КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій

Кафедра технологій управління

Спеціальність 122 – Комп'ютерні науки,
освітня програма «Інформаційна аналітика та впливи»

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему:

" Розробка методу інтелектуального аналізу та прогнозування відтоку клієнтів**"**

Студента 2–го курсу групи ІАВ–21

Прохорчука Дмитра Сергійовича
(прізвище, ім'я, по батькові)

(підпис студента)

Науковий керівник:

Зарицький Олег Володимирович
(науковий ступінь, вчене звання)

(прізвище, ім'я, по батькові)

(дата) (підпис)

Попередній захист:

(Висновок: «До захисту в Екзаменаційній комісії»)

Завідувач кафедри
технологій управління (підпис) (прізвище, ініціали) (дата)

Київ – 2025

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій

Кафедра технологій управління
Освітньо–кваліфікаційний рівень Магістр
Спеціальність 122 – Комп'ютерні науки
Освітня програма Інформаційна аналітика та впливи

ЗАТВЕРДЖУЮ
Завідувач кафедри
Професор Віктор МОРОЗОВ
_____
«____» _____2025 року

З А В Д А Н Н Я
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Студент Прохорчук Дмитро Сергійович

Група ІАВ–21

1. Тема кваліфікаційної роботи

Розробка методу інтелектуального аналізу та прогнозування відтоку клієнтів

Затверджена наказом по від «___»_____20___р. № ___.

2. Строк подання студентом готової роботи – "19" травня 2025р.

3. Цільова установка та вихідні дані до роботи

Дослідження методів інтелектуального аналізу аналізу та прогнозування відтоку клієнтів, порівняти точність їх роботи, розробка власного методу прогнозування та описати його впровадження у виробниче середовище.

4. Зміст роботи

Робота складається зі вступу, огляду літератури та існуючих рішень у досліджуваній сфері. У роботі було формалізовано поставлену задачу та проаналізовано предметну область, описано математичні моделі що використовуються або розробляються. Основний розділ присвячений детальній розробці запропонованого підходу та подальшій оцінці його ефективності. Також робота містить опис технологій розробки та впровадження отриманих результатів у практику. Завершують дослідження узагальнюючі висновки, повний список використаних наукових джерел та, за потреби, додатки з допоміжними матеріалами.

5. Перелік графічного матеріалу (слайдів)

18 рисунків , 16 слайдів презентації доповіді

6. Календарний план виконання роботи:

№ п/п	Назва частин роботи	%	Виконання роботи	
			За планом	Фактично
1.	Вибір теми дипломної роботи	3	01.10.24	01.10.24
2.	Протокол кафедри ТУ про затвердження тем дипломних робіт та призначення наукових керівників	2	27.12.24	27.12.24
3.	Формування переліку нормативних матеріалів, літератури з проблематики дипломної роботи	10	08.01.25	07.01.25
4.	Складання розгорнутого плану кваліфікаційної роботи	5	18.01.25	18.01.25
5.	Ознайомлення наукового керівника з розгорнутим планом кваліфікаційної роботи. Внесення змін.	5	19.01.25	19.01.25
6.	Підготовка розділу 1 «Основні теоретичні поняття і підходи до аналізу й прогнозування відтоку клієнтів »	10	12.02.25	13.02.25
7.	Підготовка розділу 2 «Моделі алгоритмів та методи для вирішення задачі»	14	08.03.25	08.03.25
8.	Підготовка розділу 3 «Методологія, практична реалізація та експериментальне дослідження методу відтоку клієнтів»	14	01.04.25	01.04.25
9.	Підготовка розділу 4 «Практичне застосування, автоматизація та перспективи розвитку методу прогнозування відтоку»	13	07.04.25	07.04.25
10.	Оформлення кваліфікаційної роботи. Підготовка висновків і пропозицій	15	10.04.25	10.04.25
11.	Передача кваліфікаційної роботи науковому керівникові	2	11.04.25	11.04.25
12.	Передача кваліфікаційної роботи рецензенту для рецензування	2	01.05.25	01.05.25
13.	Попередній захист кваліфікаційної роботи	5	14.05.25	14.05.25

Дата видачі завдання «____» _____20___р.

