 8 213, тобто близько 0,129 % від загального обсягу даних. Це підтверджує сильну незбалансованість задачі та пояснює необхідність використання спеціалізованих метрик оцінювання якості. У практичній частині реалізовано модель на основі LightGBM, сервіс прогнозування, симулятор потоку транзакцій, модуль моніторингу та аналітичний компонент для прийняття рішення про актуалізацію моделі.

Запропонований механізм актуалізації ґрунтується на інтегральному показнику стану системи, який враховує зміну якості моделі, дрейф вхідних даних, поведінку прогнозних оцінок і накопичення нових розмічених спостережень. Такий підхід дозволяє не реагувати на випадкові короткочасні

коливання однієї метрики, а оцінювати загальний стан моделі в поточному часовому вікні.

Експериментальна перевірка виконувалася на трьох сценаріях: natural, drift_only та degradation. У кожному сценарії було сформовано 67 моніторингових вікон на інтервалі від step 201 до step 736. У стабільному сценарії natural система не виконала жодного зайвого перенавчання, а середній PR-AUC становив 0,992. У сценарії drift_only було виконано одне перенавчання, після якого PR-AUC кандидатної моделі становив 0,963. У сценарії degradation система двічі виявила потребу в оновленні моделі, а PR-AUC після перенавчання становив 0,947 та 0,847 відповідно.

Отримані результати показали, що розроблена система здатна відрізняти нормальний стан потоку від сценаріїв деградації, своєчасно запускати перенавчання та оновлювати модель без втручання оператора. У стабільному сценарії система не генерувала помилкових спрацювань, а в сценаріях зі штучною зміною даних забезпечувала відновлення якості моделі після її погіршення.

Подальший розвиток роботи доцільно спрямувати на адаптивний підбір ваг інтегрального показника, розширення моніторингу на категоріальні ознаки, безпечне порівняння версій моделі в режимі shadow deployment або A/B-тестування, а також перевірку системи на ширшому наборі сценаріїв із різними типами дрейфу та затримки міток.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. European Central Bank. Payment statistics: first half of 2025. 2026. URL: https://www.ecb.europa.eu/press/stats/paysec/html/ecb.pis2025h1~36edd636c8.en.html (Last accessed: 14.05.2026).

2. Vanini P., Rossi S., Zvizdic E., Domenig T. Online payment fraud: from anomaly detection to risk management. *Financial Innovation*. 2023. Vol. 9. Art. 66. DOI: 10.1186/s40854-023-00470-w.

3. European Banking Authority. Final report on draft regulatory technical standards on strong customer authentication and common and secure communication under PSD2 (Article 98). EBA/RTS/2017/02. London: EBA, 2017. 121 p. URL: https://www.eba.europa.eu/regulation-and-policy/payment-services-and-electronic-money/regulatory-technical-standards-on-strong-customer-authentication-and-secure-communication-under-psd2 (Last accessed: 01.06.2026).

4. Credit card fraud detection and concept-drift adaptation with delayed supervised information / A. Dal Pozzolo et al. 2015 *International Joint Conference on Neural Networks (IJCNN)* : proceedings of the conference, Killarney, Ireland, 12–17 July 2015. Piscataway, NJ : IEEE, 2015. P. 1–8. DOI: 10.1109/IJCNN.2015.7280527.

5. 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.

6. Chen T., Guestrin C. XGBoost: a scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, CA, 2016. P. 785–794. DOI: 10.1145/2939672.2939785.

7. LightGBM: a highly efficient gradient boosting decision tree / G. Ke et al. Advances in Neural Information Processing Systems 30. Long Beach, CA, 2017. P.

3146–3154. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf (Last accessed: 16.05.2026).

8. Liu F. T., Ting K. M., Zhou Z.-H. Isolation forest. *Proceedings of the 8th IEEE International Conference on Data Mining*. Pisa, Italy, 2008. P. 413–422. DOI: 10.1109/ICDM.2008.17.

9. Saito T., Rehmsmeier M. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS ONE*. 2015. Vol. 10, No. 3. Art. e0118432. DOI: 10.1371/journal.pone.0118432.

10. Goodman B., Flaxman S. European Union regulations on algorithmic decision-making and a «right to explanation». *AI Magazine*. 2017. Vol. 38, No. 3. P. 50–57. DOI: 10.1609/aimag.v38i3.2741.

11. Lundberg S. M., Lee S.-I. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems 30*. Long Beach, CA, 2017. P. 4765–4774.

12. Ribeiro M. T., Singh S., Guestrin C. «Why should I trust you?»: explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, CA, 2016. P. 1135–1144. DOI: 10.1145/2939672.2939778.

13. A Survey on Concept Drift Adaptation / J. Gama et al. *ACM Computing Surveys*. 2014. Vol. 46, No. 4. Art. 44. DOI: 10.1145/2523813.

14. Kreuzberger D., Kühl N., Hirschl S. Machine learning operations (MLOps): overview, definition, and architecture. *IEEE Access*. 2023. Vol. 11. P. 31866–31879. DOI: 10.1109/ACCESS.2023.3262138.

15. Hidden technical debt in machine learning systems / D. Sculley et al. *Advances in Neural Information Processing Systems 28*. Montreal, Canada, 2015. P. 2503–2511.

16. NannyML Documentation. Performance estimation without targets. URL: https://nannyml.readthedocs.io/ (Last accessed: 01.06.2026).

17. Evidently AI Documentation. Data and concept drift tests. URL: https://docs.evidentlyai.com/ (Last accessed: 01.06.2026).

18. Feedzai. RiskOps platform for financial fraud prevention: product overview. URL: https://www.feedzai.com/riskops/ (Last accessed: 01.06.2026).

19. Sift. Digital trust and safety platform: product documentation. URL: https://sift.com/platform (Last accessed: 01.06.2026).

20. dos Reis D., Flach P., Matwin S., Batista G. Fast unsupervised online drift detection using incremental Kolmogorov–Smirnov test. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, CA, 2016. P. 1545–1554. DOI: 10.1145/2939672.2939836.

21. Massey F. J. Jr. The Kolmogorov–Smirnov test for goodness of fit. *Journal of the American Statistical Association*. 1951. Vol. 46, No. 253. P. 68–78. DOI: 10.1080/01621459.1951.10500769.

22. Siddiqi N. Intelligent credit scoring: building and implementing better credit risk scorecards. 2nd ed. Hoboken, NJ: John Wiley & Sons, 2017. 464 p.

23. Bifet A., Gavalda R. Learning from time-changing data with adaptive windowing. *Proceedings of the 7th SIAM International Conference on Data Mining*. Minneapolis, MN, 2007. P. 443–448. DOI: 10.1137/1.9781611972771.42.

24. Gama J., Medas P., Castillo G., Rodrigues P. Learning with drift detection. *Advances in Artificial Intelligence – SBIA 2004*. Lecture Notes in Computer Science, Vol. 3171. Berlin: Springer, 2004. P. 286–295. DOI: 10.1007/978-3-540-28645-5_29.

25. Lopez-Rojas E. A., Elmir A., Axelsson S. PaySim: a financial mobile money simulator for fraud detection. *Proceedings of the European Modeling and Simulation Symposium*. Larnaca, Cyprus, 2016. P. 249–255. URL: https://www.msc-les.org/proceedings/emss/2016/EMSS2016_249.pdf (Last accessed: 16.05.2026).

