ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА
(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерних систем і технологій
(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «бакалавр»

«Розробка системи виявлення шахрайства у фінансових транзакціях за допомогою методів класичного машинного навчання»

«Development of a system for detecting fraud in financial transactions using classical machine learning methods»

Виконала: здобувачка денної (очної) форми навчання
спеціальності 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

Кобець Вероніка Олександрівна
(прізвище, ім'я, по-батькові здобувача)

Керівник д.тех.н., проф. Михайленко В.С.
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент ст. викл. Мартинович Л.Я.
(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
комп'ютерних систем і технологій

№ ___ від ____.____. 20___ р.

Завідувач кафедри
_____ Юрій ГУНЧЕНКО
(підпис)

Захищено на засіданні ЕК № _____
протокол № __від ____.____.20___ р.

Оцінка_____/___ ___/_____
(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК
_____ Микола МАЛАКСІАНО
(підпис)

Одеса 2025

АНОТАЦІЯ

Тема роботи: *Розробка системи виявлення шахрайства у фінансових транзакціях за допомогою методів класичного машинного навчання.*

У роботі досліджено проблему виявлення шахрайства у фінансових транзакціях у контексті цифровізації, розвитку безконтактних оплат і стрімкого поширення електронних платіжних систем. За врахування висхідної тенденції фінансових збитків, пов'язаних з fraud-інцидентами, та обмежень традиційних rule-based систем, акцент зроблено на використанні класичних моделей машинного навчання для задачі бінарної класифікації.

Мета роботи – розробити систему виявлення шахрайства у фінансових транзакціях на основі реалізації та порівняння класичних моделей машинного навчання (Logistic Regression, Random Forest, XGBoost) у поєднанні з різними методами балансування даних (Random Oversampling, SMOTE, SMOTE-Tomek, ADASYN), зокрема окреслити рекомендації щодо впровадження рішень відповідно до бізнес-потреб фінансових інфраструктур.

Розроблено функціональну систему виявлення шахрайських транзакцій на основі згаданих алгоритмів із використанням сучасних стратегій ресемплінгу. Виконано попередню обробку та інженерію ознак, побудовано машинні пайплайни з `imblearn.pipeline.Pipeline`, проведено гіперпараметричну оптимізацію через `GridSearchCV`, а також оцінено моделі за релевантним набором метрик (PR-AUC, F_2-score, MCC, ROC-AUC тощо).

Отримані результати узагальнено у вигляді практичних рекомендацій щодо інтеграції в банківських інформаційних системах та окреслено перспективи подальших досліджень у сфері протидії фінансовому шахрайству.

Ключові слова: фінансове шахрайство, машинне навчання, бінарна класифікація, SMOTE, ADASYN, XGBoost, Random Forest, дисбаланс класів, PR-AUC, F_2-score.

ANNOTATION

Thesis topic: *Development of a Fraud Detection System in Financial Transactions Using Classical Machine Learning Methods.*

This thesis addresses the problem of fraud detection in financial transactions in the context of increasing digitalization, the evolution of contactless payments, and the global expansion of electronic payment systems. Due to the rising financial losses caused by fraud incidents and the shortcomings of rule-based systems, this study focuses on employing classical machine learning approaches to tackle binary classification problems in imbalanced datasets.

The main objective of the thesis is to develop a fraud detection system based on the implementation and comparison of classical machine learning models (Logistic Regression, Random Forest, XGBoost), in combination with different data balancing techniques (Random Oversampling, SMOTE, SMOTE-Tomek, ADASYN), and to formulate recommendations for practical deployment within financial infrastructures.

A functional detection system was built using the aforementioned algorithms, enhanced with advanced resampling strategies. The dataset was preprocessed and engineered using techniques such as logarithmic transformation and PCA standardization. Pipelines were constructed via `imblearn.pipeline.Pipeline`, and hyperparameter tuning was performed using `GridSearchCV`. The models were evaluated using an extended set of metrics including PR-AUC, F_2-score, MCC, and ROC-AUC.

The results are consolidated into practical implementation recommendations for banking systems and outline avenues for further research in the area of financial fraud mitigation.

Keywords: financial fraud, machine learning, binary classification, SMOTE, ADASYN, XGBoost, Random Forest, class imbalance, PR-AUC, F_2-score.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ 6
ВСТУП 7
1 ДОСЛІДЖЕННЯ ПРЕДМЕНОЇ ОБЛАСТІ, ДАНІ ТА ПІДГОТОВКА ОЗНАК……11
1.1 Шахрайство з кредитними картками: проблематика й контекст 11
1.1.1 Види шахрайських операцій із картками та їх наслідки 16
1.2 Методи виявлення шахрайства: від правил до машинного навчання ... 22
1.2.1 Правила (Rule-Based Systems) і класичний профайлінг..... 23
1.2.2 Початок застосування машинного навчання: основні підходи..... 27
1.2.3 Ансамблеві моделі та бустинг: короткий огляд..... 31
1.3 Проблема дисбалансу класів у фінансових даних та основні стратегії ресемплінгу 34
1.3.1 Random OverSampling (*ROS*) 36
1.3.2 SMOTE (*Synthetic Minority Over-sampling TEchnique*)..... 38
1.3.3 SMOTE-Tomek (комбінований підхід)..... 40
1.3.4 ADASYN (*Adaptive Synthetic Sampling*)..... 42
1.4 Набір даних «*Credit Card Fraud Detection*» (Kaggle)..... 44
1.4.1 Структура датасету: стовпці Time, V1–V28, Amount, Class 45
1.4.2 Опис анонімізованих PCA-компонент V1–V28 47
1.5 Обробка та інженерія ознак..... 49
1.6 Висновки до розділу 1 52
2 МЕТОДОЛОГІЯ ПОБУДОВИ ТА ОЦІНЮВАННЯ МОДЕЛЕЙ 53
2.1 Етапи побудови експерименту — machine learning workflow..... 53
2.2 Опис класифікаторів та налаштування гіперпараметрів 56
2.2.1 Логістична регресія (*Logistic Regression*) 56
2.2.2 Випадковий ліс та ансамблеве навчання (*Random Forest*)..... 57
2.2.3 Градієнтний бустинг (*XGBoost — eXtreme Gradient Boosting*) 59
2.3 Метрики оцінювання результатів..... 61
2.3.1 Precision (точність)..... 62
2.3.2 Recall (чутливість, повнота)..... 63
2.3.3 $F2$-score (F-міра з $\beta = 2$)..... 64

2.3.4 Коефіцієнт кореляції Метьюза (*Matthews Correlation Coefficient*).. 65
2.3.5 PR-curve (*Precision-Recall крива*) 66
2.3.6 ROC-AUC (*Area Under Receiver Operating Characteristic Curve*) ... 67
2.3.7 Мінімізація порогу (*threshold*) 68
2.4 Висновки до розділу 2 70
3 ЕКСПЕРИМЕНТИ, РЕЗУЛЬТАТИ ТА БІЗНЕС-РЕКОМЕНДАЦІЇ 72
3.1 Результати крос-валідації та порівняння моделей 72
3.1.1 Логістична регресія (*Logistic Regression*) «ресемплер + модель» ... 73
3.1.2 Випадковий ліс (*Random Fores*t) «ресемплер + модель» 76
3.1.3 Градієнтний бустинг (*eXtreme Gradient Boosting*) «ресемплер + модель» 80
3.2 Обмеження дослідження 85
3.3 Висновки до розділу 3 87
ВИСНОВКИ 88
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 90
ДОДАТОК А 97
Лістинг програми «Пайплайн системи виявлення шахрайства у фінансових транзакціях за допомогою методів класичного машинного навчання» 97

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

AI – Artificial Intelligence – штучний інтелект

AP – Average Precision – середня точність

ADASYN – Adaptive Synthetic Sampling – адаптивне синтетичне балансування

AUC – Area Under Curve – площа під кривою

BEC – Business Email Compromise – компрометація ділової електронної пошти

CV – Cross-Validation – крос-валідація

EDA – Exploratory Data Analysis – дослідницький аналіз даних

F_2-score – зважена гармонічна середня між $Precision$ та $Recall$ з $\beta = 2$

F_β – узагальнення F-міри для довільного значення β

FPR – False Positive Rate – частка хибнопозитивних передбачень

FN – False Negative – хибнонегативне передбачення

FP – False Positive – хибнопозитивне передбачення

LR – Logistic Regression – логістична регресія

ML – Machine Learning – машинне навчання

MCC – Matthews Correlation Coefficient – коефіцієнт кореляції Метьюса

PCA – Principal Component Analysis – метод головних компонент

PR-AUC – Precision-Recall AUC – площа під PR-кривою

RF – Random Forest – випадковий ліс

ROC-AUC – Receiver Operating Characteristic AUC – площа під ROC-кривою

ROS – Random Oversampling – випадкове надсемплювання

SMOTE – Synthetic Minority Over-sampling TEchnique – синтетичне надсемплювання меншості

SMOTE-Tomek – комбінація SMOTE і видалення Tomek-пар

TP – True Positive – істинно позитивне передбачення

TN – True Negative – істинно негативне передбачення

XGBoost – Extreme Gradient Boosting – алгоритм градієнтного бустингу

ROC – Receiver Operating Characteristic – характеристична крива

ВСТУП

За розвитку електронних платіжних технологій, глобалізації ринків, розробки безконтактних методів оплати та впровадження стратегії легалізації та регулювання ринку криптовалют в Україні було створено зручні умови для споживачів, але водночас розширено можливості для зловмисників, оскільки активна цифровізація сучасних фінансових систем зробила їх більш чутливими до ризиків шахрайських операцій із кредитними картками.

Згідно із даними різних аналітичних центрів, прослідковується стрімка динаміка щорічних втрат фінансових установ та їхніх клієнтів від fraud-інцидентів, що сягають десятків і сотень мільярдів доларів [1]. Таким чином, побудова ефективних методів автоматизованого виявлення шахрайських транзакцій є критичною та комплексною задачею для забезпечення безпеки платіжної інфраструктури та мінімізації фінансових збитків.

Класичні rule-based експертні системи (набір жостко прописаних правил «якщо…, то… [if…, else…]») не враховують динаміки та непередбачуваності нових тенденцій та стратегій шахрайства, через що наукові дослідження та промислові IT-рішення дедалі частіше опираються на методологію машинного навчання. Важливо зазначити, що набір даних для бінарної класифікації fraud/non-fraud зазвичай ускладнений нерівномірним розподілом (наприклад, лише 0.17% транзакцій у загальному обсязі є шахрайськими). За таких обставин, класичні ML-алгоритми демонструють низьку чутливість до «міноритарного» класу, що вимагає застосування спеціалізованих технік балансування та ретельного відбору метрик для оцінювання перформансу моделей.

Ураховуючи стрімкий розвиток технологій машинного навчання (як класичного, так і квантового), апробацію різноманітних підходів до ресемплінгу (Random Under-/Over- Sampling, SMOTE, ADASYN, Tomek Links, SMOTE-Tomek тощо) і появу градієнтних бустинг-алгоритмів (XGBoost, LightGBM), ***тема порівняльного аналізу класичних ML-методів для***

детекції шахрайства у фінанансових транзакціях залишається вкрай актуальною.

Мета цієї роботи – розробити систему виявлення шахрайства у фінансових транзакціях на основі реалізації та порівняння класичних моделей машинного навчання (Logistic Regression, Random Forest, XGBoost) у поєднанні з різними методами балансування даних (Random Oversampling, SMOTE, SMOTE-Tomek, ADASYN), зокрема окреслити рекомендації щодо впровадження рішень відповідно до бізнес-потреб фінансових інфраструктур.

Для досягнення мети дослідження поставлено ***такі завдання***:

— Провести детальний огляд сучасних підходів до виявлення шахрайства з кредитними картками та методів роботи з дисбалансом класів.

— Проаналізувати набори даних типу «Credit Card Fraud Detection» (Kaggle: [2]), виконати початковий EDA для подальших інсайтів та бізнес-рекомендацій щодо збору даних, виявити й усунути дублікати, а також здійснити інженерію основних ознак (логарифмічне перетворення суми, стандартизація PCA-компонент, первинна робота зі стовпцем `Time`).

— Сформувати функціональний пайплайн із застосуванням `imblearn.pipeline.Pipeline`, що включає етапи ресемплінгу (Random OverSampler, SMOTE, SMOTE-Tomek, ADASYN), препроцесинг (`ColumnTransformer`) і класифікацію (Logistic Regression, Random Forest, XGBoost).

— Організувати підбір гіперпараметрів для кожної ML-моделі та стратегії балансування за допомогою `GridSearchCV` за врахування оптимізації метрики Average Precision (AP) у рамках стратифікованої крос-валідації.

— Оцінити найкращі конфігурації на незалежному тестовому наборі за розширеним набором метрик та встановлення пріоритезації їх розгляду (Accuracy, Precision, Recall, F_2-score, MCC, ROC-AUC, PR-

AUC) і виконати пошук порогу класифікації, що максимізує F_2-score.

— Подувати та проаналізувати Precision-Recall та ROC-AUC криві для ТОП-10 комбінацій «модель + ресемплер», а також сформувати Confusion Matrix для обраного оптимального порогу.

— Розробити рекомендації щодо практичного використання моделі у фінансових установах та розглянути перспективи подальших досліджень, виконавши інтерпретацію результатів, включно із аналізом feature importance для Random Forest та XGBoost.

Об'єктом дослідження є процес виявлення шахрайства у фінансових транзакціях у системах обробки кредитних карток.

Порівняльний аналіз класичних методів машинного навчання у поєднанні з підходами балансування даних для підвищення якості бінарної класифікації при сильно дисбалансованих фінансових даних – ***предмет тематики бакалаврської дипломної роботи***.

Опис структури кваліфікаційної роботи містить три ключових розділи, а саме:

— ***Дослідження предметної області, дані та підготовка ознак*** – проаналізовано нормативно-правову база України та масштабні інциденти та їхні наслідки для установ та клієнтів, узагальнено чинні підходи до виявлення шахрайства, розглянуто проблеми дисбалансу класів, описано набір даних та проведено попередню обробку та інженерію ключових ознак.

— ***Методологія побудови та оцінювання моделей*** – докладно розібрано архітектуру пайплайнів із ресемплінгом і препроцесингом, аргументовано вибір ключових алгоритмів (Logistic Regression, Random Forest, XGBoost), наведено процедуру підбору гіперпараметрів через `GridSearchCV`, перелічено та обґрунтовано вибір метрик оцінювання при дисбалансі.

— ***Експерименти та результати*** – презентовано результати крос-валідації та тестування на незалежному наборі даних, побудовано Precision-Recall та ROC-AUC криві, Confusion Matrix, виконано аналіз важливості фіч, обговорено інсайти в контексті практичних застосувань і обмежень дослідження, осягнуто розширення даної праці для перспектив подальшого розвитку тематики.

Крім того, кожен розділ містить підсумки за виконаним етапом, які є основою для більш глибокого занурення в наукове дослідження окремих аспектів.

1 ДОСЛІДЖЕННЯ ПРЕДМЕНОЇ ОБЛАСТІ, ДАНІ ТА ПІДГОТОВКА ОЗНАК

У цьому розділі розглянуто ключові аспекти проблеми шахрайства з кредитними картками, особливості наявних даних і підходи до їх попередньої обробки. Спочатку аналізується контекст фінансового шахрайства, види типових зловживань та їхній вплив на банки й клієнтів — як у грошовому, так і в репутаційному вимірі. Далі подано огляд еволюції методів виявлення шахрайства: від систем на основі правил до сучасних алгоритмів машинного навчання, зокрема ансамблевих підходів, таких як бустинг.

Окрема увага приділяється проблемі дисбалансу класів — типовому виклику у задачах детекції шахрайства, де частка позитивних прикладів становить менше 0.5%. Розглянуто основні техніки ресемплінгу (*ROS, SMOTE, SMOTE-Tomek, ADASYN*), а також їх переваги й обмеження.

Крім того, описано набір даних «*Credit Card Fraud Detection*» з платформи Kaggle, включно з анонімізованими ознаками V1–V28, створеними за допомогою методу головних компонент (PCA). Нарешті, подано огляд попередньої обробки даних та інженерії ознак: видалення дублікатів, масштабування суми транзакції, обробка часу та використання `ColumnTransformer` для побудови уніфікованого пайплайну обробки.

Цей розділ закладає теоретичну й практичну основу для подальшого аналізу, моделювання та оцінки систем виявлення шахрайських транзакцій.

1.1 Шахрайство з кредитними картками: проблематика й контекст

Шахрайство з кредитними картками залишається значною та динамічною загрозою для фінансових установ, продавців та споживачів усього світу. Глобальні втрати від шахрайства з картками досягли **33,83 мільярда доларів (USD)** у 2023 році, причому **США** становлять значну міру цих збитків – **14,32 мільярда доларів (USD)** [3]. Наведені показники підкреслюють

постійну вразливість платіжної системи, особливо в умовах зростання електронної комерції та впровадження цифрових гаманців. Шахраї активно вдосконалюють свої тактики, використовуючи викрадені дані карток, електронний скімінг та складні методи соціальної інженерії, посилені ШІ. Справжня вартість шахрайства виходить далеко за межі прямих грошових втрат, оскільки цей процес охоплює операційні витрати, репутаційну шкоду та підрив довіри клієнтів.

У 2024 році було розміщено **269 мільйонів** записів карток на платформах даркнету й клірнету. Дані карток без присутності (CNP) домінували серед викрадених записів, що свідчить про вплив шахрайства, що зростає, у електронній комерції [4].

Хоча чеки залишаються основною мішенню шахрайства з платежами (**63% організацій повідомили про спроби або фактичне шахрайство з чеками у 2024 році**), корпоративні/комерційні кредитні картки були ціллю для 21% організацій [5]. Кредитні картки були найчастіше ідентифікованим методом оплати у звітах про шахрайство, отриманих Федеральною торговою комісією (FTC) у 2024 році [6].

Відповідно до табл. 1.1 «Втрати від шахрайства з картками 2022 - 2023» робимо висновок про позитивний відносний відсотковий приріст фінансових збитків протягом року.

Таблиця 1.1 – Втрати від шахрайства з картками 2022 – 2023

Категорія втрат	2022 (млрд USD)	2023 (млрд USD)	Зміна (%)
Глобальні втрати від шахрайства з картками	33,45 [3]	33,83 [3]	+1,1%
Втрати від шахрайства з картками у США	13,61 [3]	14,32 [3]	+5,2%
Втрати від CNP (оцінка)	41 [7]	48 [7]	+17,1%

Відсоток споживачів США, які повідомили про втрату грошей через шахрайство, зріс з 27% у 2023 році до 38% у 2024 році [6]. Атаки із захопленням облікових записів (Account Takeover, ATO) збільшилися на 24% порівняно з попереднім роком, при цьому 24% із загальної кількості споживачів стали їхніми жертвами відносно до 18% у 2023 році. Середня вартість претензій щодо компрометації ділової електронної пошти (Business Email Compromise, BEC) різко зросла до 84 тис. доларів США у 2022 році до 183 тис. доларів США у 2023 році, поряд із 65% збільшенням виявлених глобальних відкритих втрат від шахрайства BEC. Атаки з викраденням облікових даних, що походять від фішингових кампаній, різко зросли на 703% у другій половині 2024 року [7].

Дані з табл. 1.2 «Розподіл основних категорій загального фінансового шахрайства за сумою втрат (2023 – 2024)» чітко вказують на прискорене зростання втрат від шахрайства із року в рік у різних категоріях та органах, що звітують. Послідовна висхідна тенденція свідчить про те, що поточні механізми запобігання фінансових махінацій та fraud-детекції є або недостатніми, або їх випереджає еволюційна складність алгоритмів методології незаконного збагачення кібер-шахраями.

Таблиця 1.2 – Розподіл основних категорій загального фінансового шахрайства за сумою втрат (2023 – 2024)

Категорія шахрайства	Загальні втрати (2023)	Загальні втрати (2024)	Ключові Підкатегорії / Середні втрати	Зміна %
Загальне Глобальне Шахрайство	$485.6 млрд [8]	$1.03 трлн [9]	–	+112%

Продовження таблиці 1.2

Захоплення облікових записів (АТО)	$12.7 млрд [13]	$15.6 млрд [13]	–	+22.8%
Загальне Шахрайство Споживачів США (FTC)	$10+ млрд [6]	$12.5 млрд [6]	Інвестиційні шахрайства: $5.7 млрд (2024) [14], шахрайство з видаванням себе за іншу особу: $2.95 млрд (2024) [10]	+25% [10]
Інтернет-Злочинність (ФБР, США)	$12.5 млрд [12]	$16 млрд [11]	Шахрайство з інвестиціями (криптовалюта): $6.5 млрд (2024) [11]	+33% [11]
Шахрайство з Ідентифікацією та Шахрайство (США)	$43 млрд [13]	$47 млрд [13]	Традиційне шахрайство з ідентифікацією: $27 млрд (2024) [13], шахрайство: $20 млрд (2024) [13]	+9.3%
Шахрайство з новими обліковими записами	$5.3 млрд [13]	$6.2 млрд [13]	–	+17%

Продовження таблиці 1.2

Компрометація ділової електронної пошти (BEC)	$6.7 млрд (глоб. прогн.) [8]	$183 тис. (середня вартість претензії, 2023) [7]	–	+65% (глоб. виявлені втрати) [7]
Шахрайство з картками (Світове)	$33.83 млрд [3]	N/A	–	+1.1% (2022-2023) [3]
Службове шахрайство (глоб.)	$3.1 млрд (загальні виявлені, 2022-2023) [14]	$145 тис. (середні втрати) [14]	Привласнення активів: $120,000 (середня) [15], шахрайство з фінансовою звітністю: $766 тис. (середня) [15]	+24% (середні втрати, 2022-2024) [14]

Надзвичайно низький показник відшкодування глобальних втрат від шахрайства (лише 4% відшкодовано) висвітлює критичну слабкість у контексті фінансового моніторингу компенсації збитків, тобто на кожні 100 доларів США, втрачених від шахрайства, 96 грошових одиниць є втраченими без повернення [16]. Відповідні наслідки для фінансових структур свідчать про необхідність розробки стратегічного бізнес-фокусу на боротьбі із причинами виникнення шахрайства – інвестування в надійні та гнучкі механізми попереднього дослідження транзакцій в умовах реального часу. Розслідування та нормо-правовий контроль процесу відшкодування з точки зору юридичного супроводу та підтвердження є не ефективним рішенням з точки зору збереження довіри громадськості та здатності фінансової системи

захищати активи та забезпечувати справедливість. За ігнорування розробки новітніх систем кібербезпеки управління фінансовими ризиками наслідком є зменшення залучення до послуг відповідних структур через почуття незахищеності.

Прогнозовані 400 млрд USD глобальних втрат від шахрайства з картками протягом наступного десятиліття свідчать про те, що такі неправомірні фінансові дії з кредитними картками є не тимчасовою проблемою, а вкоріненим та прогресивним викликом, що вимагає стійких, довгострокових проєктних капіталовкладень та адаптації [17].

1.1.1 Види шахрайських операцій із картками та їх наслідки

Розглянемо ключові види шахрайства з кредитними картками:

а) Шахрайство без наявності картки (Card-Not-Present, CNP):
   1) найдорожчий вид шахрайства для 61% компаній [18].
   2) у 2020 році на нього припало майже 7 з 10 втрат від фінансових махінацій для торговців та еквайрів, що становило $19,43 млрд у всьому світі [19].
   3) пряма кореляція між зростанням електронної комерції та домінуванням шахрайства CNP вказує на те, що цифрова трансформація, хоча і є вигідною, за своєю суттю розширює простір для атак цього типу шахрайства.
   4) фінансові установи й торговці повинні надавати пріоритет передовим методам автентифікації, таким як 3D Secure 2.0, токенізація, AI/ML [19], які можуть забезпечити безпеку транзакцій без фізичної картки.

