

**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: «Методи машинного навчання та нейронних мереж для
виявлення шахрайства у банківських операціях»**

Виконала: студентка IV курсу, групи КА–22,
Кокунова Анастасія Юріївна _____

Керівник: професор, д.т.н.,
Недашківська Надія Іванівна _____

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

Консультант з нормоконтролю: асистент, доктор філософії,
Канцедал Георгій Олегович _____

Рецензент: доцент кафедри системного проектування НН ІПСА, к.т.н.,
Безносик Олександр Юрійович _____

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

Студентка _____

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 студентці
Кокуновій Анастасії Юріївні

1. Тема роботи «Методи машинного навчання та нейронних мереж для виявлення шахрайства у банківських операціях», керівник роботи доц., д.т.н. Недашківська Надія Іванівна, затверджені наказом по університету від 27.05.2026 р. №1944–с
2. Термін подання студентом роботи «10» червня 2026 р.
3. Вихідні дані до роботи: набір даних Bank Account Fraud (BAF)
4. Зміст роботи: огляд предметної області, методи бінарної класифікації та виявлення аномалій, метрики оцінювання результатів моделей, побудова моделей та аналіз результатів, функціонально–вартісний аналіз програмного продукту.
5. Перелік ілюстративного матеріалу: моделі та методи виявлення аномалій, ознаки набору даних, описова статистика набору даних, найкращі гіперпараметри для моделей, показники якості оцінки класифікації, SHAP аналіз для найкращої моделі.

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 10.04.2026

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Формулювання теми дослідження	10.04.2026	Виконано
2	Вивчення літератури за темою роботи	15.04.2026	Виконано
3	Написання першого розділу	25.04.2026	Виконано
4	Написання другого розділу	6.05.2026	Виконано
5	Розробка програмного продукту та аналіз результатів	13.05.2026	Виконано
6	Написання третього розділу	17.05.2026	Виконано
7	Написання економічної частини	29.05.2026	Виконано
8	Підготовка презентації до захисту	03.06.2026	Виконано
9	Оформлення роботи відповідно до нормоконтролю	06.06.2026	Виконано
10	Попередній захист дипломної роботи	10.06.2026	Виконано

Студентка

Анастасія КОКУНОВА

Керівник

Надія НЕДАШКІВСЬКА

РЕФЕРАТ

Дипломна робота містить: 106 с., 18 рис., 2 додатки, 13 табл., 25 джерел.

КЛАСИФІКАЦІЯ, МАШИННЕ НАВЧАННЯ, НЕЙРОННА МЕРЕЖА, ВИЯВЛЕННЯ АНОМАЛІЙ, БАНКІВСЬКЕ ШАХРАЙСТВО, БІНАРНА КЛАСИФІКАЦІЯ, НЕЗБАЛАНСОВАНІ ДАНІ, АНСАМБЛЕВІ МЕТОДИ, ДИНАМІЧНИЙ ВИБІР ПОРОГУ.

У роботі розглянуто сучасні підходи до виявлення банківського шахрайства на етапі реєстрації нових клієнтів. Досліджено роботу алгоритмів класифікації в умовах суттєвого дисбалансу класів на практичній задачі ідентифікації шахрайства безпосередньо на етапі обробки заявок на відкриття банківських рахунків.

Метою дослідження є розробка та порівняльний аналіз моделей машинного навчання для ефективної та точної класифікації шахрайських заявок на відкриття банківських рахунків.

Об'єктом дослідження – це процес ідентифікації шахрайських дій (аномалій) у банківських операціях на етапі реєстрації нових клієнтів.

Предметом дослідження є математичні методи та моделі машинного навчання (зокрема Random Forest, SVM, MLP, XGBoost, Isolation Forest) для розв'язання задачі класифікації банківського шахрайства на основі багатовимірних поведінкових ознак.

Програмною реалізацією було обрано мову Python у середовищі Kaggle.

Результатом роботи є навчені та проаналізовані предиктивні моделі.

ABSTRACT

Bachelor thesis: 106 pages, 18 figures, 13 tables, 2 appendices, 25 references.

CLASSIFICATION, MACHINE LEARNING, NEURAL NETWORK, ANOMALY DETECTION, BANK FRAUD, BINARY CLASSIFICATION, UNBALANCED DATA, ENSEMBLE METHODS, DYNAMIC THRESHOLD SELECTION.

The paper examines modern approaches to detecting banking fraud at the stage of new customer registration. The work of classification algorithms under conditions of significant class imbalance is investigated on the practical problem of fraud identification directly at the stage of processing applications for opening bank accounts.

The aim of the research is to develop and benchmark machine learning models for efficient and accurate classification of fraudulent bank account opening applications.

The object of the study is the process of identifying fraudulent actions (anomalies) in banking transactions at the stage of registering new clients.

The subject of the research is mathematical methods and machine learning models (in particular, Random Forest, SVM, MLP, XGBoost, Isolation Forest) for solving the problem of classifying banking fraud based on multidimensional behavioral features.

The Python language in the Kaggle environment was chosen as the software implementation.

The result of the work is trained and analyzed predictive models.

ЗМІСТ

ВСТУП	9
РОЗДІЛ 1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ	10
1.1 Сучасний стан та тенденції банківського шахрайства.....	10
1.1.1 Індустріалізація шахрайства	10
1.1.2 Синтетичні ідентичності	11
1.1.3 Застосування генеративного штучного інтелекту	11
1.1.4 Обмеження традиційних систем моніторингу	12
1.2 Основи визначення та терміни.....	12
1.3. Машинне навчання в системах виявлення фроду	13
1.4 Постановка задачі.....	16
1.5 Висновки до розділу 1	17
РОЗДІЛ 2. МЕТОДИ І МОДЕЛІ БІНАРНОЇ КЛАСИФІКАЦІЇ ТА ВИЯВЛЕННЯ АНОМАЛІЙ, МЕТРИКИ ОЦІНЮВАННЯ РЕЗУЛЬТАТІВ МОДЕЛЕЙ	19
2.1 Моделі	19
2.1.1 Random Forest	19
2.1.2 SVM з RBF ядром.....	22
2.1.3 XGBoost.....	25
2.1.4 MLP.....	28
2.1.5 Isolation Forest.....	31
2.2 Метрики оцінки якості класифікації	34
2.2.1 Confusion Matrix	34

	7
2.2.2 Precision–Recall Area Under Curve (PR AUC).....	35
2.2.3 F1–score	36
2.2.4 Matthews Correlation Coefficient (MCC)	36
2.2.5 Accuracy.....	37
2.3 Висновки до розділу 2	38
РОЗДІЛ 3. ПРОЕКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ, ПОБУДОВА МОДЕЛЕЙ ТА АНАЛІЗ РЕЗУЛЬТАТІВ	39
3.1 Архітектура інтелектуальної системи.....	39
3.2 Опис використаних програмних засобів	41
3.2.1 Python.....	41
3.2.2 Краудсорсингова платформа Kaggle	43
3.3 Набір даних Bank Account Fraud (BAF) Tabular Dataset Suite	44
3.4 Навчання та оцінка моделей	50
3.4.1 Розбиття даних та крос–валідація	50
3.4.2 Попередня обробка ознак.....	51
3.4.3 Специфіка навчання моделей	51
3.4.4 Оцінка якості та динамічний вибір порогу	53
3.5. Аналіз результатів.....	53
3.6 Висновки до розділу 3	64
РОЗДІЛ 4 ФУНКЦІОНАЛЬНО–ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ	65
4.1 Постановка задачі проектування	65
4.2 Обґрунтування функцій програмного продукту.....	66
4.3 Обґрунтування системи параметрів програмного продукту	69
4.4 Аналіз експертного оцінювання параметрів	72

	8
4.5 Аналіз рівня якості варіантів реалізації функцій.....	75
4.6 Економічний аналіз варіантів розробки ПП.....	77
4.7 Вибір кращого варіанту ПП за техніко–економічним рівнем.....	80
4.8 Висновки до розділу 4	81
ВИСНОВКИ.....	82
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	84
ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ	88
ДОДАТОК В. ГРАФІЧНИЙ МАТЕРІАЛ ДО ЗАХИСТУ	98

ВСТУП

В умовах стрімкої цифровізації фінансового сектору проблема шахрайства в банківських операціях набуває критичного значення, завдаючи суттєвих економічних та репутаційних збитків установам. Одним із найвразливіших етапів взаємодії клієнта з банком є процес відкриття нових рахунків, адже успішна шахрайська реєстрація відкриває зловмисникам шлях до подальших нелегальних фінансових транзакцій. Для ефективної протидії цим загрозам на ранніх етапах традиційних методів контролю вже недостатньо, що зумовлює необхідність впровадження інтелектуальних систем аналізу даних.

У цій роботі досліджено сучасні моделі бінарної класифікації та виявлення аномалій, роботу яких алгоритмічно протестовано на прикладній задачі ідентифікації шахрайства безпосередньо на етапі обробки заявок на відкриття банківських рахунків.

Об'єктом дослідження – це процес ідентифікації шахрайських дій (аномалій) у банківських операціях на етапі реєстрації нових клієнтів.

Предметом дослідження є математичні методи та моделі машинного навчання (зокрема Random Forest, SVM, MLP, XGBoost, Isolation Forest) для розв'язання задачі класифікації банківського шахрайства на основі багатовимірних поведінкових ознак.

У комплексі застосування цих інструментів спрямоване на забезпечення високої точності детекції шахрайських дій, мінімізуючи хибні спрацьовування та підвищуючи загальну надійність банківської системи.

РОЗДІЛ 1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Сучасний стан та тенденції банківського шахрайства

В умовах стрімкої цифровізації фінансового сектору проблема шахрайства в банківських операціях набула масштабу глобальної загрози, завдаючи суттєвих економічних та репутаційних збитків установам. Глобальний перехід до онлайн-банкінгу, розвиток відкритих фінансових екосистем (Open Banking) та впровадження інфраструктури миттєвих платежів суттєво розширили площу атак для кіберзлочинців. Завдяки цим змінам, швидкість транзакцій зросла, що пропорційно скоротило часове вікно, доступне банкам для виявлення та блокування підозрілих операцій.

Сьогодні одним із найвразливіших етапів взаємодії клієнта з банком є процес відкриття нових рахунків. Успішна шахрайська реєстрація на етапі цифрового онбордингу відкриває зловмисникам легітимізований шлях до цілого спектра подальших нелегальних дій: оформлення беззаставних кредитів, формування мереж транзитних рахунків-мулів для відмивання коштів та обходу санкційних обмежень.

1.1.1 Індустріалізація шахрайства

Трансформація фінансового шахрайства перейшла від ізольованих атак окремих хакерів до складної індустріалізованої екосистеми. У тіньовому сегменті інтернету сформувався повноцінний ринок послуг, де зловмисники можуть купувати готові інструменти для автоматизації атак, бази скомпрометованих персональних даних та ботнети для масової подачі заявок на відкриття рахунків [1]. Це значно знижує технічний поріг входу для нових злочинців та експоненціально збільшує кількість спроб фроду.

1.1.2 Синтетичні ідентичності

Особливу загрозу на етапі створення рахунків становить використання синтетичних ідентичностей. Це специфічна злочинна практика, за якої реальні дані існуючих осіб, наприклад, викрадені номери соціального страхування чи податкові коди індивідів, які рідко перевіряють свою кредитну історію, компілюються з вигаданою інформацією (адресами, іменами, номерами телефонів) [2].

На відміну від класичної крадіжки особистості, коли жертва швидко помічає підозрілу активність, синтетичні профілі здатні роками уникати виявлення. Зловмисники поступово “виращують” кредитний рейтинг таких синтетичних клієнтів, здійснюючи дрібні легітимні транзакції, після чого реалізують масштабну атаку з виведення кредитних коштів.

1.1.3 Застосування генеративного штучного інтелекту

Зловмисники масово імплементують великі мовні моделі LLM та інструменти генеративного штучного інтелекту для масштабування атак, що проявляється у наступних напрямках:

- 1) створення високоякісних копій паспортів, водійських посвідчень та довідок про доходи, які візуально не відрізняються від оригіналів;
- 2) використання технологій дипфейків для генерації синтетичних облич та голосів дозволяє шахраям успішно проходити процедури “Liveness detection” під час відеоідентифікації клієнта;
- 3) створення персоналізованих та лінгвістично бездоганих фішингових сценаріїв для виманювання у реальних клієнтів їхніх автентифікаційних даних для подальшого перехоплення контролю над рахунком (Account Takeover).

1.1.4 Обмеження традиційних систем моніторингу

Тривалий час фінансовий сектор покладався на експертні системи виявлення шахрайства, функціонування яких базувалося на жорстких детермінованих правилах (Rule-based systems). Проте в умовах сучасних цифрових тенденцій такі підходи виявляють низку критичних обмежень. Одним із ключових недоліків є високий рівень хибнопозитивних спрацьовувань (false positives): статичні алгоритми неминуче блокують значну частку легітимних транзакцій, що суттєво погіршує клієнтський досвід та генерує додаткові операційні витрати на проведення ручної верифікації [3].

Традиційні системи здатні ідентифікувати виключно відомі патерни зловживань, залишаючись вразливими та неефективними перед новітніми, нестандартними векторами атак. Зрештою, на тлі стрімкого зростання обсягів даних та ускладнення нелінійних взаємозв'язків між атрибутами клієнтів (такими як поведінкові маркери, геолокаційні дані, цифрові відбитки пристроїв), ручне калібрування та масштабування систем правил втрачає свою аналітичну ефективність.

1.2 Основи визначення та терміни

Для коректного формування теоретичного та практичного базису роботи необхідно формалізувати ключові терміни предметної області:

- 1) банківське шахрайство (Фрод) – сукупність навмисних протиправних дій, спрямованих на нелегітимне заволодіння фінансовими активами, продуктами або конфіденційними даними банку чи його клієнтів із використанням обману, зловживання довірою або технологічних маніпуляцій;
- 2) шахрайство при відкритті рахунку (Application Fraud) – тип фінансового злочину, що полягає в дистанційному або фізичному поданні заявки на

- отримання банківських послуг із використанням викрадених, підроблених або синтетичних персональних даних;
- 3) бінарна класифікація – задача керованого машинного навчання, математична мета якої полягає в розподілі об'єктів дослідження виключно на дві взаємовиключні категорії;
 - 4) незбалансована вибірка – набір даних, у якому класифікаційні категорії представлені вкрай непропорційно;
 - 5) ознаки (Features) – структурована багатовимірна сукупність кількісних та якісних характеристик об'єкта (наприклад, вік, фінансовий стан, технічні дані пристрою, поведінкові патерни), які аналізуються моделлю для прогнозування цільової змінної;
 - 6) аномалія – спостереження в наборі даних, яке статистично значущо відхиляється від очікуваної нормальної поведінки генеральної сукупності, виступаючи маркером потенційного ризику.

1.3. Машинне навчання в системах виявлення фроду

Вирішенням зазначених алгоритмічних та операційних проблем став концептуальний перехід фінансових установ до використання методів машинного навчання. На відміну від жорстко запрограмованої експертної логіки, парадигма машинного навчання базується на здатності алгоритмів самостійно та емпірично видобувати знання з надвеликих масивів історичних даних. Цей підхід дозволяє неявно виявляти приховані, багатовимірні та суттєво нелінійні закономірності, які принципово неможливо формалізувати за допомогою традиційних каскадів інструкцій розгалуження. Застосування інтелектуальних систем аналізу даних докорінно трансформує архітектуру банківського комплаєнсу та ризик-менеджменту, зміщуючи фокус із реактивного реагування на предиктивну аналітику.

Однією з фундаментальних переваг ML-алгоритмів є їхня проактивність та висока здатність до узагальнення. Замість реагування на ізольовані індикатори небезпеки, математичні моделі комплексно оцінюють загальний контекст поведінкового профілю клієнта. Це наділяє системи здатністю ідентифікувати аномальні відхилення від еталонної поведінки генеральної сукупності. Важливість цього аспекту підкреслюється у дослідженнях О.Р. Хомич та Н.І. Недашківської [8], які зазначають, що виявлення аномалій методами інтелектуального аналізу даних дозволяє ефективно фіксувати нетипові патерни серед масивів транзакційної інформації без опори на заздалегідь відомі правила. Саме такий підхід є критично важливим для виявлення атак нульового дня [4], оскільки система реагує на сам факт математичного відхилення від норми.

Таким чином банківська установа отримує можливість ідентифікувати новітні, ще не задокументовані шахрайські схеми до того моменту, як вони будуть алгоритмізовані та додані до баз даних сигнатур службою безпеки.

Ця предиктивна здатність безпосередньо спирається на ефективність обробки багатовимірних просторів ознак. Сучасний процес взаємодії клієнта з банком, зокрема на етапі відкриття рахунку, генерує розгалужену матрицю даних, що охоплює не лише базові соціально-демографічні чи фінансові метрики, але й глибинні технічні параметри [5].

Алгоритми машинного навчання здатні консолідовано обробляти сотні предикторів, включаючи цифрові відбитки пристроїв, параметри мережевого підключення, геолокаційні флуктуації та кінематику взаємодії з користувацьким інтерфейсом. Завдяки цьому виявляються неочевидні кореляції між, на перший погляд, незалежними атрибутами заявника, де ізольована ознака не викликає підозр, але їхня специфічна комбінація сигналізує про високу ймовірність фроду.

З точки зору операційної ефективності, критичною перевагою є оптимізація балансу помилок класифікації першого та другого роду. На зміну

категоричному бінарному висновку систем правил приходять імовірнісні оцінки ризику (fraud score), які обчислюються для кожної індивідуальної транзакції чи заявки. Практичні аспекти реалізації такого механізму розглядаються у роботі Н.Є. Негуляєва та Н.І. Недашківської [7], присвяченій розробці архітектури антифрод-системи для ідентифікації шахрайських карткових транзакцій. Дослідники демонструють, що спеціалізовані ML-моделі здатні в режимі реального часу аналізувати високошвидкісні потоки карткових платежів, формуючи точний скоринговий бал для кожної фінансової операції.

Такий підхід забезпечує фінансовим установам гнучкість у налаштуванні порогів прийняття рішень відповідно до поточного ризик-апетиту банку. Це дозволяє динамічно калібрувати чутливість системи, максимізуючи виявлення істинно шахрайських дій при одночасному жорсткому стримуванні рівня хибних блокувань, що гарантує безшовний досвід для легітимних клієнтів і знижує навантаження на підрозділи ручної верифікації [6].

Крім того, архітектура ML-систем характеризується високою адаптивністю завдяки механізмам безперервного навчання. Враховуючи, що шахрайські стратегії підлягають постійній еволюції, статичні алгоритми неминуче та швидко втрачають свою точність. Натомість моделі машинного навчання можуть ітеративно донавчатися на нових верифікованих інцидентах фроду під час регулярних пакетних оновлень або в режимі реального часу.

Завдяки цій властивості система захисту уникає алгоритмічної деградації, синхронно еволюціонуючи разом із динамічними змінами у ландшафті кіберзагроз та підтримуючи стабільно високі показники повноти виявлення.

1.4 Постановка задачі

Метою роботи є порівняльне дослідження та практична реалізація сучасних математичних моделей машинного навчання для вирішення задачі бінарної класифікації і виявлення аномалій з метою ефективної ідентифікації фроду на етапі відкриття банківських рахунків на основі аналізу багатовимірних поведінкових ознак заявників в умовах суттєво незбалансованих наборів даних.

Дано:

- 1) I – вхідний набір даних (записи з датасету Bank Account Fraud), де кожна операція представлена вектором ознак (сума, час, локація, тип операції тощо);
- 2) M – мітки розмітки, що позначають приналежність транзакції до класу шахрайських (1) або легітимних (0).

W – вагові коефіцієнти класів для корекції незбалансованості вибірки.

Нехай $f(I)$ – функція, що класифікує транзакції, реалізована за допомогою п'яти підходів:

- 1) $f_{RF}(I)$ – випадковий ліс (Random Forest) для забезпечення стабільності та інтерпретованості через оцінку важливості ознак;
- 2) $f_{SVM}(I)$ – метод опорних векторів з RBF-ядром для нелінійної класифікації;
- 3) $f_{MLP}(I)$ – багатошаровий перцептрон для аналізу ієрархічних зв'язків;
- 4) $f_{XGB}(I)$ – градієнтний бустинг (XGBoost) для послідовної мінімізації помилок класифікації на незбалансованих даних;
- 5) $f_{IF}(I)$ – ізоляційний ліс (Isolation Forest) для виявлення аномалій без вчителя, що ізолює спостереження на основі їхньої відмінності від нормального патерну.

Тоді задача виявлення фроду полягає у знаходженні функцій:

- 1) $C = f(I)$ – визначення прогнозного класу спостереження (шахрайська або легітимна);
- 2) $L(C, M)$ – мінімізація функції втрат з урахуванням ваг W для досягнення максимальної точності розпізнавання аномалій.

Критерії якості для ефективності класифікації:

- 1) Precision–Recall AUC – ключовий показник якості в умовах сильного дисбалансу класів;
- 2) F1–score – метрика для оцінки балансу між пропущеним фродом (Recall) та хибними спрацюваннями (Precision);
- 3) Matthews Correlation Coefficient (MCC) – показник стійкості моделі, що враховує всі типи прогнозів (TP, TN, FP, FN) для виключення зміщень на незбалансованих даних;
- 4) Confusion matrix – для аналізу помилок першого та другого роду;
- 5) Accurasy – для порівняння з наведеними вище метриками якості.

Для досягнення усіх поставлених задач в цій роботі використовуються такі інструменти:

- 1) мова програмування Python – для швидкої розробки і прототипування моделей;
- 2) краудсорсингова платформа Kaggle – для отримання доступу до набору даних (Bank Account Fraud), використання безкоштовних хмарних обчислювальних потужностей платформи під час навчання та тестування ресурсомістких моделей.

1.5 Висновки до розділу 1

У цьому розділі було досліджено сучасний стан кіберзагроз у фінансовому секторі, зокрема тенденції індустріалізації банківського шахрайства та критичні обмеження традиційних систем моніторингу на базі статичних правил. Визначено, що для ефективної протидії новим векторам

атак необхідним є перехід до предиктивної аналітики на основі методів машинного навчання.

Застосування цих технологій дозволяє комплексно аналізувати багатовимірні поведінкові профілі клієнтів та адаптивно ідентифікувати приховані аномалії навіть в умовах суттєвого дисбалансу класів. На базі формалізованого понятійного апарату було сформульовано загальну постановку задачі дослідження, яка полягає у порівняльному аналізі інтелектуальних моделей для розпізнавання фроду на етапі цифрового онбордингу.

РОЗДІЛ 2. МЕТОДИ І МОДЕЛІ БІНАРНОЇ КЛАСИФІКАЦІЇ ТА ВИЯВЛЕННЯ АНОМАЛІЙ, МЕТРИКИ ОЦІНЮВАННЯ РЕЗУЛЬТАТІВ МОДЕЛЕЙ

2.1 Моделі

У цьому підрозділі розглянуто теоретичні засади та математичний апарат алгоритмів машинного навчання, обраних для розв'язання поставленої задачі. Зокрема, детально описано ансамблеві моделі, а саме випадковий ліс (Random Forest) та екстремальний градієнтний бустинг (eXtreme Gradient Boosting, XGBoost), метод опорних векторів із застосуванням радіального базисного ядра (Support Vector Machine with Radial Basis Function kernel, SVM with RBF), а також архітектуру штучної нейронної мережі у вигляді багатошарового перцептрона (Multilayer Perceptron, MLP). Крім того, як окремий підхід для пошуку нетипових патернів у даних, досліджено алгоритм виявлення аномалій – ізольований ліс (Isolation Forest).