ДОДАТОК А ЛІСТИНГ КОДУ

base/data.py

```
"""PaySim loading and feature preparation.

The dataset has roughly 6.36M transactions over 743 simulated hours
(the `step` column). Fraud rate is about 0.13%, so the problem is
heavily imbalanced. We keep things simple: drop identifier columns,
one-hot the `type` column, add a couple of derived balance-diff
features, and split the data chronologically by `step`.
"""

import pandas as pd


# Columns we feed to the model. `type` is expanded into one-hot
# dummies below, the rest are numeric.
NUMERIC_COLS = [
    "amount",
    "oldbalanceOrg",
    "newbalanceOrig",
    "oldbalanceDest",
    "newbalanceDest",
    "balance_diff_orig",
    "balance_diff_dest",
]
TYPE_VALUES = ["CASH_IN", "CASH_OUT", "DEBIT", "PAYMENT", "TRANSFER"]
TYPE_COLS = [f"type_{t}" for t in TYPE_VALUES]
FEATURE_COLS = NUMERIC_COLS + TYPE_COLS


def load_paysim(path):
    """Read PaySim CSV and drop columns we will not use."""
    df = pd.read_csv(path)
    df = df.drop(columns=["nameOrig", "nameDest", "isFlaggedFraud"])
    return df


def prepare_features(df):
    """Add derived features and one-hot `type`. Returns a new dataframe
    with columns: FEATURE_COLS + ["isFraud", "step"]."""
    df = df.copy()
    df["balance_diff_orig"] = df["oldbalanceOrg"] - df["newbalanceOrig"]
    df["balance_diff_dest"] = df["newbalanceDest"] - df["oldbalanceDest"]

    # Force all five type columns to exist even if a window misses one.
    dummies = pd.get_dummies(df["type"], prefix="type")
    for col in TYPE_COLS:
        if col not in dummies.columns:
            dummies[col] = 0
    dummies = dummies[TYPE_COLS].astype(int)
```

```
    extra_cols = ["isFraud", "step"]
    if "phase" in df.columns:
        extra_cols.append("phase")
    out = pd.concat([df[NUMERIC_COLS], dummies, df[extra_cols]], axis=1)
    return out


def train_stream_split(df, train_until_step=200):
    """Chronological split. Steps 1..train_until_step go to training,
    everything after goes to the streaming part of the experiment."""
    train = df[df["step"] <= train_until_step].reset_index(drop=True)
    stream = df[df["step"] > train_until_step].reset_index(drop=True)
    return train, stream
```

base/model.py

```
import sys
from pathlib import Path

import numpy as np
from sklearn.metrics import average_precision_score

VENDOR_DIR = Path(__file__).resolve().parents[1] / ".vendor"
if VENDOR_DIR.exists():
    sys.path.append(str(VENDOR_DIR))

from lightgbm import LGBMClassifier

from data import FEATURE_COLS


def train_model(X, y):
    """Train the fraud classifier. X is a dataframe with FEATURE_COLS."""
    y = y.astype(int)
    pos = int(y.sum())
    neg = max(len(y) - pos, 1)
    scale_pos_weight = neg / max(pos, 1)

    model = LGBMClassifier(
        objective="binary",
        n_estimators=200,
        learning_rate=0.05,
        num_leaves=31,
        subsample=0.8,
        colsample_bytree=0.8,
        random_state=42,
        scale_pos_weight=scale_pos_weight,
        n_jobs=1,
        verbose=-1,
    )
    model.fit(X[FEATURE_COLS], y)
    return model


def score(model, X):
    """Predicted probability of fraud."""
    return model.predict_proba(X[FEATURE_COLS])[:, 1]
```

```
def evaluate(model, X, y):
    """PR-AUC and recall at the top 1% scored transactions. PR-AUC is the
    primary metric for this kind of heavy class imbalance."""
    probs = score(model, X)
    pr_auc = average_precision_score(y, probs)

    # Recall@top-1%: of all frauds, how many are in the top 1% by score.
    k = max(1, int(len(probs) * 0.01))
    top_idx = np.argsort(probs)[-k:]
    recall_top1 = float(y.iloc[top_idx].sum()) / max(1, int(y.sum()))

    return {"pr_auc": float(pr_auc), "recall_top1pct": float(recall_top1)}
```

base/pipeline.py

```
import json
from pathlib import Path

import joblib
import numpy as np
import pandas as pd

from program.data import (
    FEATURE_COLS,
    NUMERIC_COLS,
    prepare_features,
    train_stream_split,
)
from program.model import evaluate, score, train_model
from program.monitoring import (
    THRESHOLD,
    WEIGHTS,
    confidence_signal,
    data_readiness_signal,
    drift_signal,
    integral_score,
    quality_signal,
)


# Window size in PaySim "steps" (one step = one simulated hour).
# 24 hours = one day per window.
WINDOW_STEPS = 24

# How many windows of delay before we get the labels for a window
# (models the lag between a transaction and its chargeback).
LABEL_DELAY = 2

# Train a candidate only if at least this many newly labelled
# transactions have accumulated, regardless of S_t.
MIN_LABELS_FOR_RETRAIN = 1000


def _window_bounds(stream_df):
    """Yield (idx, step_start, step_end) for each window in the stream."""
```

```
    step_min = int(stream_df["step"].min())
    step_max = int(stream_df["step"].max())
    idx = 0
    s = step_min
    while s <= step_max:
        yield idx, s, s + WINDOW_STEPS - 1
        idx += 1
        s += WINDOW_STEPS


def run_experiment(df, output_dir):
    output_dir = Path(output_dir)
    models_dir = output_dir / "models"
    models_dir.mkdir(parents=True, exist_ok=True)

    features = prepare_features(df)
    train, stream = train_stream_split(features, train_until_step=200)

    # Hold out the last 20% of train as the reference window.
    split_at = int(len(train) * 0.8)
    train_fit = train.iloc[:split_at].reset_index(drop=True)
    train_ref = train.iloc[split_at:].reset_index(drop=True)

    # ------- initial model -------
    current_model = train_model(train_fit, train_fit["isFraud"])
    model_version = 0
    joblib.dump(current_model, models_dir / f"model_v{model_version}.joblib")

    ref_metrics = evaluate(current_model, train_ref, train_ref["isFraud"])
    pr_auc_reference = ref_metrics["pr_auc"]
    reference_scores = score(current_model, train_ref)
    # Drift reference can move over time (it is replaced after every
    # successful retrain). Quality / score references move together.
    drift_reference = train_ref
    current_pr_auc = pr_auc_reference  # best estimate we have right now

    print(f"[init] model v0 PR-AUC on reference = {pr_auc_reference:.4f}, "
          f"recall@1% = {ref_metrics['recall_top1pct']:.4f}")

    # ------- streaming loop -------
    monitoring_rows = []
    retraining_rows = []

    # Buffer of windows whose labels are not yet released.
    pending = []
    # All labelled stream windows we have seen so far (collected as a
    # list of dataframes and concatenated lazily before retraining).
    labeled_windows = []
    new_labeled_since_retrain = 0

    for w_idx, s_start, s_end in _window_bounds(stream):
        window = stream[(stream["step"] >= s_start) & (stream["step"] <= s_end)]
        if len(window) == 0:
            continue
        window = window.reset_index(drop=True)

        scores = score(current_model, window)
```

```
        pending.append((w_idx, window))

        # Release labels of the window that is now LABEL_DELAY old.
        released_quality = None
        if len(pending) > LABEL_DELAY:
            old_idx, old_window = pending.pop(0)
            released_metrics = evaluate(
                current_model, old_window, old_window["isFraud"]
            )
            released_quality = released_metrics["pr_auc"]
            current_pr_auc = released_quality  # most recent measurable quality
            labeled_windows.append(old_window)
            new_labeled_since_retrain += len(old_window)

        # Compute signals.
        d_quality = quality_signal(released_quality, pr_auc_reference)
        d_drift = drift_signal(window, drift_reference, NUMERIC_COLS)
        d_conf = confidence_signal(scores, reference_scores)
        d_data = data_readiness_signal(new_labeled_since_retrain)
        s_t = integral_score({
            "quality": d_quality, "drift": d_drift,
            "conf": d_conf, "data": d_data,
        })

        window_phase = (window["phase"].mode().iat[0]
                        if "phase" in window.columns and len(window) else "stream")

        monitoring_rows.append({
            "window_idx": w_idx,
            "step_start": s_start,
            "step_end": s_end,
            "phase": window_phase,
            "n_tx": len(window),
            "score_mean": float(np.mean(scores)),
            "d_quality": d_quality,
            "d_drift": d_drift,
            "d_conf": d_conf,
            "d_data": d_data,
            "s_t": s_t,
            "released_pr_auc": released_quality if released_quality is not None else
np.nan,
            "model_version": model_version,
        })

        # ------- retraining decision -------
        if s_t >= THRESHOLD and new_labeled_since_retrain >= MIN_LABELS_FOR_RETRAIN:
            labeled_buffer = pd.concat(labeled_windows, ignore_index=True)
            combined = pd.concat([train_fit, labeled_buffer], ignore_index=True)
            combined["isFraud"] = combined["isFraud"].astype(int)
            candidate = train_model(combined, combined["isFraud"])

            # Validate the candidate on the most recently released window.
            val_window = labeled_windows[-1].copy()
            val_window["isFraud"] = val_window["isFraud"].astype(int)
            cand_metrics = evaluate(candidate, val_window, val_window["isFraud"])
            cur_metrics = evaluate(current_model, val_window, val_window["isFraud"])
```

```
            decision = "accepted" if cand_metrics["pr_auc"] >= 0.98 *
cur_metrics["pr_auc"] else "rejected"
            new_version = model_version + 1 if decision == "accepted" else
model_version

            retraining_rows.append({
                "window_idx": w_idx,
                "s_t": s_t,
                "candidate_pr_auc": cand_metrics["pr_auc"],
                "current_pr_auc": cur_metrics["pr_auc"],
                "decision": decision,
                "new_model_version": new_version,
            })
            print(f"[window {w_idx}] S_t={s_t:.3f} -> retrain candidate "
                  f"PR-AUC {cand_metrics['pr_auc']:.4f} vs current
{cur_metrics['pr_auc']:.4f} "
                  f"({decision})")

            if decision == "accepted":
                current_model = candidate
                model_version = new_version
                joblib.dump(current_model, models_dir /
f"model_v{model_version}.joblib")
                # Refresh references so future signals are computed
                # against the new model's behaviour and against the new
                # input distribution it was trained on.
                drift_reference = labeled_buffer.tail(min(len(labeled_buffer),
20000)).reset_index(drop=True)
                reference_scores = score(current_model, drift_reference)
                new_ref_metrics = evaluate(current_model, drift_reference,
drift_reference["isFraud"])
                pr_auc_reference = new_ref_metrics["pr_auc"]
                current_pr_auc = pr_auc_reference

            new_labeled_since_retrain = 0

    # ------- save artifacts -------
    monitoring_df = pd.DataFrame(monitoring_rows)
    retraining_df = pd.DataFrame(retraining_rows)
    monitoring_df.to_csv(output_dir / "monitoring_log.csv", index=False)
    retraining_df.to_csv(output_dir / "retraining_events.csv", index=False)

    summary = {
        "n_windows": int(len(monitoring_df)),
        "n_retraining_events": int(len(retraining_df)),
        "n_accepted": int((retraining_df["decision"] == "accepted").sum()) if
len(retraining_df) else 0,
        "mean_pr_auc_released":
float(monitoring_df["released_pr_auc"].mean(skipna=True)) if len(monitoring_df) else
None,
        "initial_pr_auc_reference": pr_auc_reference,
        "weights": WEIGHTS,
        "threshold": THRESHOLD,
        "window_steps": WINDOW_STEPS,
        "label_delay": LABEL_DELAY,
    }

    if "phase" in monitoring_df.columns:
```

```
        phase_summary = {}
        for ph, sub in monitoring_df.groupby("phase"):
            phase_summary[ph] = {
                "n_windows": int(len(sub)),
                "mean_s_t": float(sub["s_t"].mean()),
                "mean_d_quality": float(sub["d_quality"].mean()),
                "mean_d_drift": float(sub["d_drift"].mean()),
                "mean_d_conf": float(sub["d_conf"].mean()),
                "mean_pr_auc_released":
float(sub["released_pr_auc"].mean(skipna=True)),
                "n_retrains_triggered": int(
                    (retraining_df["window_idx"].isin(sub["window_idx"])).sum()
                ) if len(retraining_df) else 0,
            }
        summary["per_phase"] = phase_summary

    with open(output_dir / "summary.json", "w") as f:
        json.dump(summary, f, indent=2)

    print(f"[done] windows={summary['n_windows']}
retrains={summary['n_retraining_events']} "
          f"accepted={summary['n_accepted']}")
    print(f"[done] artifacts written to {output_dir}")
```