б) Шахрайство з новими рахунками:
   1) втрати від шахрайства з новими рахунками досягли $6,2 млрд у 2024 році, порівняно з $5,3 млрд у 2023 році [13]. Такий вид

фінансових махінацій становить 90% усіх випадків шахрайства з кредитними картками.

2) перехід від використання викрадених реальних ідентифікаторів до створення синтетичних і сфабрикованих даних.
3) висвітлення необхідності інвестування у вдосконалення рішень для перевірки особистості, які можуть виявляти аномалії як ц структурованих, так і неструктурованих (фото згенеровані AI) даних ідентифікації та поведінкових патернах під час процесу реєстрації користувача.

в) Шахрайство із захопленням чинних рахунків (Account TakeOver, ATO):
1) 83% організацій зіткнулися принаймні з одним випадком захоплення рахунків за останній рік [7].
2) дані FinCEN показують, що банки США подали понад 178 тис. звітів про підозрілу діяльність, пов’язану з ATO, у 2024 році, що на 36% більше, ніж у 2023 році [20]. ATO є рушійним фактором, що сприяє іншим фінансовим злочинам, дозволяючи шахраям ініціювати перекази (Zelle, wire, ACH), додавати себе як уповноважених користувачів кредитних карток та змінювати дані рахунків, ефективно отримуючи повний контроль над фінансовим життям жертви [21].
3) ATO – блокер для інших видів шахрайства, що робить цей вид махінації критичною точкою. Це означає, що фінансові установи повинні надавати пріоритет надійній багатофакторній автентифікації, відбиткам пристроїв та поведінковій аналітиці для виявлення тонких аномалій, що вказують на спробу ATO, навіть за використання чинних облікових даних.

г) Електронний скімінг / Скімінг платіжних карток:
1) кількість заражень електронними скімерами Magecart зросла втричі з 2023 року до 11 тис. унікальних доменів електронної

комерції у 2024 році, що підкреслює поширеність атак цифрового скімінгу [4].

2) ФБР повідомило, що афери з електронним скімінгом призводять до щорічних втрат понад $1 млрд для власників карток та фінансових установ [22]. Зростання безконтактних транзакцій і транзакцій «tap-to-pay» призводить до зміни векторів атак скімінгу, що змушує шахраїв адаптувати свої методи.
3) перехід від традиційних фізичних скімерів на банкоматах/POS до більш складного цифрового електронного скімінгу (Magecart) або нових вразливостей у безконтактних технологіях [4].

д) Шахрайство з поверненням платежів (Chargeback Fraud):

1) очікується, що світовий обсяг повернення платежів зросте на 24% з 2025 по 2030 рік, досягнувши 324 млн транзакцій щорічно. Фінансовий вплив глобальних повернень платежів зросте з $33,79 млрд у 2025 році до $41,69 млрд у 2028 році [23].
2) За кожен $1 шахрайства американські торговці несуть середні витрати в розмірі $4,61, що підкреслює значні непрямі витрати, пов'язані з поверненням платежів [24].
3) фінансові установи потребують складної аналітики для класифікації справжнього шахрайства та неправомірного використання першою стороною грошових активів, аби розробити стратегії, що балансують запобігання махінаціям з досвідом клієнтів та лояльністю (надмірна бюрократизація та суворість → відмова клієнтів).

У табл. 1.3 «Ключові види шахрайства з кредитними картками: тенденції та втрати» зазначено підсумки щодо грошових збитків відповідно до класифікації фінансових махінацій за часових рамок 2023 – 2024 років, а також зі статистикою прогнозування даних найближчого десятиліття.

Таблиця 1.3 «Ключові види шахрайства з кредитними картками: тенденції та втрати»

Тип шахрайства	Ключова тенденція/статистика	Втрати (млрд/млн USD)	Періодизація
Шахрайство без фізичної картки (CNP)	Найдорожчий тип для 61% компаній [18]. 3-кратне зростання заражень Magecart e-skimmer у 2024 [4]. 269 млн викрадених записів карток у 2024 (переважно CNP) [4].	$19.43 млрд (глобально, 2020) [19].	2020, 2024
Шахрайство з новими рахунками	139 569 зареєстрованих випадків у Q1 2025 [25].Становить 90% усіх випадків шахрайства з кредитними картками [25].	$6.2 млрд (2024), зростання з $5.3 млрд (2023) [13]. $4.6 млрд не закритих залишків від синтетичних ідентифікаторів (США, 2024).	2023, 2024, Q1 2025

Продовження таблиці 1.3

Електронний скімінг	Глобальний ринок: $3.4 млрд (2024), прогноз до $5.3 млрд (2030) [25].	Понад $1 млрд (щорічно) для власників карток та ФУ (ФБР) [22].	2024, 2030 (прогноз)
Захоплення чинних рахунків (АТО)	Зростання атак на 24% у 2024 [7]. 83% організацій зіткнулися з АТО за останній рік [7]. 178 тис. SARs, пов'язаних з АТО, подано банками США у 2024 (+36% з 2023) [20].	$15.6 млрд (2024), зростання з $12.7 млрд (2023) [13].	2023, 2024
Шахрайство з поверненням платежів (Chargeback)	Глобальний обсяг зросте на 24% (2025-2028) до 324 млн транзакцій [23]. ~45% обсягу повернень платежів [18].	$33.79 млрд (2025), прогноз до $41.69 млрд (2028) [23].	2025, 2028 (прогноз)

1.1.2 Економічна репутація

Для фінансових установ кожен долар, втрачений через шахрайство, обходиться в $4,23 [7]. Цей множник підкреслює, що прямі втрати посилюються значними непрямими витратами за врахування розслідування, зусилля з відшкодування (компенсація) та серйозної шкоди для економічної репутації. Практично третина фінансових організацій (31%) повідомила про прямі витрати від шахрайства, що перевищують $1 млн у 2025 році, що є помітним зростанням порівняно з чвертю 2024 році [26].

Значні штрафи з порушенням правил боротьби з відмиванням грошей (AML), хоча й не є прямими втратами від шахрайства, становлять суттєві фінансові санкції для установ через недотримання вимог відповідного нормо-правового контролю та врегулювання. Глобальні заходи з примусового виконання AML склали $4,6 млрд у 2024 році [27]. Основні штрафи в 2024 році включали TD Bank ($3,1 млрд), Klarna Bank ($50 млн), Nordea Bank ($35 млн), Starling Bank ($29 млн) [28]. Розмір і частота штрафів за AML демонструють, що регулятори все більше нетерпимі до слабких контролів (див. табл. 1.4), а також є критичними факторами рушійної сили руйнації економічної репутації фінансових структур.

Таблиця 1.4 «Основні інциденти фінансового шахрайства»

Назва інциденту	Основний тип шахрайства	Ключова організація / особа	Оцінені фінансові втрати / вплив	Рік (роки) виявлення / викриття
Lehman Brothers	Бухгалтерське шахрайство	Lehman Brothers	$50 млрд прихованих позик	2008
Берні Медофф	Схема Понці	Берні Медофф	$64,8 млрд втрат інвесторів	2008
Enron	Бухгалтерське шахрайство	Enron	$74 млрд втрат акціонерів	2001
WorldCom	Бухгалтерське шахрайство	WorldCom	$180 млрд втрат інвесторів	2002
Satyam	Бухгалтерське шахрайство	Satyam	50 млрд рупій фальсифікацій	2009

Продовження таблиці 1.4

Wirecard	Бухгалтерське шахрайство, відмивання грошей	Wirecard	€1.9 млрд відсутніх / €30 млрд збитків	2020
FTX (Сем Бенкман-Фрід)	Привласнення цифрових активів	Сем Бенкман-Фрід	$8 млрд викрадених коштів клієнтів / $11 млрд конфіскації	2019 – 2022
Danske Bank	Відмивання грошей	Danske Bank	€200 млрд підозрілих транзакцій / $2 млрд штрафу	2007 – 2015
UBS (Квеку Адоболі)	Несанкціонована торгівля	Квеку Адоболі	$2,3 млрд втрат	2011

1.2 Методи виявлення шахрайства: від правил до машинного навчання

Цей підрозділ описує еволюцію виявлення шахрайства: від традиційних систем, заснованих на правилах, та класичного профайлінгу (базовий статистичний аналіз) до трансформаційного впливу машинного навчання (прикладна статистика), зокрема досліджень фундаментальних підходів та передових ансамблевих методів, таких як бустинг, який розширює можливості детекції фінансових махінацій за допомогою ітеративного вдосконалення та синергетичних комбінацій моделей.

ШІ та генеративний ШІ революціонізують fraud-детекцію, але одночасно використовуються шахраями для створення більш переконливих фішингових електронних листів, текстових повідомлень та клонування голосу (діпфейків) [29]. Згадки про шкідливі інструменти ШІ, такі як FraudGPT, зросли на 200% на рік у даркнеті, знижуючи бар'єр для входу в складні атаки соціальної інженерії [30, 31]. Широке впровадження ШІ фінансовими установами для виявлення шахрайства паралельно із прогресивним використанням ШІ шахраями для більш складних атак створює «гонку озброєнь», де технологічні досягнення з одного боку швидко вимагають контр-здобутків з іншого. Це означає, що впровадження ШІ для захисту є не одноразовим рішенням, а вимогою безперервних інвестицій для підтримання темпу «холодної війни».

Успіх BEC та шахрайства з видаванням себе за іншу особу ілюструє силу психологічної маніпуляції. Захист повинен виходити за рамки суто технічних заходів — необхідним є комплексний підхід, що поєднує передові системи виявлення на основі з безперервним, інноваційним навчанням для співробітників та клієнтів щодо розпізнавання тактик соціальної інженерії. «Людський брандмауер» рівноцінний технологічному, особливо з огляду на високий рівень повторного використання паролів та ефективність фішингу.

1.2.1 Правила (Rule-Based Systems) і класичний профайлінг

Системи детекції шахрайства, на основі правил, працюють за принципом «якщо…то [if…else]», виявляючи фінансові махінації на основі набору заздалегідь визначених умов або атрибутів, а саме: незвичайні часові мітки (часовий період між здійсненими транзакціями), номери рахунків, типи транзакцій та відповідні грошові суми [32]. База даних складається із бази правил та бази знань, які складаються фахівцями з безпеки, операційними працівниками та фінансовими стейкголдерами на основі відомих історичних моделей шахрайства, що становить основу експертної системи.

Поширені приклади правил включають позначення транзакцій, що відбуваються за межами звичайного географічного розташування користувача, раптові викиди (outliers) розподілу активності на рідко використовуваних рахунках або великі платежі, отримані з кількох нещодавно створених рахунків [32]. Система функціонує як детектор, ідентифікуючи одні транзакції із легітимною позначкою або як ті, що є підозрілими до фінансових правопорушень на основі правил та знань.

Серед переваг rule-based систем варто зазначити наступні тези:

— Швидкість. Завдяки низькій алгоритмічній складності вони можуть швидко сканувати та виявляти шахрайство, забезпечуючи швидкий скринінг транзакцій.

— Крім того, наведена стратегія пропонує високу прозорість. Рішення, прийняті системами, заснованими на правилах, легко інтерпретувати — кожну позначену транзакцію можна зіставити безпосередньо із конкретним правилом, яке було порушено. Це корисно для банківського аудиту, дотримання нормативних вимог та розуміння помилкових спрацьовувань або пропусків [33].

— Легкість інтеграції моделі, а також простота її впровадження. Розробка такого ПЗ вимагає мінімального досвіду, що створює умови для невеликих організацій або тих бізнесів, що на початкових засадах мають обмежені ресурси.

Недоліками таких стратегій є наступні твердження:

— Сліпі зони та відсутність адаптивності. Правила виявляють лише відомі патерни та не можуть виявляти нові в ході роботи системи, оскільки статичний характер обмежує функціонал в умовах невизначеності для потреб кризового аналізу та управління стратегіями детекції фінансових правопорушень.

— Жорсткість правил часто призводить до високої кількості помилкових визначень (хибнопозитивні або хибнонегативні результати), коли легітимні транзакції позначаються як підозрілі,

і навпаки. Вимірювання метрик precision та recall вимагають дорогого та трудомісткого ручного перегляду, спричиняючи незручності для клієнтів та продуктових і фінансових аналітиків. Загальна вартість деплойменту та підтримки традиційних систем може бути значно вищою в довгостроковій перспективі — вплив не лише на прямі втрати від шахрайства, але й на операційні бюджети та досвід клієнтів (client or user experience). Таким чином, «простота» впровадження не дорівнює «економічній ефективності» в поточних операціях.

- Крім того, правила вимагають постійних ручних оновлень та уточнень, щоб не відставати від мінливих методів шахраїв, що підкреслює фундаментальну неадекватність статичних, заздалегідь визначених правил проти інтелектуального та адаптивного супротивника, що створює постійний попит на більш динамічні та складні методи виявлення.
- Rule-based ПЗ важко справляється зі схемами шахрайства, які передбачають аналіз кількох взаємопов’язаних факторів або багатовимірних даних [34].
- Ризик оверфіттингу відповідно до історичних даних. Хоча системи, основані на правилах, не «навчаються» на даних, як моделі машинного навчання, проте такі ЕС часто розроблені на базі історичних патернах шахрайства [34].

Хоча досвід у галузі є вирішальним для початкового встановлення правил та **класичного профайлінгу**, така залежність від вимог часу, потреб бізнесу, стрімкого прогресу перформансу LLM та CV (ШІ та науки про дані), висхідної динаміки розвитку методологій як соціальної інженерії, так і кібершахрайства, вносить невід’ємні вразливості, що спираються на обмеження систем, зазначені вище.

Класичний профайлінг, який часто використовує статистичні методи, має на меті виявити аномалії або викиди в даних транзакцій шляхом

порівняння поточної поведінки зі встановленими еталонами, тенденціями або нормами, що базуються на отриманих історичних моделях [34]. Ключова задача полягає в тому, щоб виявити нерозпізнані та несанкціоновані дії, що призводять до фінансових втрат за мінімізації false positives / false negatives. Основою класичного профайлінгу є комплексний підхід, що враховує наступні парадигми:

— **Статистичний аналіз** включає такі методи, як регресійний аналіз, кластерний аналіз та різні статистичні тести для позначення підозрілих дій — історичні дані як встановлення норм.

— **Поведінковий профайлінг** встановлює базові показники для типової поведінки та звичок кожного клієнта. Моніторингу підлягають такі точки даних, як IP-адреси, інформація про місцезнаходження, використовувані пристрої, VPN/проксі, конфігурації системи та браузера, методи оплати, час входу в систему, значення транзакцій та звичайні моделі покупок [35]. Раптові відхилення від цієї встановленої базової лінії викликають сповіщення.

— **Аналіз груп рівних** (*Peer Group Analysis*) — метод, який виявляє окремі об'єкти (наприклад, рахунки кредитних карток), які починають поводитися інакше, ніж об'єкти, до яких вони раніше були подібними.

— **Аналіз точок перелому** (*Break Point Analysis*) — на відміну від аналізу груп рівних (PGA), цей метод працює на рівні окремого рахунку, виявляючи "точки перелому" або спостереження, де визначається аномальна поведінка для певного рахунку з часом.

Методологія має тотожні обмеження до rule-based систем, зокрема варто зауважити ще один суттєвий нюанс – це сувора вимога до якості даних: повинні бути чистими, добре підготовленими та повними. Викиди, відсутні значення або зашумлені дані можуть значно вплинути на ефективність та точність моделі.

1.2.2 Початок застосування машинного навчання: основні підходи

Машинне навчання (ML) забезпечує масштабовані та адаптивні підходи до виявлення шахрайських дій, аналізуючи великі обсяги даних для ідентифікації складних, прихованих закономірностей у фінансових транзакціях. Ці системи навчаються на історичних даних, поступово вдосконалюючи свою здатність до ідентифікації шахрайства, адаптуючись до нових умов та змін у поведінці зловмисників [36]. Крім того, вони відповідають сучасним вимогам щодо захисту персональних даних, зокрема GDPR та CCPA [37].

Навчання з учителем (*supervised learning*) базується на використанні мічених даних, у яких транзакції вже класифіковані як «шахрайські» або «чесні» [36]. Моделі отримують як вхідні характеристики транзакцій, так і відповідні мітки, що дозволяє алгоритмам виявляти закономірності та кореляції, які потім використовуються для класифікації нових, раніше не спостережуваних випадків.

Серед найбільш поширених алгоритмів навчання з учителем можна виділити [36]:

- **Логістична регресія** (*Logistic Regression*) — оцінює ймовірність належності до одного з двох класів (наприклад, шахрайство/не шахрайство) на основі незалежних змінних.
- **Наївний баєсівський класифікатор** (*Naïve Bayes Classifier*) — визначає ймовірність шахрайства, ґрунтуючись на байєсівській теоремі. Ефективний для невеликих наборів даних, однак поступається за точністю логістичній регресії при великому обсязі даних.
- **Дерева рішень** (*Decision Trees*) — використовують ієрархічну структуру для аналізу характеристик транзакції та прийняття рішень .

— **Випадковий ліс** (*Random Forest*) — агрегує множину дерев рішень, що дозволяє ефективно виявляти нелінійні взаємозв’язки між змінними.

— **XGBoost** — ансамблевий метод градієнтного бустингу, що покращує точність, послідовно навчаючи дерева з урахуванням помилок попередніх моделей.

— **Метод опорних векторів** (*Support Vector Machine*) — ефективний для великих обсягів даних, зокрема при виявленні шахрайства з кредитними картками, хоча й має високу обчислювальну складність.

— **K-найближчих сусідів** (*K-Nearest Neighbors*) — класифікує транзакції, порівнюючи їх із найближчими відомими прикладами, демонструючи високу точність, але обмежену інтерпретованість.

Алгоритми **навчання без учителя** (*unsupervised learning*) працюють із немічеными даними, самостійно групуючи транзакції в кластери за подібністю ознак [36]. Аномальні або нетипові зразки, що відхиляються від основної кластерної структури, можуть свідчити про потенційне шахрайство. Цей підхід особливо цінний у ситуаціях, коли неможливо отримати анотовані дані або виникають нові, раніше невідомі сценарії шахрайства. Зокрема, кластеризація (наприклад, алгоритм *K-means*) дозволяє виявляти відхилення від нормальної поведінки.

Навчання з підкріпленням, або винагородою (*Reinforcement Learning*) ґрунтується на ітеративному процесі взаємодії з середовищем, де алгоритм отримує зворотний зв’язок у вигляді винагороди або штрафу, поступово вдосконалюючи свою стратегію дій (див. рис. 1.1 – рис. 1.2). Особливістю є відсутність потреби в мічених даних, що забезпечує широку застосовність в умовах обмеженої інформації про поточні шахрайські схеми.

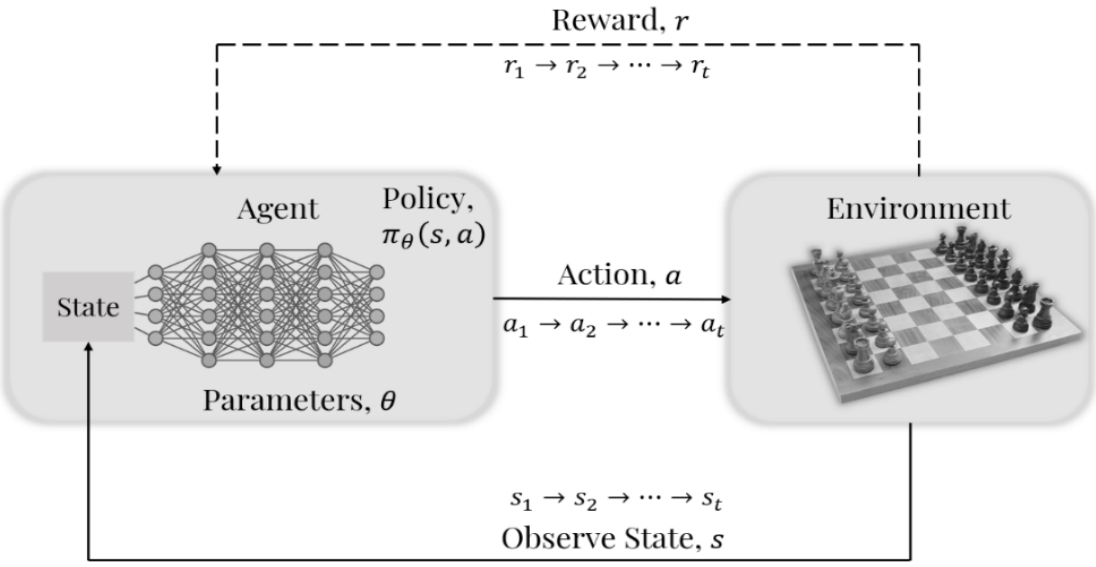

Рисунок 1.1 – Парадигма Reinforcement learning

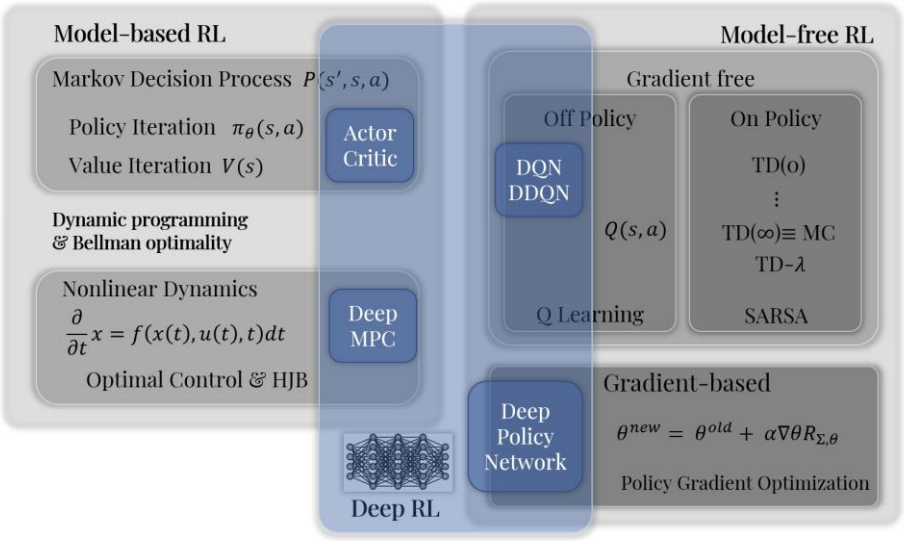

Рисунок 1.2 – Огляд методів навчання Reinforcement learning

Нейронні мережі вирізняються здатністю до моделювання складних, нелінійних взаємозв'язків у великих і багатовимірних наборах даних. Вони здатні працювати як у контексті навчання з учителем, так і без нього. Крім того, такий метод є основною в умовах підкріплення в контексті глибокого навчання з «винагородою» (див. рис. 1.2). Одним з ефективних архітектурних рішень у сфері виявлення шахрайства є автокодери, які дозволяють знаходити

аномальні зразки шляхом виявлення відхилень між вхідними та відновленими даними [36].

Машинне навчання забезпечує низку істотних переваг порівняно з традиційними, заснованими на правилах або статистичних методах виявлення шахрайства:

- Гнучкість та адаптивність — на відміну від статичних систем із жорстко закладеними правилами, алгоритми ML здатні до самонавчання та оперативного оновлення моделей у відповідь на нові типи шахрайства. Це дозволяє переходити від реактивного до проактивного виявлення аномалій, включаючи нетипові та новітні шахрайські схеми, які могли б залишитися непоміченими у рамках ручного аналізу або фіксованих правил [36].
- Обробка різнорідних даних — традиційні підходи, як правило, опрацьовують лише структуровані транзакційні дані. Натомість, ML дозволяє інтегрувати як структуровані, так і неструктуровані дані (наприклад, текстові повідомлення, зображення, документи) за допомогою суміжних технологій, таких як обробка природної мови (NLP) та комп'ютерний зір. Це особливо важливо в умовах цифровізації, коли фінансові дані стають високовимірними, складними й нестандартними. Традиційні методи стикаються із проблемою так званого прокляття розмірності [38], у той час як ML ефективно масштабується й адаптується до зростання обсягу даних
- ML-системи забезпечують вищу точність, знижуючи кількість помилкових спрацьовувань за рахунок аналізу складних багатофакторних залежностей. На відміну від бінарних рішень, притаманних правилам, вони можуть повертати імовірнісну оцінку ризику шахрайства, що дозволяє гнучкіше реагувати на виявлені загрози [39].