2.1.1 Random Forest

Випадковий ліс (Random Forest) – це ансамблевий алгоритм машинного навчання, класична математична формалізація якого була запропонована Лео Брейманом у 2001 році у співавторстві з Адель Катлер. Варто зазначити, що теоретичні передумови цієї методології, зокрема використання методу випадкових підпросторів для стохастичних лісів, були закладені ще в роботах Тін Кам Хо у 1995 році.

Концептуальним базисом даного підходу є побудова суперпозиції з множини незалежних базових предикторів – класифікаторів або регресорів, реалізованих у вигляді дерев рішень. З позицій статистичної теорії навчання,

ізолювані незсічені дерева рішень демонструють надмірну чутливість до шуму у навчальній вибірці. Це неминуче призводить до проблеми перенавчання та генерує надзвичайно високу дисперсію моделі. Алгоритм випадкових лісів нівелює цей фундаментальний недолік шляхом агрегування прогнозів великої кількості таких дерев.

Для гарантування структурної гетерогенності та мінімізації залежності базових моделей у межах ансамблю, метод спирається на дві ключові парадигми.

1. Технологію беггінгу: вносить збурення у простір спостережень шляхом генерації підвбірок із поверненням.
2. Метод випадкових підпросторів: здійснює проекцію даних на випадково обрані підмножини ознак під час формування кожного вузла розгалуження.

Синергетичний ефект цих двох підходів спрямований на примусову декореляцію базових дерев. Згідно з теоремою ансамблювання, зменшення коефіцієнта коваріації між окремими моделями дозволяє асимптотично знизити загальну дисперсію композиції, не спричиняючи при цьому критичного зростання її зміщення. Таким чином досягається оптимальний баланс зміщення та дисперсії (bias–variance tradeoff), внаслідок чого формується робастна предиктивна система. Вона характеризується високою узагальнюючою здатністю під час роботи з новими (небаченими) даними та демонструє виняткову резистентність до перенавчання.

Алгоритм роботи:

- 1) для побудови B дерев створюється B вибірок розміром N з навчального набору даних, відомі як бутстреп–вибірки (Bootstrap Sampling), шляхом випадкового вибору з поверненням;
- 2) для кожної вибірки будується дерево рішень. На кожному вузлі дерева алгоритм випадковим чином обирає $m \leq p$ ознак, де p – загальна

кількість ознак. Для класифікації зазвичай беруть $m = \sqrt{p}$, для регресії $m = \frac{p}{3}$;

- 3) серед m обраних ознак знаходиться оптимальна ознака та точка розщеплення, яка максимізує зменшення домішок. Найчастіше використовується індекс Джині (Gini index);
- 4) дерева будуються до максимальної глибини (без прунінгу). Кінцевий прогноз формується шляхом мажоритарного голосування для класифікації, або усереднення для регресії.

Архітектура Random Forest зображена на рисунку 2.1.

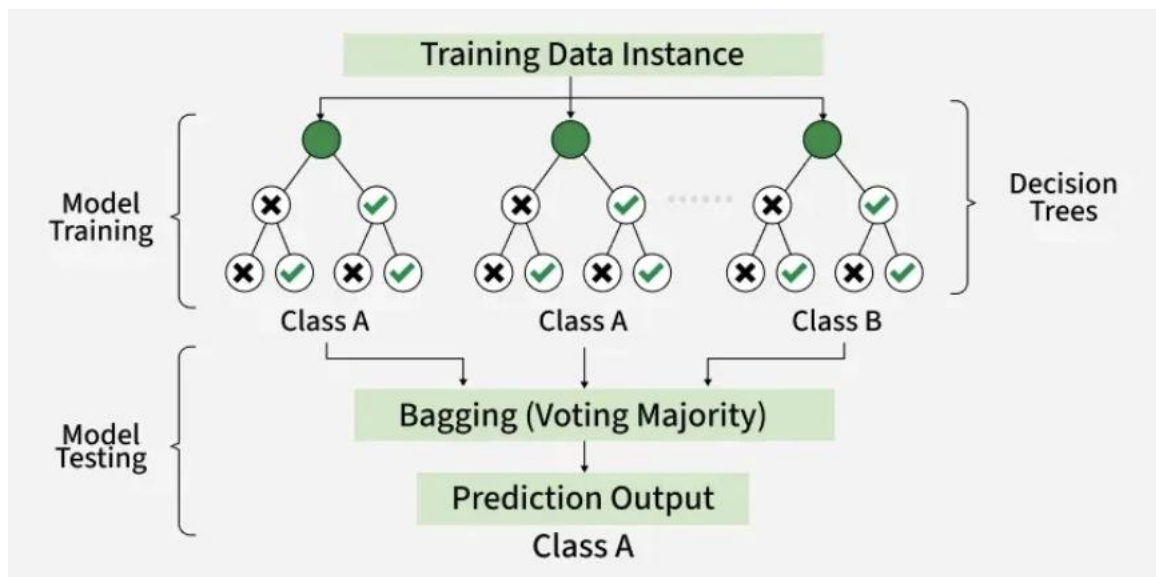


Рисунок 2.1 – Архітектура Random Forest

Прогноз ансамблю для регресії [9, с.589]:

$$\hat{f}_{rf}^B(x) = \frac{1}{B} \sum_{b=1}^B T(x; \theta_b),$$

де x – вхідний вектор;

T – дерево рішень;

θ_b – параметри b -го дерева (вибір ознак та розщеплення).

Зменшення дисперсії за рахунок ансамблювання залежить від кореляції ρ між деревами [9, с.588]:

$$\text{Var}(\hat{f}_{rf}^B(x)) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

Критерій розщеплення (Індекс Джині) [9, с.309]:

$$G = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

де $\hat{p}_{mk}$ – частка спостережень класу k у вузлі m .

2.1.2 SVM з RBF ядром

Метод опорних векторів (Support Vector Machine, SVM) – це алгоритм машинного навчання, теоретичне підґрунтя якого базується на теорії статистичного навчання та принципі мінімізації структурного ризику. Фундаментальна концепція базового класифікатора SVM полягає у знаходженні оптимальної розділювальної гіперплощини у просторі ознак, яка забезпечує максимальну геометричну відстань між об'єктами різних класів. Максимізація цієї відстані оптимізує узагальнюючу здатність моделі, тим самим мінімізуючи ризик перенавчання на тренувальних даних.

Проте, оскільки в реальних прикладних задачах досліджувані багатовимірні вибірки вкрай рідко є повністю лінійно роздільними у своєму вихідному просторі, класичний метод модифікується за допомогою «ядрового трюку» (kernel trick). Цей математичний прийом, що спирається на теорему Мерсера, дозволяє здійснювати нелінійне відображення векторів вхідних даних у новий, абстрактний простір ознак значно вищої розмірності, де

побудова лінійної межі стає можливою. Унікальність трюку полягає в тому, що він дозволяє обійти проблему «прокляття розмірності», оскільки всі обчислення відбуваються без прямого визначення координат об'єктів у новому просторі.

Серед широкого спектра доступних ядеркових функцій, радіальна базисна функція (Radial Basis Function, RBF), також відома як гаусове ядро, де-факто є стандартом для моделювання складних нелінійних залежностей. Ключовою математичною особливістю RBF-ядра є те, що воно неявно здійснює проєкцію вхідних векторів у нескінченновимірний гільбертів простір. На відміну від поліноміальних ядер, які обмежені заданим ступенем, RBF діє як міра локальної схожості. Воно оцінює близькість між спостереженнями на основі їхньої евклідової відстані, експоненційно зменшуючи вплив об'єктів один на одного в міру їх віддалення. Це наділяє модель винятковою гнучкістю: вплив кожного навчального прикладу є локалізованим, що дозволяє алгоритму формувати високоеластичні, замкнені нелінійні межі прийняття рішень, здатні ізолювати складні кластери даних незалежно від їхньої топології.

Алгоритм роботи:

- 1) замість обчислення скалярного добутку ознак у вихідному просторі, використовується функція ядра, що неявно переносить дані у нескінченновимірний простір;
- 2) алгоритм розв'язує задачу квадратичного програмування, щоб знайти опорні вектори (точки, найближчі до межі прийняття рішень), мінімізуючи норму вектора ваг $||\omega||^2$;
- 3) вводиться гіперпараметр C для контролю балансу між максимізацією зазору та штрафуванням за помилки класифікації.

Принцип розділення класів за допомогою оптимальної гіперплощини в алгоритмі SVM зображено на рисунку 2.2.

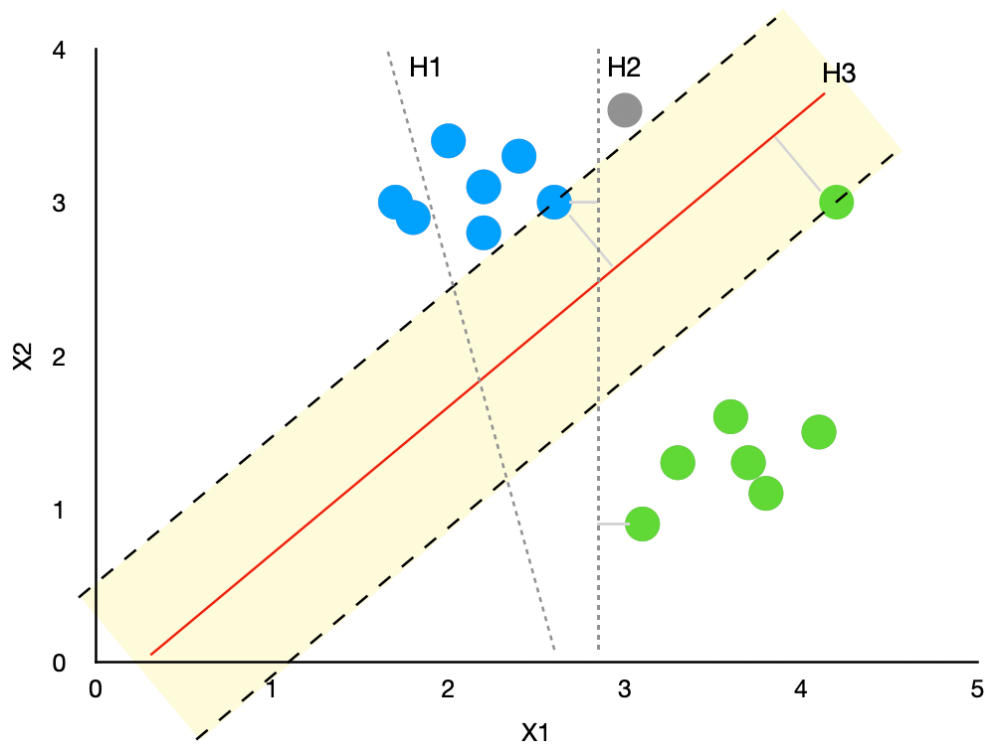


Рисунок 2.2 – Принцип роботи алгоритму SVM

Функція радіального базисного ядра (RBF) [9, с.428–431]:

$$K(x, x') = \exp \left(-\frac{\|x - x'\|^2}{2\sigma^2} \right) = \exp \left(-\gamma \|x - x'\|^2 \right),$$

де γ – параметр, що контролює радіус впливу одного спостереження.

Двоїста задача оптимізації [9, с.417–421]:

$$\max_{\alpha} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

За умов: $0 \leq \alpha_i \leq C$ та $\sum_{i=1}^N \alpha_i y_i = 0$.

Функція прогнозування для нового об'єкта x [9, с.417–421]:

$$f(x) = \text{sign}\left(\sum_{i=1}^N \alpha_i y_i K(x, x_i) + b\right)$$

2.1.3 XGBoost

XGBoost (eXtreme Gradient Boosting) – це масштабована та апаратно–оптимізована імплементація парадигми градієнтного бустингу над вирішальними деревами (Gradient Boosting Decision Trees, GBDT). Теоретичне підґрунтя методу базується на концепції ансамблювання слабких алгоритмів, які послідовно інтегруються в єдину предиктивну модель, де кожен наступний елемент ансамблю спрямований на мінімізацію залишків або корекцію помилок попередніх. Відмінною рисою даної архітектури є оптимізація функції втрат за допомогою наближення Тейлора другого порядку. Такий підхід дозволяє використовувати інформацію не лише про градієнт, але й про другу похідну цільової функції, що забезпечує швидшу збіжність та точніше визначення оптимального кроку в просторі функцій.

Ключовою характеристикою, що відрізняє цю реалізацію від класичних підходів до градієнтного бустингу, є глибока інтеграція системних і алгоритмічних вдосконалень. Зокрема, алгоритм містить вбудовані механізми для ефективної обробки розріджених даних (sparsity-aware split finding). Це дозволяє моделі автоматично адаптувати напрямок розгалуження у вузлах прийняття рішень за наявності пропущених значень або значної частки нульових ознак. На обчислювальному рівні архітектура реалізує розпаралелювання процесу побудови дерев на етапі пошуку оптимальних точок розщеплення ознак, використовує оптимізацію доступу до кеш–пам'яті мікропроцесора (cache-aware computation) та підтримує обробку масивів даних, які перевищують обсяг оперативної пам'яті системи.

Важливим структурним компонентом методу є апіорна спрямованість на запобігання перенавчанню за рахунок суворої регуляризації цільової

функції. Контроль складності базових моделей гарантує високу узагальнюючу здатність ансамблю. Завдяки комбінації обчислювальної ефективності та стійкості до шуму, алгоритм виявляє високу результативність у задачах складної багатовимірної класифікації, зокрема в умовах значного дисбалансу класів розподілу цільової змінної. Архітектура здатна ідентифікувати латентні нелінійні закономірності в даних без вимоги до попередньої глибокої математичної трансформації простору ознак.

Алгоритм роботи:

- 1) модель починає роботу з базового прогнозу (наприклад, константи);
- 2) на кроці t для кожного об'єкта i обчислюються перша g_i та друга h_i похідні функції втрат щодо поточного прогнозу;
- 3) будується нове дерево $f_t(x)$, що намагається мінімізувати помилки попереднього кроку. Розщеплення у вузлах визначається за допомогою критерію “Gain” (виграш від розщеплення), який включає регуляризацию;
- 4) прогнози оновлюються: $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta f_t(x_i)$, де η – швидкість навчання (learning rate).

Архітектура XGBoost зображена на рисунку 2.3.

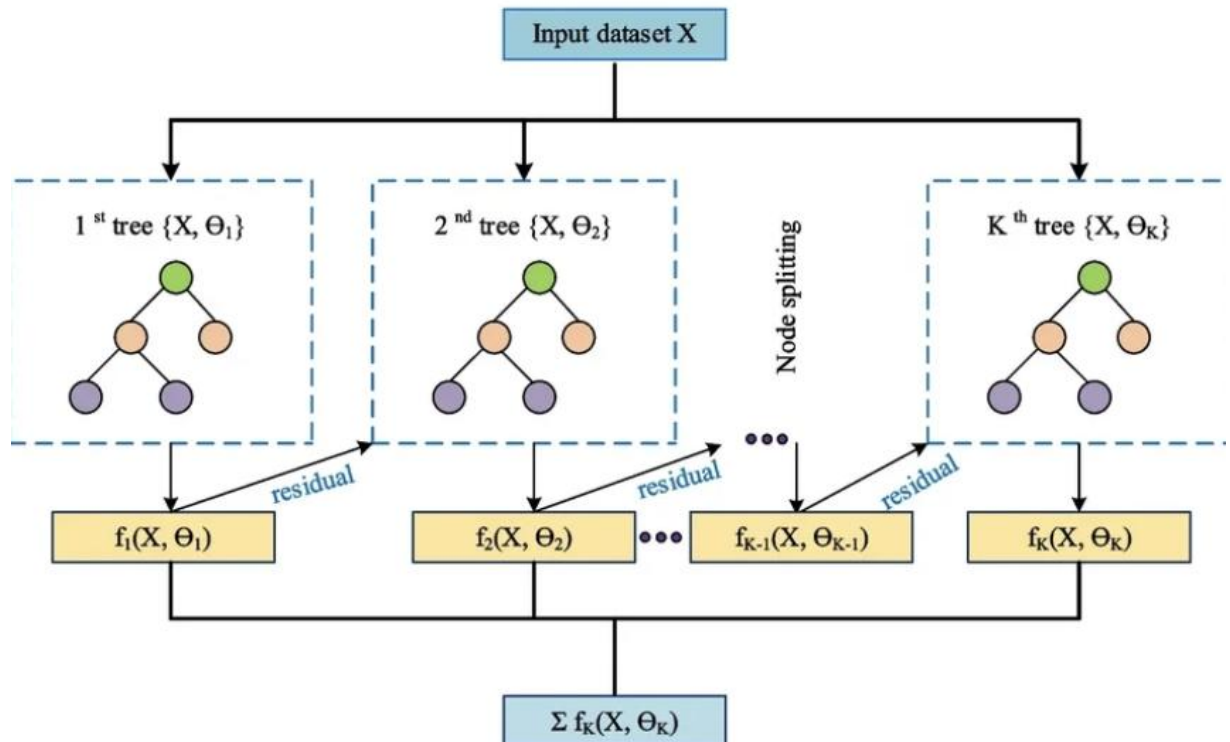


Рисунок 2.3 – Архітектура XGBoost

Цільова функція з наближенням Тейлора 2-го порядку [12, с.786]:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^n \left(g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right) + \Omega(f_t),$$

де $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ та $h_i = \partial_{\hat{y}^{(t-1)}}^2 l(y_i, \hat{y}^{(t-1)})$

Штраф за складність дерева (Регуляризація) [12, с.786]:

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T \omega_j^2,$$

де T – кількість листків, ω_j – ваги в листках.

Оптимальна вага листка j [12, с.787]:

$$\omega_j^* = - \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda'}$$

Критерій вибору найкращого розщеплення (Gain) [12, с.787]:

$$Gain = \frac{1}{2} \left(\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} + \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda} \right) - \gamma$$

2.1.4 MLP

Багатошаровий перцептрон (Multilayer Perceptron, MLP) – це штучна нейронна мережа прямого поширення сигналів (Feedforward Neural Network, FNN). Історично та концептуально ця модель стала еволюційним розвитком класичного одношарового перцептрона, подолавши його нездатність розв'язувати задачі з лінійно нероздільними даними (наприклад, класичну проблему XOR).

Архітектурно MLP є повнозв'язним орієнтованим графом, що структурно диференціюється на три типи компонентів: рецепторний вхідний шар (input layer), один або декілька прихованих шарів (hidden layers), та вихідний шар (output layer). Характерною ознакою такої топології є те, що кожен вузол (нейрон) поточного шару має зважений синаптичний зв'язок з усіма вузлами наступного шару, при цьому будь-які цикли або зворотні зв'язки в межах мережі повністю відсутні.

Висока виразна здатність багатошарового перцептрона має строге математичне обґрунтування, відоме як теорема універсальної апроксимації (зокрема, теорема Цибенка). Відповідно до неї, нейронна мережа прямого поширення із щонайменше одним прихованим шаром, що містить скінченну кількість нейронів із нелінійною функцією активації, здатна з будь-якою

наперед заданою точністю апроксимувати довільну неперервну функцію на компактній підмножині простору R^n .

Фундаментальну роль у цьому процесі відіграють саме нелінійні активаційні функції прихованих шарів. Вони дозволяють мережі здійснювати складні нелінійні просторові перетворення вхідного вектора ознак, послідовно відображаючи його в нові, більш абстрактні приховані простори. Завдяки цьому формуються такі репрезентації даних, за яких початково складна нелінійна задача стає лінійно розв'язною на етапі вихідного шару. Конфігурація самого вихідного шару (кількість нейронів та тип їхньої активації) суворо детермінується специфікою поставленої задачі: наприклад, для регресійного аналізу використовується лінійна активація, для бінарної класифікації це логістична функція (Sigmoid), а для оцінки ймовірностей у мультикласових задачах функція Softmax.

Алгоритм роботи:

- 1) вхідний вектор x проходить через шари мережі, на кожному з яких обчислюється лінійна комбінація входів з ваговими коефіцієнтами матриці W , додається зміщення b , і результат пропускається через нелінійну функцію активації (ReLU, Sigmoid, Tanh);
- 2) на вихідному шарі обчислюється функція втрат, яка показує розбіжність між прогнозом $\hat{y}$ і справжнім y ;
- 3) використовуючи ланцюгове правило диференціювання, обчислюються градієнти функції втрат по відношенню до кожної ваги та зміщення в мережі (від останнього шару до першого);
- 4) ваги коригуються за допомогою алгоритму оптимізації у напрямку, антиколінеарному градієнту.

Схема нейронної мережі (а) та схема нейрону (б) зображені на рис. 2.4.

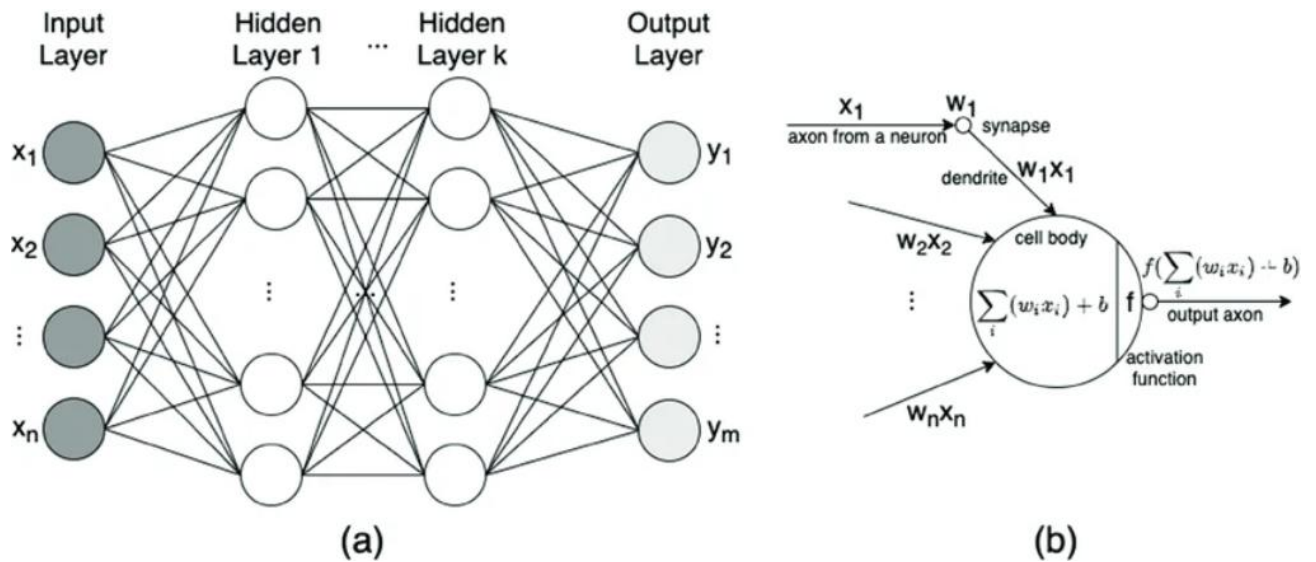


Рисунок 2.4 – Схема нейронної мережі (а) та схема нейрону (b) [16]

Операція на k -му шарі:

$$h^{(k)} = g^{(k)}(W^{(k)}h^{(k-1)} + b^{(k)}),$$

де $g^{(k)}$ – функція активації.

Функція втрат (Крос-ентропія) [10, с.174]:

$$J(\theta) = -E_{x,y \sim \hat{p}_{data}} \log p_{model}(y|x; \theta)$$

Оновлення ваг методом градієнтного спуску [10, с.291]:

$$\theta \leftarrow \theta - \epsilon \nabla_{\theta} J(\theta),$$

де ϵ – швидкість навчання (learning rate).