base/monitoring.py

```
import numpy as np
from scipy.stats import ks_2samp


WEIGHTS = {
    "quality": 0.40,
    "drift":   0.30,
    "conf":    0.15,
    "data":    0.15,
}
THRESHOLD = 0.40

# Number of newly labelled transactions considered enough to retrain on.
DATA_TARGET = 2000

UNCERTAINTY_LOW = 0.4
UNCERTAINTY_HIGH = 0.6


def quality_signal(pr_auc_current, pr_auc_reference):
    if pr_auc_current is None or pr_auc_reference is None or pr_auc_reference <= 0:
        return 0.0
    drop = 1.0 - pr_auc_current / pr_auc_reference
    return float(np.clip(drop, 0.0, 1.0))


def drift_signal(window_df, reference_df, numeric_cols):
    """Average KS statistic across numeric features."""
    if len(window_df) == 0:
        return 0.0
    stats = []
```

```
    for col in numeric_cols:
        stat, _ = ks_2samp(reference_df[col].values, window_df[col].values)
        stats.append(stat)
    return float(np.clip(np.mean(stats), 0.0, 1.0))


def confidence_signal(scores_current, scores_reference):
    if len(scores_current) == 0 or len(scores_reference) == 0:
        return 0.0
    current_share = np.mean(
        (scores_current >= UNCERTAINTY_LOW) & (scores_current <= UNCERTAINTY_HIGH)
    )
    reference_share = np.mean(
        (scores_reference >= UNCERTAINTY_LOW) & (scores_reference <=
UNCERTAINTY_HIGH)
    )
    return float(np.clip(abs(current_share - reference_share), 0.0, 1.0))


def data_readiness_signal(n_new_labeled, target=DATA_TARGET):
    """Fraction of the labelled-data buffer that is filled."""
    return float(min(n_new_labeled / target, 1.0))


def integral_score(signals, weights=None):
    w = weights or WEIGHTS
    return float(
        w["quality"] * signals["quality"]
        + w["drift"] * signals["drift"]
        + w["conf"] * signals["conf"]
        + w["data"] * signals["data"]
    )
```

base/demo_scenario.py

```
import numpy as np


NORMAL_PHASE_END = 350
DRIFT_PHASE_END = 550

AMOUNT_MULTIPLIER = 4.0          # used by `degradation` (combined with fraud
injection)
DRIFT_ONLY_MULTIPLIER = 8.0      # used by `drift_only` — stronger shift so D_drift
alone is enough to cross τ
PAYMENT_FRAUD_RATE = 0.03        # fraction of PAYMENT transactions flipped in
`degradation`

MONEY_COLS = [
    "amount",
    "oldbalanceOrg",
    "newbalanceOrig",
    "oldbalanceDest",
    "newbalanceDest",
]
```

```
def _add_phases(df):
    """Add a three-phase label used in scripted experiments."""
    df = df.copy()

    df["phase"] = "train"
    df.loc[(df["step"] > 200) & (df["step"] <= NORMAL_PHASE_END), "phase"] = "normal"
    df.loc[(df["step"] > NORMAL_PHASE_END) & (df["step"] <= DRIFT_PHASE_END),
"phase"] = "drift"
    df.loc[df["step"] > DRIFT_PHASE_END, "phase"] = "recovery"
    return df


def _apply_amount_shift(df, multiplier=AMOUNT_MULTIPLIER):
    shifted = df["phase"].isin(["drift", "recovery"])
    df.loc[shifted, MONEY_COLS] = df.loc[shifted, MONEY_COLS] * multiplier
    return df


def _inject_payment_fraud(df, seed=0):
    rng = np.random.default_rng(seed)
    shifted = df["phase"].isin(["drift", "recovery"])
    payment_shifted = shifted & (df["type"] == "PAYMENT")
    n_payment = int(payment_shifted.sum())
    n_to_flip = int(n_payment * PAYMENT_FRAUD_RATE)
    if n_to_flip > 0:
        pool = df.loc[payment_shifted].nlargest(n_to_flip * 3, "amount").index
        chosen = rng.choice(pool, size=n_to_flip, replace=False)
        df.loc[chosen, "isFraud"] = 1
    return df


def build_scenario(df, mode="degradation", seed=0):
    df = _add_phases(df)

    if mode == "natural":
        return df
    if mode == "drift_only":
        return _apply_amount_shift(df, multiplier=DRIFT_ONLY_MULTIPLIER)
    if mode == "degradation":
        df = _apply_amount_shift(df, multiplier=AMOUNT_MULTIPLIER)
        df = _inject_payment_fraud(df, seed=seed)
        return df

    raise ValueError(f"Unknown scenario mode: {mode}")
```