— Автоматизація виявлення шахрайства знижує ризик людських помилок, підвищуючи відповідність нормативно-правовим актам. Це особливо актуально в умовах постійного посилення регуляторних вимог, зокрема в Європі та США.
— Моделі машинного навчання демонструють високі можливості масштабування. У міру зростання обсягу даних, їх продуктивність, як правило, зростає — завдяки поглибленому вивченню структур даних і вдосконаленню ознак [39].
— Водночас використання складних моделей, таких як глибокі нейронні мережі, супроводжується зниженням прозорості. Це створює виклики для фінансових установ, яким необхідно обґрунтовувати рішення перед регуляторами, аудиторами та клієнтами [40]. Таким чином, інтерпретованість моделей виявлення шахрайства є визначальним науковим викликом і ключовою тенденцією у сучасних дослідженнях [41].

1.2.3 Ансамблеві моделі та бустинг: короткий огляд

Ансамблеві методи відіграють ключову роль у підвищенні точності та надійності виявлення шахрайства шляхом об’єднання результатів кількох окремих моделей прогнозування. Вони забезпечують значні переваги порівняно з традиційними одноетапними підходами завдяки використанню колективної сили різних алгоритмів [41].

Базова ідея ансамблевого підходу полягає в тому, що комбінація прогнозів кількох моделей здатна компенсувати індивідуальні недоліки кожної з них, зменшуючи вплив упередженості (*bias*) і дисперсії (*variance*), характерних для окремих моделей. Такий підхід дозволяє підвищити загальну здатність алгоритму, зменшити рівень помилкових спрацьовувань та краще адаптуватися до динамічних змін у шаблонах шахрайства [42].

Ключові ансамблеві техніки:

— **Бейгінг** (*Bootstrap aggregating*) — у цьому підході кілька версій базової моделі навчаються незалежно на різних випадкових підмножинах вихідних даних (метод бутстрепу). Кінцевий прогноз формується шляхом усереднення (для регресії) або голосування більшості (для класифікації). *Random forest* є типовим прикладом бейгінгу, де формуються множинні дерева рішень, що забезпечують підвищену стійкість до перенавчання та покращену загальну точність [42].

— **Стекінг** (*Stacking*) — цей метод поєднує прогнози різних базових моделей (можливо, різного типу) у «мета-модель», яка навчається на їхніх виходах і формує фінальний прогноз. Таким чином, стекінг дозволяє оптимально використати гетерогенність моделей.

Бустинг — це послідовний ансамблевий метод, у якому кожна нова модель навчається з урахуванням помилок попередньої. Основна мета полягає в поступовому зменшенні помилок класифікації шляхом зосередження на складних прикладах, які були неправильно класифіковані раніше [43]. Цей підхід покращує здатність моделі до узагальнення та особливо ефективний при роботі з незбалансованими вибірками, характерними для задач виявлення шахрайства.

До основних реалізацій алгоритмів бустингу належать:

а) **AdaBoost** (*Adaptive Boosting*) — один з перших та найвідоміших алгоритмів бустингу, який формує ансамбль із послідовних слабких учнів (наприклад, *decision stum*ps — дерева з одним розгалуженням) [44]:

1) Після кожної ітерації ваги неправильно класифікованих прикладів збільшуються, що змушує наступну модель зосереджуватись саме на складних транзакціях.
2) У сфері виявлення шахрайства AdaBoost ефективно виявляє тонкі закономірності, які важко розпізнати звичайним

класифікаторам, особливо при роботі з незбалансованими даними. Його гнучкість також дозволяє використовувати різноманітні слабкі моделі, включно з SVM і нейронними мережами [43].

б) **Gradient Boosting Machine** (GBM):

1) На відміну від AdaBoost, який змінює ваги, GBM мінімізує функцію втрат за допомогою градієнтного спуску. Кожен новий слабкий учень прогнозує залишки (похибки) попередньої моделі, що дозволяє поступово покращувати точність.
2) GBM активно застосовується у фінансовій сфері, зокрема для оцінки ризиків і детектування шахрайських транзакцій, демонструючи вищу точність порівняно з фіксованими правилами [43].

Серед переваг алгоритмів бустингу слід виокремити наступні характеристики:

— Висока прогностична точність, оскільки ітеративний процес дозволяє зменшувати похибки на кожному етапі, поступово наближаючись до оптимального рішення.
— Алгоритми бустингу надають більшої ваги важливим ознакам, що дозволяє ефективно виявляти складні та приховані закономірності у даних.
— Ефективність при роботі з незбалансованими вибірками — шахрайські дії часто є рідкісними подіями. Завдяки зміні ваг, бустинг дозволяє моделі зосереджуватись саме на цих рідкісних, але критично важливих випадках [43].
— Бустинг є відносно стійким до викидів, що особливо актуально у фінансових наборах даних.

— Окрім виявлення шахрайства, бустинг широко використовується в галузях, пов’язаних з медичною діагностикою, кредитним скорингом, рекомендаційними системами тощо [42].

Обмеженнями методології є наступні викладені наступним чином:

— Через послідовний характер побудови моделей бустинг повільніший за паралельні методи, що обмежує його масштабованість на великих даних.

— Хоча бустинг загалом є стійким, наявність надмірного шуму або помилкових анотацій у навчальній вибірці може призводити до перенавчання.

— Через послідовну природу навчання масштабування на масиви з мільйонами транзакцій потребує значних обчислювальних ресурсів.

— Використання великої кількості ітерацій або надмірно складних моделей може призвести до втрати здатності узагальнювати, особливо на нових даних, що є ризиком перенавчання. Проте за комплексного аналізу за допомогою множини метрик можна реалізувати більш детальний репорт щодо перформансу конкретної моделі відповідно до вимог точності бізнес-контекстів.

— Необхідність точного добору гіперпараметрів (кількість ітерацій, швидкість навчання, глибина дерева тощо) вимагає ретельної оптимізації для запобігання як *overfitting*, так і *underfitting* [43].

1.3 Проблема дисбалансу класів у фінансових даних та основні стратегії ресемплінгу

Дисбаланс класів є однією з найсерйозніших проблем у сфері застосування методів машинного навчання для виявлення фінансового шахрайства. У більшості реальних фінансових наборів даних частка шахрайських транзакцій є вкрай низькою, часто становлячи лише 0,173% від

загального обсягу записів [2]. Така значна асиметрія у розподілі класів призводить до того, що традиційні моделі машинного навчання схильні до упередженості на користь мажоритарного класу, що зумовлює високу частку хибнонегативних результатів — тобто пропущених випадків шахрайства.

Ця проблема має не лише технічні, а й критично важливі економічні наслідки, оскільки кожна пропущена шахрайська транзакція може спричинити прямі фінансові втрати, а також репутаційні ризики для фінансових установ. Відповідно, для забезпечення високої чутливості систем виявлення шахрайства необхідне впровадження спеціалізованих підходів до обробки дисбалансованих даних, серед яких ключову роль відіграють стратегії **ресемплінгу**.

У межах цієї дипломної роботи здійснено детальний аналіз впливу сильного дисбалансу класів на якість прогнозування та розглянуто основні стратегії ресемплінгу, зокрема:

- Random OverSampling (ROS) — метод випадкового дублювання прикладів міноритарного класу.
- SMOTE (*Synthetic Minority Over-sampling TEchnique*) — техніка синтетичного надсемплінгу шляхом інтерполяції.
- SMOTE-Tomek — комбінований підхід, що поєднує генерацію нових точок та очищення даних від межових аномалій.
- ADASYN (*Adaptive Synthetic Sampling*) — адаптивна модифікація SMOTE, яка зосереджується на важких для класифікації прикладах.

Для кожного з методів надається математичне пояснення принципів роботи, перелік переваг і недоліків, а також практичні рекомендації щодо доцільності їх використання в системах виявлення фінансового шахрайства. Розуміння та ефективне застосування цих підходів є критично важливим для побудови точних, стійких та адаптивних моделей у контексті реального банківського чи страхового середовища.

1.3.1 Random OverSampling (*ROS*)

Random OverSampling — найпростіший метод балансування класів, що полягає у випадковому дублюванні зразків з класу меншини з поверненням до досягнення бажаного співвідношення між кількостями зразків класів. Усі зразки меншини мають рівну ймовірність бути обраними для реплікації [45] (див. рис. 1.3 – 1.4).

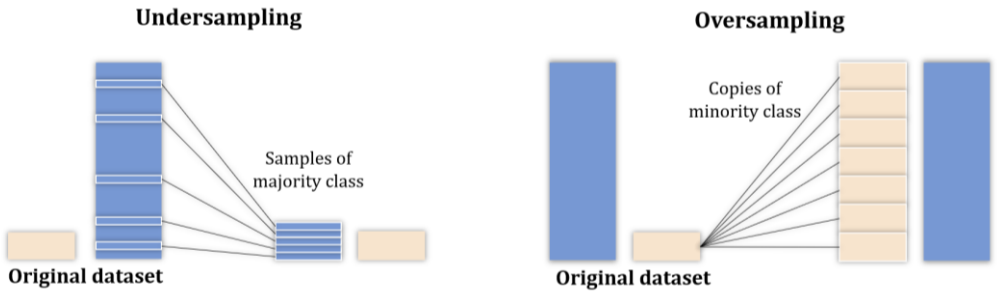

Рисунок 1.3 – Найпростіші методології балансування класів

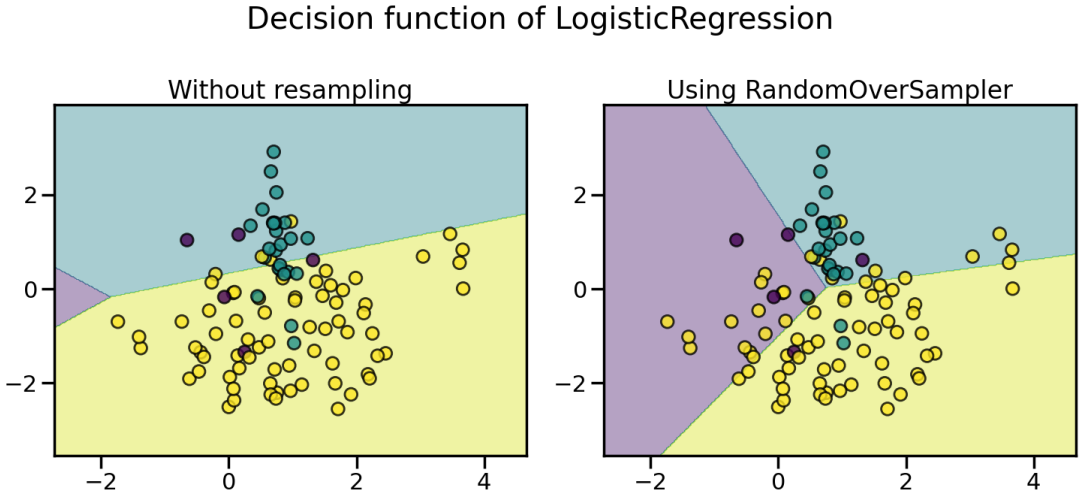

Рисунок 1.4 – Порівняння функції прийняття рішень класифікатора (тут: логістична регресія), навченої на основі оригінальної та збалансованої вибірки за допомогою Random OverSampling [45]

Нехай $X_{min} = \{x_1, \dots, x_{N_{min}}\}$ — множина зразків міноритарного класу, тоді N_{max} — кількість зразків мажоритарного класу.

Тоді нові зразки створюються вибіркою з повторенням:

$$X_{new} = \{x_{i_k}\}_{k=1}^{N_{max}-N_{min}} \quad (1.1)$$

де $i_k \sim \mathcal{U}\{1, 2, \dots, N_{min}\}$.

Після цього розмір класу меншини стає $N'_{min} = N_{max}$.

Попри простоту реалізації, ROS має низку істотних недоліків, які можуть суттєво впливати на ефективність моделей виявлення шахрайства. Один з ключових ризиків полягає в перенавчанні (*overfitting*) — дублювання чинних зразків міноритарного класу збільшує ймовірність того, що модель буде запам'ятовувати повторювані шаблони, замість того щоб вивчати узагальнювані ознаки. У результаті це може призвести до того, що модель демонструє високу точність на тренувальних даних, але істотно втрачає продуктивність на реальних або невідомих шахрайських транзакціях.

Крім того, ROS не генерує нову інформацію, тобто не вносить жодної різноманітності у простір ознак, що потенційно обмежує здатність моделі до узагальнення. У високовимірних просторах це стає особливо проблематичним, а саме: просте дублювання не сприяє поліпшенню розділення класів, яке є критичним у випадках із багатьма ознаками.

Ще одним практичним недоліком є збільшення обсягу навчального набору, що може призвести до підвищеного часу навчання, особливо при використанні ресурсомістких алгоритмів, таких як градієнтний бустинг чи глибокі нейронні мережі. У разі наявності викидів або шумових зразків у міноритарному класі, ROS також має ризик посилення цих небажаних впливів, оскільки дублює їх без фільтрації [47].

Ці обмеження вказують на важливий компроміс між простотою та ефективністю у виборі методів аугментації даних. Хоча ROS є інтуїтивно зрозумілим і легко реалізується, він не здатен повною мірою відобразити складність реального розподілу класів. Натомість більш складні методи, такі як SMOTE або ADASYN, намагаються розв'язати ці проблеми шляхом генерації синтетичних прикладів, що враховують локальну структуру даних.

Однак це досягається ціною вищої обчислювальної складності, а також можливого створення нереалістичних або артефактних зразків.

Таким чином, вибір конкретної стратегії ресемплінгу має базуватися не лише на прагненні до симетрії класів, але й на здатності методу забезпечити достовірне відтворення характеристик міноритарного класу при прийнятному рівні ресурсних та імплементаційних витрат. У випадку швидко змінних шахрайських патернів **доцільніше обирати ті методи, які вводять нову (навіть синтетичну) інформацію в простір ознак** — з огляду на необхідність кращої адаптації до нових сценаріїв.

1.3.2 SMOTE (*Synthetic Minority Over-sampling TEchnique*)

SMOTE генерує синтетичні зразки міноритарного класу шляхом інтерполяції між чинними зразками та їх k-найближчими сусідами в просторі ознак. Це розширює область, зайняту класом меншини, і зменшує ризик перенавчання порівняно з простим дублюванням (ROS) [48] (див. рис. 1.5).

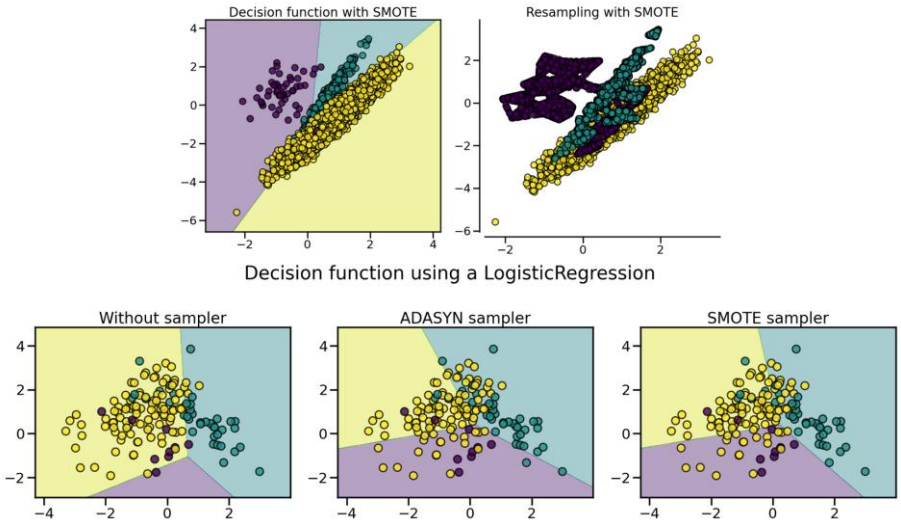

Рисунок 1.5 – Моделювання техніки балансування SMOTE; порівняння функції прийняття рішень класифікатора (тут: логістична регресія), навченої на основі оригінальної та збалансованої вибірки за допомогою SMOTE та ADASYN [45]

Для кожного $x_i \in X_{min}$ знайти множину k найближчих сусідів у класі меншини:

$$NN_k\ (x_i) = \{x_i^1, \dots, x_i^k\} \tag{1.2}$$

Щоб створити один синтетичний зразок, треба випадково обрати один із сусідів x_i^j і згенерувати:

$$x_{new} = x_i + \lambda(x_i^j - x_i) \tag{1.3}$$

де $\lambda \sim \mathcal{U}(0,1)$.

Надалі алгоритм повторює вище зазначений крок необхідну кількість разів (зазвичай задається бажаним коефіцієнтом OVER-SAMPLING).

Метод SMOTE синтезує нові, але схожі приклади, що дозволяє доповнити навчальну вибірку новою інформацією та сприяє кращому навчанню моделей для виявлення шахрайських транзакцій. Такий підхід дає змогу уникнути перенавчання, характерного для простого оверсемплінгу [50]. SMOTE ефективно збалансовує вибірку, забезпечуючи алгоритм достатньою кількістю прикладів шахрайства, що, у свою чергу, знижує упередженість моделі щодо мажоритарного класу [49]. Метод виявляється особливо цінним у ситуаціях з екстремальним дисбалансом класів, значно підвищуючи здатність моделі виявляти рідкісні випадки шахрайства. Результати досліджень також свідчать, що поєднання SMOTE з класифікаційними алгоритмами, зокрема Random Forest, може забезпечити кращі результати у задачах виявлення шахрайства.

Разом з тим, генерація синтетичних зразків призводить до збільшення загального обсягу даних, що може подовжити час навчання моделі [49]. У випадку, коли міноритарний клас містить шум або викиди, SMOTE має потенціал посилити ці недоліки, генеруючи нові приклади на основі шумних або нетипових точок. Крім того, метод може створювати нереалістичні або надмірно узагальнені зразки, особливо в умовах наявності викидів у вихідних даних, що ускладнює точне визначення меж між класами [51]. SMOTE є чутливим до шуму та може демонструвати знижені результати при

застосуванні до складних наборів даних. Зокрема, він не завжди ефективно працює з прикладами, розташованими поблизу межі рішень, що може призводити до помилкової класифікації.

Ці обмеження висвітлюють проблему так званого «синтетичного реалізму». Незважаючи на мету SMOTE — створення «нових, але схожих прикладів» задля уникнення дублювання наявних даних, існує занепокоєння щодо можливості генерації нереалістичних або надмірно узагальнених зразків та посилення шуму. Це вказує на фундаментальну проблему — синтетичні дані, хоч і збільшують представлення міноритарного класу, повинні адекватно відображати базовий розподіл реальних шахрайських транзакцій. У разі якщо синтетичні приклади не є реалістичними, вони можуть ввести хибні закономірності або шум, що заважатиме здатності моделі до генералізації у виявленні справжнього шахрайства. У відповідь на ці виклики розробляються більш складні гібридні методи, такі як ***SMOTE-Tomek*** або підходи, засновані на генеративних змагальних мережах (GAN), які спрямовані на підвищення якості та достовірності синтетичних даних, розширюючи межі досліджень за межі простої інтерполяції.

1.3.3 SMOTE-Tomek (комбінований підхід)

SMOTE-Tomek — це гібридний метод, що поєднує дві техніки:

- SMOTE для поповнення міноритарного класу синтетичними зразками.
- Tomek links для очищення «шумових» або граничних зразків, що лежать дуже близько між класами. Після SMOTE знаходять усі парні зв’язки Томека та видаляють з них зразки більшості, щоб краще визначити межу між класами [52, 53].

Математичну основу SMOTE було описано у підрозділі 1.3.2, тому розглянемо принцип роботи алгоритму Tomek links: пара (x_i, x_j), де $x_i \in$

$X_{min}, x_j \in X_{max}$, є Tomek link, якщо вони є взаємно найпершими сусідами (див. рис. 1.6):

$$d(x_i, x_j) = \min_{x_k \neq x_i} d(x_i, x_k) = \min_{x_k \neq x_j} d(x_j, x_k) \quad (1.4)$$

де d — метрика відстані (евклідова). З кожної такої пари видаляють саме x_j (зразок більшості).

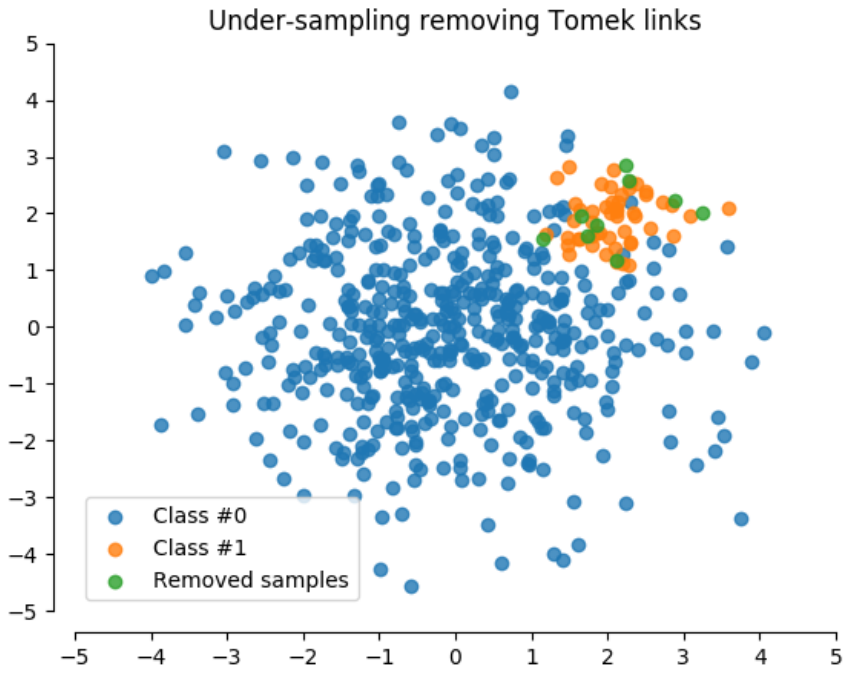

Рисунок 1.6 – Результат роботи етапу очищення граничних зв'язків за застосування методу Tomek Links [54]

Хоча етап очищення є корисним для покращення якості даних, видалення екземплярів мажоритарного класу все ж може призвести до втрати потенційно цінної інформації. Двоетапний підхід, який передбачає застосування SMOTE з подальшим очищенням за допомогою методу Tomek Links, супроводжується підвищеними обчислювальними витратами порівняно з використанням окремих методів ресемплінгу. Крім того, ефективність такого підходу може виявитися чутливою до вибору параметрів як для SMOTE (зокрема, кількість найближчих сусідів k), так і для алгоритму визначення пар Tomek Links.

1.3.4 ADASYN (*Adaptive Synthetic Sampling*)

ADASYN адаптивно генерує більше синтетичних зразків для тих прикладів меншини, які є «важчими» (тобто біля яких у k-сусідів є більше зразків більшості). Це зосереджує зусилля на регіонах простору, де модель має більшу ймовірність помилитись, й таким чином підвищує загальну стійкість класифікатора [55].

Поетапно проілюстровану математичну основу методу балансування даних на базі наступного алгоритму:

— Для кожного $x_i \in X_{min}$ порахувати кількість сусідів більшості серед k найближчих усього набору Δ_i.

— Обчислити відносне співвідношення:

$$r_i = \frac{\Delta_i}{k}, \hat{r_i} = \frac{r_i}{\sum_j r_j}, \sum_i \hat{r_i} = 1 \quad (1.5)$$

— Визначити загальну кількість синтетичних зразків:

$$G = (N_{max} - N_{min}) \times \beta \quad (1.6)$$

де $\beta \in [0, 1]$ — бажаний коефіцієнт балансу.