Керівник роботи: д.т.н., доцент кафедри технологій управління Зарицький Олег Володимирович

_____
(підпис)

Завдання прийняв до виконання студент групи ІАВ–21 Прохорчук Дмитро Сергійович

_____
(підпис)

ЗМІСТ

Перелік умовних скорочень ... 9
ВСТУП ... 10
РОЗДІЛ 1 ... 14
ОСНОВНІ ТЕОРЕТИЧНІ ПОНЯТТЯ ТА ПІДХОДИ ДО АНАЛІЗУ Й ПРОГНОЗУВАННЯ ВІДТОКУ КЛІЄНТІВ ... 14
1.1 Визначення термінів ... 14
1.2 Проблема відтоку клієнтів: сутність, масштаби та значення для бізнесу ... 16
1.2.1 Актуальність та масштаби проблеми ... 16
1.2.2 Основні причини та типи відтоку ... 17
1.2.3 Важливість прогнозування відтоку для прийняття управлінських рішень ... 19
1.3 Роль інтелектуального аналізу даних у вирішенні проблеми відтоку клієнтів ... 23
1.3.1 Обмеження традиційних підходів та переваги застосування Data Science для розуміння та управління відтоком ... 23
1.3.2 Основні завдання інтелектуального аналізу даних в контексті прогнозування та управління відтоком клієнтів ... 25
1.4 Аналіз іноземних та вітчизняних науково-інформаційних джерел ... 26
1.4.1 Огляд ключових наукових публікацій та еволюція підходів до прогнозування відтоку ... 27
1.4.2 Сучасні тенденції та відкриті питання у дослідженнях прогнозування відток ... 29
1.5 Постановка завдання ... 30
1.6 Висновки ... 31
РОЗДІЛ 2 ... 33
МОДЕЛІ АЛГОРИТМІВ ТА МЕТОДИ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ ... 33
2.1 Класифікація методів прогнозування відтоку клієнтів ... 33
2.2 Описові та діагностичні методи аналізу поведінки клієнтів ... 34
2.2.1 Когортний аналіз для дослідження динаміки утримання ... 34
2.2.2 RFM-аналіз для сегментації клієнтів за активністю та цінністю ... 36
2.3 Статистичні моделі прогнозування ... 39
2.3.1 Логістична регресія ... 39
2.3.2 Варіації логістичної регресії для аналізу відтоку ... 40
2.3.3 Моделі вживаності ... 41
2.4.2 Ансамблеві методи на основі дерев рішень ... 44
1.4.2 Метод опорних векторів ... 47
1.4.3 Нейронні мережі ... 48
2.5 Вибір алгоритмів прогнозування відтоку ... 49
2.6 Висновки ... 51
РОЗДІЛ 3 ... 52
МЕТОДОЛОГІЯ, ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ МЕТОДУ ПРОГНОЗУВАННЯ ВІДТОКУ КЛІЄНТІВ ... 52
3.1 Методологія проведення експериментального дослідження. ... 52
3.2 Опис алгоритму раннього прогнозування відтоку та методології його оцінки ... 55
3.3 Опис технологій ... 59
3.4 Характеристика набору даних та дослідницький аналіз ... 61
3.5 Побудова ознак ... 66
3.6 Навчання моделі ... 69

3.7 Оцінка натренованих моделей 72
3.8 Висновки 75
РОЗДІЛ 4 77
4.1 Концепція практичного застосування розробленого методу «Early churn Alert» 77
4.2 Концепція розгортання інфраструктури алгоритму в Google Cloud 79
4.3 Перспективи подальших досліджень 82
4.4 Висновки 84
ЗАГАЛЬНІ ВИСНОВКИ 86
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 88
ДОДАТОК А 92

АНОТАЦІЯ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра технологій управління
Спеціальність 122 –
Комп'ютерні науки,
освітня програма "Інформаційна аналітика та впливи"

Дипломна робота магістра Прохорчука Дмитра Сергійович.

Тема роботи – «Розробка методу інтелектуального аналізу та прогнозування відтоку клієнтів».

Мета дипломної роботи магістра – підвищити рівень утримання клієнтів за допомогою методів інтелектуального аналізу та прогнозування відтоку.

Об'єкт дослідження – процес відтоку клієнтів.

Предмет дослідження – методи інтелектуального аналізу та прогнозування, спрямовані на прогнозування цього процесу.