services/api/main.py

```
import json
import os
import sys
from datetime import datetime
from pathlib import Path

import joblib
import pandas as pd
from fastapi import FastAPI
```

```
from pydantic import BaseModel

sys.path.insert(0, "/app")
from data import FEATURE_COLS, prepare_features

MODEL_DIR = Path(os.getenv("MODEL_DIR", "/shared/models"))
LOG_DIR = Path(os.getenv("LOG_DIR", "/shared/logs"))
LOG_DIR.mkdir(parents=True, exist_ok=True)
EVENTS_LOG = LOG_DIR / "events.jsonl"


class State:
    model = None
    version = 0
    path = None


def latest_model_path():
    files = sorted(MODEL_DIR.glob("model_v*.joblib"),
                   key=lambda p: int(p.stem.split("v")[-1]))
    if not files:
        raise FileNotFoundError(f"no model_v*.joblib in {MODEL_DIR}")
    return files[-1]


def load_model():
    p = latest_model_path()
    State.model = joblib.load(p)
    State.version = int(p.stem.split("v")[-1])
    State.path = str(p)
    print(f"[api] loaded {p.name} (v{State.version})", flush=True)


class TxRequest(BaseModel):
    step: int
    type: str
    amount: float
    oldbalanceOrg: float
    newbalanceOrig: float
    oldbalanceDest: float
    newbalanceDest: float
    isFraud: int = 0
    phase: str = "stream"


app = FastAPI()
load_model()


@app.get("/health")
def health():
    return {"status": "ok", "model_version": State.version, "model_path": State.path}


@app.post("/predict")
def predict(req: TxRequest):
    df = pd.DataFrame([req.dict()])
```

```
    X = prepare_features(df)
    score = float(State.model.predict_proba(X[FEATURE_COLS])[:, 1][0])
    event = {
        "ts": datetime.utcnow().isoformat(),
        "step": req.step,
        "phase": req.phase,
        "features": {c: float(X[c].iloc[0]) for c in FEATURE_COLS},
        "isFraud": int(req.isFraud),
        "score": score,
        "model_version": State.version,
    }
    with open(EVENTS_LOG, "a") as f:
        f.write(json.dumps(event) + "\n")
    return {"score": score, "model_version": State.version}


@app.post("/reload")
def reload_model():
    load_model()
    return {"status": "reloaded", "model_version": State.version}
```

services/api/Dockerfile

```
FROM python:3.11-slim

WORKDIR /app

RUN apt-get update && apt-get install -y --no-install-recommends \
    libgomp1 \
 && rm -rf /var/lib/apt/lists/*

COPY base/ /app/
COPY services/api/requirements.txt /app/api-requirements.txt
RUN pip install --no-cache-dir -r /app/api-requirements.txt

COPY services/api/main.py /app/main.py

EXPOSE 8000
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

services/monitor/main.py

```
"""Monitor service: reads events log, computes signals per window, decides
retrain."""
import json
import os
import sys
import time
from pathlib import Path

import joblib
import numpy as np
import pandas as pd
import requests
from sklearn.metrics import average_precision_score

sys.path.insert(0, "/app")
```

```
from data import FEATURE_COLS, NUMERIC_COLS
from model import train_model, evaluate, score as model_score
from monitoring import (
    WEIGHTS, THRESHOLD,
    quality_signal, drift_signal, confidence_signal, data_readiness_signal,
    integral_score,
)

API_URL = os.getenv("API_URL", "http://api:8000")
SHARED = Path("/shared")
DATA_DIR = SHARED / "data"
MODEL_DIR = SHARED / "models"
LOG_DIR = SHARED / "logs"
LOG_DIR.mkdir(parents=True, exist_ok=True)

EVENTS_LOG = LOG_DIR / "events.jsonl"
MONITORING_CSV = LOG_DIR / "monitoring.csv"
RETRAIN_CSV = LOG_DIR / "retrain_events.csv"

WINDOW_STEPS = int(os.getenv("WINDOW_STEPS", "24"))
LABEL_DELAY = int(os.getenv("LABEL_DELAY", "2"))
MIN_LABELS_FOR_RETRAIN = int(os.getenv("MIN_LABELS_FOR_RETRAIN", "1000"))
POLL_INTERVAL = float(os.getenv("POLL_INTERVAL", "2.0"))


def init_csv(path, headers):
    if not path.exists():
        pd.DataFrame(columns=headers).to_csv(path, index=False)


def append_csv(path, row):
    pd.DataFrame([row]).to_csv(path, mode="a", header=False, index=False)


def wait_for_api():
    for _ in range(60):
        try:
            r = requests.get(f"{API_URL}/health", timeout=1.5)
            if r.status_code == 200:
                return
        except Exception:
            pass
        time.sleep(1)


def load_reference():
    ref_df = pd.read_csv(DATA_DIR / "reference_window.csv")
    ref_scores = np.load(DATA_DIR / "reference_scores.npy")
    with open(DATA_DIR / "reference_metrics.json") as f:
        ref_metrics = json.load(f)
    train_fit = pd.read_csv(DATA_DIR / "train_fit.csv")
    return ref_df, ref_scores, ref_metrics["pr_auc"], train_fit


def read_events():
    if not EVENTS_LOG.exists():
        return pd.DataFrame()
```

```
    rows = []
    with open(EVENTS_LOG) as f:
        for line in f:
            line = line.strip()
            if not line:
                continue
            try:
                rec = json.loads(line)
            except Exception:
                continue
            r = dict(rec["features"])
            r["step"] = rec["step"]
            r["phase"] = rec["phase"]
            r["isFraud"] = rec["isFraud"]
            r["score"] = rec["score"]
            r["model_version"] = rec["model_version"]
            rows.append(r)
    return pd.DataFrame(rows)


def current_model_path():
    files = sorted(MODEL_DIR.glob("model_v*.joblib"),
                   key=lambda p: int(p.stem.split("v")[-1]))
    return files[-1]


def main():
    print(f"[mon] WINDOW_STEPS={WINDOW_STEPS} LABEL_DELAY={LABEL_DELAY} "
          f"MIN_LABELS={MIN_LABELS_FOR_RETRAIN}", flush=True)

    wait_for_api()

    init_csv(MONITORING_CSV,
             ["window_idx", "step_start", "step_end", "phase", "n_tx",
              "d_quality", "d_drift", "d_conf", "d_data", "s_t",
              "released_pr_auc", "model_version"])
    init_csv(RETRAIN_CSV,
             ["window_idx", "s_t", "candidate_pr_auc", "current_pr_auc",
              "decision", "new_model_version"])

    drift_reference, reference_scores, pr_auc_reference, train_fit = load_reference()
    print(f"[mon] reference loaded: ref rows={len(drift_reference)}, "
          f"baseline PR-AUC={pr_auc_reference:.4f}", flush=True)

    processed = set()              # window_idx that have been emitted
    pending = []                   # [(w_idx, window_df), ...]
    labeled_windows = []
    new_labels_since_retrain = 0

    win_iter = 0
    while True:
        events = read_events()
        if len(events) == 0:
            time.sleep(POLL_INTERVAL)
            continue

        step_min = int(events["step"].min())
```

```
        step_max = int(events["step"].max())

        # window starts anchored to step_min, step grid = WINDOW_STEPS
        s = step_min
        w_idx = 0
        while s <= step_max:
            s_start, s_end = s, s + WINDOW_STEPS - 1
            if w_idx not in processed:
                window = events[(events["step"] >= s_start) & (events["step"] <=
s_end)]
                # only emit when simulator has moved past this window
                if len(window) > 0 and step_max > s_end + 1:
                    processed.add(w_idx)

                    scores_w = window["score"].values
                    pending.append((w_idx, window.copy()))

                    released_q = None
                    if len(pending) > LABEL_DELAY:
                        old_idx, old_w = pending.pop(0)
                        if old_w["isFraud"].sum() > 0:
                            released_q = float(average_precision_score(
                                old_w["isFraud"].values, old_w["score"].values))
                        labeled_windows.append(old_w)
                        new_labels_since_retrain += len(old_w)

                    d_q = quality_signal(released_q, pr_auc_reference) if released_q
is not None else 0.0
                    d_d = drift_signal(window, drift_reference, NUMERIC_COLS)
                    d_c = confidence_signal(scores_w, reference_scores)
                    d_data_v = data_readiness_signal(new_labels_since_retrain)
                    s_t = integral_score({"quality": d_q, "drift": d_d,
                                          "conf": d_c, "data": d_data_v})

                    phase_v = window["phase"].mode().iat[0] if len(window) else
"stream"
                    mv = int(window["model_version"].iloc[-1])

                    append_csv(MONITORING_CSV, {
                        "window_idx": w_idx, "step_start": s_start, "step_end":
s_end,
                        "phase": phase_v, "n_tx": len(window),
                        "d_quality": round(d_q, 6), "d_drift": round(d_d, 6),
                        "d_conf": round(d_c, 6), "d_data": round(d_data_v, 6),
                        "s_t": round(s_t, 6),
                        "released_pr_auc": (round(released_q, 6) if released_q is not
None else ""),
                        "model_version": mv,
                    })
                    pr_str = f"{released_q:.4f}" if released_q is not None else "n/a"
                    print(f"[mon] w{w_idx} step={s_start}-{s_end} phase={phase_v} "
                          f"n={len(window)} S_t={s_t:.3f} PR-AUC={pr_str}",
flush=True)

                    if s_t >= THRESHOLD and new_labels_since_retrain >=
MIN_LABELS_FOR_RETRAIN:
                        print(f"[mon] retrain triggered at window {w_idx}
(S_t={s_t:.3f})", flush=True)
```