— Для кожного x_i кількість нових зразків $g_i = \hat{r_i} G$.

— Згенерувати g_i зразків за SMOTE-формулами 1.2 та 1.3.

Адаптивна природа ADASYN дозволяє зосереджувати зусилля на тих областях, де нові зразки є найбільш потрібними, а саме для складно класифікованих екземплярів міноритарного класу. Це може позитивно впливати на продуктивність моделі, особливо у задачах виявлення рідкісних і критично важливих подій, таких як фінансове шахрайство. ADASYN демонструє певну стійкість до шуму в даних, оскільки механізм генерації синтетичних зразків базується на оцінці щільності, що знижує ймовірність посилення шумових прикладів. Метод легко інтегрується в чинні конвеєри машинного навчання та може бути використаний у поєднанні з широким спектром класифікаційних алгоритмів. Завдяки орієнтації на щільність,

ADASYN зменшує ризик перенавчання та покращує здатність моделі до генералізації [56].

Фундаментальна ідея ADASYN — це генерація синтетичних прикладів на основі оцінки щільності та розподілу складності серед екземплярів міноритарного класу. Таким чином, на відміну від рівномірних або випадкових стратегій ресемплінгу (як-от *Random OverSampling* чи базовий *SMOTE*), ADASYN реалізує інтелектуальний, контекстно-залежний підхід до синтезу. Не всі представники міноритарного класу однаково важливі для навчання, і фокус має бути на генерації нових прикладів у регіонах, де модель стикається з найбільшими труднощами (мінімальна еквклідова відстань між зразками різних класів). Цей механізм є важливою відповіддю на виклики, пов'язані з «прокляттям розмірності» та складністю реальних фінансових даних, де межі класів часто є нелінійними (див. рис. 1.7).

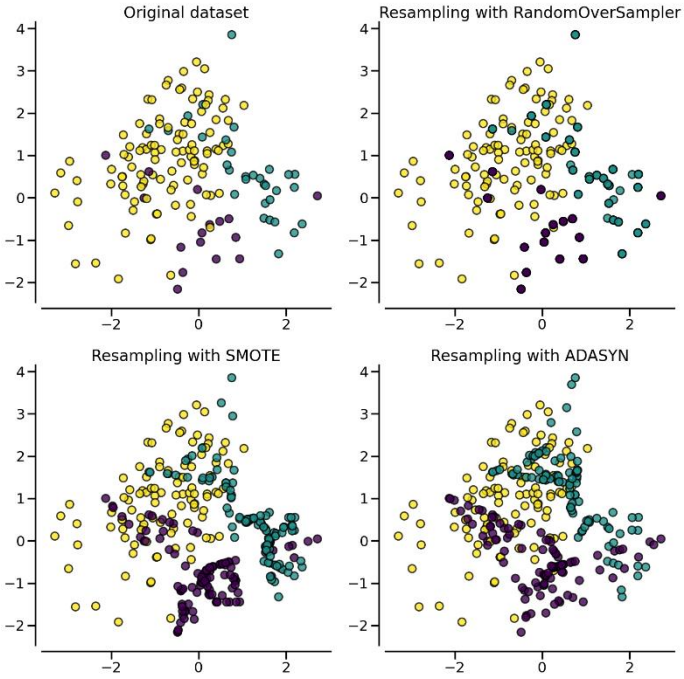

Рисунок 1.7 – Порівняльний аналіз технік балансування даних (акцент на різницю перформансу SMOTE та ADASYN) [45]

Незважаючи на переваги, застосування ADASYN може супроводжуватись високими обчислювальними витратами, особливо при

обробці великих наборів даних. У випадку необережної реалізації, надмірна генерація синтетичних прикладів може все ще призводити до перенавчання. З метою мінімізації цього ризику необхідне ретельне налаштування гіперпараметрів алгоритму. Крім того, ADASYN може іноді генерувати нерелевантні або непридатні для навчання синтетичні зразки в певних ситуаціях [57].

1.4 Набір даних «*Credit Card Fraud Detection*» (Kaggle)

Набір даних [2] містить транзакції, здійснені за допомогою кредитних карток у вересні 2013 року європейськими власниками карток (набір історичних даних). У цьому датасеті представлені транзакції, що відбулися протягом двох днів, серед яких виявлено **492 випадки шахрайства з загальної кількості у 284807 транзакцій**. *Набір даних є значно незбалансованим*: позитивний клас (шахрайство) становить лише 0,172% від усіх транзакцій (див. рис. 1.8).

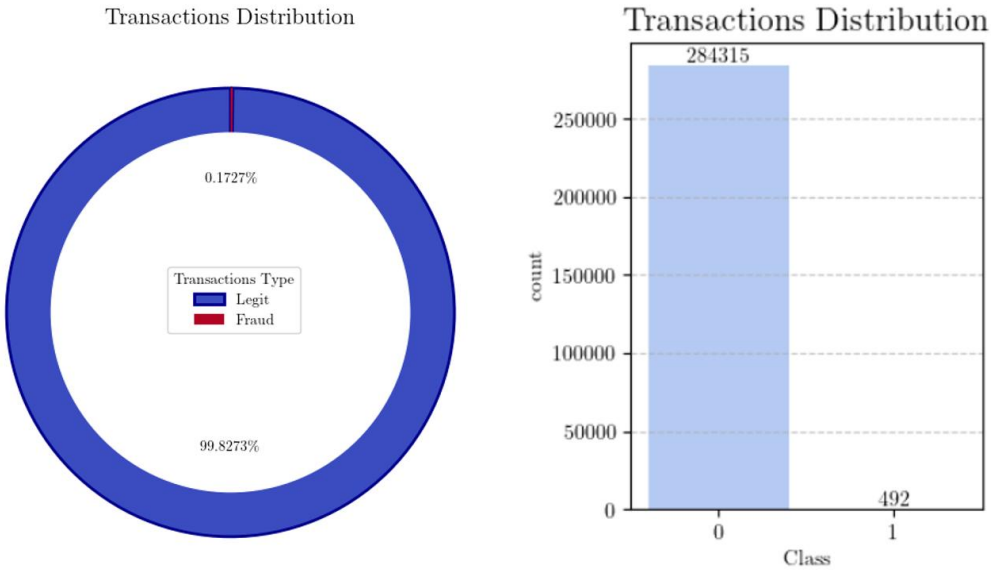

Рисунок 1.8 – Donut та bar chart, що ілюструє співвідношення класів (fraud/non-fraud) фінансових транзакцій у наведеному наборі даних — крайня незбалансованість

Дані було зібрано та проаналізовано в рамках дослідницької співпраці між компанією Worldline та групою машинного навчання [58] Université Libre de Bruxelles у сфері майнінгу великих даних і виявлення шахрайства.

1.4.1 Структура датасету: стовпці Time, V1–V28, Amount, Class

Набір містить лише числові змінні, отримані в результаті застосування методу головних компонент (PCA).

Через конфіденційність дослідники не можуть надати оригінальні ознаки та додаткову інформацію щодо даних (див. рис. 1.9).

На ринку праці це нормальна практика — у фінансових контрактах здебільшого ознаки (*features*) або зашифровані, або не опубліковані через NDA:

- Ознаки `V1, V2, … V28` — це головні компоненти, отримані за допомогою PCA (*Principal Component Analysis*). Єдині ознаки, які не були перетворені за допомогою PCA, — це `Time` та `Amount`.
- Ознака `Time` містить інформацію про кількість секунд, що минули від першої транзакції до кожної наступної транзакції у наборі даних.
- Ознака `Amount` — це сума транзакції, і вона може використовуватися, наприклад, для залежного від прикладів навчання з урахуванням вартості.
- Ознака `Class` є цільовою змінною і приймає значення 1 у разі шахрайства та 0 в інших випадках.

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502

5 rows × 31 columns

Рисунок 1.9 – Демонстрація перших п’яти рядків набору даних *Credit Card Fraud Detection* (31 колонка та 284807 рядків)

На основі дослідження розподілів `Time` та `Amount` можна розглянути перспективу подальших напрацювань з метою розробки більш специфічної системи метрик на основі EDA, а не лише результативності моделей машинного навчання (критерії оцінки перформансу методів). Тобто для більш точного розпізнавання шахрайських транзакцій додатково варто створити фільтри-детектори для ґрунтовної продуктової аналітики та виведення бізнес-інсайтів з даних (див. рис. 1.10 – 1.13).

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

Рисунок 1.10 – Перевірка чистоти даних (наявність пропущених значень), структур даних кожної з колонок

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	...	2.8
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16	1.487313e-15	-5.556467e-16	1.213481e-16	-2.406331e-15	...	1.6
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00	...	7.3
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01	...	-3.4
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01	...	-2.2
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02	...	-2.9
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-01	...	1.8
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.559499e+01	...	2.7

8 rows × 31 columns

Рисунок 1.11 – Описова таблиця, що характеризує статистичний опис векторів колонок (наприклад, максимальної кількістю секунд від поточної транзакції до попередньої — це 172792 секунд, тобто ≈ 48 годин; або менше 25% всіх транзакцій мали період від поточної транзакції до попередньої менше 54201 секунди ≈ 15 годин)

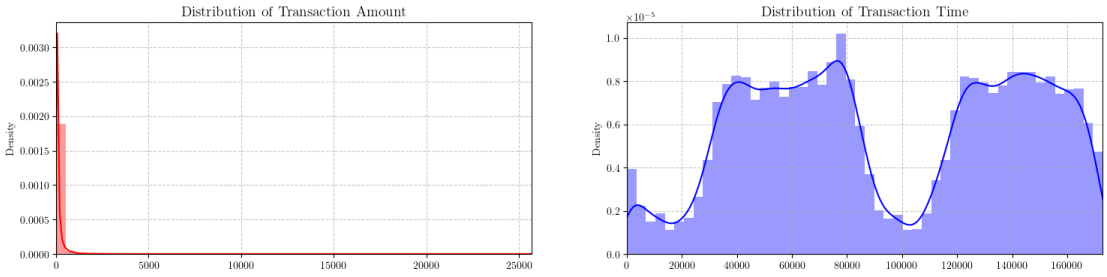

Рисунок 1.12 – Розподіли даних за `Amount` та `Time`

	Class	transactions_count	avg_amount	avg_time
1	1	473	123.871860	80450.513742
0	0	283253	88.413575	94835.058093

Рисунок 1.13 – Зведена таблиця середніх значень `Amount` та `Time` відповідно до класів транзакцій: fraud (1) та non-fraud (0)

1.4.2 Опис анонімізованих PCA-компонент V1–V28

За візуалізації розподілів анонімізованих PCA-компонент за допомогою box plots та розрахунків коефіцієнтів асиметрії відповідно до кожного вектора

(колонки) даних було ідентифіковано суттєву кількість викидів (див. рис. 1.14).

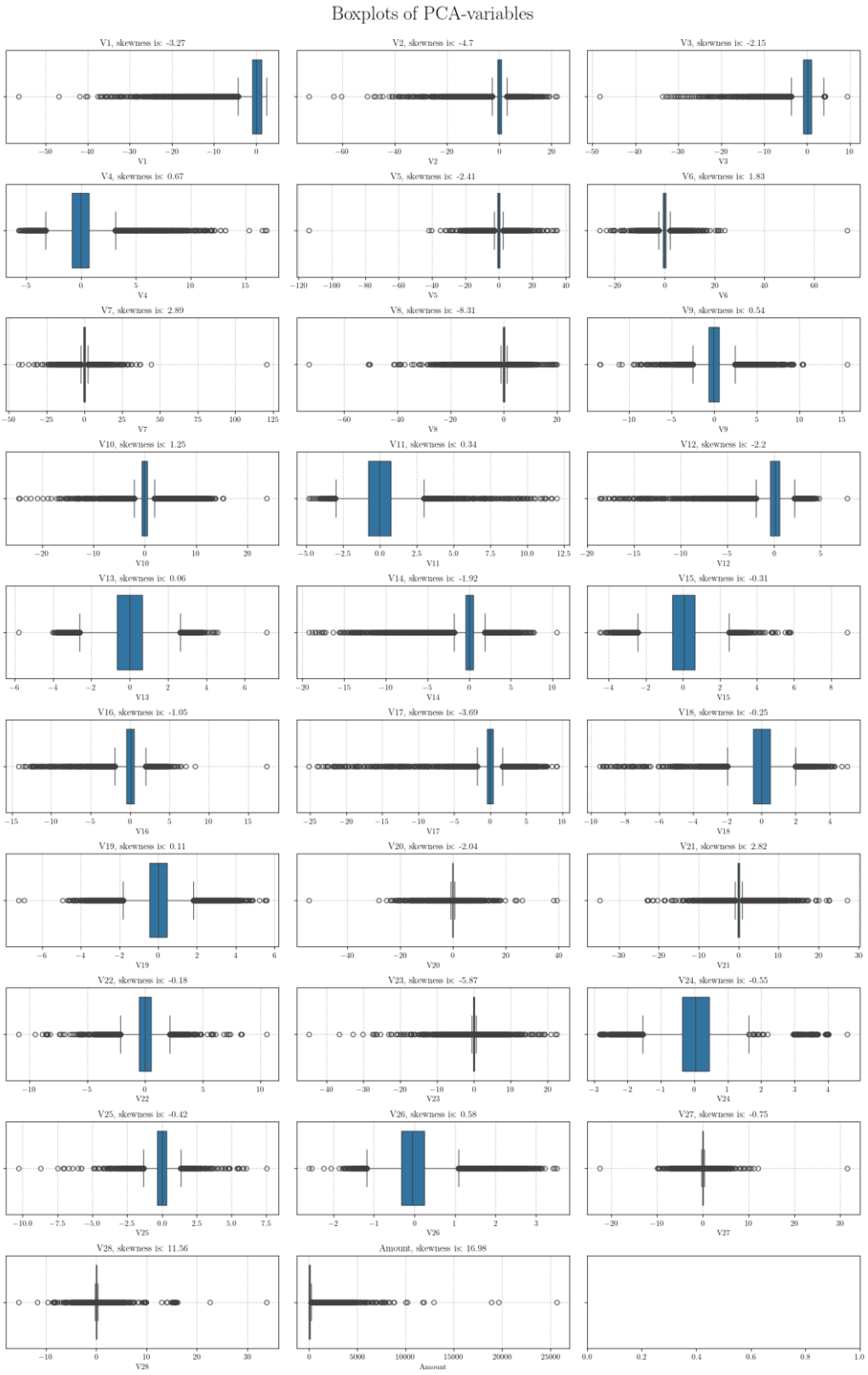

Рисунок 1.14 – Візуалізація розподілів PCA-компонент

У ході проведення обробки даних було виведено твердження щодо недоцільності очищення набору від аномалій. Для проведення експериментів, тренування та тестування моделей машинного навчання варто залишити датасет без змін, оскільки після застосування методу Interquartile Range (IQR) [59] видалено 31685 значення, з яких 466 є шахрайськими транзакціями. Тобто в оновленому датасеті залишаються лише 26 фінансових махінацій, що є критично малою кількістю для ресемплінгу (див. рис. 1.15).

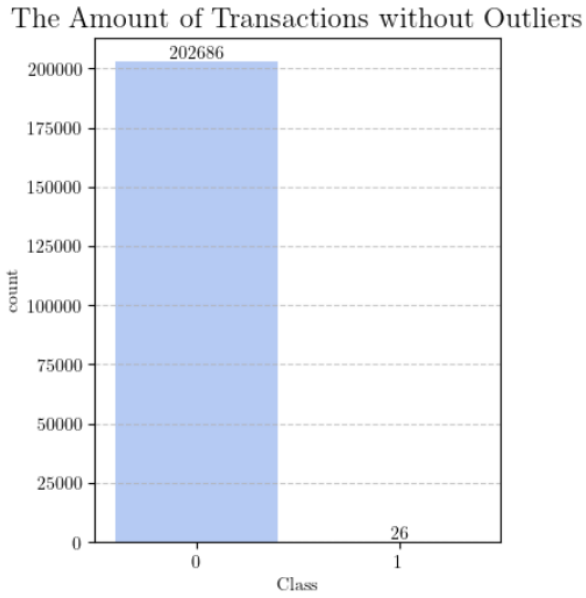

Рисунок 1.15 – Кількість значень набору даних відповідно до класів транзакцій після видалення викидів за допомогою IQR

1.5 Обробка та інженерія ознак

Розглянемо вибір методів очищення та трансформації сирих даних, опис фокус-рішень та інструментів, які застосовуються для підготовки вхідних фіч до навчання моделей.

Видалення дублікатів і базова інспекція даних має стратегічну мету гарантувати цілісність тренувального набору, уникнути витоку інформації та спотворення результатів ресемплінгу й крос-валідації. Навіть поодинокі дублікати (ідентичні транзакції) можуть потрапити одночасно у тренувальний

та валідаційний фолди, що спричинить некоректно завищені метрики через *data leakage* (див. рис. 1.16).

```
Loading dataset …
Initial shape: (284807, 31); Positive class frequency: 0.173%

Duplicate rows: 1081
Shape after dropping duplicates: (283726, 31)

Train size: (226980, 30)  Test size: (56746, 30)
Train fraud rate: 0.167%  Test fraud rate: 0.167%
```

Рисунок 1.16 – Завантаження набору даних, базова інспекція та поділ даних на тренувальні та тестові сети за принципом 80/20

Масштабування суми транзакції (логарифм `log1p` + `StandardScaler`) має зменшити вплив викидів та забезпечити нормалізований розподіл ознаки `Amount`, щоб алгоритми (особливо лінійні та бустинг) працювали стабільніше. Сума транзакції має високий «хвіст» (десятки й сотні тисяч), що викликає екстремальні значення у функції втрат і уповільнює збіжність (див. рис. 1.14). Логарифмічне перетворення `(log(Amount + 1))` стискає динамічний діапазон, зменшуючи вплив «викидів». Стандартизація (центрування $\mu = 0$, $\sigma = 1$) гарантує, що жодна ознака не домінує за масштабом над іншими, що критично в `LogisticRegression`.

За обробка стовпця `Time` збережено часовий контекст транзакцій як базову ознаку, тобто відкладено всі складні часові перетворення на окремий експеримент та відповідний *time-series forecasting*. `Time` (секунди від першої транзакції) може корелювати з поведінковими патернами шахрайства (нічні «сплати», вихідні дні). Додаткові перетворення («година доби», «день тижня») варто впроваджувати після первинного аналізу: спочатку перевірити всі базові характеристики та провести статистичні тести для доведення гіпотез, потім— збагачувати. За допомогою `ColumnTransformer` для комбінованого

препроцесингу централізовано всі трансформації в єдиному об’єкті, уникнено повторення коду, гарантовано правильне застосування трансформерів у складі крос-валідаційного пайплайну (див. рис. 1.17).

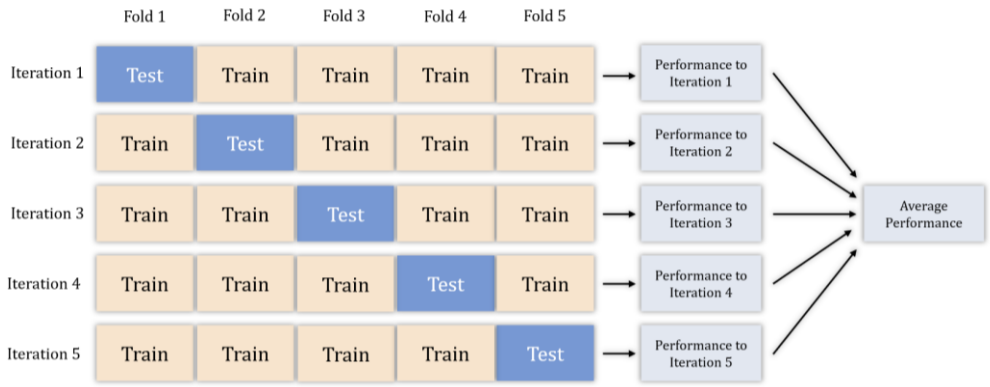

Рисунок 1.17 – Принцип роботи кросс-валідаційного процесу

Варто звернути увагу, що ресемплінг застосовано виключно на тренувальному (валідаційному) наборі даних, оскільки основною задачею є уникнення data leakage та водночас тестування перформансу моделі на базі реального бізнес-кейсу незбалансованих даних, через що реселінг забезпечено на кожній ітерації кросс-валідації за оптимізації та розрахунку AP (*Average Precision*) та мінімізації порогу для F_2-score.

Замість ручного відбору стовпців і застосування трансформерів по черзі, `ColumnTransformer` дозволяє чітко вказати:

— До колонки `Amount` застосувати лог+стандартизацію.

— До колонок `V1…V28` — лише стандартизацію.

— `Time` — пропустити без змін.

— Під час крос-валідації (`GridSearchCV` + `imblearn.Pipeline`) усі ці кроки відбуваються локально в межах *train-fold*, тому виключається витік інформації з *validation*.

Мінімальна конфігурація гарантує легку розширюваність — додавання нових груп ознак або нових трансформерів відбувається в одному місці.

1.6 Висновки до розділу 1

У цьому розділі було представлено систематизований огляд проблематики виявлення шахрайства з кредитними картками, розглянуто характерні типи шахрайських операцій, їх наслідки для фінансових установ та користувачів, а також економічні й репутаційні ризики. Проаналізовано еволюцію підходів до виявлення шахрайства — від експертних правил до сучасних методів машинного навчання, з особливим акцентом на ансамблеві алгоритми як одні з найефективніших рішень у цій сфері.

Особливу увагу приділено проблемі дисбалансу класів, яка є критичною в задачах фінансового шахрайства через вкрай низьку частку позитивних (шахрайських) транзакцій. Розглянуто різні стратегії ресемплінгу, включно з ROS, SMOTE, SMOTE-Tomek та ADASYN, проаналізовано їхні переваги, обмеження та доцільність застосування в контексті складних даних. Зазначено, що вибір методу ресемплінгу має ґрунтуватися на специфіці задачі, структурі даних та обчислювальних обмеженнях.

Крім того, було охарактеризовано відкритий набір даних «*Credit Card Fraud Detection*» з платформи Kaggle, що є репрезентативним джерелом для дослідження даної проблематики. Проведено опис його структури, зокрема анонімізованих компонент V1–V28, отриманих за допомогою PCA.

Нарешті, висвітлено підхід до попередньої обробки та інженерії ознак, а саме: видалення дублікатів, масштабування числових даних, попередню трансформацію часових характеристик та використання `ColumnTransformer` для комплексної підготовки ознак, що створює необхідне підґрунтя для побудови якісної та стійкої моделі виявлення шахрайських транзакцій у наступних етапах дослідження.

2 МЕТОДОЛОГІЯ ПОБУДОВИ ТА ОЦІНЮВАННЯ МОДЕЛЕЙ

У цьому розділі детально викладено системний підхід до побудови моделей машинного навчання для виявлення шахрайства у фінансових транзакціях. Зокрема, здійснено стратифіковане розбиття вибірки з урахуванням балансу класів та сформовано пайплайни, що включають етапи ресемплінгу, попередньої обробки ознак і класифікації.

2.1 Етапи побудови експерименту — machine learning workflow

На основі класичного workflow ML-проєкту (див. рис. 2.1) було розроблено свій власний концепт та робочий фрейм для визначення результатів на кожному з етапів імплементації системи виявлення шахрайства у фінансових транзакціях.

Рисунок 2.1 – Класичний алгоритм роботи ML-проєкту

Під час розробки концепції та проєкту пайплайну було вирішено наступні питання (див. рис. 2.2):

а) Формування навчального і тестового наборів:

1) Стратифіковане розбиття (`StratifiedTrain/Test Split`).

2) Збереження пропорції класів у Train і Test.