2.1.5 Isolation Forest

Ізолюючий ліс (Isolation Forest) – це метод машинного навчання без вчителя (unsupervised learning), заснований на ансамблях дерев ухвалення рішень та спеціально розроблений для виявлення викидів у багатовимірних масивах даних. На відміну від більшості класичних підходів до пошуку аномалій (таких як статистичне профілювання, методи на основі щільності або обчислення відстаней), які фокусуються на побудові моделі «нормальної» поведінки з подальшим виявленням відхилень, цей алгоритм спирається на принципово іншу парадигму. Isolation Forest радикально відрізняється від інших методів тим, що він не профілює нормальні точки даних, а безпосередньо ізолює аномалії.

Такий підхід є обчислювально ефективним завдяки експлуатації двох фундаментальних властивостей нетипових спостережень у будь-якому наборі даних. По-перше, аномалії зазвичай є статистичною меншістю (кількісно менші). По-друге, вони суттєво відрізняються своїми просторовими ознаками від стандартних екземплярів. У сукупності ці характеристики роблять їх значно більш вразливими до процедури «ізоляції» – процесу відокремлення конкретної точки від решти об'єктів вибірки шляхом побудови ортогональних зрізів у просторі ознак.

Це випадкове розділення створює помітні коротші шляхи для аномалій (рисунок 2.5), оскільки:

- 1) менша кількість випадків аномалій призводить до меншої кількості розділів – коротші шляхи в структурі дерева (a);
- 2) випадки з відмінними значеннями атрибутів більш ймовірно будуть розділені на ранніх етапах поділу (b).

Як наслідок, нормальні дані, що формують щільні кластери, вимагають значної кількості ітерацій розбиття простору для відокремлення кожної окремої точки. Натомість аномалії відсікаються від загального масиву вже на

перших рівнях ієрархії, що робить довжину шляху природною та математично обґрунтованою метрикою для визначення ступеня аномальності об'єкта ще до проведення складних обчислень.

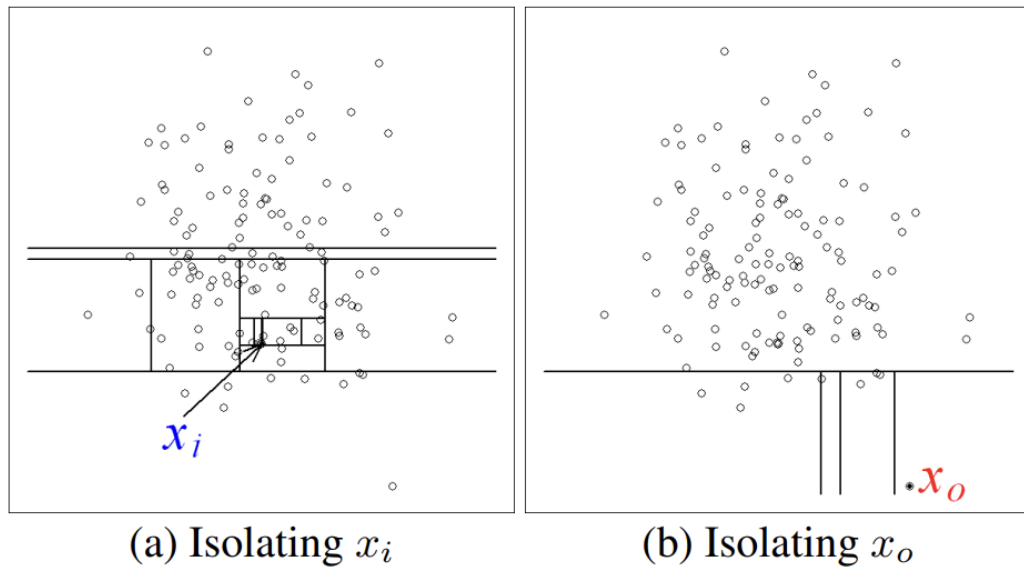


Рисунок 2.5 – Точки, ізольовані алгоритмом Isolation Forest. (a) нормальна точка вимагає дванадцяти випадкових поділів для ізоляції (b) аномалія вимагає лише чотирьох поділів для ізоляції [13]

Алгоритм роботи:

- 1) алгоритм будує двійкове дерево, випадковим чином обираючи ознаку з набору даних;
- 2) для обраної ознаки випадково генерується точка розщеплення між мінімальним та максимальним значеннями цієї ознаки;
- 3) процес повторюється рекурсивно доти, доки кожне спостереження не буде ізольоване в окремому листі (або поки не буде досягнуто ліміт глибини);
- 4) оскільки аномалії знаходяться далеко від щільних скупчень даних і потребують менше розщеплень для ізоляції, довжина шляху від кореня до листка стає мірою “нормальності”, де короткий шлях свідчить про високу ймовірність аномалії.

Архітектура Isolation Forest зображена на рисунку 2.6.

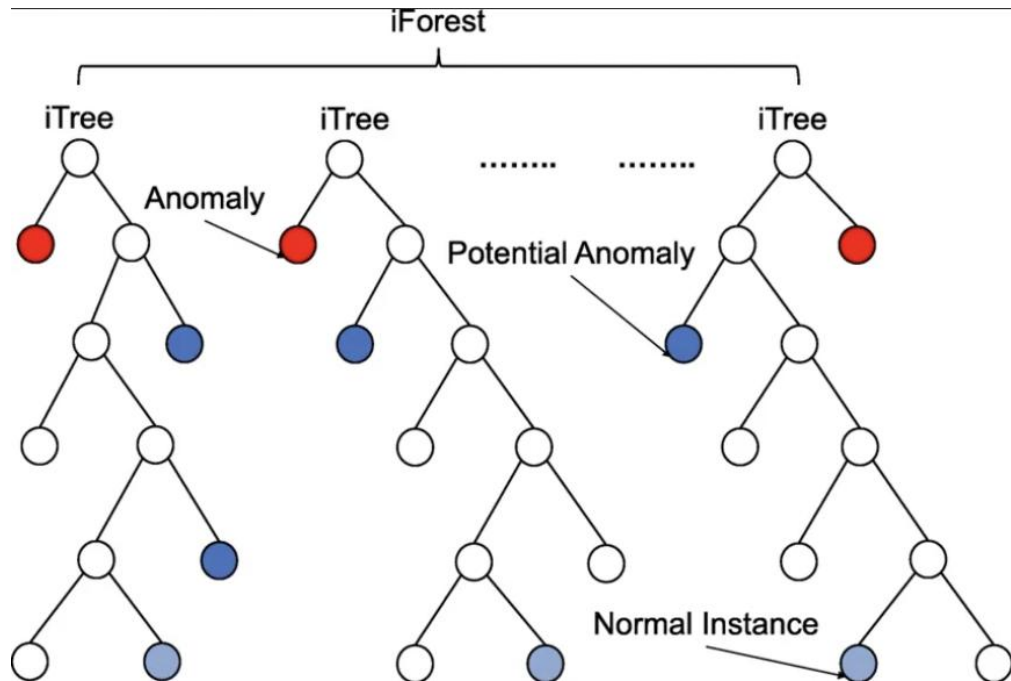


Рисунок 2.6 – Архітектура Isolation Forest

Довжина шляху $h(x)$ – це кількість ребер, які проходить спостереження x від кореневого вузла до листка в одному $iTree$. Оскільки процес розщеплення еквівалентний структурі бінарного дерева пошуку (BST), середня довжина шляху для n об'єктів оцінюється як [13, рівняння 1]:

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n},$$

де $H(i) \approx \ln \ln(i) + 0.5772157$ (стала Ейлера–Маскероні) – гармонічне число.

Оцінка аномальності (Anomaly Score) для спостереження x [13, рівняння 2]:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}},$$

де $E(h(x))$ – математичне сподівання довжини шляху $h(x)$ по всьому ансамблю з B дерев.

Інтерпретація результату:

- 1) якщо $s(x, n) \rightarrow 1$, то x є аномалією (шлях дуже короткий: $E(h(x)) \rightarrow 0$);
- 2) якщо $s(x, n) \rightarrow 0.5$, то x є нормальною точкою (шлях близький до середнього: $E(h(x)) \rightarrow c(n)$);
- 3) якщо $s(x, n) \rightarrow 0$, точка глибоко захована в кластері нормальних даних.

2.2 Метрики оцінки якості класифікації

Метрики якості це кількісні параметри, що відображають конкретні характеристики роботи моделей. Вони слугують інструментом для зіставлення ефективності різних алгоритмів та допомагають ідентифікувати джерела виникнення помилок.

2.2.1 Confusion Matrix

Матриця помилок – це таблична структура, яка дозволяє візуалізувати продуктивність алгоритму. Кожен рядок матриці представляє екземпляри фактичного класу, тоді як кожен стовпець представляє екземпляри передбаченого класу (або навпаки).

$$Confusion\ Matrix = \begin{pmatrix} TP & FN \\ FP & TN \end{pmatrix},$$

де TP – об'єкти позитивного класу, які модель правильно ідентифікувала як позитивні;

TN – об'єкти негативного класу, правильно класифіковані як негативні;

FP – об'єкти негативного класу, які модель помилково віднесла до

позитивного (помилка I роду);

FN – об'єкти позитивного класу, які модель пропустила і віднесла до негативного (помилка II роду).

Наявність матриці помилок є обов'язковою для детального аналізу поведінки моделі на даному наборі даних. Вона дозволяє не просто отримати агреговану оцінку (як скалярне значення), а побачити структурну природу помилок алгоритму: чи схильна модель до генерації хибних відповідей (високий рівень FP), чи навпаки, пропускає цільові класи (високий рівень FN).

2.2.2 Precision–Recall Area Under Curve (PR AUC)

Precision–Recall Area Under Curve (PR AUC) – це інтегральна метрика, яка оцінює якість бінарної класифікації шляхом аналізу залежності між точністю (Precision) та повнотою (Recall) при зміні порогу прийняття рішення. Математично PR AUC визначається як інтеграл від функції точності за повнотою:

$$PR AUC = \int_0^1 P(R) dR,$$

де $P(R)$ – функція точності (Precision) в залежності від повноти (Recall) R .

На практиці цей інтеграл часто апроксимується за допомогою інтерполяції або обчислюється як зважена сума на дискретних проміжках.

Враховуючи специфіку набору даних (наявність дисбалансу класів), використання PR AUC є критично важливим. На відміну від класичної ROC AUC, яка враховує специфічність (оцінку мажоритарного класу), крива PR фокусується виключно на міноритарному (цільовому) класі. Це дозволяє об'єктивно оцінити здатність моделі виявляти рідкісні події без спотворення результатів за рахунок великої кількості істинно негативних класифікацій.

2.2.3 F1–score

F1–міра є гармонійним середнім між точністю (Precision) та повнотою (Recall). Ця метрика розроблена для консолідації двох часто конфліктуючих показників у єдине значення, що дозволяє комплексно оцінити ефективність класифікатора [13].

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN}$$

Ця метрика була додана до пайплайну оцінки, оскільки для даного набору даних однаково важливо мінімізувати як хибні спрацьовування (помилки I роду), так і пропуски цільових об'єктів (помилки II роду).

Гармонічне середнє, на відміну від арифметичного, сильно штрафує модель, якщо хоча б один із показників (Precision або Recall) наближається до нуля, що робить F1–score надійним індикатором збалансованості моделі.

2.2.4 Matthews Correlation Coefficient (MCC)

Коефіцієнт кореляції Метьюза (Matthews Correlation Coefficient) – це метрика, яка використовується для оцінки якості бінарної класифікації та враховує всі чотири елементи матриці помилок. Вона розглядається як дискретна версія коефіцієнта кореляції Пірсона. Значення MCC варіюються від -1 (повна невідповідність) до $+1$ (ідеальна класифікація), де 0 означає випадкове передбачення [13].

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

МСС додано як одну з найбільш стійких статистичних метрик для нашого набору даних. Оскільки вона враховує пропорції всіх чотирьох категорій результатів (істинних та хибних), МСС дає високий бал лише тоді, коли модель демонструє хороші результати як на мажоритарному, так і на міноритарному класах. Це ідеальний індикатор для підтвердження того, що високі показники інших метрик не є артефактом незбалансованості вибірки.

2.2.5 Accuracy

Точність (Ассурасу) вимірює загальну частку правильних прогнозів моделі (як позитивних, так і негативних) серед усієї кількості досліджуваних випадків [13].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

У межах даної роботи метрика Ассурасу включена не для того, щоб слугувати основним критерієм оцінки якості моделі, а виключно як базовий показник для порівняння. Враховуючи специфіку набору даних, а саме структурний дисбаланс класів, використання Ассурасу для прийняття рішень щодо вибору оптимальної моделі є хибним підходом з наукової точки зору (так званий «парадокс точності»).

Якщо мажоритарний клас становить, наприклад, 90% вибірки, примітивна модель, яка класифікує всі об'єкти як мажоритарні, матиме значення Ассурасу на рівні 0.90. Проте така модель є абсолютно нездатною розв'язувати поставлене завдання (виявлення цільового міноритарного класу).

Тому Ассурасу наведена тут для повноти картини, проте висновки щодо ефективності алгоритмів базуються на більш робастних метриках, описаних у попередніх підрозділах.

2.3 Висновки до розділу 2

У цьому розділі було проведено теоретичний огляд кожної з моделей, обраних для розв’язання поставленої задачі бінарної класифікації. Зокрема, розглянуто принципи роботи та специфічні особливості застосування низки алгоритмів машинного навчання, серед яких Random Forest, метод опорних векторів з RBF-ядром, XGBoost, багатошаровий перцептрон (MLP) та Isolation Forest.

Окрему увагу приділено формуванню системи метрик для об’єктивної оцінки якості предиктивних моделей в умовах наявного дисбалансу класів. Використання таких робастних інтегральних та статистичних показників, як PR AUC, F1-міра та коефіцієнт кореляції Метьюза (MCC), забезпечує стійкість оцінювання та дозволяє сфокусуватися на здатності алгоритмів виявляти цільовий міноритарний клас. Водночас застосування матриці помилок допомагає структурно проаналізувати природу хибних спрацьовувань та пропусків, тоді як метрика Ассигасу слугує виключно базовим орієнтиром, ілюструючи вплив «парадоксу точності».

РОЗДІЛ 3. ПРОЕКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ, ПОБУДОВА МОДЕЛЕЙ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Архітектура інтелектуальної системи

Опис архітектури інформаційної системи відображає концептуальну структуру взаємодії її функціональних блоків, які забезпечують повний життєвий цикл обробки даних від моменту надходження заявки до формування фінального вердикту. Головною метою побудови такої архітектури є створення модульної, масштабованої та відмовостійкої системи, здатної в реальному часі аналізувати багатовимірні поведінкові ознаки потенційних клієнтів і мінімізувати фінансові та репутаційні ризики банківської установи.

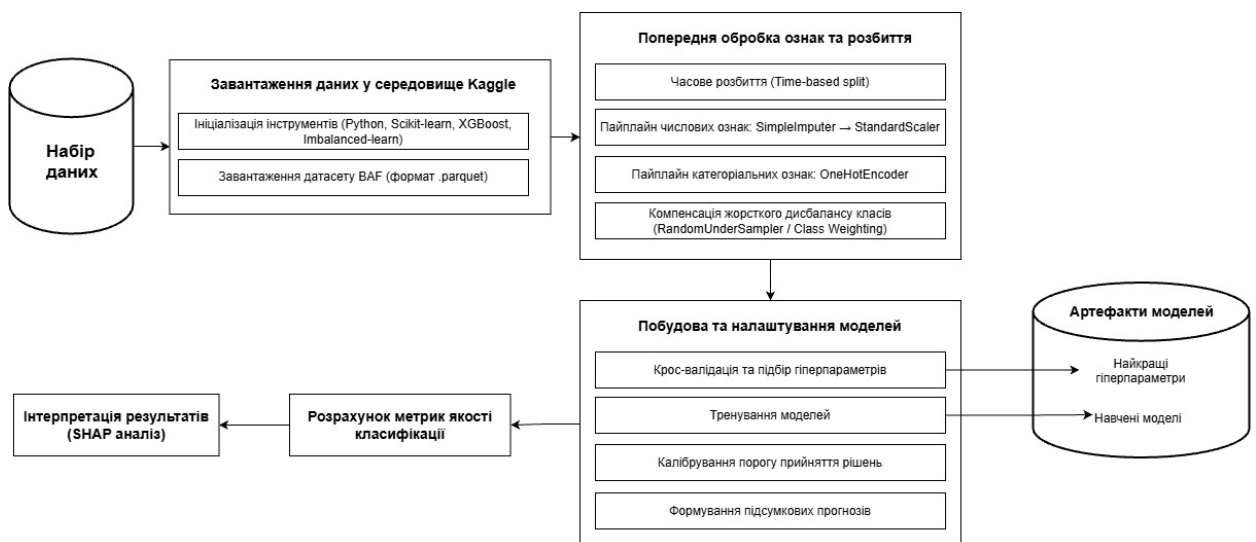


Рисунок 3.1 – Архітектура системи

Першим базовим компонентом системи є модуль збору та попередньої обробки даних, куди безпосередньо надходить потік вхідних заявок. На цьому етапі реалізовано індивідуальні пайплайни (pipelines) очищення та трансформації ознак. Процес обробки включає специфічні алгоритми

заповнення пропущених значень, зокрема методи лінійної інтерполяції та прямого заповнення (Forward Fill), які демонструють високу стабільність на фінансових часових рядах і запобігають спотворенню структури даних. Окрім цього, модуль здійснює автоматичне кодування категоріальних змінних та масштабування числових метрик. Важливою архітектурною особливістю цього блоку є чітке часове розбиття даних (temporal split), що повністю виключає можливість витоку інформації з майбутнього при навчанні та валідації моделей.

Центральне місце в архітектурі посідає аналітичне ядро, що складається з комплексу навчених моделей бінарної класифікації та виявлення аномалій. Модульна структура ядра дозволяє інтегрувати різні типи алгоритмів, серед яких випадковий ліс (Random Forest), машини опорних векторів (SVM), багатошаровий перцептрон (MLP) та Isolation Forest. Завдяки проведеній оптимізації гіперпараметрів та оцінці критеріїв ефективності, основним робочим інструментом системи виступає модель градієнтного бустингу XGBoost, яка показала найвищу прогностичну здатність. Програмна реалізація передбачає можливість паралельного оцінювання або швидкої заміни класифікатора в разі концептуального зсуву в потоці шахрайських схем.

Для забезпечення практичної цінності системи у реальних банківських процесах архітектура містить модуль динамічного пост-процесингу та прийняття рішень. Замість статичного розділення класів тут впроваджено алгоритм адаптивного підбору порогу класифікації. Його функціонування підпорядковане жорсткому бізнес-обмеженню — рівень хибнопозитивних спрацьовувань (FPR) не повинен перевищувати 5%, що гарантує збереження лояльності легітимних клієнтів та запобігає безпідставному блокуванню їхніх заявок. На виході цей блок генерує остаточне рішення щодо верифікації транзакції чи анкетних даних.

Невід'ємною складовою архітектурного рішення є підсистема інтерпретації результатів моделювання, побудована на основі розрахунку

значень SHAP (SHapley Additive exPlanations). Цей інструмент інтегрований для забезпечення прозорості (explainability) прийнятих рішень, що є критично важливим для відділів ризик-менеджменту та відповідності регуляторним вимогам. Модуль SHAP у реальному часі декомпонує фінальний прогноз моделі, виділяючи ключові предиктори, які вплинули на зміщення оцінки в бік шахрайства. Зокрема, архітектурно забезпечено візуалізацію та логування впливу таких факторів, як житловий статус заявника та характеристики використовуваної операційної системи, що дозволяє аналітикам банку чітко розуміти логіку роботи алгоритмів штучного інтелекту.

3.2 Опис використаних програмних засобів

Даний підрозділ присвячено опису програмно-апаратного забезпечення та інструментальних середовищ, задіяних у ході виконання практичної частини роботи. У ньому обґрунтовується вибір базових технологій для обробки даних, побудови прогностичних моделей та оцінки їхньої ефективності, а також розглядаються платформи, що слугують джерелом емпіричної бази для тестування розроблених алгоритмічних рішень.

3.2.1 Python

Python – це високорівнева об'єктно-орієнтована мова програмування загального призначення, яка є одним із провідних стандартів у галузі системного аналізу, науки про дані та машинного навчання. Завдяки лаконічному синтаксису та наявності великої екосистеми спеціалізованих бібліотек, Python забезпечує повний цикл розробки аналітичних моделей: від збору та попередньої обробки даних до побудови, налаштування та валідації складних предиктивних алгоритмів.

Для реалізації практичної частини дослідження та побудови моделей інтелектуального аналізу було використано набір спеціалізованих бібліотек мови Python, які забезпечують реалізацію п'яти основних алгоритмів класифікації та виявлення аномалій, а також супутніх процесів обробки даних.

3.2.1.1 Бібліотека Scikit-learn

Scikit-learn є базовою бібліотекою для машинного навчання в Python, що надає широкий спектр алгоритмів для навчання з учителем та без учителя.

З цієї бібліотеки було використано модулі для реалізації основних методів, наведені нижче.

1. RandomForestClassifier – алгоритм випадкового лісу.
2. IsolationForest – алгоритм ізоляційного лісу.
3. SVC – класифікатор на основі методу опорних векторів.
4. MLPClassifier – класифікатор на основі багатошарового перцептрона, що є класом штучних нейронних мереж прямого поширення.

Окрім безпосередньо алгоритмів, модуль sklearn забезпечує допоміжні процеси:

- 1) попередня обробка – використовується StandardScaler для стандартизації числових ознак, OneHotEncoder для кодування категоріальних змінних, SimpleImputer для обробки пропущених значень та ColumnTransformer для об'єднання цих кроків у єдиний конвеєр перетворень;
- 2) оцінка та налаштування – для пошуку оптимальних гіперпараметрів застосовується RandomizedSearchCV, а для крос-валідації часових рядів TimeSeriesSplit. Оцінка якості моделей здійснюється за допомогою метрик, стійких до дисбалансу класів (average_precision_score, f1_score, matthews_corrcoef, а також confusion_matrix).

3.2.1.2 Бібліотека XGBoost

XGBoost (eXtreme Gradient Boosting) – це оптимізована розподілена бібліотека градієнтного бустингу. У роботі використовується XGBClassifier, що реалізує алгоритми машинного навчання в рамках фреймворку Gradient Boosting. Метод послідовно будує ансамбль слабких моделей (зазвичай дерев рішень), кожна з яких мінімізує помилки попередників, що забезпечує високу точність прогнозування та ефективну обробку розріджених даних.

3.2.1.3 Бібліотека Imbalanced–learn

Спеціалізована бібліотека, орієнтована на вирішення проблеми сильного дисбалансу класів у наборах даних, що є типовою проблемою при виявленні шахрайства. Використовується RandomUnderSampler для збалансування навчальної вибірки шляхом випадкового вилучення зразків мажоритарного класу. Об'єднання кроків семплінгу та моделювання забезпечується класом Pipeline з модуля imblearn.pipeline.

3.2.2 Краудсорсингова платформа Kaggle

Kaggle – це міжнародна краудсорсингова платформа для спеціалістів з Data Science та машинного навчання, дочірня компанія Google. У контексті наукових досліджень та розробок платформа виконує роль глобального репозиторію даних, середовища для розробки та майданчика для проведення змагань з предиктивної аналітики.