```
                        buf = pd.concat(labeled_windows, ignore_index=True)
                        combined = pd.concat([train_fit, buf], ignore_index=True)
                        combined["isFraud"] = combined["isFraud"].astype(int)
                        candidate = train_model(combined, combined["isFraud"])

                        val = labeled_windows[-1].copy()
                        val["isFraud"] = val["isFraud"].astype(int)
                        cand_m = evaluate(candidate, val, val["isFraud"])
                        cur_path = current_model_path()
                        cur_model = joblib.load(cur_path)
                        cur_m = evaluate(cur_model, val, val["isFraud"])

                        decision = ("accepted" if cand_m["pr_auc"] >= 0.98 *
cur_m["pr_auc"]
                                    else "rejected")
                        new_v = int(cur_path.stem.split("v")[-1]) + (1 if decision ==
"accepted" else 0)
                        append_csv(RETRAIN_CSV, {
                            "window_idx": w_idx, "s_t": round(s_t, 6),
                            "candidate_pr_auc": round(cand_m["pr_auc"], 6),
                            "current_pr_auc": round(cur_m["pr_auc"], 6),
                            "decision": decision, "new_model_version": new_v,
                        })
                        print(f"[mon] candidate PR-AUC={cand_m['pr_auc']:.4f} "
                              f"vs current {cur_m['pr_auc']:.4f} -> {decision}",
flush=True)

                        if decision == "accepted":
                            new_path = MODEL_DIR / f"model_v{new_v}.joblib"
                            joblib.dump(candidate, new_path)
                            try:
                                requests.post(f"{API_URL}/reload", timeout=5)
                                print(f"[mon] api reloaded to v{new_v}", flush=True)
                            except Exception as e:
                                print(f"[mon] api reload failed: {e}", flush=True)
                            drift_reference = buf.tail(min(len(buf),
20000)).reset_index(drop=True)
                            reference_scores = model_score(candidate,
drift_reference)
                            new_ref_m = evaluate(candidate, drift_reference,
                                                 drift_reference["isFraud"])
                            pr_auc_reference = new_ref_m["pr_auc"]

                        new_labels_since_retrain = 0
                s += WINDOW_STEPS
                w_idx += 1

        time.sleep(POLL_INTERVAL)


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

services/monitor/Dockerfile

```
FROM python:3.11-slim
```

```
WORKDIR /app

RUN apt-get update && apt-get install -y --no-install-recommends \
    libgomp1 \
 && rm -rf /var/lib/apt/lists/*

COPY base/ /app/
COPY services/monitor/requirements.txt /app/mon-requirements.txt
RUN pip install --no-cache-dir -r /app/mon-requirements.txt

COPY services/monitor/main.py /app/main.py

CMD ["python", "-u", "main.py"]
```

services/simulator/main.py

```
"""Replay PaySim subset by step, send each row to api /predict at controlled rate."""
import os
import sys
import time
from pathlib import Path

import pandas as pd
import requests

sys.path.insert(0, "/app")
from demo_scenario import build_scenario

API_URL = os.getenv("API_URL", "http://api:8000")
DATA_PATH = Path(os.getenv("DATA_PATH", "/shared/data/paysim_subset.csv"))
SCENARIO = os.getenv("SCENARIO", "degradation")
RATE_TPS = float(os.getenv("RATE_TPS", "100"))
QUICK = os.getenv("QUICK", "0") == "1"
SEED = int(os.getenv("SEED", "0"))
QUICK_FRACTION = float(os.getenv("QUICK_FRACTION", "0.10"))


def wait_for_api(retries=60):
    for _ in range(retries):
        try:
            r = requests.get(f"{API_URL}/health", timeout=1.5)
            if r.status_code == 200:
                return
        except Exception:
            pass
        time.sleep(1)
    raise RuntimeError(f"api at {API_URL} did not become ready")


def main():
    print(f"[sim] waiting for api at {API_URL} ...", flush=True)
    wait_for_api()

    print(f"[sim] reading {DATA_PATH}", flush=True)
    df = pd.read_csv(DATA_PATH)
    df = build_scenario(df, mode=SCENARIO, seed=SEED)
```

```
    # streaming portion only (training was step <= 200)
    df = df[df["step"] > 200].reset_index(drop=True)

    if QUICK:
        df = (
            df.groupby("step", group_keys=False)
              .apply(lambda g: g.sample(frac=QUICK_FRACTION, random_state=SEED))
              .sort_values("step")
              .reset_index(drop=True)
        )
        print(f"[sim] QUICK mode: {len(df)} rows ({QUICK_FRACTION*100:.0f}% per
step)", flush=True)
    else:
        print(f"[sim] FULL mode: {len(df)} rows", flush=True)

    print(f"[sim] scenario={SCENARIO}, rate={RATE_TPS} tps", flush=True)
    interval = 1.0 / RATE_TPS if RATE_TPS > 0 else 0.0

    sent, errors = 0, 0
    t0 = time.time()
    last_step = -1
    for _, row in df.iterrows():
        payload = {
            "step": int(row["step"]),
            "type": str(row["type"]),
            "amount": float(row["amount"]),
            "oldbalanceOrg": float(row["oldbalanceOrg"]),
            "newbalanceOrig": float(row["newbalanceOrig"]),
            "oldbalanceDest": float(row["oldbalanceDest"]),
            "newbalanceDest": float(row["newbalanceDest"]),
            "isFraud": int(row["isFraud"]),
            "phase": str(row.get("phase", "stream")),
        }
        try:
            requests.post(f"{API_URL}/predict", json=payload, timeout=3)
            sent += 1
        except Exception as e:
            errors += 1
            if errors <= 5:
                print(f"[sim] err: {e}", flush=True)

        s = int(row["step"])
        if s != last_step:
            print(f"[sim] step={s} sent={sent} errors={errors} "
                  f"elapsed={time.time()-t0:.1f}s", flush=True)
            last_step = s

        if interval:
            time.sleep(interval)

    print(f"[sim] DONE: sent={sent} errors={errors} elapsed={time.time()-t0:.1f}s",
          flush=True)


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

services/simulator/Dockerfile

```
FROM python:3.11-slim

WORKDIR /app

COPY base/demo_scenario.py /app/
COPY services/simulator/requirements.txt /app/sim-requirements.txt
RUN pip install --no-cache-dir -r /app/sim-requirements.txt

COPY services/simulator/main.py /app/main.py

CMD ["python", "-u", "main.py"]
```

services/dashboard/main.py

```
"""Streamlit dashboard. Reads shared/logs/monitoring.csv + retrain_events.csv."""
from pathlib import Path

import pandas as pd
import plotly.graph_objects as go
import streamlit as st
from streamlit_autorefresh import st_autorefresh

SHARED = Path("/shared")
LOG_DIR = SHARED / "logs"
MONITORING_CSV = LOG_DIR / "monitoring.csv"
RETRAIN_CSV = LOG_DIR / "retrain_events.csv"

# Light cool palette — readable on soft gray-blue background, not harsh.
COLORS = {
    "s_t": "#2563eb",
    "threshold": "#64748b",
    "retrain": "#7c3aed",
    "d_quality": "#0284c7",
    "d_drift": "#4f46e5",
    "d_conf": "#0891b2",
    "d_data": "#475569",
    "pr_auc": "#0d9488",
    "grid": "#e2e8f0",
    "text": "#334155",
    "muted": "#64748b",
}

PLOT_LAYOUT = dict(
    paper_bgcolor="rgba(0,0,0,0)",
    plot_bgcolor="#ffffff",
    font=dict(family="Inter, system-ui, sans-serif", color=COLORS["text"], size=13),
    title=dict(font=dict(size=16, color="#0f172a")),
    margin=dict(l=48, r=24, t=48, b=40),
    hoverlabel=dict(bgcolor="#ffffff", font_color=COLORS["text"],
bordercolor=COLORS["grid"]),
    xaxis=dict(
        showgrid=True,
        gridcolor=COLORS["grid"],
        linecolor=COLORS["grid"],
        tickfont=dict(color=COLORS["muted"]),
        title_font=dict(color=COLORS["text"]),
```

```
    ),
    yaxis=dict(
        showgrid=True,
        gridcolor=COLORS["grid"],
        linecolor=COLORS["grid"],
        tickfont=dict(color=COLORS["muted"]),
        title_font=dict(color=COLORS["text"]),
    ),
    legend=dict(
        bgcolor="rgba(255,255,255,0.85)",
        bordercolor=COLORS["grid"],
        borderwidth=1,
        font=dict(color=COLORS["text"]),
    ),
)