б) Створення пайплайнів з ресемплерами та препроцесингом:

1) Опис `imblearn.pipeline.Pipeline` і порядок етапів (*sampler → preprocessor → classifier*).
2) Конфігурація ресемплерів: none (без реселпера), ROS, SMOTE, SMOTE-Tomek, ADASYN.

в) Опис класифікаторів і налаштування гіперпараметрів:

1) Logistic Regression з `class_weight="balanced"`.
2) Random Forest з `class_weight="balanced"`.
3) XGBoost (`tree_method="hist"`, `eval_metric="logloss"`)
4) DummyClassifier як контрольний базовий рівень.

г) Пошук гіперпараметрів через `GridSearchCV`:

1) Використання `StratifiedKFold` (`CV_SPLITS = 5`) – див. рис. 1.17.
2) Оптимізація за метрикою *Average Precision* (AP).
3) Налаштування пошукових грідів для LR, RF, XGB.

д) Метрики оцінювання результатів:

1) Average Precision (PR-AUC) та його значущість у дисбалансі.
2) ROC-AUC для ранжування «score».
3) Precision, Recall, F^2 (F_β з $\beta = 2$) — мотиви вибору F_2 .
4) MCC (Matthews Correlation Coefficient) для збалансованого оцінювання.
5) Accuracy@0.5 як додатковий інформативний, але не ключовий показник.

е) Допоміжні функції оцінювання:

1) `evaluate_on_test` для обчислення усіх метрик та підготовки даних для графіків.

2) `find_best_threshold` для пошуку оптимального порогу F_2 на тесті.

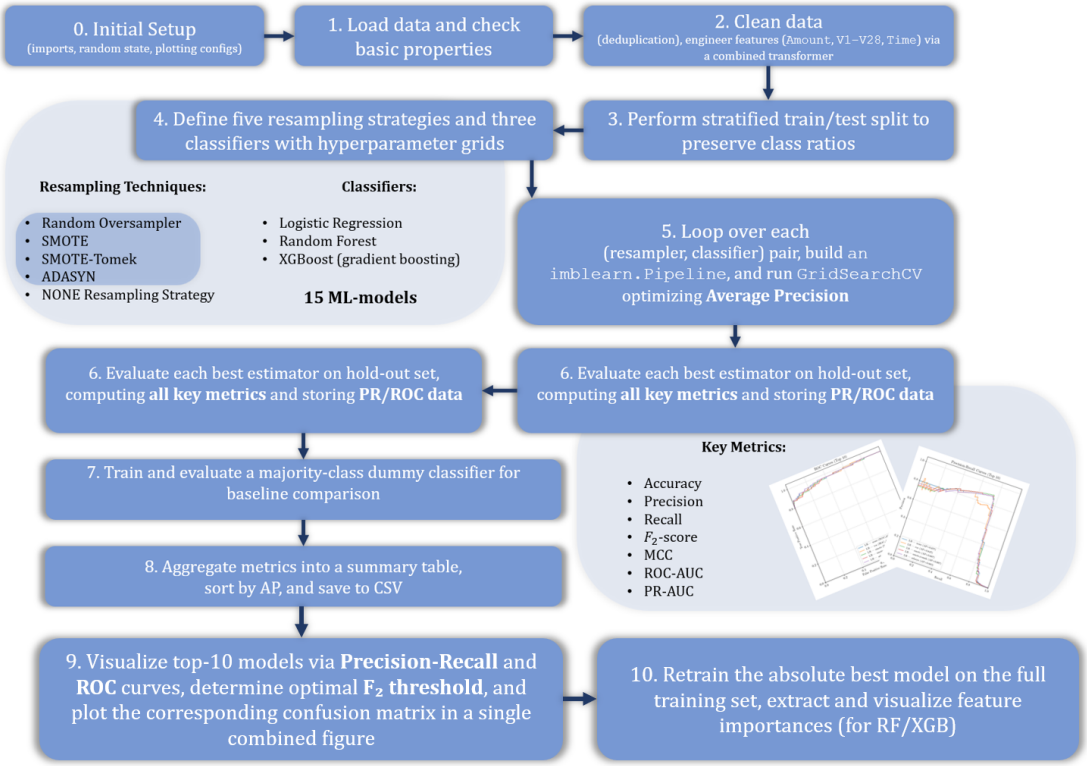

Рисунок 2.2 – Етапи розробки системи fraud-детекцій у фінансових транзакціях

Зауважимо, що, окрім застосування технік балансування даних у стратегії «*sampler → preprocessor → classifier*», було проведено експерименти щодо результативності показників і метрик перформансу моделей машинного навчання без ресемплювання тренувальної вибірки, тобто створено ситуацію для вимушеного *overfitting*, аби проаналізувати ключові ідентифікатори порівняно із моделями, що натреновані коректно — на виході маємо систему, що основана на 15 ML-моделях.

2.2 Опис класифікаторів та налаштування гіперпараметрів

У цьому підрозділі здійснено детальний аналіз трьох ключових моделей машинного навчання, застосовуваних для задачі виявлення шахрайських транзакцій: логістичної регресії, випадкового лісу та XGBoost. Розглянуто математичні основи кожного алгоритму, їх застосування у сфері протидії шахрайству, специфіку роботи з незбалансованими даними через механізми вагування класів, а також особливості налаштування гіперпараметрів.

2.2.1 Логістична регресія (*Logistic Regression*)

Логістична регресія — це лінійна модель для бінарної класифікації, що оцінює ймовірність належності спостереження до класу $y = 1$ за допомогою сигмоїдної функції:

$$P(y = 1 \,|\mathbf{x}) = \sigma(\mathbf{w^T x} + b) = \frac{1}{1 + e^{-(\mathbf{w^T x} + b)}} \tag{2.1}$$

де:

— $\mathbf{x}$ — це вектор ознак,

— $\mathbf{w}, b$ — це параметри моделі,

— $\sigma(z)$ — це сигмоїдна функція.

Функція втрат — логістична (log-loss):

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)] \tag{2.2}$$

де p_i — імовірність, передбачена для прикладу i.

Логістична регресія є базовим класифікатором у задачах виявлення шахрайства, оскільки забезпечує інтерпретовані результати і дозволяє моделювати ймовірність ризикової транзакції. Вона дозволяє налаштовувати поріг для спрацювання (наприклад, $P > 0.7$), що критично важливо у системах реального часу [60].

У scikit-learn параметр `class_weight="balanced"` коригує ваги класів пропорційно до їхньої зворотної частоти в даних:

$$w_j = \frac{N}{k \cdot N_j} \tag{2.3}$$

де:

— N — це загальна кількість прикладів,

— k — це кількість класів,

— N_j — це кількість прикладів класу j.

Ці ваги включаються у функцію втрат, збільшуючи штраф за помилки на міноритарному класі.

Гіперпараметри дозволяють керувати регуляризацією, що є ключовим для уникнення перенавчання, особливо на високовимірних або розріджених даних, відображені у табл 2.1 «Гіперпараметри логістичної регресії (logistic regression) для Grid Search».

Таблиця 2.1 – Гіперпараметри логістичної регресії (*logistic regression*) для `GridSearch`

Параметр	Опис	Значення для коду
`C`	Зворотній коефіцієнт регуляризації (L2)	`[0.01, 0.1, 1, 10, 100]`
`penalty`	Тип регуляризації	`['l2']`
`solver`	Оптимізатор	`['liblinear', 'lbfgs']`
`class_weight`	Балансування ваг класів	`['balanced']`

2.2.2 Випадковий ліс та ансамблеве навчання (*Random Forest*)

Random Forest — ансамбль M lдерев рішень, кожне з яких навчається на бутстреп-вибірці:

— Для кожного дерева вибирається підмножина даних $D_m \subset D$ із заміною.

— На кожному вузлі дерева випадково вибирається підмножина ознак $F' \subset F$ для побудови розбиття.

Кінцеве рішення:

$$\hat{y} = \text{mode}(h_1(\mathbf{x}), h_2(\mathbf{x}), \dots, h_M(\mathbf{x})) \quad (2.4)$$

де h_m — передбачення m-ого дерева.

Random Forest демонструє високу точність у задачах виявлення шахрайства, ефективно моделюючи складні, нелінійні закономірності та взаємодії між ознаками [61]. Завдяки стійкості до шуму та викидів, він підходить для реальних транзакційних даних.

Параметр `class_weight="balanced"` змінює ваги при розрахунку імпюрності вузлів (наприклад, ентропії або критерію Джині) [62]:

$$\text{Gini}_\text{w} = \sum_{j=1}^{k} w_j p_j (1 - p_j) \quad (2.5)$$

`balanced_subsample` застосовує ваги класів до кожної бутстреп-вибірки окремо [63], підвищуючи точність міноритарного класу.

Відповідно до [63] маємо наступні гіперпараметри для Grid Search для методу Random Forest, що дозволяють краще обробляти клас меншини (див. табл. 2.2 «Гіперпараметри методу випадкових лісів (random forest) для Grid Search»)

Таблиця 2.2 – Гіперпараметри методу випадкових лісів (*random forest*) для `GridSearch`

Параметр	Опис	Значення для коду
`n_estimators`	Кількість дерев у лісі	`[100, 200, 300]`
`max_depth`	Максимальна глибина дерева	`[None, 10, 20, 30]`
`min_samples_split`	Мінімальна кількість зразків для поділу вузла	`[2, 5, 10]`

Продовження таблиці 2.2

`class_weight`	Ваги класів для компенсації дисбалансу	`['balanced', 'balanced_subsample']`
`min_samples_leaf`	Мінімальна кількість зразків у листі	`[1, 2, 4]`
`max_features`	Кількість ознак для вибору при поділі	`['sqrt', 'log2']`

2.2.3 Градієнтний бустинг (XGBoost — eXtreme Gradient Boosting)

XGBoost реалізує градієнтний бустинг дерев, де кожна нова модель f_t навчається для мінімізації похідної функції втрат:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} \ell\left(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)\right) + \Omega(f_t) \tag{2.6}$$

де:

— ℓ — це функція втрат (наприклад, logloss),

— $\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum w_j^2$ — регуляризаційний терм із L1/L2 штрафами,

— T — це кількість листів у дереві.

Параметр `scale_pos_weight` — це математично обґрунтований механізм балансування функції втрат у XGBoost. Його використання дозволяє значно покращити продуктивність моделі у випадках, коли важливо виявляти рідкісні, але критично важливі випадки, як-от фінансове шахрайство. Збалансована логістична функція втрат для бінарної класифікації із врахуванням ваги позитивного класу має вигляд:

$$\mathcal{L}_{balanced} = -\sum_{i=1}^{n} [\text{sw} \cdot y_i \log(\widehat{p_i}) + (1 - y_i) \log(1 - \widehat{p_i})] \tag{2.7}$$

де:

— $y_i \in \{0, 1\}$ — це істинна мітка класу для зразка i,

— $\widehat{p_i} \in (0, 1)$ — це передбачена ймовірність позитивного класу,

— sw — це ваговий коефіцієнт для позитивного класу (`scale_pos_weight`),

— n — це загальна кількість прикладів у навчальній вибірці.

`tree_method="hist"` — це побудова дерева за гістограмами, що знижує обчислювальні витрати, а `eval_metric="logloss"` — це логарифмічна функція втрат, що надає більш точну оцінку ймовірнісної класифікації [64].

XGBoost вирізняється здатністю працювати з великими обсягами та високовимірними даними, ефективно виявляючи складні шаблони шахрайства навіть за сильного дисбалансу. Паралелізація та автоматична обробка пропущених значень роблять його придатним для масштабованих систем реального часу.

XGBoost дуже чутливий до налаштувань параметрів, зокрема `learning_rate`, `scale_pos_weight` та `max_depth`, що визначають глибину та швидкість навчання, особливо у задачах із сильним дисбалансом [64, 65] (див. табл. 2.3 «Гіперпараметри градієнтного бустингу (XGBoost) для Grid Search»).

Таблиця 2.3 – Гіперпараметри градієнтного бустингу (*XGBoost*) для `Grid Search`

Параметр	Опис	Значення для коду
`n_estimators`	Кількість дерев у бустингу	`[100, 200, 300]`
`max_depth`	Максимальна глибина дерева	`[3, 5, 7]`
`Learning_rate`	Коефіцієнт навчання (μ)	`[0.01, 0.1, 0.3]`

Продовження таблиці 2.3

`subsample`	Частка навчальної вибірки для кожного дерева	`[0.5, 0.7, 1.0]`
`colsample_bytree`	Частка ознак для використання при побудові дерева	`[0.5, 0.7, 1.0]`
`scale_pos_weight`	Ваговий коефіцієнт для класу шахрайства	`[1, 10, 25, 50]`
`tree_method`	Алгоритм побудови дерева	`['hist']`
`eval_metric`	Метрика оцінки якості моделі	`['logloss']`

2.3 Метрики оцінювання результатів

У цій частині наведено деталізований опис кожної із метрик у контексті задачі класифікації (зокрема — детекції шахрайства), а також пояснення, що таке Precision-Recall та ROC-AUC (*Area Under Receiver Operating Characteristic Curve*) криві. У ході добору метрик оцінки результативності ML-моделей було вирішено розширити стандартний фреймворк підбору перформанс-характеристик для незбалансованої бінарної класифікації [66] (див. рис. 2.3).

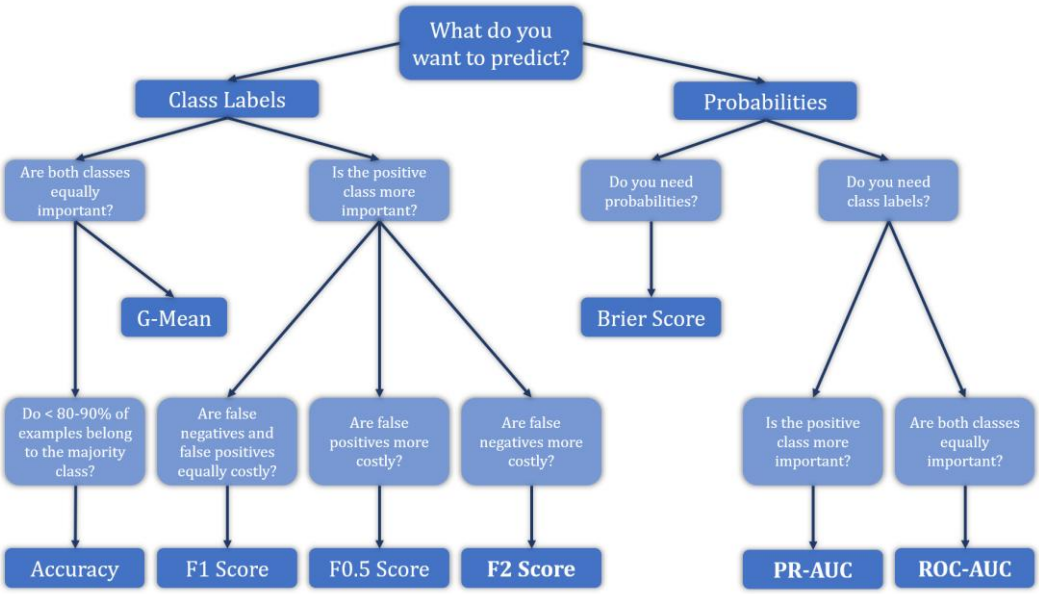

Рисунок 2.3 – Паттерн добору метрик для незбалансованої класифікації

Крім того, описано проблематику мінімізації порогу (*threshold*) у контексті класифікації — зниження граничного значення ймовірності (score), при якому модель починає позначати спостереження як «позитивний» (у нашому випадку – «шахрайська транзакція»).

2.3.1 Precision (точність)

$Precision$ (точність) — це відношення числа правильно передбачених позитивних прикладів (True Positives, TP) до загальної кількості прикладів, класифікованих моделлю як позитивні (тобто TP + FP, де FP — False Positives):

$$Precision = \frac{TP}{TP + FP} \tag{2.8}$$

$Precision$ показує, наскільки «чистими» є ті детекції, які модель позначила як «шахрайські» (див. рис. 2.4).

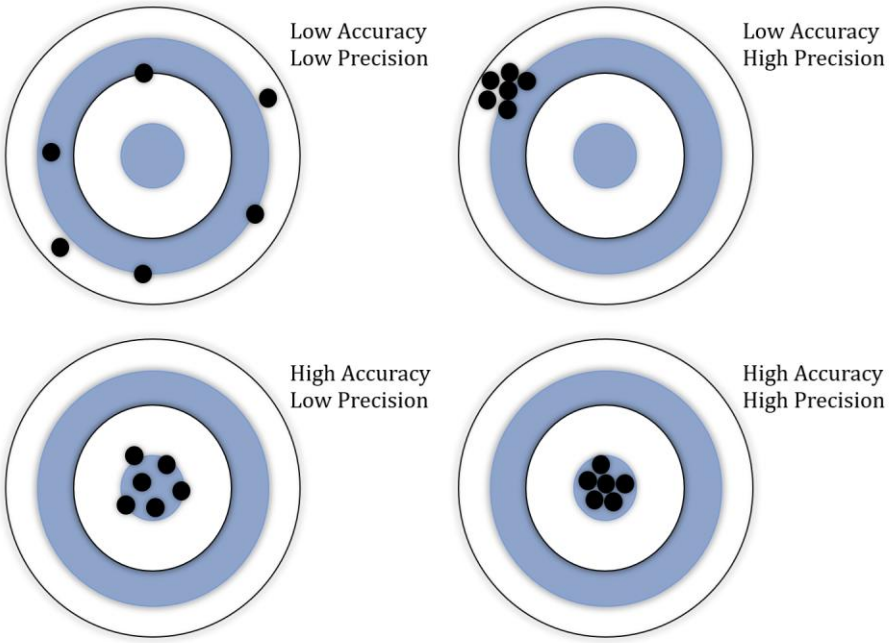

Рисунок 2.4 – Ілюстративна інтерпретація різниці між метриками $Accuracy$ (у контексті дисбалансу не є пріоритетним показником якості моделі) та $Precision$

Якщо $Precision = 0.90$, це означає: із 100 транзакцій, які модель позначила як шахрайські, 90 дійсно були шахрайськими, а 10 — помилково.

Висока точність важлива в ситуаціях, коли не можна дозволити надмірну кількість «хибних підозр» (*False Positives*). Наприклад, якщо кожен False Positive призводить до блокування легітимної транзакції або вимагає ручної перевірки, дуже бажано тримати $Precision$ максимально високим.

2.3.2 Recall (чутливість, повнота)

$Recall$ (чутливість або повнота, англ. *sensitivity*) — це відношення числа правильно передбачених позитивних прикладів (TP) до загальної кількості реальних позитивних (TP + FN, де FN — False Negatives):

$$Recall = \frac{TP}{TP + FN} \tag{2.9}$$

$Recall$ показує, яку частину з усіх фактично шахрайських транзакцій модель змогла знайти (див. рис. 2.5).

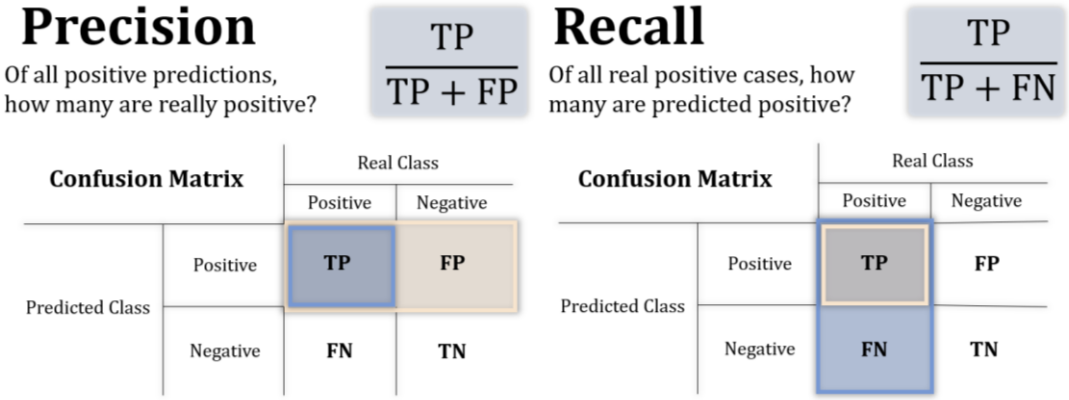

Рисунок 2.5 – Зіставлення метрик $Precision$ та $Recall$ для візуального пояснення різниці інтерпретації розрахунків

Якщо $Recall = 0.80$, це означає: з 100 реальних шахрайських транзакцій модель виявила 80, але пропустила 20.

Висока чутливість важлива в ситуаціях, коли критично важливо не пропустити якомога більше кейсів шахрайства, навіть якщо це призведе до збільшення кількості хибнопозитивних (FP). Наприклад, коли пропущена шахрайська транзакція може завдати великої фінансової шкоди, варто жертвувати точністю ($Precision$) на користь $Recall$.

2.3.3 F_2-score (F-міра з $\beta = 2$)

$\boldsymbol{F_2 - score}$ — це узагальнена F-міра, яка об'єднує $Precision$ і $Recall$ у єдину метрику, але з ваговим акцентом на $Recall$ ($\beta = 2$). Узагальнена F-міра визначається як:

$$F_\beta - score = (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall} \qquad (2.10)$$

де β^2 визначає, наскільки важливішим є $Recall$ порівняно з $Precision$.

Для $\beta = 2$ формула має наступний вигляд:

$$F_2 - score = 5 \cdot \frac{Precision \cdot Recall}{(4 \cdot Precision) + Recall} \quad (2.11)$$

Оскільки $\beta = 2$, у цій метриці $Recall$ має у чотири рази більшу вагу, ніж $Precision$. Тобто ми «штрафуємо» модель сильніше за пропущені випадки (False Negatives), ніж за хибні детекції (False Positives).

$F_2 - score \approx 1$ свідчить про те, що точність, і чутливість високі, проте досягнення високого F_2 вимагає особливої уваги до $Recall$.

Зазначена метрика використовується, коли пропущена шахрайська транзакція (FN) несе значно більші збитки, ніж зайва перевірка легітимної (FP).

2.3.4 Коефіцієнт кореляції Метьюза (*Matthews Correlation Coefficient*)

MCC (Коефіцієнт кореляції Метьюза) — збалансована метрика для двокласової класифікації, яка враховує всі чотири елементи матриці невідповідностей (TP, TN, FP, FN):

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)}(TP + FN)(TN + FP)(TN + FN)} \quad (2.12)$$

MCC дає значення в інтервалі від −1 до +1:

- $MCC = +1$ означає ідеальну класифікацію (всі TP, TN правильно, немає FP, FN).
- $MCC = 0$ означає, що класифікація не краща, ніж випадковий вибір.
- $MCC = -1$ означає повністю протилежні передбачення (усі класи переплутані).

MCC особливо корисний у задачах з дисбалансом класів, бо навіть коли dataset містить багато TN, він «штрафує» модель за високі FP і FN.

На відміну від простих $Precision/Recall$, MCC дає єдине число, що враховує як помилки одного класу, так і другого.

$MCC \approx 0.85$ означає дуже високу узгодженість із реальними мітками, незважаючи на дисбаланс (наприклад, 99.8 % TN та 0.2 % TP).

Якщо модель лише відкидає всі транзакції як легітимні ($Recall$ шахрайського класу $= 0$, $Precision$ шахрайського класу $= 0$), то MCC буде ≈ 0 (або навіть від'ємним, якщо є певні помилки у протилежний бік).