Сутність платформи Kaggle полягає в наданні дослідникам та аналітикам доступу до стандартизованих, високоякісних наборів даних з різних предметних областей. Це дозволяє науковцям проводити емпіричні

дослідження, тестувати та порівнювати алгоритми машинного навчання на реальних репрезентативних даних.

Архітектура платформи включає:

- 1) сховище даних – відкритий каталог структурованих та неструктурованих даних, які проходять модерацію спільнотою. Для багатьох наукових робіт, пов'язаних з інтелектуальним аналізом, ці датасети є основним джерелом емпіричної бази;
- 2) хмарне середовище обчислень (Kaggle Notebooks) – інтегроване середовище розробки, яке дозволяє виконувати скрипти Python та R безпосередньо в браузері, надаючи доступ до обчислювальних потужностей (включно з GPU та TPU) без необхідності локального налаштування інфраструктури;
- 3) систему бенчмаркінгу – стандартизовані метрики оцінювання та відкриті таблиці лідерів (Leaderboards) дозволяють об'єктивно порівнювати ефективність різних математичних моделей та алгоритмічних підходів на єдиній тестовій вибірці.

Використання ресурсів Kaggle в академічних дослідженнях забезпечує верифікованість результатів, оскільки доступність вихідних даних дозволяє іншим дослідникам відтворювати експерименти та перевіряти заявлену ефективність розроблених систем чи моделей класифікації.

3.3 Набір даних Bank Account Fraud (BAF) Tabular Dataset Suite

Набір даних Bank Account Fraud (BAF) Tabular Dataset Suite, офіційно представлений на конференції NeurIPS 2022, є першим публічно доступним, великомасштабним та реалістичним набором табличних даних, призначеним для оцінки алгоритмів машинного навчання у сфері фінансової безпеки. Набір був згенерований на основі реальних анонімізованих даних банківського сектору. Важливою характеристикою датасету є те, що він фокусується

виключно на виявленні шахрайства під час відкриття нових банківських рахунків (онбордингу).

Для створення BAF розробники застосували сучасні методи генерації синтетичних даних на основі архітектури CTGAN (Conditional Generative Adversarial Networks). Цей підхід дозволив зберегти статистичні властивості та складність реальних даних, водночас забезпечивши строгу конфіденційність клієнтів шляхом додавання шуму та попереднього кодування ознак.

Повний набір складається з шести різних синтетичних варіантів датасету. Розробники пропонують одну базову версію (Base) та п'ять модифікацій (Variants I–V) для стрес-тестування моделей.

В рамках даної роботи використовується базова версія набору даних. Цей варіант є еталонним бенчмарком, який максимально точно відтворює початковий реальний розподіл даних без внесення додаткових штучних зміщень або маніпуляцій з характеристиками суб'єктів. Однією з ключових характеристик датасету є екстремальний дисбаланс класів. Частка шахрайських заявок у цьому наборі становить лише близько 1.1% від загального обсягу даних.

Набір Base має 1000000 спостережень та 32 ознаки. Ознаки поділяються на цільову змінну, захищені атрибути та ознаки безпосередньо самої заявки.

Частина ознак набору даних Bank Account Fraud (BAF) та їх опис вказані у таблиці 3.1.

Таблиця 3.1 – Частина ознак набору даних Bank Account Fraud (BAF) та їх опис

Назва ознаки	Опис
fraud_bool	Цільова змінна. Бінарний індикатор шахрайства, де 1 означає шахрайську заявку, а 0 легітимну.

Продовження таблиці 3.1

Назва ознаки	Опис
customer_age	Захищений атрибут. Вікова категорія заявника, подана з кроком у 10 років
income	Захищений атрибут. Квантиль рівня доходу заявника, діапазон від 0.1 до 0.9
employment_status	Статус зайнятості особи під час подання заявки
employment_status	Статус зайнятості особи під час подання заявки
session_length_in_minutes	Загальна тривалість сесії користувача під час оформлення заявки на рахунок
keep_alive_session	Бінарна ознака того, чи підтримувалася активність сесії заявником
prev_address_months_count	Кількість місяців проживання за попередньою адресою (значення –1 вказує на відсутність інформації)
current_address_months_count	Кількість місяців проживання заявника за поточною адресою
bank_months_count	Кількість місяців з моменту відкриття попереднього рахунку в установі (значення –1, якщо клієнт новий)
intended_balcon_amount	Запланована сума первинного балансу під час відкриття рахунку
proposed_credit_limit	Запропонований банком кредитний ліміт для нового клієнта
has_other_cards	Бінарна ознака наявності в клієнта інших платіжних карток
foreign_request	Індикатор того, чи був зроблений запит на відкриття рахунку з-за кордону

Продовження таблиці 3.1

Назва ознаки	Опис
foreign_request	Індикатор того, чи був зроблений запит на відкриття рахунку з-за кордону
month	Порядковий номер місяця від 0 до 7, у якому була подана заявка, використовується для імітації часової динаміки
name_email_similarity	Розрахована метрика подібності між вказаним іменем клієнта та назвою його електронної пошти
phone_mobile_valid	Ознака валідності вказаного мобільного номера телефону
device_fraud_count	Історична кількість шахрайських дій, які були зафіксовані з використанням даного пристрою
device_distinct_emails_8w	Кількість унікальних електронних адрес, пов'язаних з даним пристроєм за останні 8 тижнів

Описову статистику для 10 найважливіших числових ознак датасету Bank Account Fraud (BAF) Base варіант можна побачити на рисунку 3.2.

	count	mean	std \
credit_risk_score	1000000.0	130.989595	69.681812
proposed_credit_limit	1000000.0	515.850891	487.025696
customer_age	1000000.0	33.689080	12.025799
keep_alive_session	1000000.0	0.576947	0.494044
income	1000000.0	0.562696	0.290300
date_of_birth_distinct_emails_4w	1000000.0	9.503544	5.033792
name_email_similarity	1000000.0	0.493694	0.289069
device_distinct_emails_8w	1000000.0	1.018312	0.180761
has_other_cards	1000000.0	0.222988	0.416251
phone_home_valid	1000000.0	0.417077	0.493076

	min	25%	50% \
credit_risk_score	-170.000000	83.000000	122.000000
proposed_credit_limit	190.000000	200.000000	200.000000
customer_age	10.000000	20.000000	30.000000
keep_alive_session	0.000000	0.000000	1.000000
income	0.100000	0.300000	0.600000
date_of_birth_distinct_emails_4w	0.000000	6.000000	9.000000
name_email_similarity	0.000001	0.225216	0.492153
device_distinct_emails_8w	-1.000000	1.000000	1.000000
has_other_cards	0.000000	0.000000	0.000000
phone_home_valid	0.000000	0.000000	0.000000

	75%	max
credit_risk_score	178.000000	389.000000
proposed_credit_limit	500.000000	2100.000000
customer_age	40.000000	90.000000
keep_alive_session	1.000000	1.000000
income	0.800000	0.900000
date_of_birth_distinct_emails_4w	13.000000	39.000000
name_email_similarity	0.755567	0.999999
device_distinct_emails_8w	1.000000	2.000000
has_other_cards	0.000000	1.000000
phone_home_valid	1.000000	1.000000

Рисунок 3.2 – Описова статистика для 10 найважливіших числових ознак датасету Bank Account Fraud (BAF) Base варіант

Описову статистику для категоріальних ознак датасету Bank Account Fraud (BAF) Base варіант можна побачити на рисунку 3.3.

	count	unique	top	freq
payment_type	1000000	5	AB	370554
employment_status	1000000	7	CA	730252
housing_status	1000000	7	BC	372143
source	1000000	2	INTERNET	992952
device_os	1000000	5	other	342728

Рисунок 3.3 – Описова статистика для категоріальних ознак датасету Bank Account Fraud (BAF) Base варіант

З метою наближення умов моделювання до реального середовища, автори набору даних навмисно заклали часову варіативність розподілу цільової змінної. Частка шахрайських транзакцій не є статичною і демонструє виражену тенденцію до зростання в останніх місяцях спостереження.

Динаміку рівня шахрайських заявок за місяцями можна побачити на рисунку 3.4.

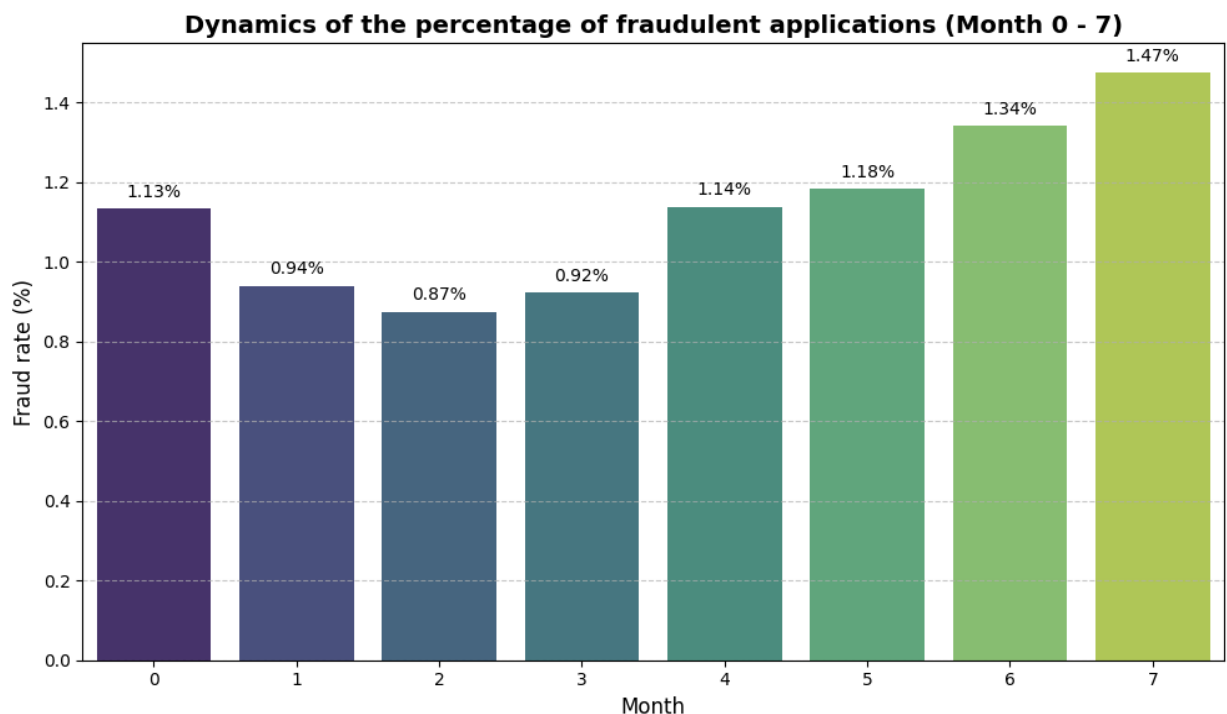


Рисунок 3.4 – Динаміка рівня шахрайських заявок за місяцями

3.4 Навчання та оцінка моделей

У даному підрозділі розглядається методологія та етапи практичної реалізації пайплайну машинного навчання, розробленого для задачі виявлення шахрайських транзакцій. Наведено комплексний опис процесу моделювання, який послідовно охоплює стратегію розбиття даних з урахуванням їхньої часової специфіки, методи попередньої обробки ознак, адаптовані під різні типи алгоритмів, а також індивідуальні підходи до налаштування та навчання моделей.

Окрему увагу приділено загальній парадигмі оцінювання якості: висвітлюються методи боротьби з жорстким дисбалансом класів та підходи до вибору критеріїв класифікації, які б задовольняли суворі вимоги до надійності антифрод-систем.

3.4.1 Розбиття даних та крос-валідація

Для запобігання витоку даних (data leakage) у майбутнє та збереження хронології транзакцій було використано часове розбиття вибірки (Time-based split) на основі змінної month (місяць):

- 1) навчальна вибірка (місяці 0–5), використовувалась для навчання моделей та пошуку оптимальних гіперпараметрів;
- 2) валідаційна вибірка (місяць 6), використовувалась для калібрування моделей, вибору оптимального порогу класифікації та налаштування параметрів алгоритму Isolation Forest;
- 3) тестова вибірка (місяць 7), виступала як повністю незалежний набір для фінальної перевірки генералізаційної здатності моделей.

Пошук оптимальних гіперпараметрів здійснювався за допомогою рандомізованого пошуку (RandomizedSearchCV), де як стратегія внутрішньої крос-валідації використовувався TimeSeriesSplit (на 3 фолди), що гарантувало

тестування моделей лише на наступних за часом спостереженнях [15]. Цільовою метрикою оптимізації було обрано PR–AUC, оскільки вона є найбільш релевантною в умовах жорсткого дисбалансу класів.

3.4.2 Попередня обробка ознак

Для забезпечення коректної роботи різних типів моделей було розроблено окремі пайплайни (Pipelines) попередньої обробки даних за допомогою ColumnTransformer. Значення -1 у числових ознаках було ідентифіковано як специфічний маркер пропущених значень та замінено на NaN.

Категоріальні ознаки для всіх алгоритмів оброблялися однаково: пропущені значення заповнювались константою “missing”, після чого застосовувалось унітарне кодування (OneHotEncoder).

Для числових ознак для математичних та метричних моделей (SVM, MLP, Isolation Forest) було застосовано імпутацію медіаною з додаванням індикатора пропусків, після чого ознаки масштабувалися за допомогою StandardScaler.

Для числових ознак для деревних ансамблів (XGBoost) застосовувалось лише масштабування (StandardScaler). Імпутація пропусків навмисно не виконувалась, оскільки алгоритм XGBoost має вбудований механізм обробки NaN значень під час побудови дерев, що дозволяє отримувати з них додатковий інформаційний сигнал.

3.4.3 Специфіка навчання моделей

До кожної моделі застосовано специфічні підходи для боротьби з дисбалансом та оптимізації обчислень.

Random Forest навчався на повному наборі даних. Проблема дисбалансу вирішувалась на рівні алгоритму за рахунок налаштування параметра `class_weight` (балансування ваг класів загальне або на рівні кожної підвибірки). Оптимізовувались глибина дерев, кількість випадкових ознак для розщеплення вузлів та мінімальна кількість об'єктів у листі.

У XGBoost для боротьби з дисбалансом розраховувався коефіцієнт `scale_pos_weight` як відношення кількості легітимних транзакцій до шахрайських. Модель оптимізувалась за метрикою PR–AUC.

Оскільки SVM (RBF) має квадратичну обчислювальну складність від розміру вибірки, перед навчанням було застосовано алгоритм випадкового недосемплювання мажоритарного класу (`RandomUnderSampler`) із співвідношенням 1:10. Так як метод опорних векторів за замовчуванням не повертає ймовірності, то після пошуку гіперпараметрів було проведено процедуру калібрування ймовірностей `CalibratedClassifierCV` на основі сигмоїдної функції/методу Платта, використовуючи незалежну валідаційну вибірку.

Навчання нейронної мережі (MLP) також супроводжувалось недосемплюванням (1:5). Для запобігання перенавчанню було активовано механізм ранньої зупинки. В процесі налаштування варіювалася архітектура прихованих шарів та коефіцієнт L2–регуляризації.

Модель Isolation Forest використовувалась у парадигмі напівкерованого навчання (`semi-supervised learning`) як метод виявлення аномалій. Модель навчалася виключно на спостереженнях нормального класу (легітимні транзакції). З міркувань продуктивності навчання проводилося на випадковій підмножині зі 100000 нормальних екземплярів. Замість класичного `RandomizedSearchCV`, оптимальний розмір підмножини для побудови дерев підбирався ітеративно шляхом оцінки функції прийняття рішень за метрикою PR–AUC на валідаційній вибірці.

3.4.4 Оцінка якості та динамічний вибір порогу

Особливістю вимог до антифрод–систем є необхідність суворого контролю кількості хибнопозитивних спрацьовувань, тобто щоб модель не блокувала легітимних клієнтів. Тому замість стандартного порогу класифікації 0.5, було реалізовано алгоритм динамічного підбору порогу згідно рекомендацій авторів набору даних.

Процедура оцінки складалась з наступних етапів:

- 1) за допомогою ROC–кривої на валідаційній вибірці розраховувався такий оптимальний поріг, за якого рівень хибнопозитивних спрацьовувань не перевищував 5% ($FPR \leq 0.05$) [15];
- 2) знайдений поріг застосовувався для бінаризації прогнозів на незалежній тестовій вибірці;
- 3) якість фінальних прогнозів оцінювалась за допомогою метрики PR–AUC, а також точкових метрик, стійких до дисбалансу;
- 4) для візуального аналізу помилок першого та другого роду для кожної моделі було побудовано матриці помилок. Фінальні результати відсортовано за значенням PR–AUC.

3.5. Аналіз результатів

Найкращі гіперпараметри для алгоритму Random Forest (RF) після навчання на наборі даних Bank Account Fraud (BAF) вказані у табл. 3.2.

Таблиця 3.2 – Найкращі гіперпараметри для алгоритму Random Forest (RF) після навчання на наборі даних Bank Account Fraud (BAF)

Параметр	Опис параметру	Значення
clf__n_estimators	Кількість базових дерев рішень в ансамблі	500
clf__min_samples_leaf	Мінімальна кількість об'єктів у термінальному вузлі (листяку)	2
clf__max_features	Розмірність випадкового підпростору ознак для пошуку спліт-критерію	log2
clf__max_depth	Гранична глибина ієрархічної структури дерева	30
clf__class_weight	Метод вагового регулювання для компенсації дисбалансу класів	balanced

Найкращі гіперпараметри для Support Vector Machine (SVM) з ядром RBF після навчання на наборі даних Bank Account Fraud (BAF) вказані у табл. 3.3.

Таблиця 3.3 – Найкращі гіперпараметри для Support Vector Machine (SVM) з ядром RBF після навчання на наборі даних Bank Account Fraud (BAF)

Параметр	Опис параметру	Значення
clf__gamma	Коефіцієнт інтенсивності впливу окремих навчальних прикладів (обернений радіус ядра)	auto
clf__C	Параметр регуляризації, що визначає компроміс між шириною роздільної смуги та помилками класифікації	1
clf__max_depth	Гранична глибина кожного дерева рішень	4
clf__learning_rate	Швидкість навчання (розмір кроку корекції помилок попередніх дерев)	0.05

Продовження таблиці 3.3

Параметр	Опис параметру	Значення
clf__gamma	Мінімальне зменшення функції втрат, необхідне для створення нового розгалуження (вузла)	0.5
clf__colsample_bytree	Частка випадково відібраних ознак для побудови кожного дерева	0.8

Найкращі гіперпараметри для багатошарового перцептрону (MLP) після навчання на наборі даних Bank Account Fraud (BAF) вказані у табл. 3.4.

Таблиця 3.4 – Найкращі гіперпараметри для багатошарового перцептрону (MLP) після навчання на наборі даних Bank Account Fraud (BAF)

Параметр	Опис параметру	Значення
clf__learning_rate_init	Початкова швидкість навчання, визначає розмір кроку оновлення вагових коефіцієнтів під час оптимізації (градієнтного спуску)	0.01
clf__hidden_layer_sizes	Структура та розмірність прихованих шарів нейронної мережі	(128, 64, 32)
clf__alpha	Коефіцієнт L2-регуляризації (штраф за великі значення ваг)	0.0001

Найкращі гіперпараметри для XGBoost після навчання на наборі даних Bank Account Fraud (BAF) вказані у табл. 3.5.

Таблиця 3.5 – Найкращі гіперпараметри для XGBoost після навчання на наборі даних Bank Account Fraud (BAF)

Параметр	Опис параметру	Значення
clf__subsample	Частка випадково відібраних об'єктів вибірки для побудови кожного дерева	0.7
clf__n_estimators	Загальна кількість дерев (ітерацій бустингу) в ансамблі	400

Найкращі гіперпараметри Isolation Forest (IF) після навчання на наборі даних Bank Account Fraud (BAF) вказані у табл. 3.6

Таблиця 3.6 – Найкращі гіперпараметри Isolation Forest (IF) після навчання на наборі даних Bank Account Fraud (BAF)

Параметр	Опис параметру	Значення
max_samples	Кількість об'єктів у кожній випадковій підвибірці для побудови дерева ізоляції	256
contamination	Очікувана частка аномальних спостережень у наборі даних.	0.01

Порівняльна характеристика ефективності моделей на тестовій вибірці набору даних BAF занесені у табл. 3.7.

Таблиця 3.7 – Порівняльна характеристика ефективності моделей на тестовій вибірці набору даних BAF

Model	Accuracy	PR-AUC	F1-score	MCC	Оптимальний поріг
XGBoost	0.955144	0.216674	0.255910	0.280240	0.809660
SVM (RBF)	0.943486	0.180813	0.213988	0.244820	0.019436
MLP	0.945468	0.177273	0.215421	0.243558	0.449798
Random Forest	0.961959	0.155795	0.239785	0.246346	0.118877
Isolation Forest	0.940099	0.020298	0.037817	0.018588	-0.030570

Аномально високе значення Ассурасу (понад 94% у кожної з моделей) досягається переважно за рахунок правильного розпізнавання домінуючого легального класу заявок, маскуючи при цьому низьку якість виявлення самих махінацій.

Низькі абсолютні значення метрик PR-AUC, F1-score та MCC є типовими та прийнятними для задач виявлення шахрайства із заявками (Application Fraud) в умовах екстремального дисбалансу класів. Оскільки частка шахрайських дій у генеральній сукупності є мізерною, навіть за умови жорсткого утримання частки хибнопозитивних спрацьовувань на низькому рівні ($FPR \leq 5\%$), абсолютна кількість легітимних клієнтів, помилково класифікованих як шахраї, все одно значно перевищує кількість істинно виявлених зловмисників. Це математично зумовлює стрімке падіння показника Precision, що автоматично тягне за собою зниження F1-score та площі під кривою PR-AUC. Подібні показники об'єктивно відображають реальну складність проблеми: табличні дані у сфері банківського шахрайства є вкрай зашумленими та складними для моделювання, тому моделі, що здатні

працювати за таких жорстких обмежень, є практично значущими для бізнесу, незважаючи на номінально низькі метрики [15].

За результатами тестування, найкращою моделлю виявився алгоритм XGBoost, який досяг найвищих значень за всіма цільовими метриками. Така перевага пояснюється здатністю градієнтного бустингу послідовно фокусуватися на складнороздільних прикладах міноритарного класу під час навчання, що робить його ефективнішим за нейромережевий чи метричний підходи.

Оскільки Isolation Forest є методом навчання без вчителя, він не використовує відомі мітки цільового класу для налаштування, а класифікує об'єкти виключно за їх ізольованістю в просторі ознак. Проте специфіка шахрайства під час відкриття банківських рахунків полягає в тому, що зловмисники навмисно маскують свої заявки під анкети легальних клієнтів, через що такі спостереження рідко виступають явними структурними викидами. Відповідно, без цілеспрямованих вказівок “вчителя” алгоритм виявився нездатним відрізнити ці приховані патерни махінацій від нормальної варіативності даних, що й призвело до критичного падіння якості класифікації.

Побудовано матриці неточностей для всіх п'яти моделей. Шахрайство визначено як позитивний клас, а легітимні транзакції як негативний.