# CUSTOM_CSS = (CSS styles for dashboard layout — opotted)

st.set_page_config(page_title="Fraud Monitoring", layout="wide", page_icon="")
st.markdown(CUSTOM_CSS, unsafe_allow_html=True)
st.title("Моніторинг fraud-моделі")
st.caption("Сигнали моніторингу, інтегральний показник S_t, події перенавчання")

refresh_sec = st.sidebar.slider("Refresh (сек)", 1, 30, 3)
st_autorefresh(interval=refresh_sec * 1000, key="refresh")

if not MONITORING_CSV.exists():
    st.info("Очікую на дані від monitor-сервісу ...")
    st.stop()

mon = pd.read_csv(MONITORING_CSV)
ret = pd.read_csv(RETRAIN_CSV) if RETRAIN_CSV.exists() else pd.DataFrame()

if len(mon) == 0:
    st.info("Поки нема жодного вікна.")
    st.stop()

last = mon.iloc[-1]
status = "RETRAIN" if last["s_t"] >= 0.4 else "OK"
status_class = "status-retrain" if status == "RETRAIN" else "status-ok"

c1, c2, c3, c4 = st.columns(4)
c1.metric("Вікон оброблено", len(mon))
c2.metric("Retrain подій", len(ret))
c3.metric("Поточна model_version", int(last["model_version"]))
with c4:
    st.markdown(
        f'<div class="status-card">'
        f'<p class="status-label">Статус</p>'
        f'<p class="status-value {status_class}">{status}</p>'
        f"</div>",
        unsafe_allow_html=True,
    )


def plot_layout(**overrides):
    layout = dict(PLOT_LAYOUT)
```

```
    layout.update(overrides)
    return layout


def add_retrain_lines(fig, ret):
    for _, e in ret.iterrows():
        fig.add_vline(
            x=int(e["window_idx"]),
            line_color=COLORS["retrain"],
            line_width=2,
            line_dash="dot",
            annotation_text=f"retrain → v{int(e['new_model_version'])}",
            annotation_position="top",
            annotation=dict(font=dict(color=COLORS["retrain"], size=11)),
        )


# 1. S_t timeline
fig1 = go.Figure()
fig1.add_scatter(
    x=mon["window_idx"],
    y=mon["s_t"],
    mode="lines+markers",
    name="S_t",
    line=dict(color=COLORS["s_t"], width=2.5),
    marker=dict(size=6, color=COLORS["s_t"], line=dict(width=1, color="#ffffff")),
)
fig1.add_hline(
    y=0.4,
    line_dash="dash",
    line_color=COLORS["threshold"],
    annotation_text="threshold τ=0.4",
    annotation_position="bottom right",
    annotation=dict(font=dict(color=COLORS["muted"], size=11)),
)
add_retrain_lines(fig1, ret)
fig1.update_layout(
    **plot_layout(
        title="Інтегральний показник S_t",
        height=320,
        xaxis_title="window_idx",
        yaxis_title="S_t",
        yaxis_range=[0, max(0.6, mon["s_t"].max() * 1.1)],
    )
)
st.plotly_chart(fig1, use_container_width=True)

# 2. signals breakdown
fig2 = go.Figure()
for col, color in (
    ("d_quality", COLORS["d_quality"]),
    ("d_drift", COLORS["d_drift"]),
    ("d_conf", COLORS["d_conf"]),
    ("d_data", COLORS["d_data"]),
):
    fig2.add_scatter(x=mon["window_idx"], y=mon[col], mode="lines", name=col,
line=dict(color=color, width=2))
fig2.update_layout(
```

```
    **plot_layout(
        title="Сигнали моніторингу",
        height=320,
        xaxis_title="window_idx",
        yaxis_title="value",
    )
)
st.plotly_chart(fig2, use_container_width=True)

# 3. PR-AUC over time
pr = mon[mon["released_pr_auc"].astype(str) != ""].copy()
if len(pr) > 0:
    pr["released_pr_auc"] = pr["released_pr_auc"].astype(float)
    fig3 = go.Figure()
    fig3.add_scatter(
        x=pr["window_idx"],
        y=pr["released_pr_auc"],
        mode="lines+markers",
        name="PR-AUC",
        line=dict(color=COLORS["pr_auc"], width=2.5),
        marker=dict(size=6, color=COLORS["pr_auc"], line=dict(width=1,
color="#ffffff")),
    )
    add_retrain_lines(fig3, ret)
    fig3.update_layout(
        **plot_layout(
            title="PR-AUC на розмічених вікнах",
            height=300,
            xaxis_title="window_idx",
            yaxis_title="PR-AUC",
            yaxis_range=[0, 1.05],
        )
    )
    st.plotly_chart(fig3, use_container_width=True)

if len(ret) > 0:
    st.subheader("Retrain events")
    st.dataframe(ret, use_container_width=True)

with st.expander("Сирі дані monitoring.csv"):
    st.dataframe(mon.tail(50), use_container_width=True)
```

services/dashboard/Dockerfile

```
FROM python:3.11-slim

WORKDIR /app

COPY services/dashboard/requirements.txt /app/dash-requirements.txt
RUN pip install --no-cache-dir -r /app/dash-requirements.txt

COPY services/dashboard/main.py /app/main.py
COPY services/dashboard/.streamlit /app/.streamlit

EXPOSE 8501
CMD ["streamlit", "run", "main.py", \
     "--server.address=0.0.0.0", "--server.port=8501", \
     "--server.headless=true", "--browser.gatherUsageStats=false"]
```

docker-compose.yml

```
services:
  api:
    build:
      context: .
      dockerfile: services/api/Dockerfile
    container_name: mon_api
    ports:
      - "8000:8000"
    volumes:
      - ./shared:/shared
    environment:
      MODEL_DIR: /shared/models
      LOG_DIR: /shared/logs

  monitor:
    build:
      context: .
      dockerfile: services/monitor/Dockerfile
    container_name: mon_monitor
    depends_on:
      - api
    volumes:
      - ./shared:/shared
    environment:
      API_URL: http://api:8000
      WINDOW_STEPS: ${WINDOW_STEPS:-24}
      LABEL_DELAY: ${LABEL_DELAY:-2}
      MIN_LABELS_FOR_RETRAIN: ${MIN_LABELS_FOR_RETRAIN:-1000}
      POLL_INTERVAL: ${POLL_INTERVAL:-2.0}

  simulator:
    build:
      context: .
      dockerfile: services/simulator/Dockerfile
    container_name: mon_sim
    depends_on:
      - api
    volumes:
      - ./shared:/shared
    environment:
      API_URL: http://api:8000
      DATA_PATH: /shared/data/paysim_subset.csv
      SCENARIO: ${SCENARIO:-degradation}
      RATE_TPS: ${RATE_TPS:-100}
      QUICK: ${QUICK:-0}
      QUICK_FRACTION: ${QUICK_FRACTION:-0.10}
      SEED: ${SEED:-0}

  dashboard:
    build:
      context: .
      dockerfile: services/dashboard/Dockerfile
    container_name: mon_dash
    ports:
      - "8501:8501"
```

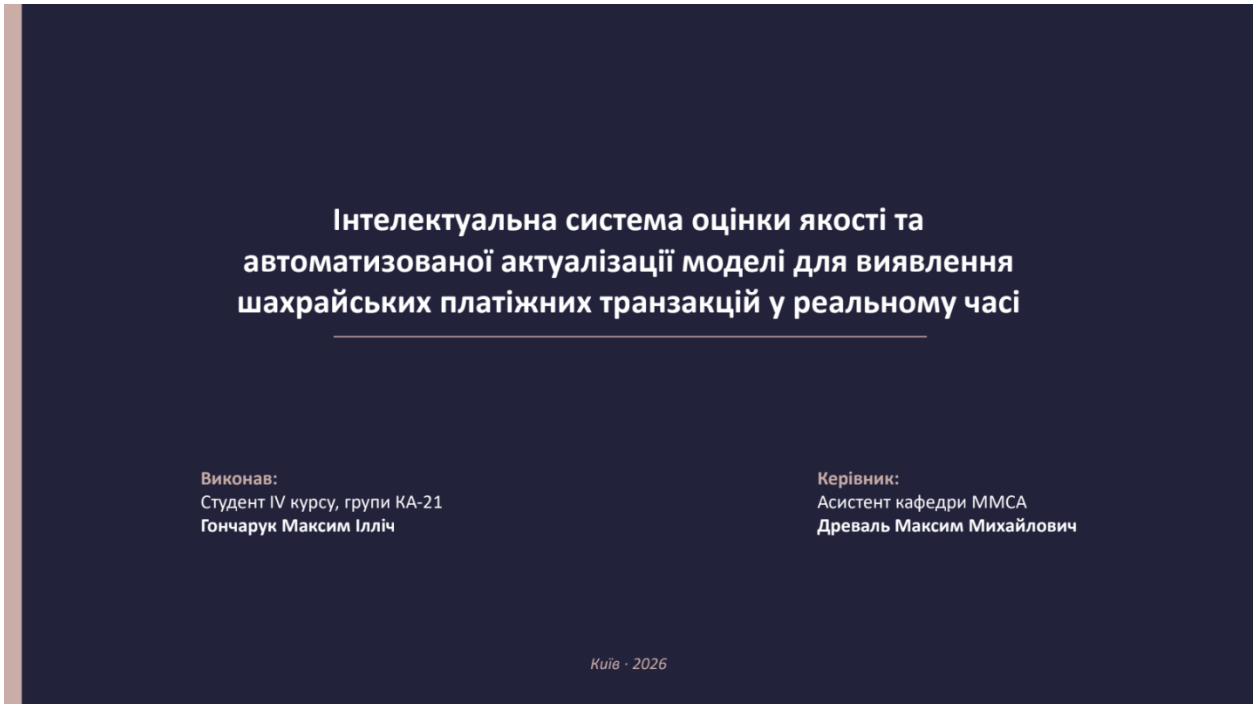
```
    volumes:
      - ./shared:/shared
```

Makefile