2.3.5 PR-curve (*Precision-Recall крива*)

Precision-Recall крива — це графік, на якому по вертикалі відкладається $Precision$, а по горизонталі — $Recall$, обчислювані для різних значень порогу (*threshold*). Замість FPR у вісь абсцисі модель показує Recall, а у вісь ординаті

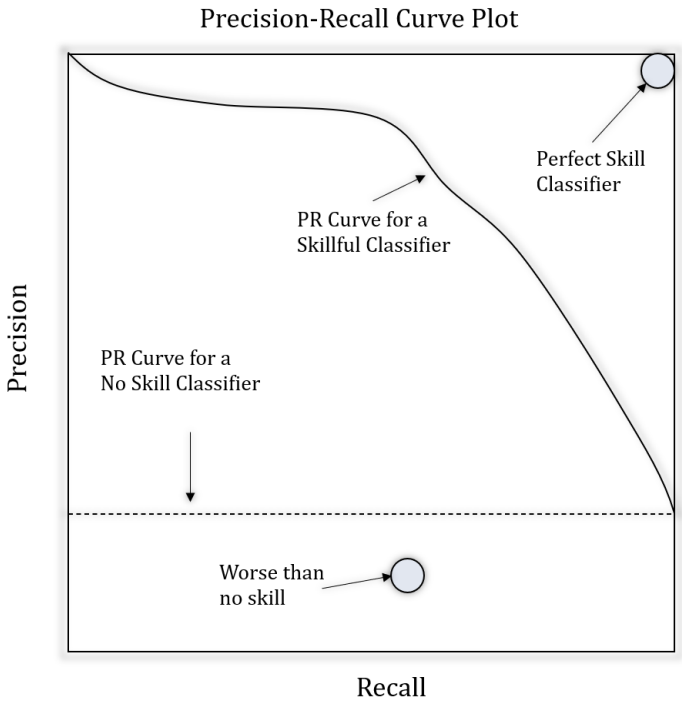

Рисунок 2.6 – Precision-Recall Curve

PR-curve демонструє, як змінюється *trade-off* між точністю та чутливістю залежно від порогу.

Якщо модель утримує високі значення $Precision$ навіть за високих $Recall$ (або навпаки), то вона якісніше відділяє позитивні приклади від негативних саме у класах-екстремумах (корисно в розріджених даних).

Площа під PR-кривою (**Average Precision, AP**) є сумарною метрикою, близькою до інтегралу $precision(recall)$. *У задачах із сильним дисбалансом AP вважається кращою характеристикою, ніж ROC-AUC, тому що вона фокусується на якості виявлення меншості.*

У разі надзвичайно дисбалансованих класів (наприклад, $\approx 0.17\,\%$ шахрайства) PR-curve дозволяє оцінити, наскільки добре модель буде працювати у зоні, де $Recall$ невеликий, але $Precision$ має бути високим.

2.3.6 ROC-AUC (*Area Under Receiver Operating Characteristic Curve*)

ROC-крива (*Receiver Operating Characteristic*) — це графік, який показує залежність True Positive Rate (TPR = Recall) від False Positive Rate (FPR = FP / (FP + TN)) при різних порогах класифікації. ROC-AUC — це площа під цією кривою (Area Under Curve). Діапазон значень: від 0.0 до 1.0 (див. рис. 2.7)

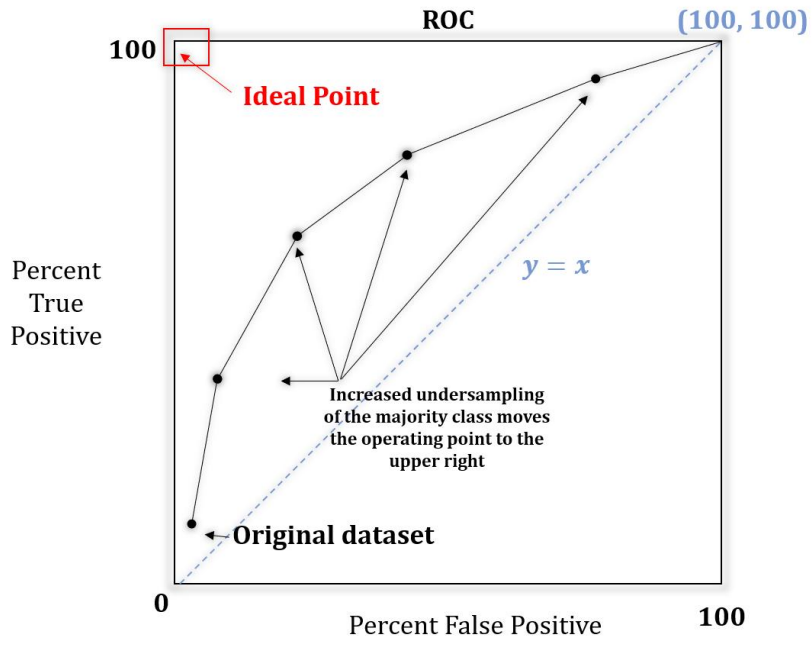

Рисунок 2.7 – Area Under Receiver Operating Characteristic Curve

Формула обчислення AUC укладається у інтегральне/сукупне підсумовування площ трапецій, які утворюють крива ROC:

- ROC-AUC показує здатність моделі розділяти приклади двох класів без прив'язки до конкретного порогу.
- Якщо $ROC - AUC = 0.5 \rightarrow$ модель нічим не краща, ніж випадковий вибір.
- Якщо $ROC - AUC = 1.0 \rightarrow$ ідеальна дискримінація (модель для кожного порогу чітко відрізняє позитивні від негативних).
- Чим ближче значення до 1.0, тим краще модель «ранжує» позитивні приклади вище негативних.

ROC-AUC крива особливо корисна, коли необхідно оцінити загальну якість частоти хибнопозитивних / хибнонегативних за весь діапазон порогів — здатність моделі апріорі розрізняти класи.

Якщо у середовищі з високою дисбалансованістю важко одночасно тримати і $high\ precision$, і $high\ recall$ на одному фіксованому порозі — ROC-AUC допомагає обрати оптимальний поріг.

2.3.7 Мінімізація порогу (*threshold*)

Більшість алгоритмів (*Logistic Regression, Random Forest, XGBoost тощо*) при передбаченні повертають не одразу мітку «0/1», а скоріше умовну ймовірність (score або p-value), що даний приклад належить до позитивного класу. Наприклад, після обробки транзакції в XGBoost можна отримати $y_{proba} = 0.37$.

Стандартно (як налаштовано за замовчуванням) беруть поріг $t = 0.5$:

- Якщо $y_{proba} \geq 0.5$, модель видає «1» (класифікує як позитивний клас, тобто «шахрайство»).
- Якщо $y_{proba} < 0.5$, модель видає «0» (класифікує як негативний — «легітимна транзакція»).

Поріг 0.5 – лише умовність. Насправді, у задачах із сильним дисбалансом класів (наприклад, 0.17 % шахрайських транзакцій) цей «класичний» 0.5 не завжди дає потрібний баланс між $Precision$ та $Recall$. За замовчуванням можна або пропускати дуже багато реального шахрайства (*якщо модель віддає скори* < 0.5), або навпаки – створювати занадто багато хибнопозитивних виявлень (*якщо модель загалом прогнозує низькі ймовірності навіть для справжніх шахрайств*).

Причини зміни (мінімізації) порогу у задачі *fraud detection*:

а) Контекст бізнес-вимог:

1) Якщо пропустити реальне шахрайство, коштує дорожче, ніж розблокувати трохи легітимних транзакцій, то доцільно знизити поріг так, щоб $Recall$ був достатньо високим (наприклад, 0.85 чи 0.90). Навіть якщо $Precision$ «просідає» до, наприклад, 0.70, зате маємо ≥ 85–90 % усіх шахрайських транзакцій.
2) Якщо, навпаки, кожен false positive (помилково заблокована) транзакція — це велика незручність (імідж банка) чи додаткові витрати на допоміжні перевірки, тоді поріг варто піідйняти (наприклад, з 0.5 до 0.7 чи 0.8), щоб $Precision$ залишався дуже високим (> 0.95), навіть якщо $Recall$ опуститься до 0.60–0.70.

б) Автоматизація VS ручний ітеративний перегляд:

1) У компанії, де є професійна служба аналітиків *manual review*, можна свідомо знизити поріг (наприклад, до 0.3), щоб спершу охопити якомога більше потенційно шахрайських операцій, навіть якщо серед них є 30–40 % FP: це просто «додаткове навантаження» на аналітика.
2) Якщо ж необхідно мінімізувати ручну роботу, поріг слід залишити або підійняти (наприклад, 0.5 → 0.6), щоб лише ті транзакції з дуже високою ймовірністю позначалися як шахрайські.

Зауважимо, що за розробки системи виявлення шахрайства у фінансових транзакціях, пошук оптимального порогу було підключено до *pipeline*. За допомогою Precision-Recall чи ROC-кривих для кожної моделі можна знайти поріг, який максимізує певну метрику (наприклад, F_2, тобто *priority на recall*).

2.4 Висновки до розділу 2

У цьому розділі було детально розглянуто методологічні аспекти побудови, налаштування та оцінювання моделей машинного навчання для задачі виявлення шахрайських транзакцій за умов екстремального дисбалансу класів. Стратифіковане розбиття вибірки забезпечило збереження реального розподілу класів у тренувальному та тестовому підмножинах, що є критично важливим для достовірної оцінки ефективності моделей.

Створено узгоджені ML-пайплайни із застосуванням `imblearn.pipeline.Pipeline`, що включають поетапне виконання ресемплінгу, попередньої обробки даних та класифікації. Реалізовано декілька варіантів ресемплінгу (опис див. у розділі 1) для покращення здатності моделей виявляти транзакції «міноритарного» класу.

У якості моделей було обрано три базові, добре інтерпретовані й промислово вживані алгоритми — *Logistic Regression, Random Forest ma XGBoost* — з урахуванням механізмів компенсації дисбалансу (`class_weight="balanced"`, спеціалізовані метрики втрат). *DummyClassifier* використано як контрольну базову модель для оцінки ефективності.

Підбір гіперпараметрів здійснено за допомогою `GridSearchCV` із використанням стратифікованої п’ятикратної крос-валідації та оптимізацією за метрикою Average Precision (AP) — найбільш придатною у випадках з розрідженим позитивним класом. Налаштовано окремі пошукові сітки для кожної моделі з урахуванням їхньої специфіки.

Для повного та зваженого оцінювання моделей використано набір інформативних метрик: $PR-AUC, ROC-AUC, Precision, Recall, F_2-score$ ($\beta = 2$), а також MCC, що дозволяє враховувати як позитивні, так і негативні передбачення. $Accuracy$ розглядався як допоміжний показник. Допоміжні функції `evaluate_on_test` і `find_best_threshold` були розроблені для автоматизованого аналізу моделей на тестовому наборі, включно з пошуком оптимального порогу класифікації за критерієм F_2.

Описана методологія створює цілісну й адаптивну основу для побудови та тестування моделей у складних умовах дисбалансованих фінансових даних та слугує основою для проведення подальших експериментів, наведених у наступному розділі.

3 ЕКСПЕРИМЕНТИ, РЕЗУЛЬТАТИ ТА БІЗНЕС-РЕКОМЕНДАЦІЇ

Цей розділ присвячено емпіричній частині дослідження, у якій проведено порівняльний аналіз комбінацій класичних моделей машинного навчання та методів балансування вибірки для задачі виявлення шахрайських фінансових транзакцій. Представлені результати отримані шляхом систематичних експериментів із використанням крос-валідації, детального аналізу метрик, графічної інтерпретації моделей і візуалізації ваг ознак.

3.1 Результати крос-валідації та порівняння моделей

У підрозділі зведено всі конфігурації типу «модель + ресемплер», результати яких занесено до таблиці `results_summary.csv`. Основним критерієм відбору є метрика **Average Precision (AP)**. Відібрано топ-10 найкращих комбінацій, для яких проведено розширене порівняння за метриками: ROC-AUC, Precision, Recall, F_2-score, MCC, Accuracy (усі при порозі 0.5). Такий підхід дозволив виявити моделі, що демонструють найкращий баланс між виявленням шахрайства та мінімізацією помилкових детекцій.

Здійснено побудову Precision-Recall і ROC-кривих для кожної з відібраних моделей у топ-10. Графіки містять підписи з AP та ROC-AUC значеннями, що дозволяє зручно інтерпретувати продуктивність систем у ранжуванні транзакцій. Для найкращої моделі також згенеровано Confusion Matrix із використанням порогу, що максимізує F_2-score. Інтерпретація графіків подана з точки зору прикладного бізнес-контексту — зокрема, важливості мінімізації пропущених шахрайських транзакцій при збереженні прийнятного рівня false positives.

Для моделей Random Forest та XGBoost виконано аналіз важливості ознак (*feature importance*). Побудовано таблиці топ-10 фіч за величиною

впливу на рішення моделей та горизонтальні діаграми (*barh*), які відображають розподіл ваг.

3.1.1 Логістична регресія (*Logistic Regression*) «ресемплер + модель»

На рис. 3.1 продемонстровано результати найкращого показника основної метрики оптимізації Average Precision (AP) під час процесу крос-валідації за тренування моделі логістичної регресії за кожною із обраних технік балансування даних.

```
=== LR | none ===
Best CV AP = 0.7538

=== LR | ros ===
Best CV AP = 0.7299

=== LR | smote ===
Best CV AP = 0.7591

=== LR | smote_tomek ===
Best CV AP = 0.7581

=== LR | adasyn ===
Best CV AP = 0.7584
```

Рисунок 3.1 – Average Precision за процесу крос-валідації тренування логістичної регресії

На рис. 3.2 наведено детальний аналіз перформансу п’яти комбінацій «Logistic Regression + різні ресемплери» з точки зору ключових метрик ($Accuracy@0.5, AP, Precision@0.5, Recall@0.5, F_2@0.5, MCC@0.5, ROC-AUC$) та практичні бізнес-рекомендації, як обрати оптимальну конфігурацію залежно від вимог фінансової інфраструктури.

	sampler	model	Accuracy@0.5	AP	Precision@0.5	Recall@0.5	F2@0.5	MCC@0.5	ROC_AUC
0	none	lr	0.974254	0.676821	0.054178	0.873684	0.217050	0.213925	0.961667
1	smote_tomek	lr	0.988598	0.667242	0.114525	0.863158	0.374088	0.312048	0.957267
2	smote	lr	0.987823	0.666996	0.105820	0.842105	0.352113	0.296018	0.957650
3	adasyn	lr	0.988845	0.663068	0.116809	0.863158	0.378928	0.315196	0.963910
4	ros	lr	0.974853	0.646878	0.055407	0.873684	0.220980	0.216426	0.961874

Рисунок 3.2 – Детальний аналіз перформансу п’яти комбінацій «Logistic Regression + різні ресемплери»

Маємо наступні стратегії аргументації прийняття рішень щодо впровадження кожної з моделей:

а) Без ресемплінгу (none) & ROS:

1) AP і ROC-AUC найвищі → модель найкраще ранжує транзакції за ймовірністю шахрайства.

2) $Precision \approx 5\ \%$ — **майже всі позначені «шахрайські» виявляються false positives!**

3) $Recall \approx 87\ \%$ — багато шахрайських транзакцій виявляються, але ціною величезної кількості хибних спрацьовувань.

4) Бізнес-контекст:

— Не підходить, якщо кожне хибне блокування вимагає додаткової операційної витрати (додаткові дзвінки клієнтам, зайве ручне розслідування).

— Категорично не рекомендовано інтегрувати моделі, що навчені на НЕ збалансованих даних, оскільки наслідком такого рішення є ідентифікація кожної із транзакцій на користь мажоритарного класу, оскільки дані мали 99% легітимних транзакцій.

б) SMOTE:

1) $Precision \approx 10.6\ \%$, $Recall \approx 84.2\ \%$, $F_2 \approx 0.352$.
2) Подвоєння точності позначень шахрайства при незначному зниженні $recall$.
3) Бізнес-контекст:
   — Підійде, коли важливо зменшити навантаження на ручну перевірку (половина спрацювань уже FP замість 95 %).
   — Водночас захоплює понад 80 % реального шахрайства.

в) SMOTE-Tomek:

1) $Precision \approx 11.5\ \%, Recall \approx 86.3\ \%, F_2 \approx 0.374$.
2) Комбінація синтетики та видалення «шуму» забезпечує кращий баланс: відносно високий $precision$ і збереження $recall$.
3) Бізнес-контекст:
   — Оптимальний вибір, коли потрібно стримати фальшиві спрацювання (~ 1 з 9 позначених транзакцій реально шахрайська) і водночас не втратити багато справжніх шахрайств.
   — Підходить для середніх фінансових інфраструктур із помірними ресурсами на ручну розбірку.

г) ADASYN:

1) $Precision \approx 11.7\ \%, Recall \approx 86.3\ \%, F_2 \approx 0.379$ (найвищий F_2 і MCC ≈ 0.315).
2) Адаптивна генерація синтетичних зразків фокусується на «складних» регіонах простору ознак.
3) Бізнес-контекст:
   — Рекомендовано, якщо головним пріоритетом є максимальна чутливість ($recall$) із достатньо високою точністю (~12 %).

— Ідеально для великих банків або платіжних агрегаторів, де пропустити шахрайство (FN) набагато дорожче, ніж обробити додаткові FP.

На рис. 3.3 показано перформанс ML-моделей за PR-curve, ROC-AUC curve та Confusion Matrix для найкращої моделі за AP та розрахованим оптимальним порогом.

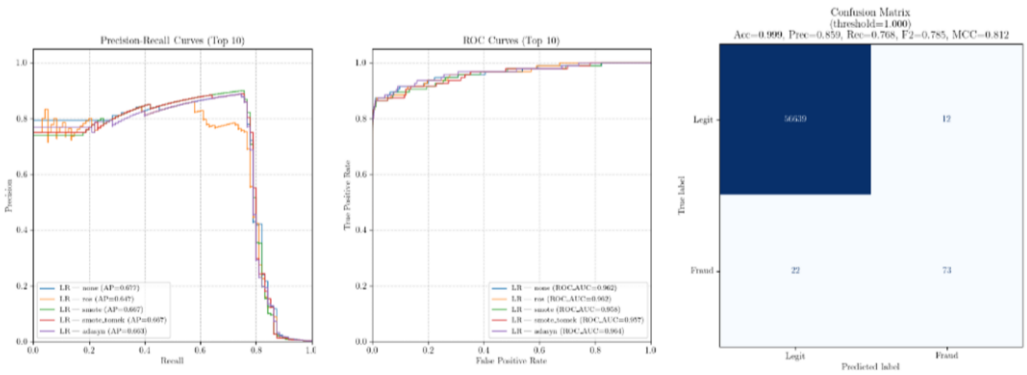

Рисунок 3.3 – PR-curve, ROC-AUC curve та Confusion Matrix для найкращої моделі за AP та розрахованим оптимальним порогом за класифікації логістичної регресії

3.1.2 Випадковий ліс (*Random Forest*) «ресемплер + модель»

На рис. 3.4 системою занотовано найкращий результат показника-оптимізатора AP відповідно до певного фолду крос-валідації за тренування класифікатора методу випадкового лісу (random forest).

За таблицею зведених даних за метриками $Accuracy@0.5, AP, Precision@0.5, Recall@0.5, F_2@0.5, MCC@0.5, ROC-AUC$, де 0.5 — значення порогу, за кожною моделлю на рис. 3.5 проаналізовано доцільність впровадження та інтегрування цього компоненту системи до фінансової IT-структури.

```
=== RF | none ===
Best CV AP = 0.8387

=== RF | ros ===
Best CV AP = 0.8444

=== RF | smote ===
Best CV AP = 0.8403

=== RF | smote_tomek ===
Best CV AP = 0.8435

=== RF | adasyn ===
Best CV AP = 0.8435
```

Рисунок 3.4 – Average Precision за процесу крос-валідації тренування випадкового лісу (ансамблеве навчання)

	sampler	model	Accuracy@0.5	AP	Precision@0.5	Recall@0.5	F2@0.5	MCC@0.5	ROC_AUC
0	ros	rf	0.999471	0.821335	0.922078	0.747368	0.776805	0.829890	0.953479
1	smote	rf	0.999454	0.815490	0.872093	0.789474	0.804721	0.829486	0.968567
2	none	rf	0.999471	0.807976	0.957746	0.715789	0.753880	0.827740	0.922315
3	smote_tomek	rf	0.999471	0.807850	0.882353	0.789474	0.806452	0.834362	0.960627
4	adasyn	rf	0.999454	0.802950	0.890244	0.768421	0.790043	0.826827	0.968516

Рисунок 3.5 – Детальний аналіз перформансу п’яти комбінацій «Random Forest + різні ресемплери»

Відповідно до табл. 3.1 «Бізнес-контекст інтеграції Random Forest моделей до фінансової IT-системи» виведено бізнес-юз-кейс та рекомендації щодо кожної комбінації.

Зазначимо, що до рекомендаційного експерт-аналізу не включено моделі без ресемплера, оскільки такий класифікатор даватиме хибнонегативні результати, тобто ідентифікуватиме легітимні транзакції, як фінансове правопорушення.

Таблиця 3.1 – Бізнес-контекст інтеграції Random Forest моделей до фінансової IT-системи

Sampler	Business Use-Case	Рекомендація
ROS	Оптимальний AP (ранжування)Висока здатність правильно розставляти пріоритети розслідування ($AP \approx 0.82$).	Рекомендовано як основну модель у банках середнього рівня: мінімізує помилки ранжування, забезпечуючи 92 % $precision$ при 75 % $recall$, AP найвища серед усіх.
SMOTE	Максимальний Recall. Переважна мета — ловити якомога більше шахрайства ($Recall \approx 79$ %)	Ідеальний варіант для anti-fraud команд, де важливо зловити майже 80 % шахрайства, а ~13 % FP ($Precision \approx 87$ %) можна обробити вручну.
SMOTE-Tomek	Найкращий F_2 та MCC. Баланс між захопленням шахрайства й мінімізацією FP.	Рекомендовано як «головну» стратегію: $F_2 = 0.8065$ (найвище), $MCC = 0.8344$ → гармонійний компроміс $Precision \approx 88$ % і $Recall \approx 79$ %, оптимально для банків з помірним ризиком.
ADASYN	Адаптивний фокус на складних випадках. Підвищення чутливості в «складних» регіонах.	Використовувати для детальнішої перевірки транзакцій із високим ризиком за допомогою ADASYN, особливо, коли важливо виявити нетривіальні сусідства шахрайських прикладів.

На рис. 3.6 зображено інтерпретацію отриманих графіків та їх значення для бізнесу за допомогою візуалізацій PR-curve, ROC-AUC curve та Confusion Matrix для найкращого перформансу із оптимальним порогом (тут: $t = 0.245$).

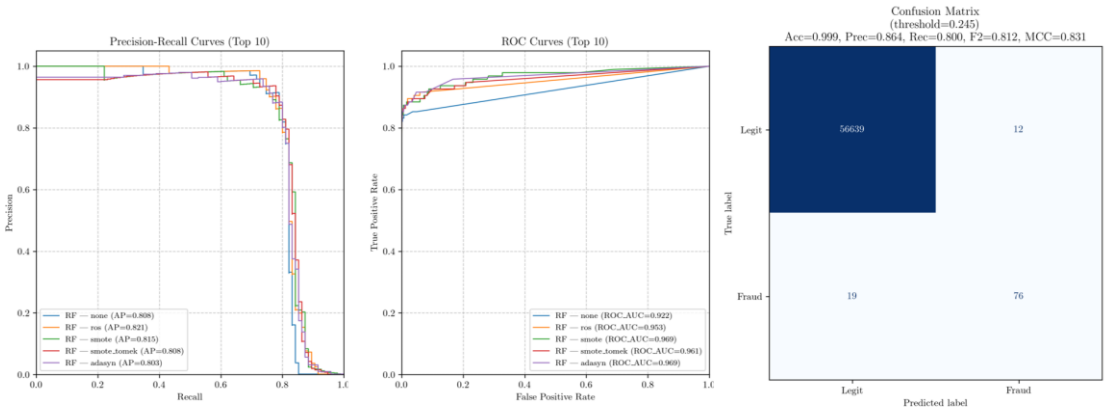

Рисунок 3.6 – PR-curve, ROC-AUC curve та Confusion Matrix для найкращої моделі за AP та розрахованим оптимальним порогом за класифікації випадковим лісом

Для подальших досліджень з точки зору дослідницького аналізу даних слід спиратися на *feature importance* обраного класифікатора для деталізації системи та додаткового моделювання на основі тих ознак, які найбільше впливають на детекцію класу меншини (див. рис. 3.7).