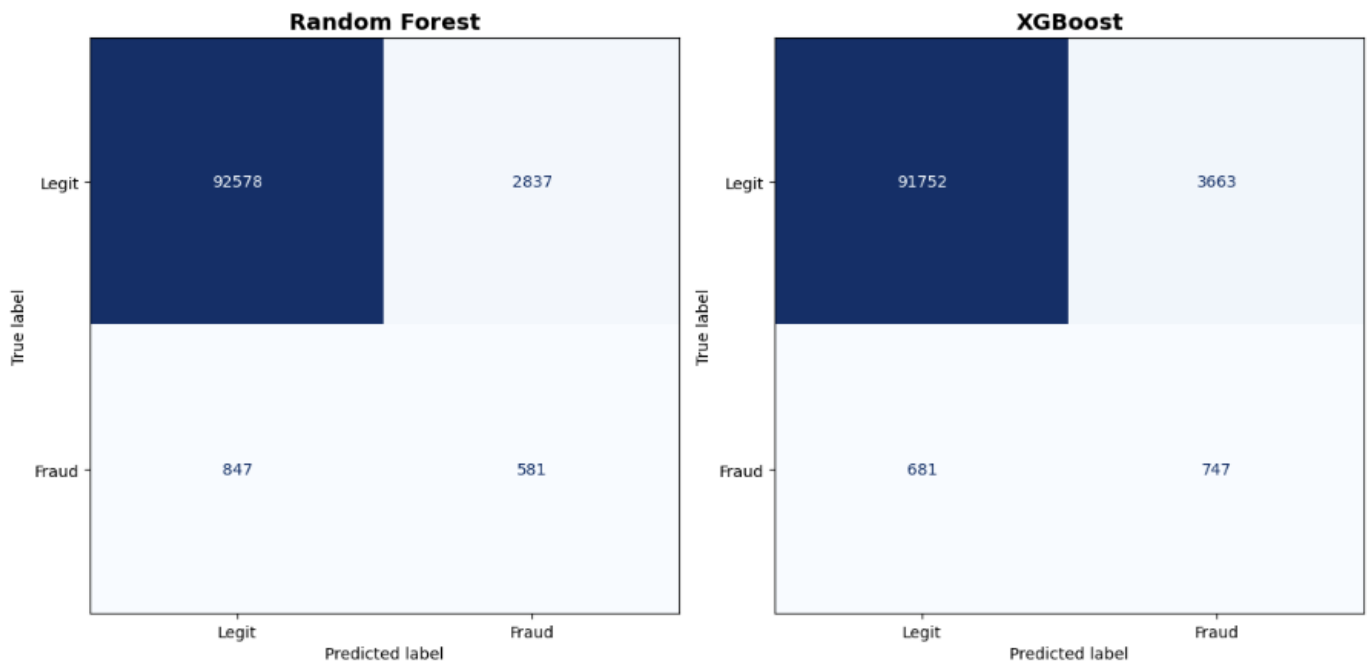


Рисунок 3.5 – Матриці неточностей для Random Forest та XGBoost

З рисунку 3.5 видно, що алгоритм XGBoost демонструє вищу чутливість у виявленні цільового класу шахрайства, класифікувавши 747 транзакцій як істинно позитивні (TP) порівняно з 581 у моделі Random Forest. Відповідно, XGBoost характеризується меншою кількістю хибно негативних результатів (FN = 681 проти 847 у Random Forest), що вказує на його кращу здатність мінімізувати критичний пропуск реальних аномалій. Водночас модель Random Forest виявляє дещо вищу специфічність, забезпечуючи більшу кількість істинно негативних класифікацій легітимних транзакцій (TN = 92578) та знижений рівень хибно позитивних спрацьовувань (FP = 2837 порівняно з 3663 у XGBoost).

З огляду на специфіку предметної області, де пріоритетом є максимізація повноти виявлення шахрайства, XGBoost виглядає більш доцільним та ефективним рішенням, незважаючи на супутнє зростання похибки першого роду.

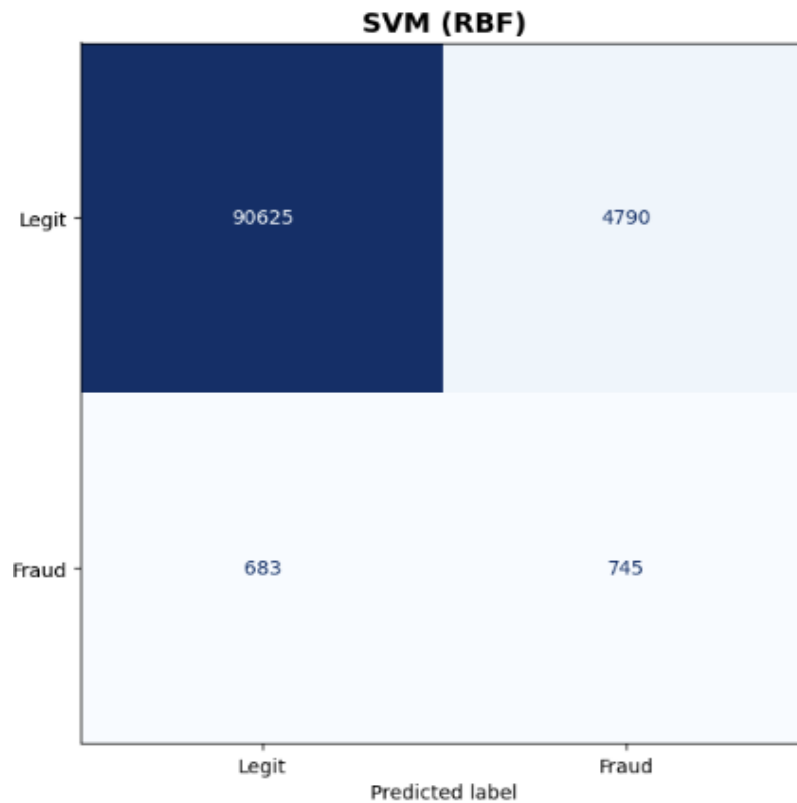


Рисунок 3.6 – Матриця неточностей для SVM з RBF-ядром

З рисунку 3.6, модель SVM (RBF) демонструє значну кількість хибнопозитивних спрацьовувань ($FP = 4790$), що вказує на невисоку точність (precision) класифікатора при ідентифікації шахрайства на тлі великого масиву легітимних даних. Водночас співвідношення істинно позитивних ($TP = 745$) та хибнонегативних ($FN = 683$) результатів свідчить про помірну повноту (recall) моделі, оскільки вона пропускає майже половину реальних фродових випадків.

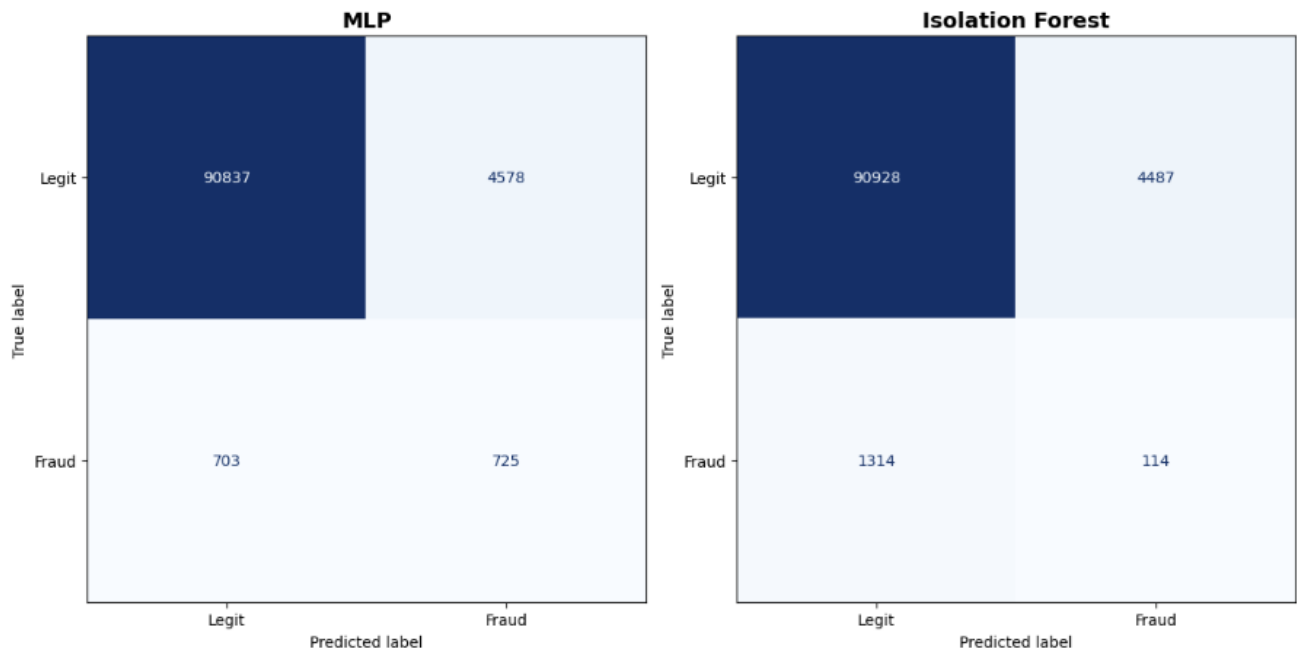


Рисунок 3.7 – Матриці неточностей для MLP та Isolation Forest

З рисунку 3.7, модель MLP значно ефективніше ідентифікує цільовий клас шахрайства, демонструючи суттєво вищий показник істинно позитивних результатів ($TP = 725$) порівняно з Isolation Forest ($TP = 114$), хоча це і супроводжується незначним зростанням кількості хибнопозитивних помилок ($FP = 4578$ проти 4487 у Isolation Forest). Водночас алгоритм Isolation Forest характеризується критично високим рівнем хибнонегативних спрацьовувань ($FN = 1314$), пропускаючи переважну більшість реальних аномалій, що робить його менш придатним для даного набору даних незважаючи на точність визначення легітимних транзакцій ($TN = 90928$).

SHAP (SHapley Additive exPlanations) аналіз застосовується для забезпечення локальної та глобальної інтерпретованості моделі шляхом точної кількісної оцінки впливу кожної окремої змінної на фінальне прогнозування [17].

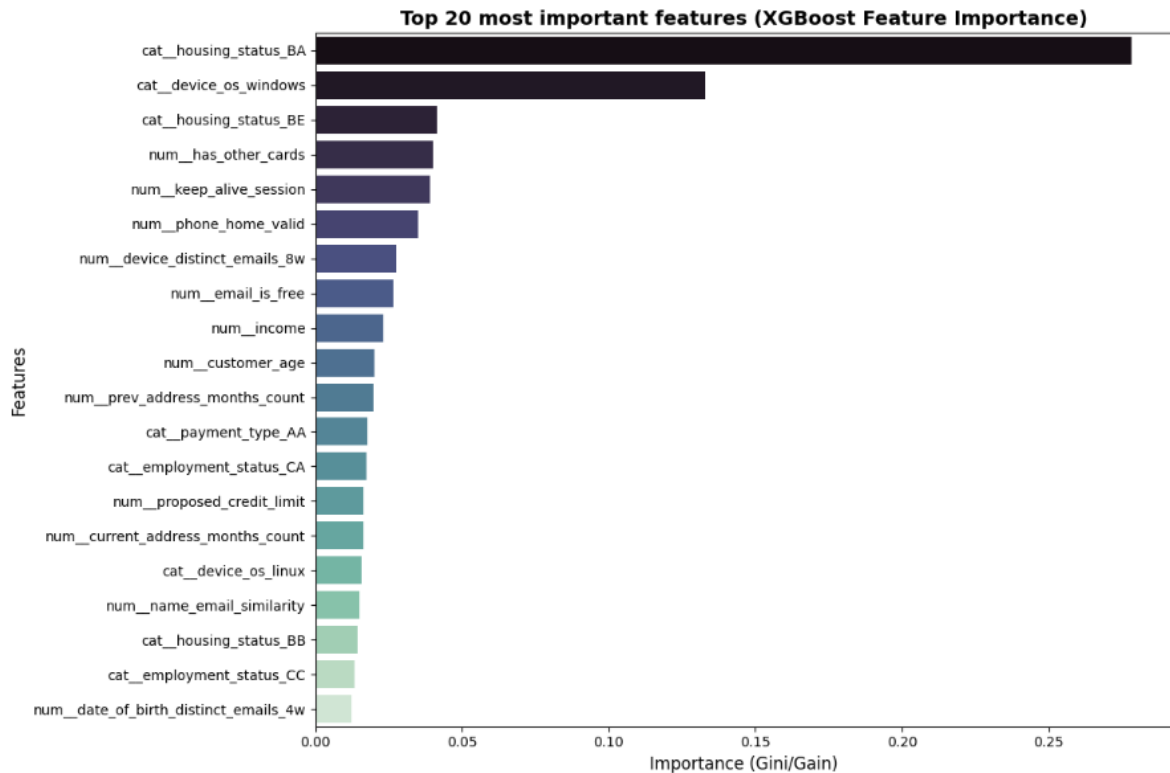


Рисунок 3.8 – Ранжування топ–20 найважливіших предикторів алгоритму XGBoost

Як видно з графіка на рисунку 3.8, який візуалізує глобальну важливість ознак на основі метрики Gini/Gain для моделі XGBoost, ключовий внесок у здатність моделі розділяти класи роблять саме категоріальні змінні. Найвагомішим предиктором виявилася ознака `cat_housing_status_BA`, що відображає специфічний статус проживання клієнта (найчастіше це категорія орендованого житла або проживання з батьками). Другою за інформативністю виступає змінна `cat_device_os_windows`, яка фіксує використання операційної системи Windows під час транзакції чи подачі заявки.

Розподіл значень SHAP для оцінки локального та глобального впливу атрибутів на результати класифікації алгоритмом XGBoost можна побачити на рисунку 3.9.

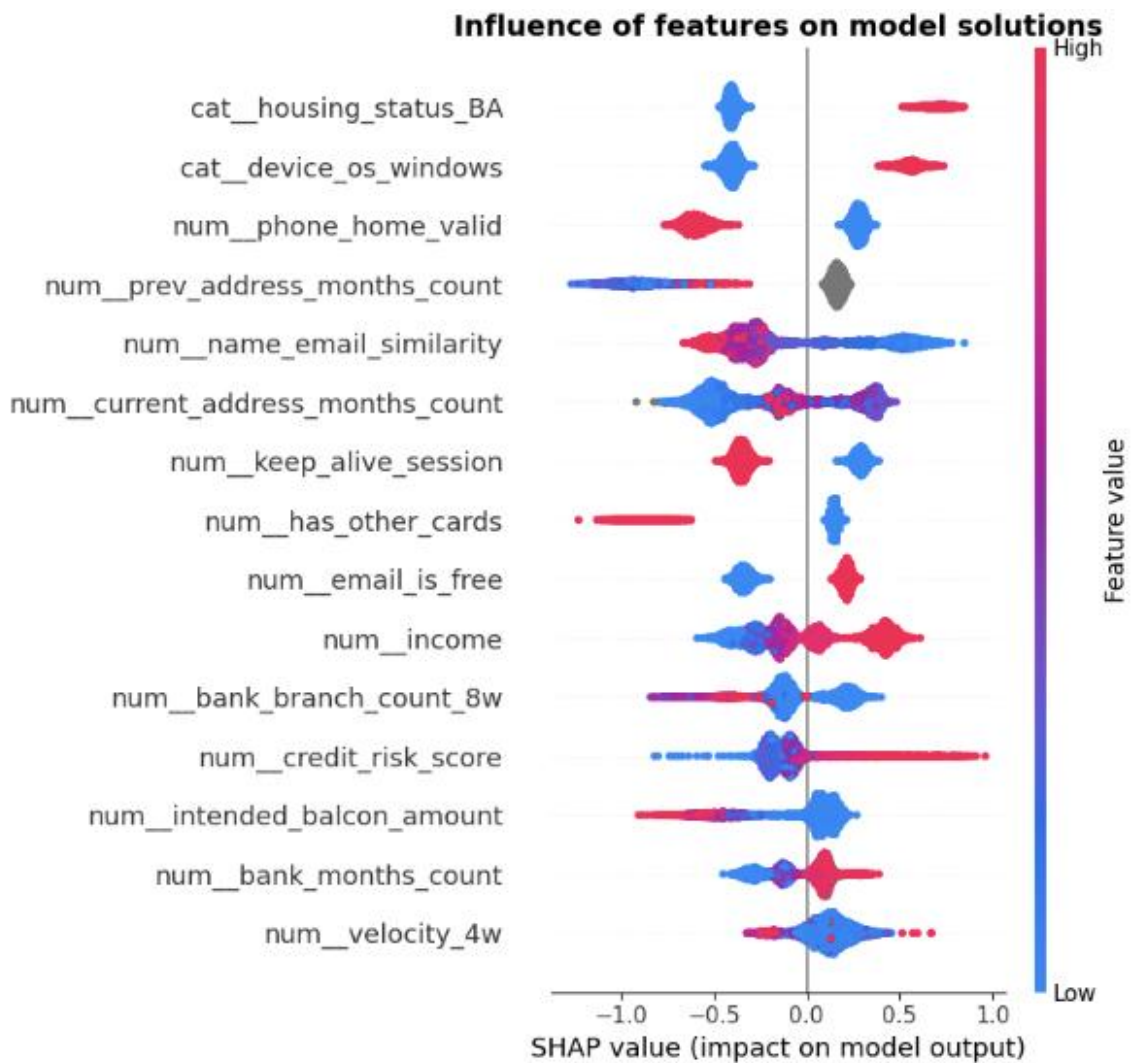


Рисунок 3.9 – Розподіл значень SHAP для оцінки локального та глобального впливу атрибутів на результати класифікації алгоритмом XGBoost

Найбільший прямий вплив на прогнози оптимізованої моделі XGBoost мають ознаки `cat_housing_status_BA` та `cat_device_os_windows`, високі значення яких суттєво підвищують імовірність цільового класу.

Водночас для низки числових змінних, зокрема `num_phone_home_valid` та `num_name_email_similarity`, спостерігається виражена зворотна залежність, оскільки саме їх низькі значення дають значний позитивний внесок у вихід моделі. З поведінкової точки зору низька подібність імені та пошти пояснюється тим, що шахраї, особливо під час масових атак, часто застосовують автоматично згенеровані випадкові поштові скриньки, які лексично не збігаються з іменами створених синтетичних ідентичностей.

Також показовою є ознака `num__has__other__cards`, високі показники якої виступають потужним стримувальним фактором, різко зсуваючи SHAP-значення в негативний бік. Загальний розподіл щільності вказує на наявність нелінійних впливів, де екстремальні значення певних предикторів (наприклад, `num__credit__risk__score`) утворюють довгі “хвости”, що свідчить про їхню вирішальну роль у класифікації граничних спостережень.

3.6 Висновки до розділу 3

У третьому розділі реалізовано підхід до побудови зазначених моделей машинного навчання. Для забезпечення надійності оцінок та уникнення витoku даних у майбутнє застосовано часове розбиття вибірки, а також розроблено індивідуальні пайплайни попередньої обробки ознак залежно від специфіки обраних алгоритмів. Процес навчання охопив моделі Random Forest, XGBoost, SVM, MLP та Isolation Forest із застосуванням методів оптимізації гіперпараметрів і адаптивних підходів до вирішення жорсткого дисбалансу класів. З метою мінімізації безпідставного блокування легітимних клієнтів був реалізований алгоритм динамічного підбору порогу класифікації, який гарантував рівень хибнопозитивних спрацьовувань не більше 5%.

Аналіз результатів було здійснено на основі оцінки моделей на незалежних тестових даних за допомогою метрик оцінки якості. Для найкращої моделі (XGBoost) також проведено SHAP аналіз, що дозволило забезпечити локальну та глобальну інтерпретованість моделі шляхом точної кількісної оцінки впливу кожної окремої змінної на фінальне прогнозування.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО–ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, розробленого для ідентифікації шахрайства під час відкриття нових банківських рахунків з використанням методів машинного навчання.

Щоб визначити найбільш раціональний шлях побудови модуля, здійснюється порівняння альтернативних варіантів його реалізації. Остаточний вибір базується на пошуку балансу між фінансовими витратами та технічними метриками. Головним інструментом для такого оцінювання обрано метод функціонально–вартісного аналізу (ФВА).

Функціонально–вартісний аналіз є дієвим підходом для об'єктивного розрахунку фактичної собівартості створення ІТ–продукту. Його застосування дозволяє оптимізувати бюджет проєкту завдяки відбору найдоцільніших алгоритмічних рішень. Ключове завдання ФВА у цьому контексті полягає у забезпеченні максимальної цінності системи для фінансової установи (у вигляді надійного захисту від грошових та репутаційних втрат) за умови раціоналізації витрат на її розробку.

Аналіз проводиться поетапно: спочатку визначаються основні та допоміжні функції продукту, формуються варіанти їх виконання, після чого на основі технічних параметрів та розрахованої собівартості обирається остаточний варіант реалізації, який і буде впроваджений.

4.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко–економічного аналізу розробки системи ідентифікації шахрайства під час

відкриття нових банківських рахунків. Оскільки рішення щодо проектування та реалізації алгоритмів машинного навчання впливають на ефективність та швидкодію всієї системи, кожна окрема підсистема (наприклад, модулі збору даних, обробки незбалансованих вибірок та бінарної класифікації) має задовольняти загальні критерії якості. Тому фактичний аналіз являє собою оцінку функцій програмного продукту, призначеного для збору, попередньої обробки багатовимірних клієнтських даних та виявлення маркерів ризику в нових заявках.

Основні технічні вимоги до програмного продукту є наступними:

- 1) функціонування на стандартному обчислювальному обладнанні банку без необхідності залучення дороговартісних вузькоспеціалізованих апаратних потужностей;
- 2) зручність інтерфейсу та зрозумілість (інтерпретованість) результатів роботи моделей для співробітників служби безпеки банку;
- 3) висока швидкість обробки даних та виявлення аномалій у режимі реального часу (безпосередньо під час подачі заявки клієнтом);
- 4) можливість зручного масштабування (при зростанні потоку реєстрацій) та легкість обслуговування (зокрема, донавчання та оновлення моделей);
- 5) мінімальні витрати на інтеграцію та впровадження програмного продукту в існуючу IT-інфраструктуру фінансової установи.

4.2 Обґрунтування функцій програмного продукту

Головна функція F – розробка програмного продукту, який вирішує задачу виявлення шахрайства на основі аналізу клієнтських даних та надає оцінку ймовірності шахрайських дій.