Наукова новизна роботи – запропоновано комбіновану метрику втрат, яка зважує бізнес-вартість хибних позитивів і негативів, а також авторський алгоритм, котрий поєднує графовий аналіз взаємодій із градієнтним бустингом і забезпечує підвищення ROC-AUC на 10-15% порівняно з класичними baseline-моделями. Уперше продемонстровано застосування SHAP-цінностей на рівні окремих транзакцій для формування персоналізованих рекомендацій зі втручання.

Практичне значення одержаних результатів:

Реалізація запропонованої системи у виробничому середовищі як наслідок дає змогу:

– застосування розробленого методу дозволяє компаніям здійснювати більш точне та, що важливо, раннє ранжування клієнтської бази за

індивідуальним рівнем ризику відтоку, що створює можливості для своєчасного та ефективного превентивного втручання;

– можливість чіткої концентрації зусиль на найбільш ризикові сегменти клієнтської бази дозволяє уникнути невиправданих витрат на масові промо-акції, що охоплюють лояльних клієнтів.;

– впровадження розробленого алгоритму, що інтегрує графові ознаки взаємодій, сприяє ідентифікації неочевидних шаблонів поведінки, які можуть передувати відтоку, тим самим підвищуючи ефективність персоналізованих retention-кампаній.

– використання запропонованої комбінованої економічної метрики надає бізнесу інструмент для оцінки та вибору прогностичних моделей не лише за стандартними метриками точності, але й з урахуванням реальної економічної вартості помилок, що веде до більш обґрунтованих управлінських рішень.

– отримувати інтерпретовані висновки, які підтримують рішення менеджерів із лояльності та дозволяють оптимізувати пропозиції для клієнтів.

– здатність методу генерувати інтерпретовані висновки щодо ключових факторів ризику надає цінну інформаційну підтримку (наприклад, за допомогою SHAP-значень для моделі LightGBM), дозволяючи оптимізувати продуктові пропозиції та комунікаційні стратегії.

– розроблений підхід створює методологічну основу для побудови автоматизованих систем управління відтоком, які можуть суттєво підвищити оперативність реагування на зміни в поведінці клієнтів та оптимізувати використання маркетингових бюджетів.

Кваліфікаційна робота магістра складається зі вступу, основної частини, яка включає чотири розділи, висновків та списку використаних джерел. Всього налічує 124 сторінки та перелік посилань з джерел.

Ключові слова: клієнтський відтік, прогнозування відтоку, інтелектуальний аналіз даних, графовий аналіз, градієнтний бустинг, бізнес-метрики, Data Science.

Перелік умовних скорочень

CLV – Customer Lifetime Value

CAC – Customer Acquisition Cost

RR – Retention Rate

CR – Churn Rate

У.г.о – умовно грошова одиниця

Churn – Клієнтський відтік

LSTM – Long Short Term Memory

GRU – Gated Recurent Units

GBDT – Gradient Boosting Decision Trees

GDPR – General Data Protection Regulation

LGBM – Light Gradient Boosted Machine

CRM – Customer Relationship Management

NN – Neural Networks

ВСТУП

За останнє десятиліття витрати на залучення клієнтів у більшості споживчих сфер збільшились втричі, водночас темпи приросту нових клієнтів сповільнилися через перенасичення ринку та зростання конкуренції. В умовах високого рівня конкуренції важливо завчасно передбачити ризики втрати клієнтів, аналізувати причини їх відтоку та розробляти відповідні запобіжні заходи.

Розвиток «економіки за підпискою» (subscription economy) та перехід багатьох організацій та компаній моделей регулярний доходів актуалізує питання довгострокового утримання аудиторії, адже щомісячний відтік на рівні лише 7 % перешкоджає досягненню компаній «роботи в плюс».

Дані консалтингових компаній McKinsey і Bain показують - збільшення показника утримання клієнтів(retention rate) лише на 5% здатне підвищити операційний прибуток на 25–95%, а зниження показника відтоку(churn rate) на 1% у на прикладі відомої компанії зі сфери надання телекомунікаційних послуг еквівалентне економії маркетингових витрат у сотні мільйонів доларів на рік.

Разом з тим зростають очікування клієнтів щодо персоналізації надання товарів, послуг та інтегрованої багатоканальної системи взаємодії. Персональні алгоритмічні рекомендації та push-нотифікації стають буденністю, тому компанії з повільними або неінтелектуальними системами реагування на поведінкові сигнали ризикують втратити лояльність клієнтів до того, як традиційні показники ефективності (KPI) зафіксують негативні тенденції. Крім того, нові регуляторні норми (GDPR, CCPA) обмежують агресивні стратегії acquisition, зумовлюючи перехід бізнесу ціннісно-орієнтованого шляху утримання аудиторії.