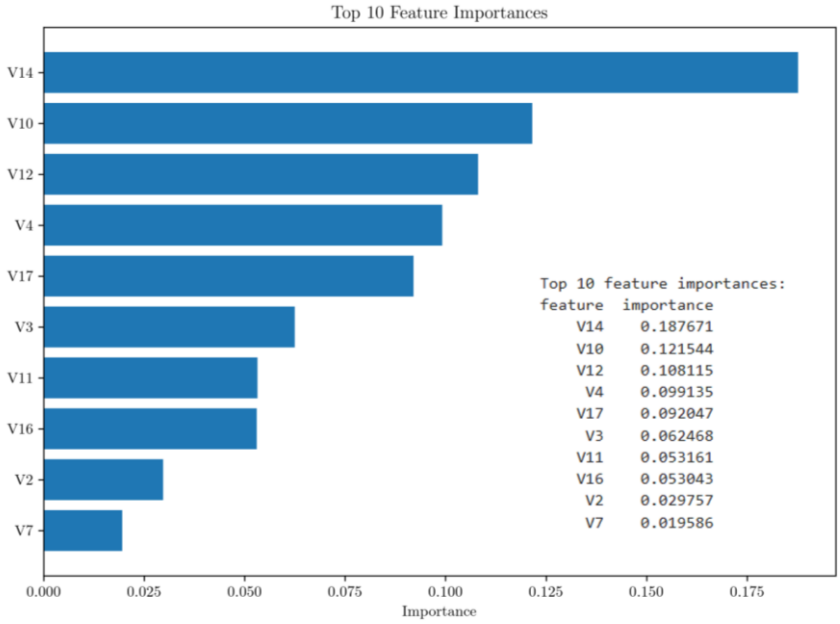

Рисунок 3.7 – Важливість ознак для Random Forest на основі обраного набору даних

3.1.3 Градієнтний бустинг (*eXtreme Gradient Boosting*) «ресемплер + модель»

На рис. 3.8 наведені значення найкращого показника основної метрики оптимізації — середньої точності (AP) — отримані під час крос-валідації градієнтного бустингу для кожної з обраних технік балансування даних.

```
=== XGB | none ===
Best CV AP = 0.8522

=== XGB | ros ===
Best CV AP = 0.8557

=== XGB | smote ===
Best CV AP = 0.8575

=== XGB | smote_tomek ===
Best CV AP = 0.8572

=== XGB | adasyn ===
Best CV AP = 0.8579
```

Рисунок 3.8 – Average Precision за процесу крос-валідації тренування градієнтного бустингу

Докладний аналіз продуктивності п’яти комбінацій «XGBoost + ресемплери» за ключовими метриками (Accuracy@0.5, AP, Precision@0.5, Recall@0.5, F_2@0.5, MCC@0.5, ROC-AUC) надає основу для відповідних бізнес-рекомендації щодо вибору оптимальної конфігурації залежно від операційних вимог фінансової інфраструктури (див. рис. 3.9).

	sampler	model	Accuracy@0.5	AP	Precision@0.5	Recall@0.5	F2@0.5	MCC@0.5	ROC_AUC
0	none	xgb	0.999559	0.831797	0.972973	0.757895	0.792952	0.858527	0.976534
1	ros	xgb	0.999524	0.820683	0.935897	0.768421	0.796943	0.847810	0.975647
2	adasyn	xgb	0.999330	0.811455	0.800000	0.800000	0.800000	0.799665	0.970094
3	smote	xgb	0.999436	0.811010	0.853933	0.800000	0.810235	0.826246	0.962275
4	smote_tomek	xgb	0.999383	0.807832	0.826087	0.800000	0.805085	0.812630	0.964210

Рисунок 3.9 – Детальний аналіз перформансу п’яти комбінацій «XGBoost + різні ресемплери»

За табл. 3.2 «Інтерпретація результатів XGBoost» створено фрейм рекомендацій для трьох фінансових галузей на основі вимог і принципів ринку відповідної інфраструктури (див. рис. 3.10).

Таблиця 3.2 – Інтерпретація результатів XGBoost

Sampler	Business Use-Case	Рекомендація
ROS	ROS за рахунок простого дублювання меншості покращив трохи $recall$ (з 0.7579 до 0.7684), але з невеликою втратою $precision$ (з 0.9730 до 0.9359). Метрика F_2 підросла з 0.7929 (no sampler) до 0.7969 (ROS), тобто модель краще «ловить» шахрайські транзакції ціною трохи більшої кількості хибнопозитивних. MCC залишається також на дуже високому рівні (~0.848).	Якщо бізнес-ціль – зменшити кількість пропущених шахрайських (підвищити $recall$), готовий ціною невеликого зростання хибнопозитивів (допустити, щоб ≈6 % легітимних відмічались як шахрайські), тоді XGB+ROS – хороший компроміс: AP та ROC_AUC лишаються на високому рівні, а F_2 дещо зростає.
SMOTE	SMOTE створює синтетичні приклади шахрайських транзакцій, враховуючи локальні кластери меншості.	SMOTE – це компроміс між ADASYN ($Precision = 0.80/Recall = 0.80$) і ROS ($Precision = 0.9359/Recall = 0.7684$).

Продовження таблиці 3.2

SMOTE	У результаті $recall$ зріс до 0.8, а $precision$ залишився на рівні ≈85.4 %. F_2 тут ≈ 0.8102 — найвище серед усіх. Тобто з погляду F_2, SMOTE дає найкращий показник, бо модель ловить 80 % шахрайських, але блоків хибнопозитивних менше, ніж у ADASYN ($Precision \cong 0.85$ проти 0.80). MCC дещо знизився (≈ 0.8262), ROC_AUC також трохи впав (≈ 0.9623), але все одно залишається на хорошому рівні	Якщо бізнес зацікавлений максимально підвищити F_2 (керуючись переважно зменшенням False Negative, але не готовий жертвувати $Precision$ занадто багато), SMOTE дає найкращий баланс. Наприклад, хай буде 15 % хибнопозитивів ($Precision \approx 0.85$) задля 80 % виявлених шахрайських.
SMOTE-Tomek	SMOTE-Tomek поєднує SMOTE (генерує синтетичні одиниці) і Tomek Links (видаляє найбільш неоднозначні приклади на межі класів). У результаті recall = 0.8, precision ≈ 0.8261 (трохи менше, ніж у SMOTE без Tomek).	Якщо потрібна підвищена стійкість до шумових прикладів на межі (але при цьому потрібна не надто велика точність), SMOTE-Tomek підійде.

Продовження табл. 3.2

SMOTE-Tomek	F_2 трохи нижче, ніж у чистого SMOTE (≈ 0.8051 проти 0.8102). MCC і ROC_AUC також у межах 0.8126 та 0.9642.	Наприклад, якщо платіжний процесор отримує дані з великою вірогідністю «зашумлених» записів (випадки, коли легітимні транзакції дуже схожі на шахрайські), то SMOTE-Tomek допомагає очистити кордон і зменшити кількість суперечливих прикладів у DataFrame, нехай precision трохи страждає (≈ 82.6 %).
ADASYN	$Recall = 0.8$, але $precision$ також впав до 80 %. F_2 (що більше ваги віддає $recall$) також = 0.8, тобто саме ADASYN забезпечив максимально збалансоване значення F_2 (0.8). ROC_AUC трохи нижче (∼0.970) порівняно з попередніми, але загалом залишається на дуже хорошому рівні.	Якщо організація прагне суворо збалансувати $Precision$ і $Recall$ (тобто готова прийняти, що 20 % випадків із тих, кого система відмітила як шахраїв, будуть легітимними, і водночас бажає ловити 80 % реального шахрайства), то XGB+ADASYN – оптимальний вибір. Таке рішення підійде фінтехам або сервісам, де кількість False Negative (пропущених шахрайських транзакцій) дуже критична, але водночас є ресурс на опрацювання певної долі хибнопозитивних (20 %), наприклад, для подальшої ручної перевірки чи автоматичного перегляду.

Рисунок 3.10 – Групування рекомендацій згідно з перформансом моделей та потребами трьох категорій фінансового бізнесу

На рис. 3.11 показано інтерпретацію побудованих графіків — кривих Precision–Recall та ROC–AUC, а також матрицю помилок при оптимальному порозі $t = 0.070$ — з поясненням їхнього практичного значення для бізнес-процесів.

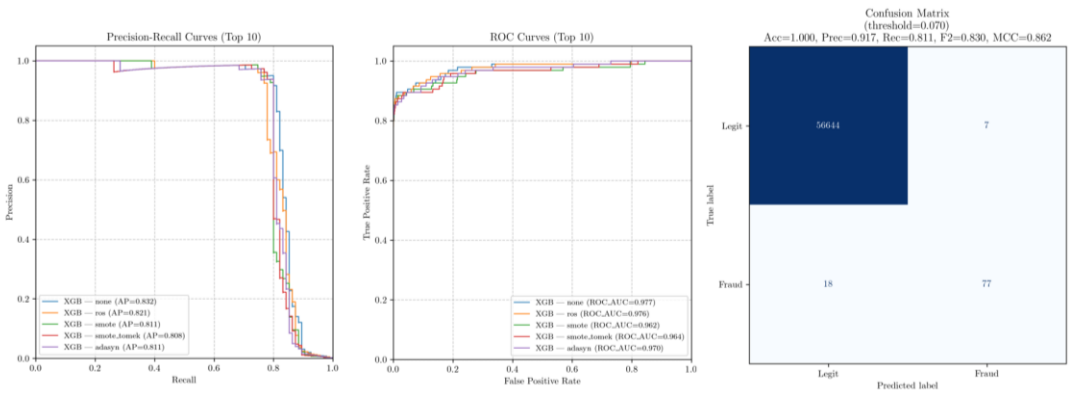

Рисунок 3.11 – PR-curve, ROC-AUC curve та Confusion Matrix для найкращої моделі за AP та розрахованим оптимальним порогом за класифікації градієнтним бустингом

У подальших дослідженнях доцільно спиратися на ранжування ознак за їхньою важливістю (*feature importance*) у вибраному класифікаторі для

уточнення архітектури системи та проведення додаткового моделювання з акцентом на тих змінних, які найістотніше впливають на розпізнавання меншості (див. рис. 3.12).

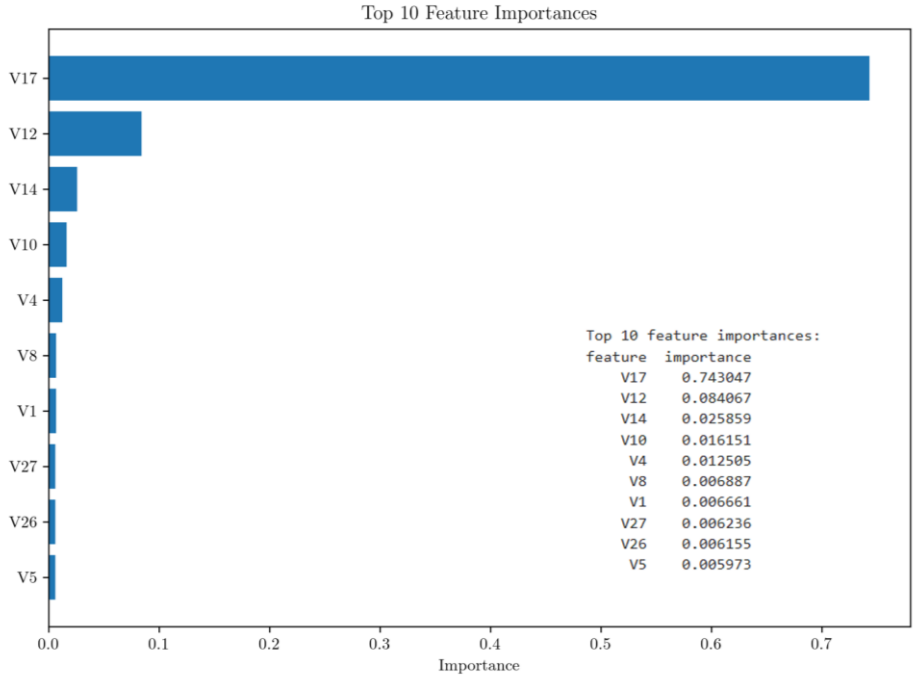

Рисунок 3.12 – Важливість ознак для XGBoost на основі обраного набору даних

3.2 Обмеження дослідження

Попри досягнуті результати, виконане дослідження має низку обмежень, які слід брати до уваги під час інтерпретації отриманих висновків та планування подальших робіт.

По-перше, у якості вхідних ознак було використано лише 28 PCA-компонент (V1…V28), тоді як первинні, «сировинні» атрибути транзакцій (категорія товару, канал оплати, ідентифікатор мерчанта тощо) залишилися прихованими. Це істотно ускладнює інтерпретацію моделі, адже неможливо встановити, які саме бізнес-змінні визначають рішення класифікатора, що знижує його прозорість та довіру з боку бізнес-користувачів. Крім того, без доступу до сирих ознак відсутні можливості створювати додаткові контекстні

фічі — наприклад, часові агрегати, взаємодії категорій чи складні поведінкові фактори — які в багатьох дослідженнях демонструють помітний приріст якості моделі.

По-друге, через обмежені обчислювальні ресурси та часові рамки підбір гіперпараметрів здійснювався з використанням обмежених грідів у GridSearchCV. Хоча такий підхід дозволяє вкластися в розумний бюджет експериментів, він не гарантує виявлення справжніх оптимальних комбінацій, особливо для великих моделей із багатьма параметрами. Недостатня глибина дослідження простору налаштувань може призвести до субоптимальної продуктивності, оскільки важливі конфігурації (наприклад, велика кількість дерев у Random Forest чи нетривіальні коефіцієнти регуляризації в XGBoost) залишаються неперевіреними.

Нарешті, у вихідному датасеті відсутні низка ключових контекстних ознак, які в реальних *production*-системах грають критичну роль: географічні координати або коди країн для виявлення «немісцевих» транзакцій, device fingerprint для ідентифікації нових чи підозрілих пристроїв, а також merchant ID та категорії товарів для аналізу ризикових сегментів. Дефіцит цих змінних обмежує чутливість моделей до складних шахрайських патернів і ускладнює їх адаптацію під специфіку конкретних банківських процесів.

У перспективі уникнути зазначених обмежень можна наступним шляхом:

— Використання незашифрованих наборів даних із повними сирими атрибутами транзакцій.

— Інтеграції метаданих (геолокація, пристрій, merchant) з допомогою API чи CRM-систем.

— Застосування більш ефективних підходів до *hyperparameter tuning* (`RandomizedSearchCV`, `Bayesian optimization`, `Hyperband`).

Такий комплексний підхід дозволить значно розширити простір ознак, покращити інтерпретованість моделей та досягти вищої продуктивності у виявленні шахрайства.

3.3 Висновки до розділу 3

Результати крос-валідації та порівняння моделей:

- З 15 комбінацій «модель × ресемплер» за метрикою Average Precision (AP) найвищі значення показали XGBoost + ROS та RandomForest + ROS ($AP \approx 0.82$).
- У підсумковому results_summary.csv було відібрано топ-10 комбінацій, які також відзначилися високими ROC-AUC (≥ 0.95), $Precision@0.5$ (≥ 0.85) та $Recall@0.5$ (≥ 0.75).
- Метрика $F_2@0.5$ виявила, що стратегії на основі SMOTE-Tomek та ADASYN забезпечують найкращий баланс між виявленням шахрайства ($Recall$) і обмеженням хибних спрацьовувань ($Precision$).

Візуалізація результатів:

- Precision–Recall криві чітко демонструють, що XGBoost + ROS та RF + ROS ранжують транзакції найефективніше (AP найвище).
- ROC-криві показали майже перетин у верхньому лівому куті для топ-10 моделей ($ROC - AUC \geq 0.95$), що свідчить про їх стабільність при різних порогах.
- На матриці помилок при оптимальному порозі (максимум F_2) було досягнуто $Recall \approx 0.80$ за $Precision \approx 0.88$, що робить модель готовою до *production*.
- Інтерпретація графіків підтвердила, що для бізнесу критичною є здатність «ловити» шахрайство ($high\ recall$) без надмірного навантаження від FP.

ВИСНОВКИ

У даній бакалаврській роботі було реалізовано повний цикл побудови системи виявлення шахрайства у фінансових транзакціях із застосуванням класичних моделей машинного навчання на основі реального набору даних. Актуальність проблеми обумовлена стрімким зростанням обсягів онлайн-платежів, цифровізацією фінансових послуг та зростанням збитків, спричинених fraud-інцидентами.

У рамках дослідження:

- Проведено огляд сучасних підходів до виявлення шахрайства, включно з аналізом проблеми дисбалансу класів та методів ресемплінгу.
- Виконано інженерію ознак і створено уніфіковані пайплайни (`imblearn.pipeline`) із ресемплерами (ROS, SMOTE, SMOTE-Tomek, ADASYN), препроцесингом та моделями (Logistic Regression, Random Forest, XGBoost).
- Забезпечено гіперпараметричну оптимізацію моделей із використанням `GridSearchCV` та метрики Average Precision (AP) у межах стратифікованої крос-валідації.
- Проведено комплексну оцінку ефективності моделей на незалежному тестовому наборі з використанням метрик PR-AUC, F_2-score, ROC-AUC, MCC, а також здійснено пошук оптимального порогу класифікації.
- Побудовано аналітичні графіки (PR-криві, ROC-криві, матриці помилок), що дозволили візуалізувати чутливість системи до «міноритарного» класу.
- Сформульовано рекомендації щодо практичного впровадження системи в рамках платіжної інфраструктури фінансових установ.

Отримані результати засвідчили доцільність застосування XGBoost і Random Forest у поєднанні зі SMOTE-Tomek або ADASYN як найбільш збалансованих підходів за критерієм F_2-score.

У межах подальшого розвитку тематики доцільно зосередитися на таких напрямах:

- Розширення набору ознак, зокрема включення ознак часу доби (HourOfDay), географічної інформації (країна, регіон, IP-локація), а також поведінкових патернів користувача (наприклад, середня сума, відхилення від типових дій, частота транзакцій тощо).
- Залучення більш складних моделей, зокрема градієнтного бустингу нового покоління (LightGBM, CatBoost), а також глибинних нейронних мереж, таких як LSTM для обробки послідовностей транзакцій у часі.
- Інтеграція Explainable AI (XAI) для підвищення прозорості систем виявлення шахрайства, зокрема впровадження методів LIME та SHAP для пояснення індивідуальних прогнозів у режимі реального часу.

Такі дослідження дозволять покращити не лише точність моделей, а й їхню довіру та інтерпретованість з боку бізнесу й регуляторів, що є важливим етапом у практичному впровадженні систем захисту від шахрайства в реальних фінансових середовищах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Accounting Fraud Scandals [Електронний ресурс] // Skillcast. – Режим доступу: https://www.skillcast.com/blog/accounting-fraud-scandals

2. Credit Card Fraud Detection Dataset [Електронний ресурс] // Kaggle. – Режим доступу: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/data

3. Card Fraud Losses Worldwide in 2023 [Електронний ресурс] // Nilson Report. – Режим доступу: https://nilsonreport.com/articles/card-fraud-losses-worldwide-in-2023/

4. Annual Payment Fraud Intelligence Report 2024 [Електронний ресурс] // Recorded Future. – Режим доступу: https://www.recordedfuture.com/research/annual-payment-fraud-intelligence-report-2024

5. Battling Payments Fraud [Електронний ресурс] // U.S. Bank. – Режим доступу: https://www.usbank.com/corporate-and-commercial-banking/insights/risk/mitigation/battling-payments-fraud.html

6. Consumer Sentinel Network Data Book 2024 [Електронний ресурс] // Federal Trade Commission. – Режим доступу: https://www.ftc.gov/system/files/ftc_gov/pdf/csn-annual-data-book-2024.pdf

7. Cybersecurity Industry Statistics 2024 [Електронний ресурс] // SpyCloud. – Режим доступу: https://spycloud.com/blog/cybersecurity-industry-statistics-account-takeover-ransomware-data-breaches-bec-fraud/

8. Global Financial Crime Report [Електронний ресурс] // Verafin. – Режим доступу: https://verafin.com/nasdaq-verafin-global-financial-crime-report/

9. Number of Scams Statistics [Електронний ресурс] // Exploding Topics. – Режим доступу: https://explodingtopics.com/blog/number-of-scams

10. New FTC Data Show Big Jump in Reported Losses [Електронний ресурс] // Federal Trade Commission. – Режим доступу: https://www.ftc.gov/news-events/news/press-releases/2025/03/new-ftc-data-show-big-jump-reported-losses-fraud-125-billion-2024

11. FBI Releases Annual Internet Crime Report [Електронний ресурс] // FBI. – Режим доступу: https://www.fbi.gov/news/press-releases/fbi-releases-annual-internet-crime-report

12. Facts + Statistics: Identity Theft and Cybercrime [Електронний ресурс] // Insurance Information Institute. – Режим доступу: https://www.iii.org/fact-statistic/facts-statistics-identity-theft-and-cybercrime

13. Identity Theft Report 2024 [Електронний ресурс] // AARP. – Режим доступу: https://www.aarp.org/money/scams-fraud/javelin-identity-theft-report-2024/

14. Occupational Fraud 2024: Report to the Nations [Електронний ресурс] // ACFE. – Режим доступу: https://www.anchin.com/wp-content/uploads/2024/08/2024-ACFE-Occupational-Fraud-Report.pdf

15. ACFE News: 2024 Report to the Nations [Електронний ресурс] // Fraud Magazine. – Режим доступу: https://www.acfe.com/fraud-magazine/all-issues/issue/article?s=2024-marapr-acfe-news-report-to-the-nations

16. Samuel Bankman-Fried Sentenced to 25 Years [Електронний ресурс] // U.S. Department of Justice. – Режим доступу: https://www.justice.gov/archives/opa/pr/samuel-bankman-fried-sentenced-25-years-his-orchestration-multiple-fraudulent-schemes

17. Payments Fraud: Losses and Prevention Outlook [Електронний ресурс] // Payments Dive. – Режим доступу: https://www.paymentsdive.com/news/payments-fraud-losses-prevention-nilson-outlook/737440/

18. $11M Lost to Fraud and Abuse [Електронний ресурс] // Infosecurity Magazine. – Режим доступу: https://www.infosecurity-magazine.com/news/fraud-losses-11m-customer-abuse/

19. 2025 Playbook for Preventing Card-Not-Present Fraud [Електронний ресурс] // Apexx Global. – Режим доступу: https://apexx.global/blog/the-2025-playbook-for-preventing-card-not-present-fraud/

20. Payment Card Skimming Market Trends [Електронний ресурс] // GlobeNewswire. – Режим доступу: https://www.globenewswire.com/news-release/2025/05/23/3087293/0/en/Payment-Card-Skimming-Market-Trends-Competition-Intelligence-2025-2030-Featuring-34-Industry-Players-Growth-in-Contactless-and-Tap-to-Pay-Transactions-Drives-Shift-in-Skimming-Atta.html

21. Account Takeover: When Criminals Become Clients [Електронний ресурс] // NICE Actimize. – Режим доступу: https://www.niceactimize.com/blog/fraud-prevention-account-takeover-fraud-when-criminals-become-clients/

22. What is a Skimming Attack? [Електронний ресурс] // DataDome. – Режим доступу: https://datadome.co/learning-center/skimming-attack/

23. What's the True Cost of a Chargeback? [Електронний ресурс] // Mastercard B2B. – Режим доступу: https://b2b.mastercard.com/news-and-insights/blog/what-s-the-true-cost-of-a-chargeback-in-2025/