Беручи за основу цю функцію, можна виділити наступні:

$F1$ – вибір мови програмування;

$F2$ – вибір бібліотек та фреймворків машинного навчання;

$F3$ – вибір середовища розробки;

Кожна з цих функцій має декілька варіантів реалізації:

1. Функція $F1$:

A. Python

B. C++

2. Функція $F2$:

A. Scikit-Learn

B. Tensorflow

3. Функція $F3$:

A. Kaggle Notebook

B. Visual Studio Code

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

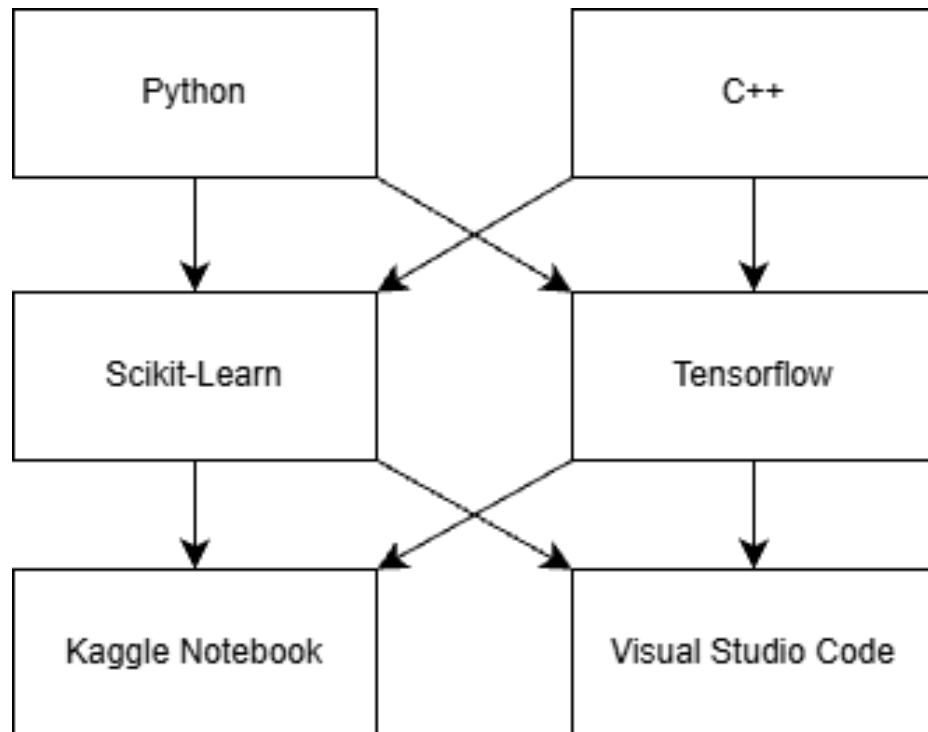


Рисунок 4.1 – Морфологічний карта системи

На основі морфологічної карти проведено аналіз переваг і недоліків

кожного варіанту через позитивно–негативну матрицю (табл. 4.1).

Таблиця 4.1 – Позитивно–негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F1	А	Широкий набір інструментів, легка робота з пайплайнами	Нижча швидкість виконання у порівнянні з компільованими мовами
	Б	Максимальна продуктивність та швидкість	Складна розробка, відсутність зручних інструментів для даної задачі
F2	А	Вбудовані методи обробки дисбалансу класів, легко інтерпретувати моделі	Обмежені можливості для побудови складних архітектур глибокого навчання
	Б	Потужність для складних нейромереж	Надлишковий для класичних табличних даних
F3	А	Інтерактивність, наявність безкоштовних хмарних ресурсів	Не підходить для розгортання моделі у продакшен
	Б	Зручний дебагінг, повноцінна інтеграція з Git	Потребує самостійного налаштування локального середовища

На основі проведеного аналізу позитивних та негативних сторін для кожної підфункції, було зроблено вибір на користь конкретних технологічних рішень:

Для *F1* перевагу отримала мова Python. Завдяки широкому набору інструментів для Data Science та надзвичайно легкій роботі з пайплайнами, вона дозволяє значно швидше реалізувати систему виявлення шахрайства. Хоча C++ забезпечує максимальну швидкість, складність розробки та відсутність зручних інструментів для розвідувального аналізу роблять його менш доцільним для цієї задачі.

Для $F2$ у першому варіанті обрано використання бібліотеки Scikit–Learn. Це рішення забезпечує наявність вбудованих методів обробки дисбалансу класів (що є критично важливим при роботі з незбалансованими транзакціями) та дозволяє легко інтерпретувати отримані моделі. Для другого варіанту розглядається TensorFlow, що надає потужні можливості для побудови складних нейромереж, хоча для класичних табличних даних цей інструмент може бути дещо надлишковим.

Для $F3$ базовим середовищем розробки обрано Kaggle Notebook. Його інтерактивність та доступ до безкоштовних хмарних ресурсів роблять його ідеальним для проведення розвідувального аналізу даних та швидкого тестування гіпотез. Оптимізована стратегія у другому варіанті передбачає перехід до Visual Studio Code, що є необхідним кроком для написання чистого продакшен коду, зручного дебагінгу та повноцінної інтеграції з Git при підготовці системи до розгортання.

За результатами аналізу сформовано два варіанти реалізації програмного продукту:

Варіант 1: $F1(a) - F2(a) - F3(a)$

Варіант 2: $F1(a) - F2(b) - F3(b)$

4.3 Обґрунтування системи параметрів програмного продукту

На основі даних, розглянутих вище, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

$X1$ – швидкодія мови програмування;

$X2$ – час навчання даних;

$X3$ – потенційний об'єм програмного коду;

$X4$ – об'єм пам'яті для обчислень та збереження даних;

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у таблиці 4.2.

Таблиця 4.1 – Основні параметри ПП

Назва параметра	Умовне позначення	Одиниці виміру	Гірше	Середнє	Краще
Швидкодія мови програмування	X1	мс	800	15000	25000
Час навчання даних	X2	хв	45	15	5
Потенційний об'єм програмного коду	X3	кількість рядків коду	1000	500	250
Об'єм пам'яті для обчислень та збереження даних	X4	МБ	1024	512	256

За даними таблиці 4.2 будуються графічні характеристики параметрів – рис.4.2 – рис. 4.5.

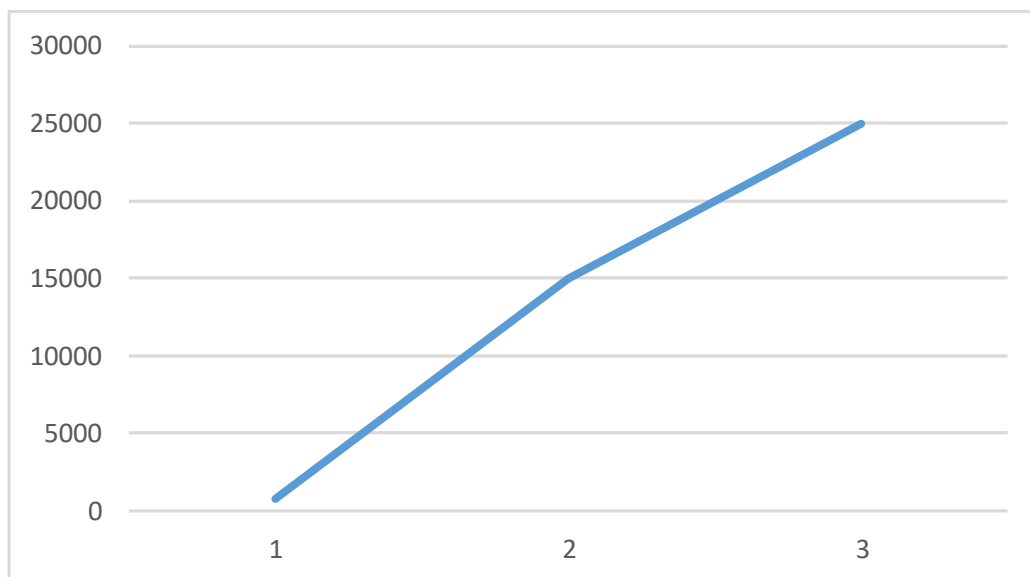


Рисунок 4.2 – X1, швидкодія мови програмування

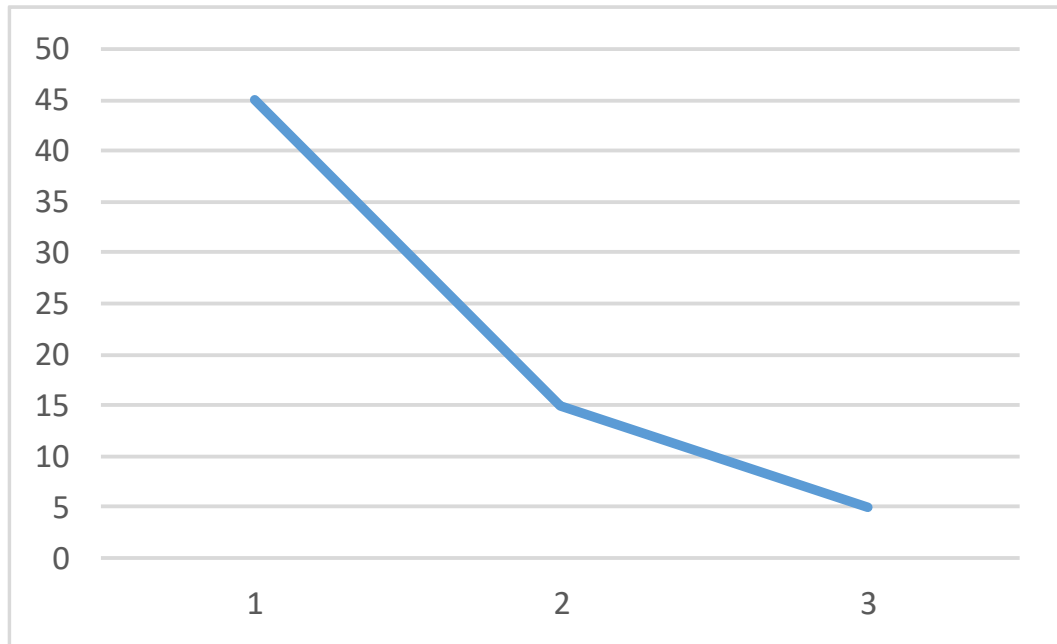


Рисунок 4.3 – X2, час навчання даних

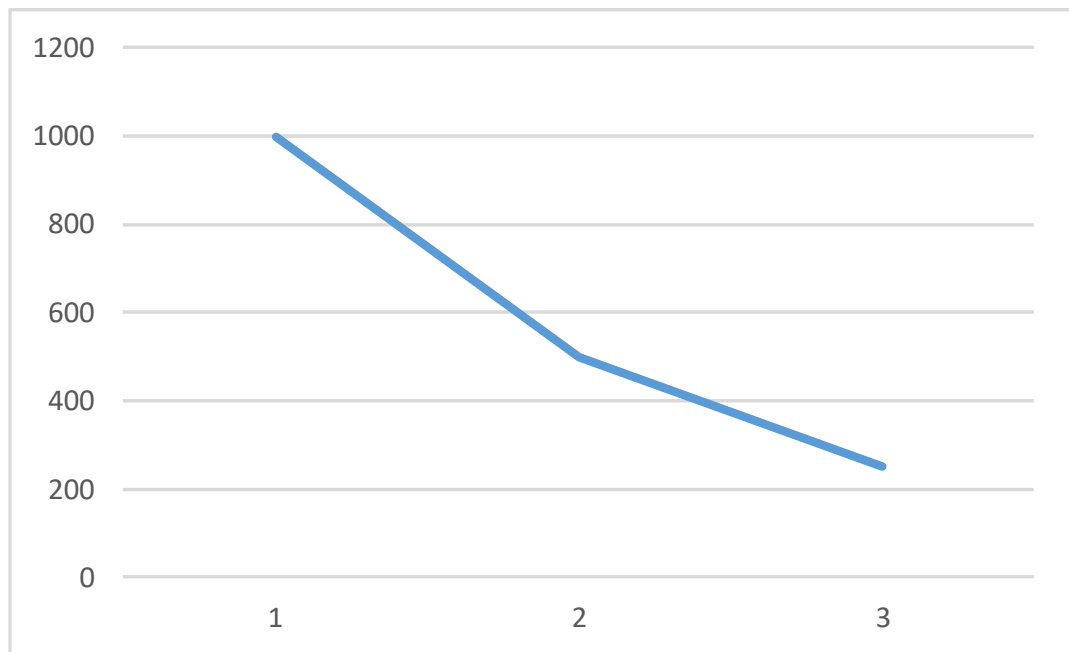


Рисунок 4.4 – X3, потенційний об'єм програмного коду

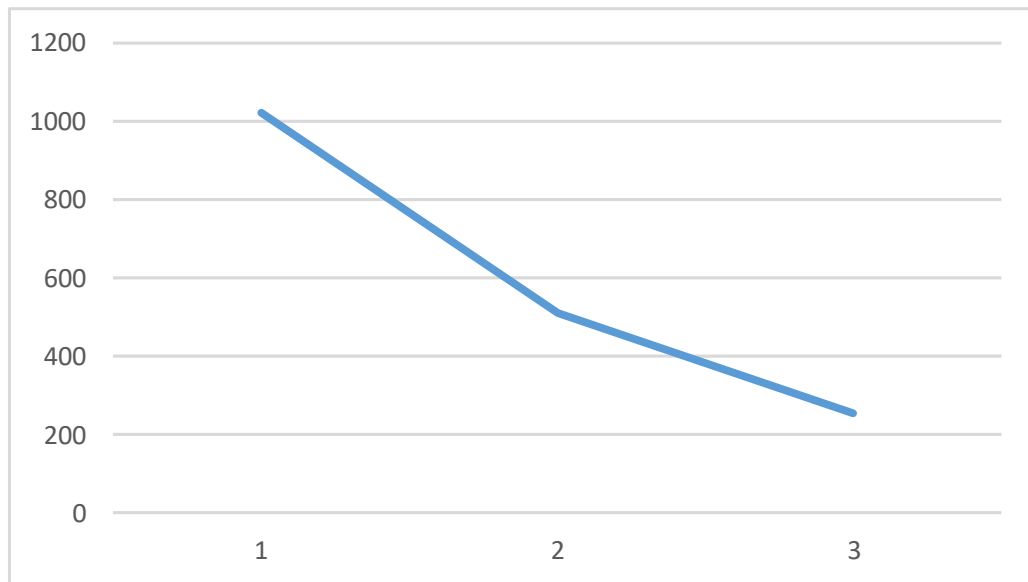


Рисунок 4.5 – X4, об'єм пам'яті для обчислень та збереження даних

4.4 Аналіз експертного оцінювання параметрів

Для визначення вагомості кожного параметра залучено експертну групу з 7 осіб – фахівців у галузі інтелектуального аналізу даних та системного аналізу. Кожен експерт незалежно присвоїв ранги параметрам від 1 до 4, де 1 – найважливіший параметр (пріоритет на точність та швидкість обчислень). Результати зведено у таблицю 4.3.

Таблиця 4.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів в R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкість мови програмування	мс	2	2	1	2	3	2	1	25	7,5	56,25
X2	Час навчання даних	хв	1	1	2	1	1	1	2	8	-9,5	90,25

Продовження таблиці 4.3

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X3	Потенційний об'єм програмного коду	кількість рядків коду	4	4	3	4	3	4	4	21	3,5	12,25
X4	Об'єм пам'яті для обчислень та збереження даних	МБ	3	3	4	3	3	3	3	16	-1,5	2,25
	Разом									70	0	161

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) загальна сума рангів:

$$R_i = \sum_{j=1}^N r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70$$

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = \frac{70}{4} = 17,5$$

в) відхилення суми рангів кожного параметра від середньої:

$$\Delta_i = R_i - T$$

Сума відхилень по всіх параметрах повинна дорівнювати 0

г) загальна сума квадратів відхилень:

$$S = \sum_{i=1}^N \Delta_i^2 = 161$$

д) коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 161}{7^2(4^3 - 4)} \Rightarrow W_k = 0,66$$

Ранжування можна вважати достатньо достовірним, оскільки знайдений коефіцієнт узгодженості наближений до нормативного значення та показує прийнятний рівень узгодженості експертних оцінок. Ступінь переваги i -го параметра над j -тим визначається числовим значенням a_{ij} та по формулі:

$$a_{ij} = \{1.5 \text{ при } X_i > X_j \text{ } 1.0 \text{ при } X_i = X_j \text{ } 0.5 \text{ при } X_i < X_j\}$$

Результати попарного порівняння зведено у таблицю 4.4.

Таблиця 4.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	>	>	>	>	>	>	>	1,5
X1 і X3	>	>	>	=	<	=	>	>	1,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X2 і X3	<	<	<	<	<	<	<	<	0,5
X2 і X4	<	<	<	=	<	<	<	<	0,5
X3 і X4	>	>	<	>	>	=	>	>	1,5

З отриманих числових оцінок переваги складемо матрицю $A = ||a_{ij}||$ та розрахуємо вагомість K_{bi} кожного параметра за наступними формулами:

$$K_{bi} = \frac{b_i}{\sum_{i=1}^n b_i}$$

$$b_i = \sum_{j=1}^N a_{ij}$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за формулами:

$$K_{Bi} = \frac{b'_i}{\sum_{i=1}^n b'_i}$$

$$b'_i = \sum_{j=1}^N a_{ij} b_j$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.	
	X1	X2	X3	X4	b_i	K_{Bi}	b_i^1	K_{Bi}^1
X1	1	1,5	1,5	1,5	5	0,3	22,25	0,29
X2	1,5	1	0,5	0,5	2,5	0,15	14,5	0,19
X3	1,5	0,5	1	1,5	4,5	0,27	20	0,26
X4	1,5	0,5	1,5	1	4,5	0,27	20	0,26
Всього:					16,5	1	76,75	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту реалізації окремо.

Коефіцієнт технічного рівня для кожного варіанта розраховується за формулою:

$$K_K(j) = \sum_{i=1}^n K_{Bi,j} B_{i,j}$$

де n – кількість параметрів;

K_{Bi} – коефіцієнт вагомості i -го параметра;

B_i – бальна оцінка i -го параметра.

Бальні оцінки визначаються з графіків (рис. 4.2 – 4.5) на основі абсолютних значень параметрів для кожного варіанту.

Варіант 1: швидкодія роботи алгоритму 250 мс, час навчання моделі 15 хв, об'єм коду 150 рядків, споживання пам'яті 300 МБ.

Варіант 2: швидкодія роботи алгоритму 400 мс, час навчання моделі 45 хв, об'єм коду 400 рядків, споживання пам'яті 1500 МБ.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F ₁	А	X1	20000	90	0,29	26,1
		X2	15	95	0,19	18
		X3	200	90	0,26	23,4
		X4	300	85	0,26	22,1
F ₂	А	X1	700	75	0,29	21,75
		X2	45	40	0,19	7,6
		X3	700	60	0,26	15,6
		X4	1000	50	0,26	13

За даними з таблиці 4.6 за формулою:

$$K_K = K_{TY} [F_{1k}] + K_{TY} [F_{2k}] + \dots + K_{TY} [F_{zk}]$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 26,1 + 18 + 23,4 + 22,1 = 88,60$$

$$K_{K2} = 21,75 + 7,6 + 15,6 + 13 = 57,95$$

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

- 1) розробка проекту програмного продукту;
- 2) розробка програмної оболонки.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як

$$T_0 = T_p \cdot K_{\Pi} \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}$$

де T_p – трудомісткість розробки ПП;

K_{Π} – поправочний коефіцієнт;

$K_{СК}$ – коефіцієнт на складність вхідної інформації;

K_M – коефіцієнт рівня мови програмування;

$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення.

Для першого завдання $T_p = 80$ людино-днів:

$$T_1 = 80 \cdot 1,8 \cdot 1 \cdot 0,8 \cdot 1 = 115,2 \text{ людино-днів}$$

Для другого завдання: $T_p = 28$ людино-днів:

$$T_2 = 28 \cdot 0,9 \cdot 1 \cdot 0,8 \cdot 1 = 20,2 \text{ людино-днів}$$

Загальна трудомісткість для кожного варіанту в людино-годинах:

$$T_I = (115,2 + 20,2 + 4,8 + 20,2) \cdot 8 = 1\,283,2 \text{ людино-годин}$$

$$T_{II} = (115,2 + 20,2 + 6,91 + 20,2) \cdot 8 = 1\,300 \text{ людино-годин}$$

Найбільш високу трудомісткість має варіант II.

У розробці беруть участь розробник з місячним окладом 20 000 грн та науковий керівник з окладом 40 000 грн. Середня погодинна ставка, рахується за формулою:

$$C_q = \frac{M}{T_m \cdot t} \text{ грн.}$$

де M – місячний оклад працівників, T_m – кількість робочих днів на місяць, t – кількість робочих годин в день.

$$C_q = \frac{20000 + 40000}{2 \cdot 21 \cdot 8} = 178,85 \text{ грн/год}$$

Заробітна плата вираховується по даній формулі:

$$C_{3П} = C_q \cdot T_i \cdot КД$$

де C_q – величина погодинної оплати праці програміста;

T_i – трудомісткість відповідного завдання;

КД – норматив, який враховує додаткову заробітну плату.

Заробітна плата з урахуванням додаткової (КД = 1,2):

$$\text{I. } C_{3П} = 178,85 \cdot 1283,2 \cdot 1,2 = 275\,400,4 \text{ грн}$$

$$\text{II. } C_{3П} = 178,85 \cdot 1300 \cdot 1,2 = 279\,006,0 \text{ грн}$$

Відрахування на єдиний соціальний внесок (22%):

$$\text{I. } C_{ВІД} = 275400,4 \cdot 0,22 = 60\,588 \text{ грн}$$

$$\text{II. } C_{ВІД} = 279006,0 \cdot 0,22 = 61\,381 \text{ грн}$$

Визначимо витрати на оплату однієї машино-години. ЕОМ обслуговує розробника з окладом 20 000 грн, коефіцієнт зайнятості 0,2:

$$C_{\Gamma} = 12 \cdot 20\,000 \cdot 0,2 = 48\,000 \text{ грн}$$

З урахуванням додаткової заробітної плати:

$$C_{\text{ВІД}}(\text{ЕОМ}) = 48\,000 \cdot (1 + 0,2) = 57\,600 \text{ грн}$$

Відрахування на соціальний внесок:

$$C_{\text{ВІД}}(\text{ЕОМ}) = 57\,600 \cdot 0,22 = 12\,672 \text{ грн}$$

Амортизаційні відрахування при нормі амортизації 25% та вартості ноутбука 40 000 грн:

$$C_A = K_{\text{ТМ}} \cdot K_A \cdot C_{\text{ПР}} = 1,15 \cdot 0,25 \cdot 40\,000 = 11\,500 \text{ грн}$$

де $K_{\text{ТМ}}$ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

K_A – річна норма амортизації;

$C_{\text{ПР}}$ – договірна ціна приладу.

Витрати на ремонт та профілактику:

$$C_p = K_{\text{ТМ}} \cdot C_{\text{ПР}} \cdot K_p = 1,15 \cdot 40\,000 \cdot 0,05 = 2\,300 \text{ грн}$$

де K_p – відсоток витрат на поточні ремонти.

Ефективний річний фонд часу роботи ПК:

$$T_{\text{ЕФ}} = (D_K - D_B - D_C - D_P) \cdot t_3 \cdot K_B = (365 - 104 - 11 - 8) \cdot 8 \cdot 0,85 = 1\,645,6 \text{ годин}$$

де D_K – календарна кількість днів у році;

D_B, D_C – відповідно кількість вихідних та святкових днів;

D_P – кількість днів планових ремонтів устаткування;

t – кількість робочих годин в день;

K_B – коефіцієнт використання приладу у часі протягом зміни.

Витрати на електроенергію (тариф 13,64 грн/кВт·год, потужність 0,2 кВт):

$$C_{\text{ЕЛ}} = T_{\text{ЕФ}} \cdot N_C \cdot K_3 \cdot C_{\text{ЕН}} = 1\,645,6 \cdot 0,2 \cdot 0,2 \cdot 13,64 = 897,84 \text{ грн}$$

де N_C – середньо-споживча потужність приладу;

K_3 – коефіцієнтом зайнятості приладу;

$C_{\text{ЕН}}$ – тариф за 1 кВт–годин електроенергії.