```
SHELL := /bin/bash

ifeq ($(shell docker compose version >/dev/null 2>&1 && echo ok),ok)
  COMPOSE := docker compose
else
  COMPOSE := docker-compose
endif

TS := $(shell date +%Y%m%d_%H%M%S)

# Shared "fast demo" params for ~30s wall time.
FAST_ENV := QUICK=1 QUICK_FRACTION=0.10 WINDOW_STEPS=8 LABEL_DELAY=1 \
            MIN_LABELS_FOR_RETRAIN=100 POLL_INTERVAL=1.5

.PHONY: help build prepare-shared demo-natural demo-drift demo-degradation \
        quick archive reset logs stop clean

help:
	@echo "Compose: $(COMPOSE)"
	@echo ""
	@echo "Targets (all use the fast demo profile):"
	@echo "  build                 build all images"
	@echo "  prepare-shared        copy data/ and models/ into shared/"
	@echo "  demo-natural          run scenario natural        (~30s)"
	@echo "  demo-drift            run scenario drift_only     (~30s)"
	@echo "  demo-degradation      run scenario degradation    (~30s)"
	@echo "  quick                 ultra-fast (2x), degradation (~15s)"
	@echo "  archive RUN=<name>    copy shared/logs/* to outputs/<name>_<ts>/"
	@echo "  reset                 wipe shared/logs/ and model_v1+ joblibs"
	@echo "  logs                  tail compose logs"
	@echo "  stop                  docker compose down"
	@echo "  clean                 stop + wipe logs + extra models"
	@echo ""
	@echo "After 'make quick' or 'make demo-*': open http://localhost:8501"

prepare-shared:
	@mkdir -p shared/data shared/models shared/logs
	@cp -n data/* shared/data/ 2>/dev/null || true
	@cp -n models/model_v0.joblib shared/models/ 2>/dev/null || true

build: prepare-shared
	$(COMPOSE) build

demo-natural: prepare-shared
	SCENARIO=natural RATE_TPS=500 $(FAST_ENV) $(COMPOSE) up

demo-drift: prepare-shared
	SCENARIO=drift_only RATE_TPS=500 $(FAST_ENV) $(COMPOSE) up

demo-degradation: prepare-shared
	SCENARIO=degradation RATE_TPS=500 $(FAST_ENV) $(COMPOSE) up

quick: prepare-shared
```

```
	SCENARIO=degradation RATE_TPS=1000 $(FAST_ENV) $(COMPOSE) up

archive:
	@if [ -z "$(RUN)" ]; then echo "Usage: make archive RUN=<name>"; exit 1; fi
	@DST=outputs/$(RUN)_$(TS); mkdir -p $$DST; \
	cp shared/logs/*.csv $$DST/ 2>/dev/null || true; \
	cp shared/logs/events.jsonl $$DST/ 2>/dev/null || true; \
	echo "archived → $$DST"

reset:
	@rm -f shared/logs/*.csv shared/logs/events.jsonl
	@find shared/models -name "model_v*.joblib" ! -name "model_v0.joblib" -delete
2>/dev/null || true
	@echo "reset done"