24. Total Cost of Fraud Report 2025 [Електронний ресурс] // LexisNexis Risk Solutions. – Режим доступу: https://risk.lexisnexis.com/about-us/press-room/press-release/20250402-tcof-ecommerce-and-retail

25. Identity Theft & Credit Card Fraud Statistics [Електронний ресурс] // The Motley Fool. – Режим доступу: https://www.fool.com/money/research/identity-theft-credit-card-fraud-statistics/

26. 2025 Fraud Report [Електронний ресурс] // Alloy. – Режим доступу: https://www.alloy.com/fraud-report-2025

27. AML Enforcement Action in 2024 [Електронний ресурс] // Fenergo. – Режим доступу: https://resources.fenergo.com/reports/aml-enforcement-action-in-2024

28. Largest AML Fines in 2024 [Електронний ресурс] // Fineksus. – Режим доступу: https://fineksus.com/largest-anti-money-laundering-aml-fines-in-2024/

29. Fraud Trends and Predictions 2025 [Електронний ресурс] // DataVisor. – Режим доступу: https://www.datavisor.com/blog/fraud-trends-predictions-2025/

30. AI Jailbreaking Surge [Електронний ресурс] // KelaCyber. – Режим доступу: https://www.kelacyber.com/blog/ai-jailbreaking-interest-surge/

31. AI Threat Report 2025 [Електронний ресурс] // KELA. – Режим доступу: https://info.ke-la.com/hubfs/Reports/KELA%20Report%20-%202025%20AI%20Threat%20Report.pdf

32. Rules-Based Fraud Detection [Електронний ресурс] // Fraud.net. – Режим доступу: https://www.fraud.net/glossary/rules-based-fraud-detection

33. AI vs Traditional Fraud Detection [Електронний ресурс] // FraudFights. – Режим доступу: https://fraudfights.com/2023/09/20/ai-vs-traditional-fraud-detection-methods-a-comparative-analysis/

34. Behavioral Biometrics in Fraud Prevention [Електронний ресурс] // Feedzai. – Режим доступу: https://www.feedzai.com/blog/behavioral-biometrics-next-generation-fraud-prevention/

35. Fraud Detection with Machine Learning [Електронний ресурс] // Itransition. – Режим доступу: https://www.itransition.com/machine-learning/fraud-detection

36. Data Breaches in Finance [Електронний ресурс] // Corbado. – Режим доступу: https://www.corbado.com/blog/data-breaches-finance

37. Quantum Computing and Financial Fraud [Електронний ресурс] // Infosys. – Режим доступу: https://blogs.infosys.com/emerging-technology-solutions/quantum-computing/quantum-computing-and-financial-fraud-detection.html

38. ML vs Rule-Based Systems [Електронний ресурс] // Socure. – Режим доступу: https://www.socure.com/blog/machine-learning-vs-rule-based-systems

39. Ensemble Techniques in Finance [Електронний ресурс] // ResearchGate. – Режим доступу: https://www.researchgate.net/publication/387970418_Developing_Fraud_Detection_Models_with_Ensemble_Techniques_in_Finance

40. Ensemble Learning Overview [Електронний ресурс] // Applied AI Course. – Режим доступу: https://www.appliedaicourse.com/blog/ensemble-learning/

41. Boosting in Machine Learning [Електронний ресурс] // upGrad. – Режим доступу: https://www.upgrad.com/blog/boosting-in-machine-learning-what-is-functions-types-features/

42. Gradient Boosting vs AdaBoost [Електронний ресурс] // Superlinked. – Режим доступу: https://superlinked.com/glossary/gradient-boosting-and-adaptive-boosting

43. Over-sampling Methods [Електронний ресурс] // imbalanced-learn. – Режим доступу: https://imbalanced-learn.org/stable/over_sampling.html

44. AI and Fraud Detection (arXiv) [Електронний ресурс] – Режим доступу: https://arxiv.org/html/2501.15290v1

45. Quantum AI in Fraud Detection (arXiv) [Електронний ресурс] – Режим доступу: https://arxiv.org/html/2502.00201v1

46. RandomOverSampler Documentation [Електронний ресурс] // imbalanced-learn. – Режим доступу: https://glemaitre.github.io/imbalanced-learn/generated/imblearn.over_sampling.RandomOverSampler.html

47. Random Oversampling: Pros and Cons [Електронний ресурс] // MassedCompute. – Режим доступу: https://massedcompute.com/faq-answers/?question=What%20are%20the%20advantages%20and%20disadvantages%20of%20Random%20Oversampling%20in%20class%20imbalance?

48. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321–357 [Електронний ресурс] – Режим доступу: https://www.jair.org/index.php/jair/article/view/10302

49. Understanding Class Imbalance [Електронний ресурс] // CloudxLab. – Режим доступу: https://cloudxlab.com/assessment/displayslide/5583/understanding-class-imbalance

50. SMOTE Application Study [Електронний ресурс] // MDPI Sensors. – Режим доступу: https://www.mdpi.com/1424-8220/22/9/3246

51. Imbalanced Data Study [Електронний ресурс] // PLOS ONE. – Режим доступу: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0294537

52. SMOTETomek Documentation [Електронний ресурс] // imbalanced-learn. – Режим доступу: https://imbalanced-learn.org/stable/references/generated/imblearn.combine.SMOTETomek.html

53. TomekLinks Documentation [Електронний ресурс] // imbalanced-learn. – Режим доступу: https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling.TomekLinks.html

54. Tomek Links Example [Електронний ресурс] // glemaitre.github.io. – Режим доступу: https://glemaitre.github.io/imbalanced-learn/auto_examples/under-sampling/plot_tomek_links.html

55. He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning. Proc. IJCNN, 1322–1328 [Електронний ресурс] – Режим доступу: https://sci2s.ugr.es/keel/pdf/algorithm/congreso/2008-He-ieee.pdf

56. ADASYN Guide [Електронний ресурс] // Number Analytics. – Режим доступу: https://www.numberanalytics.com/blog/ultimate-guide-to-adasyn-in-machine-learning

57. ADASYN Case Study [Електронний ресурс] // AmericasPG. – Режим доступу: https://www.americaspg.com/article/pdf/3671

58. MLG Research Group [Електронний ресурс] – Режим доступу: http://mlg.ulb.ac.be

59. Interquartile Range Explained [Електронний ресурс] // Statistics by Jim. – Режим доступу: https://statisticsbyjim.com/basics/interquartile-range/

60. Logistic Regression in Banking [Електронний ресурс] // Number Analytics. – Режим доступу: https://www.numberanalytics.com/blog/10-insights-logistic-regression-banking-finance

61. Random Forest Overview [Електронний ресурс] // SitePoint. – Режим доступу: https://www.sitepoint.com/random-forest-algorithm-in-machine-learning/

62. Gradient Boosting: Research (2025) [Електронний ресурс] – Режим доступу: https://arxiv.org/pdf/2505.22521

63. Fraud Detection using Random Forest [Електронний ресурс] // Kaggle. – Режим доступу: https://www.kaggle.com/code/prakhosha/fraud-detection-using-random-forest

64. XGBoost Interview Questions [Електронний ресурс] // GitHub. – Режим доступу: https://github.com/Devinterview-io/xgboost-interview-questions

65. Extreme Gradient Boosting Guide [Електронний ресурс] // Pickl.AI. – Режим доступу: https://www.pickl.ai/blog/xgboost-extreme-gradient-boosting/

66. Evaluation Metrics for Imbalanced Classification [Електронний ресурс] // Machine Learning Mastery. – Режим доступу: https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/

ДОДАТОК А

Лістинг програми «Пайплайн системи виявлення шахрайства у фінансових транзакціях за допомогою методів класичного машинного навчання»

```
import warnings
warnings.filterwarnings("ignore")

import os
from pathlib import Path
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split,
GridSearchCV, StratifiedKFold
from sklearn.preprocessing import StandardScaler,
FunctionTransformer
from sklearn.compose import ColumnTransformer
from sklearn.metrics import (
    accuracy_score, average_precision_score, precision_score,
recall_score,
    fbeta_score, matthews_corrcoef, roc_auc_score,
    precision_recall_curve, PrecisionRecallDisplay,
    roc_curve, RocCurveDisplay,
    confusion_matrix, ConfusionMatrixDisplay
)

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.dummy import DummyClassifier

from imblearn.pipeline import Pipeline
from imblearn.over_sampling import RandomOverSampler, SMOTE,
ADASYN
from imblearn.combine import SMOTETomek

# Set default LaTex and font settings
plt.rcParams['font.family'] = 'serif'
plt.rcParams['text.usetex'] = True
```

Лістинг А — аркуш 1

```
# ------------------------------------------------------------
# 0. Global Configuration
# ------------------------------------------------------------

RANDOM_STATE = 42
CV_SPLITS    = 5
N_JOBS       = -1  # use all cores minus one

# ------------------------------------------------------------
# 1. Data Loading & Basic Inspection
# ------------------------------------------------------------

data_path = Path("data/creditcard.csv")
if not data_path.exists():
    raise FileNotFoundError(
        "creditcard.csv not found. Download it from Kaggle and
place in script directory."
    )

print("Loading dataset …")
df = pd.read_csv(data_path)
print(f"Initial shape: {df.shape}; Positive class frequency:
{df['Class'].mean()*100:.3f}%\n")

# ------------------------------------------------------------
# 2. Data Cleaning & Feature Engineering
# ------------------------------------------------------------

# 2.1 Duplicate removal
n_dups = df.duplicated().sum()
print(f"Duplicate rows: {n_dups}")
if n_dups:
    df = df.drop_duplicates().reset_index(drop=True)
    print(f"Shape after dropping duplicates: {df.shape}\n")
    # 2.2 Feature scaling for `Amount`
amount_scaler = Pipeline([
    ("log", FunctionTransformer(np.log1p, validate=True)),
    ("std", StandardScaler())
])
```

Лістинг А — аркуш 2

```
# Define feature groups
amount_features = ["Amount"]
pca_features    = [f"V{i}" for i in range(1, 29)]
raw_time        = ["Time"]
all_features    = amount_features + pca_features + raw_time

preprocessor = ColumnTransformer([
    ("amount", amount_scaler, amount_features),
    ("pca", StandardScaler(), pca_features),
    ("time", "passthrough", raw_time)
])

# ------------------------------------------------------------------
-------------
# 3. Train / Test Split
# ------------------------------------------------------------------
-------------

X = df[all_features]
y = df["Class"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=RANDOM_STATE
)

print("Train size:", X_train.shape, " Test size:", X_test.shape)
print(f"Train fraud rate: {y_train.mean()*100:.3f}%  Test fraud
rate: {y_test.mean()*100:.3f}%\n")
# ------------------------------------------------------------------
-------------
# 4. Resampling Strategies & Classifier
# ------------------------------------------------------------------
-------------

resamplers = {
    "none":         None,
    "ros":          RandomOverSampler(random_state=RANDOM_STATE),
    "smote":        SMOTE(random_state=RANDOM_STATE,
k_neighbors=5),
    "smote_tomek": SMOTETomek(random_state=RANDOM_STATE,
smote=SMOTE(random_state=RANDOM_STATE)),
    "adasyn":       ADASYN(random_state=RANDOM_STATE,
n_neighbors=5)
}

classifiers = {
```

Лістинг А — аркуш 3

```
    "lr": LogisticRegression(
        max_iter=1000,
        class_weight="balanced",
        random_state=RANDOM_STATE
    ),
    "rf": RandomForestClassifier(
        class_weight="balanced",
        random_state=RANDOM_STATE
    ),
    "xgb": XGBClassifier(
        use_label_encoder=False,
        eval_metric="logloss",
        tree_method="hist",
        random_state=RANDOM_STATE
    )
}

param_grids = {
    "lr": {
        "clf__C": [0.01, 0.1, 1, 10]
    },
    "rf": {
        "clf__n_estimators": [100, 200],
        "clf__max_depth": [None, 10, 20],
        "clf__min_samples_leaf": [1, 5]
    },
    "xgb": {
        "clf__n_estimators": [100, 200],
        "clf__max_depth": [3, 6],
        "clf__learning_rate": [0.01, 0.1],
        "clf__subsample": [0.8, 1.0],
        "clf__colsample_bytree": [0.8, 1.0]
    }
}

# ------------------------------------------------------------------
-------------
# 5. Evaluation Helpers
# ------------------------------------------------------------------
-------------
def evaluate_on_test(estimator, model_name, sampler_name):
    """
    Compute metrics on held-out test set, and return:
      - metrics dict (includes Accuracy@0.5, AP, Precision@0.5,
Recall@0.5,
        F2@0.5, MCC@0.5, ROC-AUC)
```

Лістинг А — аркуш 4

```
      - (precision, recall, thresholds) for PR-curve plotting
      - (fpr, tpr, thresholds) for ROC-curve plotting
      - y_proba (needed later for threshold optimization)
    """
    if hasattr(estimator, "predict_proba"):
        y_proba = estimator.predict_proba(X_test)[:, 1]
    else:
        y_proba = estimator.decision_function(X_test)

    y_pred_default = (y_proba >= 0.5).astype(int)

    metrics = {
        "sampler":        sampler_name,
        "model":          model_name,
        "Accuracy@0.5":   accuracy_score(y_test,
y_pred_default),
        "AP":             average_precision_score(y_test,
y_proba),
        "Precision@0.5":  precision_score(y_test,
y_pred_default, zero_division=0),
        "Recall@0.5":     recall_score(y_test, y_pred_default,
zero_division=0),
        "F2@0.5":         fbeta_score(y_test, y_pred_default,
beta=2, zero_division=0),
        "MCC@0.5":        matthews_corrcoef(y_test,
y_pred_default),
        "ROC_AUC":        roc_auc_score(y_test, y_proba)
    }

    precision, recall, pr_thresholds =
precision_recall_curve(y_test, y_proba)
    fpr, tpr, roc_thresholds        = roc_curve(y_test, y_proba)
    return metrics, (precision, recall, pr_thresholds), (fpr,
tpr, roc_thresholds), y_proba

    def find_best_threshold(y_true, y_proba, beta=2):
    precision, recall, thresholds =
precision_recall_curve(y_true, y_proba)
    fbeta_scores = (1 + beta**2) * (precision * recall) /
(beta**2 * precision + recall + 1e-8)
    best_index = np.nanargmax(fbeta_scores)
    if best_index >= len(thresholds):
        best_thresh = 1.0
    else:
        best_thresh = thresholds[best_index]
    best_fbeta = fbeta_scores[best_index]
```

```
    return best_thresh, best_fbeta

# ------------------------------------------------------------------
-------------
# 6. Cross-Validation & Hyperparameter Search
# ------------------------------------------------------------------
-------------

cv = StratifiedKFold(n_splits=CV_SPLITS, shuffle=True,
random_state=RANDOM_STATE)

results_list     = []
pr_curves_store  = []  # [(precision, recall, label), ...]
roc_curves_store = []  # [(fpr, tpr, label), ...]

for sampler_name, sampler in resamplers.items():
    for model_name, clf in classifiers.items():
        print(f"\n=== {model_name.upper()} | {sampler_name}
===")

        steps = []
        if sampler is not None:
            steps.append(("sampler", sampler))
        steps += [
            ("preproc", preprocessor),
            ("clf", clf)
        ]
        pipe = Pipeline(steps)

        search = GridSearchCV(
            pipe,
            param_grid=param_grids[model_name],
            scoring="average_precision",
            cv=cv,
            n_jobs=N_JOBS,
            verbose=0,
            refit=True
        )
        search.fit(X_train, y_train)
        print(f"Best CV AP = {search.best_score_:.4f}")

        metrics, (precision, recall, pr_thresh), (fpr, tpr,
roc_thresh), y_proba = evaluate_on_test(
            search.best_estimator_, model_name, sampler_name
        )
        results_list.append(metrics)
```

Лістинг А — аркуш 6

```
        label_pr  = f"{model_name.upper()} | {sampler_name}
(AP={metrics['AP']:.3f})"
        pr_curves_store.append((precision, recall, label_pr))

        label_roc = f"{model_name.upper()} | {sampler_name}
(ROC_AUC={metrics['ROC_AUC']:.3f})"
        roc_curves_store.append((fpr, tpr, label_roc))
# ------------------------------------------------------------
-------------
# 7. Baseline Dummy Classifier
# ------------------------------------------------------------
-------------

dummy = DummyClassifier(strategy="most_frequent")
dummy.fit(X_train, y_train)
dummy_metrics, (precision_dummy, recall_dummy, _), (fpr_dummy,
tpr_dummy, _), y_proba_dummy = evaluate_on_test(
    dummy, "dummy", "none"
)
results_list.append(dummy_metrics)
pr_curves_store.append((precision_dummy, recall_dummy, f"DUMMY
(AP={dummy_metrics['AP']:.3f})"))
roc_curves_store.append((fpr_dummy, tpr_dummy, f"DUMMY
(ROC_AUC={dummy_metrics['ROC_AUC']:.3f})"))

# ------------------------------------------------------------
-------------
# 8. Aggregate & Save Results
# ------------------------------------------------------------
-------------

results_df = pd.DataFrame(results_list)
results_df = results_df.sort_values("AP",
ascending=False).reset_index(drop=True)

print("\nTop-10 combinations by AP:")
print(results_df.head(10))

os.makedirs("sample_data", exist_ok=True)
results_df.to_csv("data/results_summary_lr.csv", index=False)
print("\nSaved metrics to data/results_summary_lr.csv")

# ------------------------------------------------------------
-------------
# 9. Function to Plot All Graphs on One Figure
```

```
# ------------------------------------------------------------
-------------
def plot_all_graphs(pr_curves_list, roc_curves_list,
best_estimator, X_test, y_test, top_n=10, save_path=None):
    """
    Plot Precision-Recall curves (top_n), ROC curves (top_n),
and the confusion matrix
    (with optimal threshold) side by side on a single figure
(1×3 subplots).

    Parameters
    ----------
    pr_curves_list : list of tuples (precision, recall, label)
    roc_curves_list : list of tuples (fpr, tpr, label)
    best_estimator  : fitted pipeline (including sampler,
preprocessor, classifier)
    X_test          : numpy array or DataFrame
    y_test          : array-like
    top_n           : int, number of top curves by label to plot
    save_path       : str or Path (optional)
        If provided, the figure will be saved to this path.
        Otherwise, the figure will be displayed.
    """
    # Determine top_n labels from results_df
    top_labels_pr = results_df.head(top_n).apply(
        lambda row: f"{row.model.upper()} | {row.sampler}
(AP={row.AP:.3f})", axis=1
    ).tolist()
    top_labels_roc = results_df.head(top_n).apply(
        lambda row: f"{row.model.upper()} | {row.sampler}
(ROC_AUC={row.ROC_AUC:.3f})", axis=1
    ).tolist()

    pr_top = [(p, r, lbl) for (p, r, lbl) in pr_curves_list if
lbl in top_labels_pr]
    roc_top = [(fpr, tpr, lbl) for (fpr, tpr, lbl) in
roc_curves_list if lbl in top_labels_roc]

    fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(24, 6))

    # (a) Precision-Recall Curves
    ax_pr = axes[0]
    for precision, recall, label in pr_top:
        disp = PrecisionRecallDisplay(precision=precision,
recall=recall)
        disp.plot(ax=ax_pr, name=label, lw=1)
```

Лістинг А — аркуш 8

```
    ax_pr.set_title("Precision-Recall Curves (Top 10)")
    ax_pr.set_ylim([0.0, 1.05])
    ax_pr.set_xlim([0.0, 1.0])
    ax_pr.grid(linestyle='--', alpha=0.7)
    ax_pr.legend(fontsize="small", loc="lower left")

    # (b) ROC Curvesax_roc = axes[1]
    for fpr, tpr, label in roc_top:
        disp = RocCurveDisplay(fpr=fpr, tpr=tpr)
        disp.plot(ax=ax_roc, name=label, lw=1)
    ax_roc.set_title("ROC Curves (Top 10)")
    ax_roc.set_xlim([0.0, 1.0])
    ax_roc.set_ylim([0.0, 1.05])
    ax_roc.grid(linestyle='--', alpha=0.7)
    ax_roc.legend(fontsize="small", loc="lower right")

    # (c) Confusion Matrix at Optimal Threshold
    if hasattr(best_estimator, "predict_proba"):
        y_proba_best = best_estimator.predict_proba(X_test)[:,
1]
    else:
        y_proba_best = best_estimator.decision_function(X_test)

    best_thresh, best_fbeta = find_best_threshold(y_test,
y_proba_best, beta=2)
    y_pred_best_thresh = (y_proba_best >=
best_thresh).astype(int)

    prec_at_thresh = precision_score(y_test, y_pred_best_thresh,
zero_division=0)
    rec_at_thresh  = recall_score(y_test, y_pred_best_thresh,
zero_division=0)
    mcc_at_thresh  = matthews_corrcoef(y_test,
y_pred_best_thresh)
    acc_at_thresh  = accuracy_score(y_test, y_pred_best_thresh)

    cm = confusion_matrix(y_test, y_pred_best_thresh)
    ax_cm = axes[2]
    disp_cm = ConfusionMatrixDisplay(
        confusion_matrix=cm,
        display_labels=["Legit", "Fraud"]
    )
    disp_cm.plot(ax=ax_cm, cmap="Blues", colorbar=False)
    ax_cm.set_title(
        f"Confusion Matrix\n(threshold={best_thresh:.3f})\n"
        f"Acc={acc_at_thresh:.3f}, Prec={prec_at_thresh:.3f}, "
```

```
        f"Rec={rec_at_thresh:.3f}, F2={best_fbeta:.3f},
MCC={mcc_at_thresh:.3f}"
    )
    plt.tight_layout()
    if save_path:
        plt.savefig(save_path, dpi=300)
        print(f"Saved combined figure to {save_path}")
    else:
        plt.show()

# ------------------------------------------------------------------
-------------
# 10. Retrain & Plot for Overall Best Combination
# ------------------------------------------------------------------
-------------

best_combo = results_df.iloc[0]
print(f"\nBest overall → {best_combo['model'].upper()} |
{best_combo['sampler']} (AP={best_combo['AP']:.3f})")

best_sampler = resamplers[best_combo["sampler"]]
best_clf     = classifiers[best_combo["model"]]

pipe_steps_best = []
if best_sampler is not None:
    pipe_steps_best.append(("sampler", best_sampler))
pipe_steps_best += [("preproc", preprocessor), ("clf",
best_clf)]
best_pipe = Pipeline(pipe_steps_best)

search_best = GridSearchCV(
    best_pipe,
    param_grid=param_grids[best_combo["model"]],
    scoring="average_precision",
    cv=cv,
    n_jobs=N_JOBS,
    verbose=0,
    refit=True
)
search_best.fit(X_train, y_train)
best_pipe = search_best.best_estimator_
# Plot PR, ROC (top 10) and Confusion Matrix for best model
output_filepath = "images/all_graphs_lr.png"
plot_all_graphs(
    pr_curves_store,
    roc_curves_store,
```

```
    best_pipe,
    X_test,
    y_test,
    top_n=10,
    save_path=output_filepath
)

# ------------------------------------------------------------
-------------
# 11. Feature Importance for RF / XGB (if applicable)
# ------------------------------------------------------------
-------------

best_model = best_pipe.named_steps["clf"]
if hasattr(best_model, "feature_importances_"):
    importances = best_model.feature_importances_
    feat_imp_df = pd.DataFrame({
        "feature": all_features,
        "importance": importances
    }).sort_values("importance", ascending=False).head(10)
    print("\nTop 10 feature importances:")
    print(feat_imp_df.to_string(index=False))

    # Plot feature importances
    plt.figure(figsize=(8, 6))
    plt.barh(feat_imp_df["feature"][::-1],
feat_imp_df["importance"][::-1])
    plt.title("Top 10 Feature Importances")
    plt.xlabel("Importance")
    plt.tight_layout()
    plt.savefig("images/feature_importances_lr.png", dpi=300)
    print("Saved feature importances to
images/feature_importances_lr.png")

print("\nPipeline execution completed.")
```