Накладні витрати:

$$C_H = C_{\text{ПР}} \cdot 0,67 = 40\,000 \cdot 0,67 = 26\,800 \text{ грн}$$

Річні експлуатаційні витрати:

$$C_{\text{ЕКС}} = C_{\text{ЗП}} + C_{\text{ВІД}} + C_A + C_P + C_{\text{ЕЛ}} + C_H$$

$$C_{\text{ЕКС}} = 57\,600 + 12\,672 + 11\,500 + 2\,300 + 897,84 + 26\,800 = 111\,767,84 \text{ грн}$$

Собівартість однієї машино-години:

$$C_{\text{М-Г}} = C_{\text{ЕКС}} / T_{\text{ЕФ}} = 111\,767,84 / 1\,645,6 = 67,92 \text{ грн/год}$$

Витрати на машинний час:

$$C_M = C_{\text{М-Г}} \cdot T$$

$$\text{I. } C_M = 67,92 \cdot 1\,283,2 = 87\,154,9 \text{ грн}$$

$$\text{II. } C_M = 67,92 \cdot 1\,300 = 88\,296 \text{ грн}$$

Накладні витрати (67% від заробітної плати):

$$C_H = C_{\text{ЗП}} \cdot 0,67$$

$$\text{I. } C_H = 275\,400,4 \cdot 0,67 = 184\,518,3 \text{ грн}$$

$$\text{II. } C_H = 279\,006,0 \cdot 0,67 = 186\,934,0 \text{ грн}$$

Повна вартість розробки ПП:

$$C_{\text{ПП}} = C_{\text{ЗП}} + C_{\text{ВІД}} + C_M + C_H$$

$$\text{I. } C_{\text{ПП}} = 275\,400,4 + 60\,588 + 87\,154,9 + 184\,518,3 = 607\,661,6 \text{ грн}$$

$$\text{II. } C_{\text{ПП}} = 279\,006,0 + 61\,381 + 88\,296 + 186\,934,0 = 615\,617 \text{ грн}$$

4.7 Вибір кращого варіанту ПП за техніко-економічним рівнем

Для остаточного вибору оптимального варіанту реалізації розраховується коефіцієнт техніко-економічного рівня за формулою:

$$K_{\text{ТЕР}j} = K_{Kj} / C_{\Phi j}$$

де K_{Kj} – коефіцієнт рівня якості j -го варіанту;

$C_{\Phi j}$ – вартість розробки j -го варіанту, грн

$$K_{\text{ТЕР}1} = 88,60 / 607\,661,6 = 1,45 \cdot 10^{-4}$$

$$K_{\text{ТЕР}2} = 57,95 / 615\,617 = 9,4 \cdot 10^{-5}$$

Після виконання функціонально–вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишилися після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту.

У нього виявився найкращий показник техніко–економічного рівня якості $K_{\text{TEP1}} = 1,45 \cdot 10^{-4}$.

Цей варіант реалізації програмного продукту має такі параметри:

- 1) мова програмування Python;
- 2) бібліотека Scikit–Learn;
- 3) середовище Kaggle Notebook.

4.8 Висновки до розділу 4

Було здійснено комплексний функціонально–вартісний аналіз розроблюваного програмного забезпечення. На початковому етапі дослідження виконалось виокремлення та оцінювання ключових функцій, які безпосередньо має виконувати система. Разом із цим було чітко сформульовано набір базових параметрів та властивостей, що визначають специфіку даного продукту.

Наступним кроком стала експертна оцінка показників і порівняння якості різних підходів до реалізації функцій. Також було економічно обґрунтовано проєктні рішення: розраховано загальну трудомісткість, витрати на зарплату та інші видатки для альтернативних шляхів розробки.

Спираючись на результати проведеного технічного та фінансового дослідження, було прийнято обґрунтоване рішення та обрано найбільш оптимальний і доцільний варіант остаточної реалізації програмного продукту.

ВИСНОВКИ

В ході роботи виконано комплексний огляд предметної області, пов'язаної з виявленням банківського шахрайства на етапі реєстрації нових клієнтів в умовах суттєвого дисбалансу класів. Було детально розглянуто та проаналізовано сучасні математичні моделі та алгоритми машинного навчання для розв'язання задачі бінарної класифікації та виявлення аномалій.

Зокрема, досліджено ефективність алгоритмів Random Forest, SVM з RBF-ядром, багатопарового перцептрона (MLP) та XGBoost для навчання з учителем, а також Isolation Forest для виявлення аномалій без учителя.

Для об'єктивної оцінки якості класифікації та стійкості моделей на незбалансованих вибірках застосовувалися такі метрики, а саме: Precision–Recall AUC, F1–score, Matthews Correlation Coefficient (MCC), матриця помилок (Confusion Matrix) та Accuracy. Це дозволило всебічно порівняти результати різних методів з урахуванням мінімізації помилок першого та другого роду.

Розроблено програмний продукт на мові програмування Python з використанням обчислювальних потужностей середовища Kaggle. Проведено його тестування на багатовимірному наборі даних Bank Account Fraud, а також здійснено попередню обробку ознак та корекцію незбалансованості вибірки за допомогою вагових коефіцієнтів класів для підвищення точності моделей.

Проаналізовано результати метрик якості для кожної розробленої предиктивної моделі та встановлено, що найкращою моделлю виявився алгоритм XGBoost. В умовах жорсткого контролю рівня хибнопозитивних спрацьовувань, який не перевищував 5% , ця модель досягла найвищих значень за всіма цільовими метриками. Зокрема, для XGBoost показник PR-AUC склав 0.216674 , F1-score 0.255910 , а MCC 0.280240. Водночас алгоритм навчання без учителя Isolation Forest виявився найменш ефективним (PR-AUC

0.020298). Це пояснюється специфікою предметної області: зломисники навмисно маскують свої заявки під анкети легальних клієнтів. Відповідно, без вказівок "вчителя" алгоритм виявився нездатним відрізнити ці приховані патерни махінацій від нормальної варіативності даних. Завдяки застосуванню SHAP-аналізу було інтерпретовано результати найкращої моделі та виявлено, що ключовий внесок у здатність розділяти класи роблять категоріальні змінні. Найвагомішими предикторами виявилися статус проживання клієнта (cat_housing_status_BA) та факт використання операційної системи Windows (cat_device_os_windows).

Наприкінці проведено функціонально–вартісний аналіз програмного продукту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Sorunke O. Emerging Trends in Cybercrime and Digital Fraud: A Critical Appraisal. URL: https://www.researchgate.net/publication/400629164_Emerging_Trends_in_Cybercrime_and_Digital_Fraud_A_Critical_Appraisal (last accessed 10.05.2026)
2. Synthetic Identity Fraud. LexisNexis Risk Solutions. URL: <https://risk.lexisnexis.com/insights-resources/article/synthetic-identity-fraud>
3. Future of Financial Crime: Moving to a single, client-centric view of risk. Deloitte. URL: <https://www.deloitte.com/mt/en/services/consulting-financial/blogs/moving-to-a-single-client-centric-view-of-risk.html> (last accessed 10.05.2026)
4. Zero-Day Fraud Detection Using One-Class Classification Techniques in Finance. ResearchGate. 2026. URL: https://www.researchgate.net/publication/394171143_Zero-Day_Fraud_Detection_Using_One-Class_Classification_Techniques_in_Finance (last accessed 10.05.2026)
5. Machine Learning for Fraud Detection in Banking Systems. Computer Science Bulletin, 2026. URL: <https://www.csbulletin.com/index.php/csb/article/download/120/136> (last accessed 10.05.2026)
6. Machine Learning Driven Fraud Detection Pipelines for Digital Banking Products. Famous Journal of Artificial Intelligence and Engineering. 2026. URL: https://www.researchgate.net/publication/400019254_Machine_learning_dri

- [ven_fraud_detection_pipelines_for_digital_banking_products](#)(last accessed 10.05.2026)
7. Негуляев Н. Є., Недашківська Н. І. Антифрод-система для ідентифікації шахрайських карткових транзакцій. Системні науки та інформатика: збірник доповідей III Всеукр.науково-практичної конференції «Системні науки та інформатика», 25–29 листопада 2024 року, Київ. К., НН ІПСА КПІ ім. Ігоря Сікорського, 2024. – С. 193-198. URL: http://mmsa.kpi.ua/sites/default/files/systemni_nauky_ta_informatyka_2024.pdf
 8. Хомич О.Р., Недашківська Н.І. Виявлення аномалій методами інтелектуального аналізу даних // Системні науки та інформатика: збірник доповідей I Всеукр. науково-практичної конференції «Системні науки та інформатика», 22–29 листопада 2022 року, Київ. – К., НН ІПСА КПІ ім. Ігоря Сікорського, 2022., стор. 452 – 455. URL: <http://mmsa.kpi.ua/conferences/2290>
 9. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. 2009.
URL: <https://esl.hohoweiyu.xyz/book/The%20Elements%20of%20Statistical%20Learning.pdf> (last accessed 17.05.2026)
 10. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016.
URL: <https://www.deeplearningbook.org/> (last accessed 17.05.2026)
 11. He H., Ma Y. Imbalanced Learning: Foundations, Algorithms, and Applications. 2013. URL: [https://cyberenlightener.com/wp-content/uploads/2025/03/Imbalanced-Learning-Foundations-Algorithms-and-Applications-2013-Wiley-IEEE-Press-10.1002_9781118646106-libgen.li .pdf](https://cyberenlightener.com/wp-content/uploads/2025/03/Imbalanced-Learning-Foundations-Algorithms-and-Applications-2013-Wiley-IEEE-Press-10.1002_9781118646106-libgen.li.pdf) (last accessed 17.05.2026)
 12. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System of the 22nd ACM SIGKDD International Conference. 2016. URL: <https://dl.acm.org/doi/10.1145/2939672.2939785> (last accessed 17.05.2026)

13. Liu F. T., Ting K. M., Zhou Z. H. Isolation Forest. 2008. URL: https://www.researchgate.net/publication/224384174_Isolation_Forest (last accessed 17.05.2026)
14. Chicco D., Jurman G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genomics. 2020. URL: https://www.researchgate.net/publication/338351315_The_advantages_of_the_Matthews_correlation_coefficient_MCC_over_F1_score_and_accuracy_in_binary_classification_evaluation (last accessed 17.05.2026)
15. Turning the Tables: Biased, Imbalanced, Dynamic Tabular Datasets for ML Evaluation / S. A. Jesus et al. 2022. URL: https://proceedings.neurips.cc/paper_files/paper/2022/hash/d9696563856bd350e4e7ac5e5812f23c-Abstract-Datasets_and_Benchmarks.html (last accessed 06.05.2026)
16. Srinivasan Selvaraj. Industry 4.0 Interoperability, Analytics, Security, and Case Studies. Industry 4.0 Interoperability, Analytics, Security and Case Studies. CRC press, Taylor Francis Group, 2021. URL: https://www.researchgate.net/publication/355485828_Industry_40_Interoperability_Analytics_Security_and_Case_Studies (last accessed 20.05.2026)
17. Using SHAP Values to Explain How Your Machine Learning Model Works. URL: <https://medium.com/data-science/using-shap-values-to-explain-how-your-machine-learning-model-works-732b3f40e137> (last accessed 20.05.2026)
18. Недашківська Н.І. Інтелектуальний аналіз даних : Практикум. Навч. посіб. для студ. спеціальності 124 «Системний аналіз», освітніх програм «Системний аналіз і управління», «Системний аналіз фінансового ринку». Київ : КПІ ім. Ігоря Сікорського. 2021. 105 с. URL: <https://ela.kpi.ua/handle/123456789/53763> (дата звернення 06.05.2026)
19. NumPy. NumPy. URL: <https://numpy.org> (last accessed 30.05.2026)

20. Matplotlib – Visualization with Python. Matplotlib – Visualization with Python. URL: <https://matplotlib.org> (last accessed 30.05.2026)
21. Scikit–Learn. Model Evaluation. Scikit–Learn. URL: https://scikit-learn.org/stable/modules/model_evaluation.html (last accessed 30.05.2026)
22. Scikit–Learn. RandomForestClassifier. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html> (last accessed 30.05.2026)
23. Scikit–Learn. IsolationForest. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html> (last accessed 30.05.2026)
24. Scikit–Learn. SVC. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html> (last accessed 30.05.2026)
25. Scikit–Learn. MLPClassifier. URL: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html (last accessed 30.05.2026)

ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ

```

import pandas as pd
import numpy as np
import joblib
import os
import copy
import shap
import matplotlib.pyplot as plt
import seaborn as sns
import json

from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier, IsolationForest
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from xgboost import XGBClassifier
from sklearn.model_selection import RandomizedSearchCV, TimeSeriesSplit
from sklearn.metrics import average_precision_score, accuracy_score, f1_score, matthews_corrcoef,
confusion_matrix, ConfusionMatrixDisplay, roc_curve
import scipy.stats as stats
from sklearn.calibration import CalibratedClassifierCV
from imblearn.pipeline import Pipeline as ImbPipeline
from imblearn.under_sampling import RandomUnderSampler
path = '/kaggle/input/datasets/kokunovaanastasia/diploma-data/Base.parquet'
df = pd.read_parquet(path)

print(df.shape)
print(df.head())

plt.figure(figsize=(10, 6))
fraud_by_month = df.groupby('month')['fraud_bool'].mean() * 100

sns.barplot(x=fraud_by_month.index, y=fraud_by_month.values, palette='viridis')

```

```

plt.title('Dynamics of the percentage of fraudulent applications (Month 0 – 7)', fontsize=14,
fontweight='bold')
plt.xlabel('Month', fontsize=12)
plt.ylabel('Fraud rate (%)', fontsize=12)
plt.grid(axis='y', linestyle='—', alpha=0.7)

for index, value in enumerate(fraud_by_month.values):
    plt.text(index, value + 0.02, f'{value:.2f}%', ha='center', va='bottom', fontsize=10)

plt.tight_layout()
plt.show()
save_path = '/kaggle/working/models/'
os.makedirs(save_path, exist_ok=True)

categorical_cols = ['payment_type', 'employment_status', 'housing_status', 'source', 'device_os']
ignored_cols = ['fraud_bool', 'month']
numerical_cols = [c for c in df.columns if c not in categorical_cols + ignored_cols]

df[numerical_cols] = df[numerical_cols].replace(-1, np.nan)

df = df.sort_values(by='month').reset_index(drop=True)
train_mask = df['month'] <= 5
val_mask = df['month'] == 6
test_mask = df['month'] == 7

def get_xy(mask):
    X = df[mask].drop(columns=['fraud_bool', 'month'])
    y = df[mask]['fraud_bool']
    return X, y

X_train_raw, y_train = get_xy(train_mask)
X_val_raw, y_val = get_xy(val_mask)
X_test_raw, y_test = get_xy(test_mask)

categorical_transformer = ImbPipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),

```

```

        ('onehot', OneHotEncoder(handle_unknown='ignore', sparse_output=False))
    ])

numeric_transformer_math = ImbPipeline(steps=[
    ('imputer', SimpleImputer(strategy='median', add_indicator=True)),
    ('scaler', StandardScaler())
])

preprocessor_math = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer_math, numerical_cols),
        ('cat', categorical_transformer, categorical_cols)
    ], remainder='drop'
)

numeric_transformer_xgb = ImbPipeline(steps=[
    ('scaler', StandardScaler())
])

preprocessor_xgb = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer_xgb, numerical_cols),
        ('cat', categorical_transformer, categorical_cols)
    ], remainder='drop'
)

cv_folds = TimeSeriesSplit(n_splits=3)
scoring_metric = 'average_precision'

# Random Forest
print("Optimizing Random Forest...")

rf_pipe = ImbPipeline([
    ('prep', preprocessor_math),
    ('clf', RandomForestClassifier(random_state=42, n_jobs=-1))
])

rf_params = {

```

```

        'clf__n_estimators': [200, 400, 500],
        'clf__max_depth': [15, 25, 30],
        'clf__min_samples_leaf': [2, 5],
        'clf__max_features': ['sqrt', 'log2', 0.2, 0.5],
        'clf__class_weight': ['balanced', 'balanced_subsample']
    }

    rf_search = RandomizedSearchCV(rf_pipe, rf_params, n_iter=6, cv=cv_folds,
                                   scoring=scoring_metric, n_jobs=1, random_state=42)
    rf_search.fit(X_train_raw, y_train)

    with open(f'{save_path}rf_best_params.json', 'w') as f:
        json.dump(rf_search.best_params_, f, indent=4)
    joblib.dump(rf_search.best_estimator_, f'{save_path}rf_model_best.pkl')
    print("Random Forest saved.")

# XGBoost
print("Optimizing XGBoost...")
ratio = float(np.sum(y_train == 0) / np.sum(y_train == 1))

xgb_pipe = ImbPipeline([
    ('prep', preprocessor_xgb),
    ('clf', XGBClassifier(tree_method='hist', random_state=42, eval_metric='aucpr',
                          scale_pos_weight=ratio, n_jobs=1))
])

xgb_params = {
    'clf__n_estimators': [200, 400, 600],
    'clf__max_depth': [4, 6, 8],
    'clf__learning_rate': [0.01, 0.05, 0.1],
    'clf__subsample': [0.7, 0.8, 1.0],
    'clf__colsample_bytree': [0.7, 0.8, 1.0],
    'clf__gamma': [0, 0.5, 1, 5]
}

xgb_search = RandomizedSearchCV(xgb_pipe, xgb_params, n_iter=8, cv=cv_folds,
                                scoring=scoring_metric, n_jobs=-1, random_state=42)

```

```

xgb_search.fit(X_train_raw, y_train)

with open(f'{save_path}xgb_best_params.json', 'w') as f:
    json.dump(xgb_search.best_params_, f, indent=4)
joblib.dump(xgb_search.best_estimator_, f'{save_path}xgb_model_best.pkl')
print("XGBoost saved.")

# SVM (RBF)
print("Optimizing SVM with RBF Kernel...")

svm_pipe = ImbPipeline([
    ('prep', preprocessor_math),
    ('sampler', RandomUnderSampler(sampling_strategy=0.1, random_state=42)),
    ('clf', SVC(kernel='rbf', cache_size=2000, random_state=42))
])

svm_params = {
    'clf__C': [0.1, 1, 10, 100],
    'clf__gamma': ['scale', 'auto', 0.1, 0.01]
}

svm_search = RandomizedSearchCV(svm_pipe, svm_params, n_iter=3, cv=cv_folds,
                                scoring=scoring_metric, n_jobs=2, random_state=42)
print("Fitting SVM...")
svm_search.fit(X_train_raw, y_train)

with open(f'{save_path}svm_rbf_best_params.json', 'w') as f:
    json.dump(svm_search.best_params_, f, indent=4)

print("Calibrating SVM...")
X_val_trans = svm_search.best_estimator_.named_steps['prep'].transform(X_val_raw)

calibrated_svm = CalibratedClassifierCV(
    svm_search.best_estimator_.named_steps['clf'],
    cv='prefit',
    method='sigmoid'
)

```

```
calibrated_svm.fit(X_val_trans, y_val)
```

```
final_svm_pipe = ImbPipeline([
    ('prep', svm_search.best_estimator_.named_steps['prep']),
    ('clf', calibrated_svm)
])
```

```
joblib.dump(final_svm_pipe, f'{save_path}svm_rbf_model_best.pkl')
print("SVM (RBF) saved successfully.")
```

```
# MLP
```

```
print("Optimizing MLP...")
```

```
mlp_pipe = ImbPipeline([
    ('prep', preprocessor_math),
    ('sampler', RandomUnderSampler(sampling_strategy=0.2, random_state=42)),
    ('clf', MLPClassifier(max_iter=500, early_stopping=True, shuffle=False, random_state=42))
])
```

```
mlp_params = {
    'clf__hidden_layer_sizes': [(64, 32), (100,), (128, 64, 32)],
    'clf__alpha': [0.0001, 0.001, 0.01],
    'clf__learning_rate_init': [0.001, 0.01]
}
```

```
mlp_search = RandomizedSearchCV(mlp_pipe, mlp_params, n_iter=4, cv=cv_folds,
                                scoring=scoring_metric, n_jobs=1, random_state=42)
mlp_search.fit(X_train_raw, y_train)
```

```
with open(f'{save_path}mlp_best_params.json', 'w') as f:
    json.dump(mlp_search.best_params_, f, indent=4)
joblib.dump(mlp_search.best_estimator_, f'{save_path}mlp_model_best.pkl')
print("MLP saved.")
```

```
# Isolation Forest
```

```
print("Evaluating Isolation Forest...")
best_score = -1
```

```

best_iso_model = None

X_train_trans_math = preprocessor_math.fit_transform(X_train_raw)
X_val_trans_math = preprocessor_math.transform(X_val_raw)

normal_mask_train = (y_train == 0)
X_train_normal_trans = X_train_trans_math[normal_mask_train]

np.random.seed(42)
iso_idx = np.random.choice(X_train_normal_trans.shape[0], min(len(X_train_normal_trans), 100000),
replace=False)
X_train_iso = X_train_normal_trans[iso_idx]

for ms in [256, 512, 1024, 2048]:
    iso = IsolationForest(max_samples=ms, contamination=0.01, n_jobs=1, random_state=42)
    iso.fit(X_train_iso)

    val_scores = -iso.decision_function(X_val_trans_math)
    score = average_precision_score(y_val, val_scores)

    if score > best_score:
        best_score = score
        best_iso_model = copy.deepcopy(iso)

final_iso_pipe = ImbPipeline([
    ('prep', preprocessor_math),
    ('clf', best_iso_model)
])

with open(f'{save_path}iso_forest_best_params.json', 'w') as f:
    json.dump(best_iso_model.get_params(), f, indent=4)

joblib.dump(final_iso_pipe, f'{save_path}iso_forest_best.pkl')
print("Isolation Forest saved.")

# Найкращі гіперпараметри
print("Random Forest Best Params:", rf_search.best_params_)

```

```

print("XGBoost Best Params:", xgb_search.best_params_)
print("SVM Best Params:", svm_search.best_params_)
print("MLP Best Params:", mlp_search.best_params_)
print("Isolation Forest Best Params:", {'max_samples': best_iso_model.max_samples, 'contamination':
best_iso_model.contamination})

```

Оцінка моделей та візуалізація

```

best_models = {
    'Random Forest': joblib.load(f'{save_path}rf_model_best.pkl'),
    'XGBoost': joblib.load(f'{save_path}xgb_model_best.pkl'),
    'SVM (RBF)': joblib.load(f'{save_path}svm_rbf_model_best.pkl'),
    'MLP': joblib.load(f'{save_path}mlp_model_best.pkl'),
    'Isolation Forest': joblib.load(f'{save_path}iso_forest_best.pkl')
}

```

```

results = []

```

```

confusion_matrices = {}

```

```

print("\nEvaluating models on unseen Test Data (Month 7) with dynamic threshold (FPR <= 5%)...\n")

```

```

for name, model in best_models.items():
    if name == 'Isolation Forest':
        val_scores = -model.decision_function(X_val_raw)
        y_scores = -model.decision_function(X_test_raw)
    else:
        val_scores = model.predict_proba(X_val_raw)[:, 1]
        y_scores = model.predict_proba(X_test_raw)[:, 1]

    fpr, tpr, thresholds = roc_curve(y_val, val_scores)
    valid_fpr_indices = np.where(fpr <= 0.05)[0]

    if len(valid_fpr_indices) > 0:
        target_fpr_idx = valid_fpr_indices[-1]
    else:
        target_fpr_idx = 0

```

```

optimal_threshold = thresholds[target_fpr_idx]

y_pred = (y_scores >= optimal_threshold).astype(int)

# Metrics
acc = accuracy_score(y_test, y_pred)
pr_auc = average_precision_score(y_test, y_scores)
f1 = f1_score(y_test, y_pred)
mcc = matthews_corrcoef(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
confusion_matrices[name] = cm

results.append({
    'Model': name,
    'Accuracy': acc,
    'PR-AUC': pr_auc,
    'F1-score': f1,
    'MCC': mcc,
    'Optimal Threshold': optimal_threshold
})

results_df = pd.DataFrame(results).sort_values(by='PR-AUC', ascending=False)
print(results_df.to_string(index=False))

# Confusion Matrices
fig, axes = plt.subplots(2, 3, figsize=(18, 12))
axes = axes.flatten()

for i, (name, cm) in enumerate(confusion_matrices.items()):
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Legit', 'Fraud'])
    disp.plot(ax=axes[i], cmap='Blues', colorbar=False, values_format='d')
    axes[i].set_title(f'{name}', fontsize=14, fontweight='bold')

if len(confusion_matrices) < len(axes):
    fig.delaxes(axes[5])

```

```

plt.subplots_adjust(hspace=0.3, wspace=0.3)
plt.tight_layout()
plt.show()
# SHAP аналіз для XGBOOST
xgb_pipe = best_models['XGBoost']
prep_xgb = xgb_pipe.named_steps['prep']
xgb_model = xgb_pipe.named_steps['clf']

feature_names = prep_xgb.get_feature_names_out()

X_test_trans_xgb = prep_xgb.transform(X_test_raw)
X_test_trans_df = pd.DataFrame(X_test_trans_xgb, columns=feature_names)

plt.figure(figsize=(12, 8))
importances = xgb_model.feature_importances_
indices = np.argsort(importances)[::-1][:20]

sns.barplot(x=importances[indices], y=np.array(feature_names)[indices], palette='mako')
plt.title('Top 20 most important features (XGBoost Feature Importance)', fontsize=14, fontweight='bold')
plt.xlabel('Importance (Gini/Gain)', fontsize=12)
plt.ylabel('Features', fontsize=12)
plt.tight_layout()
plt.show()

explainer = shap.TreeExplainer(xgb_model)

X_test_sample = X_test_trans_df.sample(n=min(5000, len(X_test_trans_df)), random_state=42)
shap_values = explainer.shap_values(X_test_sample)

plt.figure(figsize=(10, 8))
plt.title('Influence of features on model solutions', fontsize=14, fontweight='bold')
shap.summary_plot(shap_values, X_test_sample, max_display=15, show=False)
plt.tight_layout()
plt.show()

```

ДОДАТОК В. ГРАФІЧНИЙ МАТЕРІАЛ ДО ЗАХИСТУ

МЕТОДИ МАШИННОГО НАВЧАННЯ ТА НЕЙРОННИХ МЕРЕЖ ДЛЯ ВИЯВЛЕННЯ ШАХРАЙСТВА У БАНКІВСЬКИХ ОПЕРАЦІЯХ

ВИКОНАЛА: СТУДЕНТКА IV КУРСУ, ГРУПИ КА-22

КОКУНОВА АНАСТАСІЯ ЮРІІВНА

КЕРІВНИК: ПРОФЕСОР, Д.Т.Н.,

НЕДАШКІВСЬКА НАДІЯ ІВАНІВНА

1



АКТУАЛЬНІСТЬ ТЕМИ ДОСЛІДЖЕННЯ

Стрімка цифровізація фінансового сектору та розвиток онлайн-банкінгу призвели до критичного зростання масштабів шахрайства, де найвразливішим етапом стало дистанційне відкриття нових рахунків. Застосування зловмисниками синтетичних ідентичностей та генеративного штучного інтелекту перетворило фінансовий фрод на високотехнологічну індустрію, з якою традиційні системи моніторингу на базі статичних правил більше не здатні впоратися.