logs:
	$(COMPOSE) logs -f

stop:
	$(COMPOSE) down

clean: stop reset
```

ДОДАТОК Б ПРЕЗЕНТАЦІЯ



Об'єкт, предмет, мета роботи

Об'єкт	Процес автоматичного виявлення шахрайських платіжних транзакцій у режимі потоку даних.
Предмет	Методи моніторингу якості та автоматичного оновлення моделей класифікації за умов зміни характеристик вхідних даних і відкладеного отримання підтверджених міток.
Мета	Розробити інтелектуальну програмну систему, що класифікує транзакції в реальному часі, відстежує зміну якості моделі та запускає її перенавчання у разі стабільного погіршення показників.

3

Основні завдання роботи

1. Проаналізувати предметну область і ризики шахрайства у платіжному потоці.
2. Формалізувати задачу моніторингу якості моделі та автоматичного перенавчання на потоці з відкладеними мітками.
3. Обрати метрики якості та сигнали виявлення дрейфу даних, концепту й розподілу скорів.
4. **Розробити інтегральний показник S_t та правила запуску оновлення моделі.**
5. Реалізувати мікросервісну систему з прогнозуванням, моніторингом і автоматичним перенавчанням.
6. Перевірити роботу системи на трьох демонстраційних сценаріях: natural, drift_only, degradation.

4

Предметна область

Рідкісний позитивний клас
Частка fraud у платіжному потоці становить близько 0,1%, тому precision і recall є чутливими до дисбалансу класів. ROC-AUC у такому випадку може давати надто оптимістичну оцінку, тому основною метрикою обрано PR-AUC.

Затримка міток
Інформація про шахрайство надходить із затримкою через chargeback або скарги користувачів. Через це якість моделі можна коректно оцінювати лише після появи підтверджених історичних даних.

Дрейф даних і концепту
З часом змінюються як розподіл ознак, так і зв'язок між ознаками та класом. Тому модель, навчена на історичних даних, поступово втрачає актуальність і потребує моніторингу.

Типи концептного дрейфу

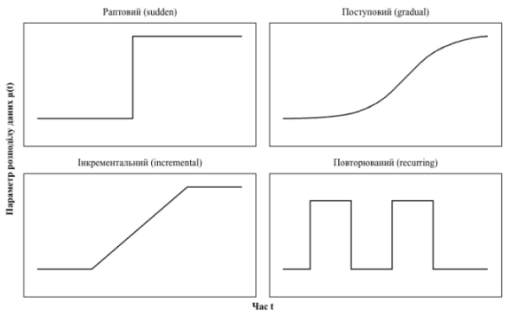


Рисунок: типи концептного дрейфу за динамікою [13, 14]

5

Вхідні дані

6,36 млн	743	8 213	0,129 %
транзакцій	часові кроки (step)	fraud-транзакцій	частка fraud

Набір даних PaySim

Тип: синтетичний набір даних, що моделює мобільні платіжні транзакції на основі реального журналу подій від африканського мобільного оператора.

Структура: 11 базових ознак, 5 типів операцій (PAYMENT, TRANSFER, CASH_OUT, CASH_IN, DEBIT), мітка isFraud.

Особливість: сильна незбалансованість класів — менше 0,13 % fraud, що визначає вибір PR-AUC як основної метрики.

Поле step: часовий індекс у годинах, дозволяє моделювати потоковий характер даних і реалістичну затримку міток.

Джерело: Lopez-Rojas E. A., Elmir A., Axelsson S. PaySim — a financial mobile money simulator for fraud detection. EMSS 2016. Bordeaux, France.

6

Методи та підходи

Метод / показник	Призначення та обґрунтування вибору
LightGBM	Базова модель класифікації — швидкий gradient boosting, добре працює з табличними ознаками й незбалансованими класами.
PR-AUC	Основна метрика якості — більш інформативна за ROC-AUC при дуже рідкісному позитивному класі.
KS-тест	Виявлення дрейфу ознак — сигнал D_drift. Чутливі до зміни розподілу числових ознак між референсним і поточним вікном.
Ковзне вікно	Оцінка якості D_quality на актуальному відрізку потоку з урахуванням затримки появи міток.
D_conf, D_data	Сигнали дрейфу розподілу скорів і накопичення нових розмічених спостережень як умови запуску перенавчання.
Інтегральний S_t	Зважена згортка сигналів з порогом τ — **комбінований критерій**, що мінімізує помилкові спрацювання.

7

Інтегральний показник S_t та правило запуску

$$S_t = w_1 \cdot D_{quality} + w_2 \cdot D_{drift} + w_3 \cdot D_{conf} + w_4 \cdot D_{data}$$

$D_{quality}$ - падіння PR-AUC поточної моделі на актуальному вікні

D_{drift} - PSI / KS-тест за ключовими числовими ознаками

D_{conf} - зміна розподілу прогнозних скорів моделі

D_{data} - обсяг нових міток у часовому вікні

Ваги: $w_1 = 0{,}40 \cdot w_2 = 0{,}30 \cdot w_3 = 0{,}15 \cdot w_4 = 0{,}15$
Поріг: $\tau = 0{,}40$ (тригер перенавчання)
Захист від помилкових: перенавчання лише за умови ≥ 100 нових міток у вікні

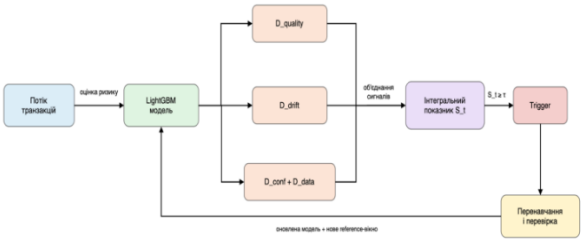


Рисунок 3.2 — Логічна схема прийняття рішення про оновлення моделі

9

Реалізація

Мова програмування Python	**ML** LightGBM · scikit-learn	**API сервісу прогнозу** FastAPI
Аналітика Streamlit · matplotlib	**Оркестрація сервісів** Docker	**Зберігання артефактів** Спільний том /shared (models, logs)

Потік даних між сервісами

1. Симулятор → POST /predict сервісу прогнозування → запис у events.jsonl.
2. Сервіс моніторингу читає events.jsonl, рахує S_t і записує monitoring.csv.
3. При $S_t > \tau$ і достатньому обсязі міток — навчання кандидата, валідація на свіжому вікні.
4. Якщо PR-AUC кандидата ≥ 0,98 від поточного — збереження моделі і виклик /reload.

9

Архітектура системи

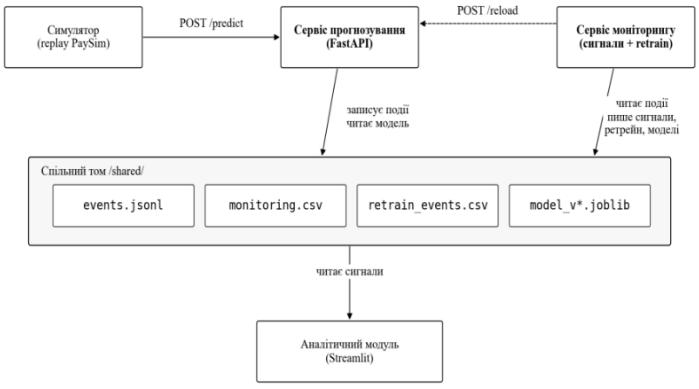


Рисунок 3.1 — Архітектура мікросервісної реалізації

Сервіс прогнозування

FastAPI · POST /predict, POST /reload. Завантажує активну модель з /shared/models.

Симулятор потоку

Підвибірка PaySim · 500 tps · 3 сценарії: natural / drift_only / degradation.

Моніторинг та актуалізація

Обчислює S_t на ковзних вікнах, запускає перенавчання й валідацію кандидата.

Аналітичний модуль

Streamlit-дашборд: журнали моніторингу, події перенавчання, графіки PR-AUC і S_t.

8

Демонстраційний сценарій перевірки

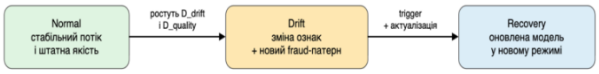


Рисунок 3.4 — Логіка фаз normal → drift → recovery

natural	Без збурень. Очікувано: 0 перенавчань.
drift_only	Множник грошових ознак ×8 у фазі drift.
degradation	Множник ×4 + 3 % нових PAYMENT-fraud.

Параметри прогону

Параметр	Значення
WINDOW_STEPS	8 значень step
LABEL_DELAY	1 вікно
MIN_LABELS_FOR_RETRAIN	100
Поріг τ	0,40
Ваги w_1, w_2, w_3, w_4	0,40 / 0,30 / 0,15 / 0,15
RATE_TPS	500 запитів/с
QUICK_FRACTION	10 %
Моніторингових вікон	67 (step 201–736)

11

Результати експериментальної перевірки

Сценарій	Вікон	Retrain	Mean S_t	Max S_t	Mean PR-AUC	PR-AUC after retrain
natural	67	0	0,200	0,306	0,992	—
drift_only	67	1	0,273	0,419	0,936	0,963
degradation	67	2	0,222	0,422	0,899	0,947 → 0,847

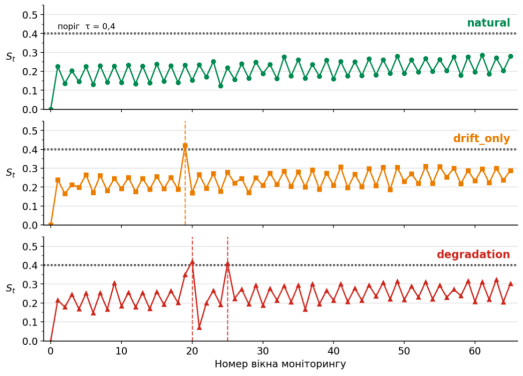


Рисунок 3.5 — Інтегральний показник S_t у часі для трьох сценаріїв

Ключове

natural: жодного хибного спрацювання, S_t нижче порогу.

drift_only: одне перенавчання на вікні 19; PR-AUC відновлено до 0,963.

degradation: два послідовних перенавчання (вікна 20 і 25) — адаптація до тривалого збурення.

12

Порівняння якості: до і після перенавчання

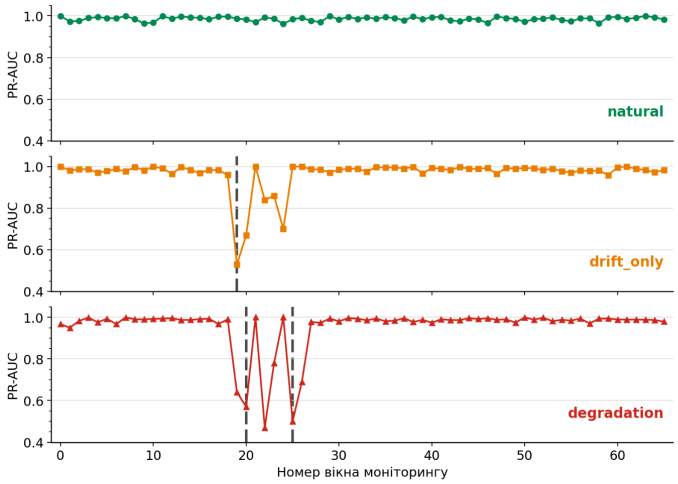


Рисунок 3.6 — PR-AUC моделі у часі для трьох сценаріїв

Спостереження

Базовий стан (natural)
PR-AUC тримається на рівні 0,96–1,0 без стійких провалів — модель стабільна.

Поява збурення (drift)
У сценаріях **drift_only** і **degradation** якість різко падає до 0,47–0,69 у вікнах 19–25, що й фіксує монітор.

Реакція системи (перенавчання)
Після перенавчання (вертикальні лінії) PR-AUC повертається до > 0,93 уже за 1–2 вікна.

Висновок
Система відрізняє природний шум від реальної деградації та вчасно відновлює якість, не реагуючи на сценарій **natural**.

Висновки та практична цінність

Висновки

- Мету роботи досягнуто: реалізовано систему класифікації + моніторингу + автоматичного перенавчання.
- Інтегральний показник стану системи комбінує чотири сигнали і мінімізує помилкові спрацювання.
- На трьох сценаріях підтверджено своєчасну реакцію на деградацію та відновлення PR-AUC вище 0,93.

Практична цінність

- Готовий мікросервісний шаблон для антифрод-команд PSP і банків.
- Розділення моделі та механізму моніторингу — заміна моделі без зміни решти системи.
- Зменшення часу реакції на деградацію без участі спеціаліста.

Подальший розвиток

- Адаптивний підбір вагових коефіцієнтів інтегрального показника.
- Розширення моніторингу на категоріальні ознаки.
- Shadow deployment та A/B-тестування кандидатних моделей.
- Ширша палітра типів дрейфу та затримки міток.

Дякую за увагу!