Саме тому впровадження сучасних методів машинного навчання є критично необхідним кроком для предиктивного виявлення неочевидних злочинних патернів, мінімізації хибних блокувань та надійного захисту активів банків.

2

ПРЕДМЕТ, ОБ'ЄКТ ТА МЕТА ДОСЛІДЖЕННЯ

Метою дослідження є розробка та порівняльний аналіз моделей машинного навчання для ефективної та точної класифікації шахрайських заявок на відкриття банківських рахунків.

Об'єктом дослідження є процес ідентифікації шахрайських дій (аномалій) у банківських операціях на етапі реєстрації нових клієнтів.

Предметом дослідження є математичні методи та моделі машинного навчання (зокрема Random Forest, SVM, MLP, XGBoost, Isolation Forest) для розв'язання задачі класифікації банківського шахрайства на основі багатовимірних поведінкових ознак.

3

BANK ACCOUNT FRAUD (BAF) (NEURIPS 2022)

Набір має 1 000 000 спостережень та 32 ознаки.

Цільова змінна — **fraud_bool**, бінарний індикатор шахрайства, де 1 означає шахрайську заявку, а 0 легітимну.

Частка шахрайських заявок становить близько 1.1% від загального обсягу даних.

Набір даних:
<https://www.kaggle.com/datasets/spgjesus/bank-account-fraud-dataset-neurips-2022>

Bank Account Fraud Dataset Suite (NeurIPS 2022)

192

Code

Download

Data CardCode (56)Discussion (6)Suggestions (0)

Base.csv (213.43 MB)

10 of 32 columns

fraud_bool

Fraud label (1 if fraud, 0 if legit)

01

income

Annual income of the applicant in quantiles. Ranges between [0, 1].

0.10.9

name_email_simil...

Metric of similarity between email and applicant's name. Higher values represent higher similarity. Ranges

01

prev_address_mo...

Number of months in previous registered address of the applicant, i.e. the applicant's previous residence, if

-1383

current_address...

Months in currently registered address of the applicant. Ranges between [-1, 406] months (-1 is a missing

-1428

custom

Applicant's per decade represent

10

# fraud_bool	# income	# name_email_simil...	# prev_address_mo...	# current_address...	# custom
0	0.3	0.986586318633834	-1	25	48
0	0.8	0.6174268862650861	-1	89	28
0	0.8	0.9967878286489232	9	14	48
0	0.6000000000000001	0.4758999462388287	11	14	38

Version 2 (1.36 GB)

Base.csv

Variant I.csv

Variant II.csv

Variant III.csv

Variant IV.csv

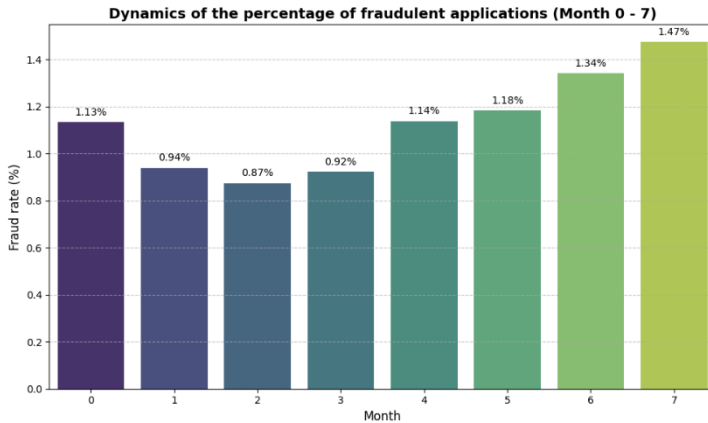
Variant V.csv

Summary

6 files

196 columns

ВХІДНІ ДАНІ



Динаміка рівня шахрайських заявок за місяцями

	count	mean	std
credit_risk_score	1000000.0	130.989595	69.681812
proposed_credit_limit	1000000.0	515.850891	487.025696
customer_age	1000000.0	33.680000	12.025799
keep_alive_session	1000000.0	0.576947	0.494044
income	1000000.0	0.562696	0.290300
date_of_birth_distinct_emails_4w	1000000.0	9.503544	5.033792
name_email_similarity	1000000.0	0.493694	0.289069
device_distinct_emails_8w	1000000.0	1.018312	0.180761
has_other_cards	1000000.0	0.222988	0.416251
phone_home_valid	1000000.0	0.417077	0.493076

	min	25%	50%
credit_risk_score	-170.000000	83.000000	122.000000
proposed_credit_limit	190.000000	200.000000	200.000000
customer_age	10.000000	20.000000	30.000000
keep_alive_session	0.000000	0.000000	1.000000
income	0.100000	0.300000	0.600000
date_of_birth_distinct_emails_4w	0.000000	6.000000	9.000000
name_email_similarity	0.000001	0.225216	0.492153
device_distinct_emails_8w	-1.000000	1.000000	1.000000
has_other_cards	0.000000	0.000000	0.000000
phone_home_valid	0.000000	0.000000	0.000000

Описова статистика для 10 найважливіших
числових ознак

ПОСТАНОВКА ЗАДАЧІ

Вхідні дані:

I – вхідний набір даних, де кожна операція представлена вектором ознак (сума, час, локація, тип операції тощо)

M – мітки розмітки, що позначають приналежність транзакції до класу шахрайських (1) або легітимних (0)

Нехай $f(I)$ – функція класифікації, реалізована такими підходами:

$f_{RF}(I)$ – Random Forest (стабільність та оцінка важливості ознак)

$f_{SVM}(I)$ – SVM з RBF-ядром (нелінійна класифікація)

$f_{MLP}(I)$ – Багатошаровий перцептрон (аналіз ієрархічних зв'язків)

$f_{XGB}(I)$ – XGBoost (последовна мінімізація помилок на незбалансованих даних)

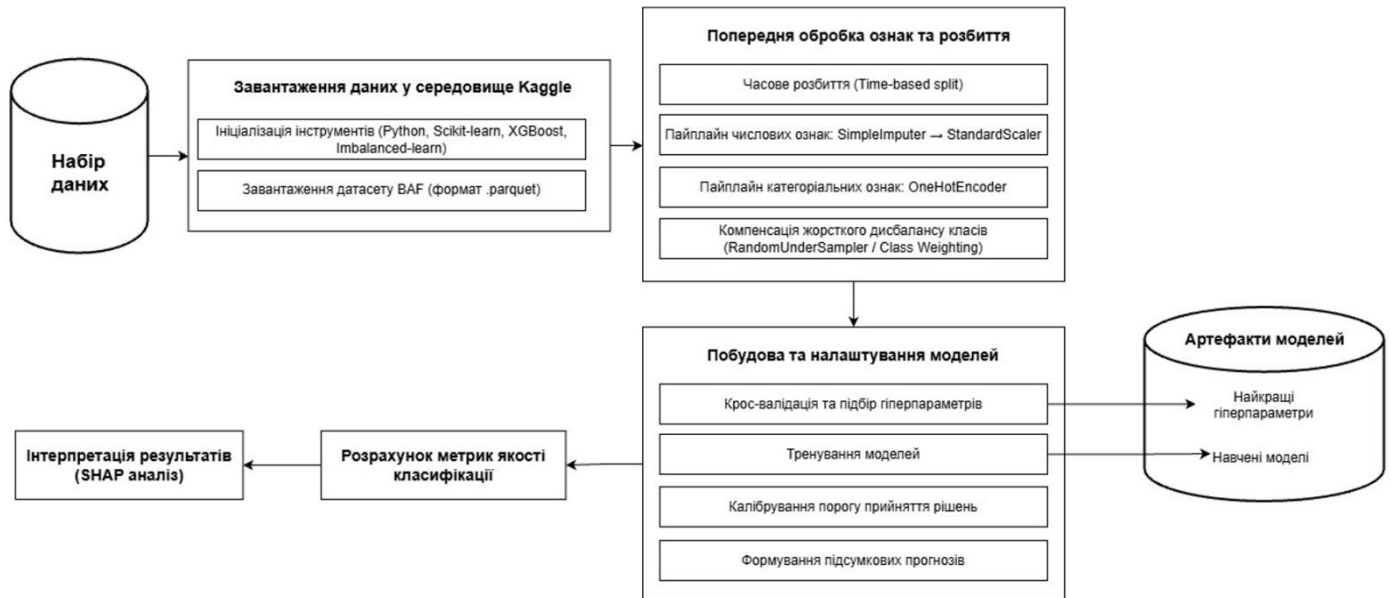
$f_{IF}(I)$ – Isolation Forest (виявлення аномалій без вчителя)

Задача виявлення фроду полягає у знаходженні:

$C = f(I)$ – визначення прогнозного класу спостереження

$L(C, M)$ – мінімізація функції втрат з урахуванням ваг W для максимальної точності розпізнавання аномалій

АРХИТЕКТУРА СИСТЕМИ



МОДЕЛІ БІНАРНОЇ КЛАСИФІКАЦІЇ ТА ВИЯВЛЕННЯ АНОМАЛІЙ

Навчання з вчителем:

Random Forest

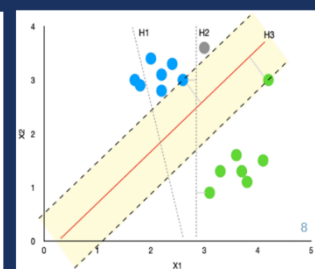
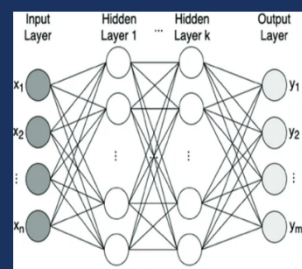
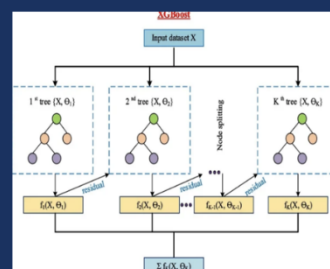
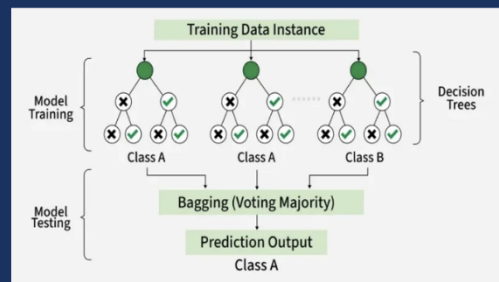
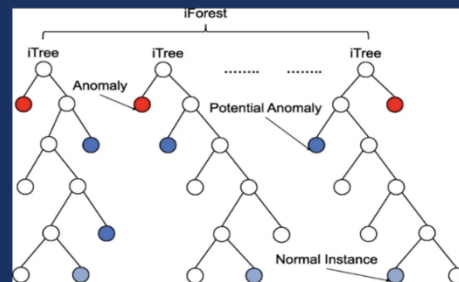
SVM (RBF)

XGBoost

MLP

Навчання без вчителя:

Isolation Forest



МЕТРИКИ ОЦІНКИ РЕЗУЛЬТАТІВ МОДЕЛЕЙ

$$PR AUC = \int_0^1 P(R) dR$$

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN}$$

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Confusion Matrix = \begin{pmatrix} TP & FN \\ FP & TN \end{pmatrix}$$

TP — кількість істинно позитивних значень.

TN — кількість істинно негативних значень.

FP — кількість невірно позитивних значень.

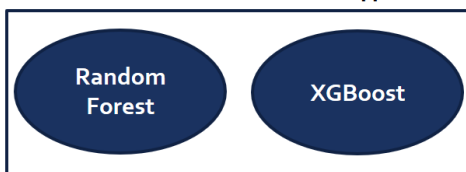
FN — кількість невірно негативних значень.

9

НАВЧАННЯ ТА ОЦІНКА МОДЕЛЕЙ

Часове розбиття (**Time-based split**) — навчання (міс. 0-5), валідація (міс. 6), тестування (міс. 7)

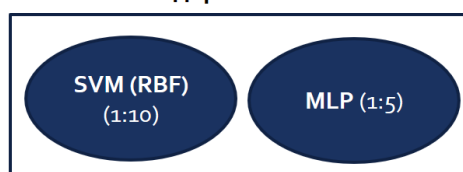
Навчання на повних даних



`class\_weight` `scale\_pos\_weight`

Компенсування дисбалансу

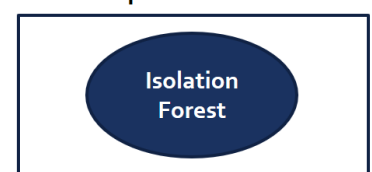
Андерсемплінг



Калібрування ймовірностей

Механізм ранньої зупинки

Напівкероване навчання



Виключно на легітимних транзакціях

Динамічний поріг — підбирався на валідаційній вибірці так, щоб рівень хибних спрацьовувань не перевищував 5% ($FPR \leq 0.05$)

10

НАЙКРАЩІ ГІПЕРПАРАМЕТРИ МОДЕЛЕЙ

Параметр	Значення
clf__n_estimators	500
clf__min_samples_leaf	2
clf__max_features	log2
clf__max_depth	30
clf__class_weight	balanced

Random Forest

Параметр	Значення
max_samples	256
contamination	0.01

Isolation Forest

Параметр	Значення
clf__subsample	0.7
clf__n_estimators	400
clf__max_depth	4
clf__learning_rate	0.05
clf__gamma	0.5
clf__colsample_bytree	0.8

XGBoost

Параметр	Значення
clf__learning_rate_init	0.01
clf__hidden_layer_sizes	(128, 64, 32)
clf__alpha	0.0001

MLP

Параметр	Значення
clf__gamma	auto
clf__C	1

SVM з ядром RBF

11

МАТРИЦІ ПОМИЛОК МОДЕЛЕЙ

TN	Legit/Legit		Legit/Fraud	
	XGBoost	91752	XGBoost	3663
	SVM (RBF)	90625	SVM (RBF)	4790
	MLP	90837	MLP	4578
	Random Forest	92578	Random Forest	2837
	Isolation Forest	90928	Isolation Forest	4487
FN	Fraud/Legit		Fraud/Fraud	
	XGBoost	681	XGBoost	747
	SVM (RBF)	683	SVM (RBF)	745
	MLP	703	MLP	725
	Random Forest	847	Random Forest	581
	Isolation Forest	1314	Isolation Forest	114
			TP	
			FP	

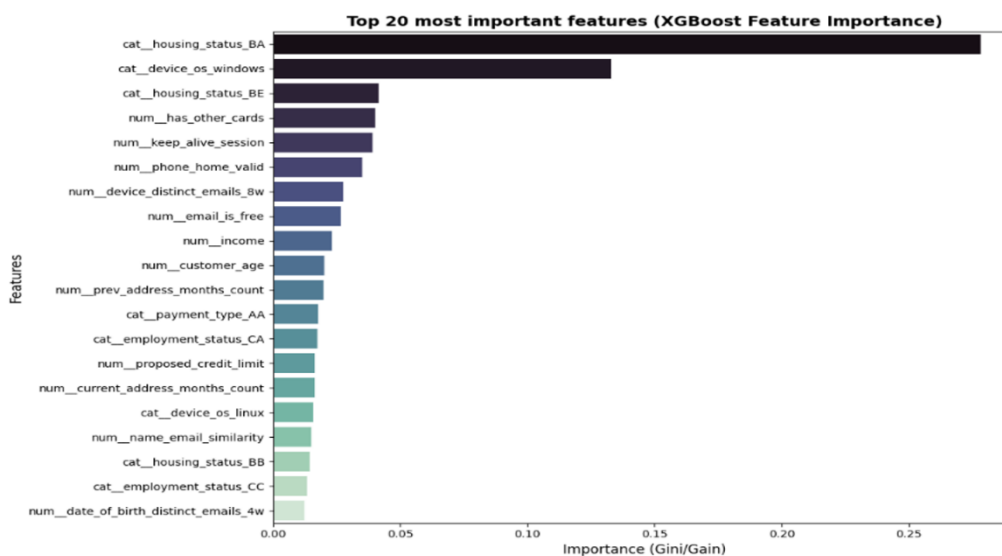
12

ЕФЕКТИВНІСТЬ МОДЕЛЕЙ ЗА МЕТРИКАМИ

Model	Accuracy	PR-AUC	F1-score	MCC	Оптимальний поріг
XGBoost	0.955144	0.216674	0.255910	0.280240	0.809660
SVM (RBF)	0.943486	0.180813	0.213988	0.244820	0.019436
MLP	0.945468	0.177273	0.215421	0.243558	0.449798
Random Forest	0.961959	0.155795	0.239785	0.246346	0.118877
Isolation Forest	0.940099	0.020298	0.037817	0.018588	-0.030570

13

НАЙВАЖЛИВІШІ ПРЕДИКТОРИ XGBOOST



статус проживання операційна система

housing_status	device_os
Current residential status for applicant. 7 possible (anonymized) values.	Operative system of device that made request. Possible values are: Windows, Macintosh, Linux, X11, or other.
BC 37%	other 34%
BB 26%	linux 33%
Other (366892) 37%	Other (324560) 32%
BA	other
BE	other
BA	macintosh
BA	linux
BC	linux
BC	windows

14

SHARP АНАЛІЗ ДЛЯ XGBOOST

Найбільший прямиий вплив

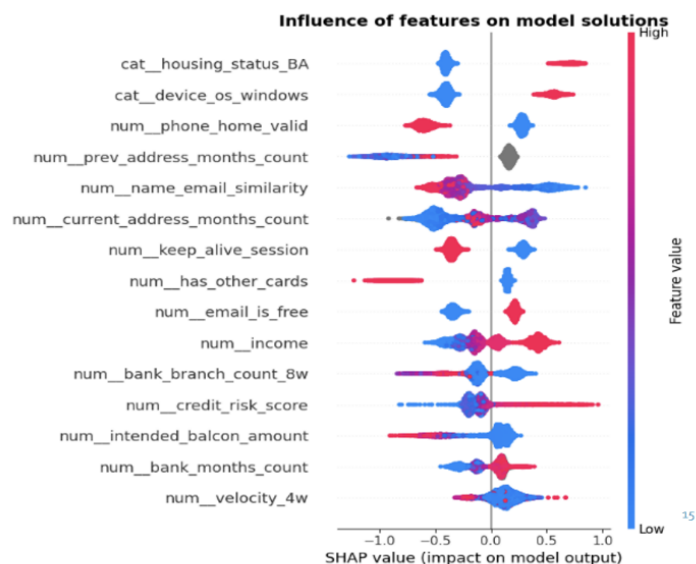
- `cat__housing_status_BA` та `cat__device_os_windows` (житловий статус; операційна система пристрою)

Зворотні індикатори

- `num__phone_home_valid` та `num__name_email_similarity` (валідність домашнього номера телефону; схожість між реальним ім'ям заявника та адресою його електронної пошти)

Предиктор із негативним впливом

- `num__has_other_cards` (чи має клієнт інші банківські картки)



ВИСНОВКИ

У роботі вирішено задачу виявлення банківського шахрайства при реєстрації клієнтів в умовах значного дисбалансу даних.

Розроблено програмний продукт мовою Python у середовищі Kaggle, який реалізує та порівнює алгоритми Random Forest, SVM, MLP, XGBoost та Isolation Forest на наборі даних Bank Account Fraud.

XGBoost став найкращою моделлю (PR-AUC: 0.217, F1-score: 0.256, MCC: 0.280) за умови обмеження хибнопозитивних спрацьовувань до 5%.

Модель навчання без учителя Isolation Forest показала найнижчу точність (PR-AUC: 0.020), оскільки без розмітки не здатна розпізнати цілеспрямоване маскування шахраїв під легальних клієнтів.

Найбільший внесок у виявлення шахрайства роблять категоріальні змінні, зокрема статус проживання клієнта та використання операційної системи Windows.

ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Подальші дослідження варто спрямувати на глибшу оптимізацію гіперпараметрів та вдосконалення алгоритмів виявлення шахрайства.

Важливим вектором є аналіз впливу конструювання нових ознак на ефективність класифікації, кластеризації та пошуку аномалій.

Через екстремальний дисбаланс класів доцільно вивчати методи з урахуванням вартості помилки (cost-sensitive learning) для точнішого калібрування динамічних порогів і максимізації PR-AUC.

Крім того, з огляду на часову залежність даних, актуальним є дослідження стійкості ансамблевих моделей до зміни поведінки шахраїв із часом (concept drift).

17

ДЯКУЮ ЗА УВАГУ!

18