В цьому контексті інтелектуальні методи до прогнозування відтоку клієнтів набувають стратегічного значення. Застосування методів машинного навчання й глибоких нейронних мереж дозволяють виявляти складні нелінійні

шаблони у великих масивах транзакційних і поведінкових даних та передбачати потенційні точки відтоку задовго до їхньої фактичної появи та запускати превентивні, персоналізовані кампанії спрямовані на утримання.

Відсутність рішень на розробку та впровадження таких систем в бізнес складову означає не лише втрату виручки, а й погіршення репутації компанії, яку дедалі складніше відновлювати у цифрову епоху миттєвих відгуків і соціальних мереж. Таким чином, розробка точного та зрозумілого для бізнесу методу прогнозування відтоку клінтів є вкрай актуальним завданням, здатним безпосередньо вплинути на фінансову стабільність і конкурентоспроможність компанії навіть за умов динамічного середовища високих маркетингових витрат.

Мета роботи – підвищити рівень утримання клієнтів за допомогою методів інтелектуального аналізу та прогнозування відтоку. Для досягнення цієї мети необхідно вирішити такі завдання:

- Проаналізувати сучасні наукові й прикладні підходи до вимірювання та прогнозування клієнтського відтоку, виокремити їх переваги та обмеження.
- Сформулювати бізнес-вимоги до системи прогнозування.
- Обрати методологію Data Science та методи проведення дослідження.
- Обрати технології для проведення дослідження.
- Зібрати, очистити та уніфікувати дані для дослідження.
- Розробити багаторівневу схему формування ознак: від простих RFM-показників до графових взаємодій і часових агрегатів.
- Побудувати, порівняти та проаналізувати кілька моделей машинного та глибокого навчання, визначивши оптимальний баланс точності та інтерпретованості.
- Оцінити економічний ефект від впровадження найкращої версії моделі.

- Провести експериментальне дослідження та оцінити ефективність розроблених алгоритмів.

Об’єкт дослідження – процес клієнтського відтоку. Предмет дослідження – методи Data Science та алгоритми машинного навчання Learning для прогнозування ймовірності відтоку.

Предмет дослідження – методи інтелектуального аналізу та прогнозування, спрямовані на прогнозування цього процесу.

Методи дослідження:

1. Аналіз літературних та галузевих джерел:

Проведено систематичний огляд наукових статей, публікацій та звітів, присвячених прогнозуванню, методів Data Science та алгоритмів машинного навчання та клієнтській аналітиці. Цей метод допоміг окреслити сучасний стан досліджень та виявити прогалини, обґрунтовують наукову новизну роботи.

2. Експериментальний аналіз на датасеті KKBox:

Для оцінки працездатності та порівняння моделей машинного навчання було використано публічний набір даних KKBox. Даний набір даних містить близько 21 млн користувацьких транзакцій та близько 400 млн записів поведінкових подій користувачів. Додаткові файли набору даних містять в собі деякі демографічні ознаки, які дозволяють побудувати повноцінний набір ознак для ефективного прогнозування. Обсяг та різноманітність обраного набору даних має забезпечити коректну перевірку алгоритмів прогнозування відтоку.

3. Моделювання та оптимізація алгоритмів машинного навчання:

Реалізовано етап моделювання та оптимізацію розроблених моделей машинного навчання для прогнозування відтоку аналізу, включаючи підбір та налаштування гіперпараметрів для досягнення оптимальної точності та стабільності прогнозу. Було проведено статистичний аналіз та проведено оцінку ефективності кожної з розроблених моделей.

4. Програмна реалізація та тестування:

Використано мови та технології програмування для розроблення алгоритмів, включаючи популярні бібліотеки та інструментів для роботи з даними та машинним навчанням.

У роботі будуть використані такі технології:

- Python
- Pandas
- Numpy
- Matplotlib
- Seaborn
- Scikit-learn
- LGBM-boost

Наукова новизна роботи – в роботі запропоновано нову композитну функцію витрат, яка поєднує крос-ентропію з економічним коефіцієнтом ваги хибних рішень, що дозволяє калібрувати модель під конкретні бізнес-пріоритети.

Крім того, розроблено гібридний алгоритм Early-Churn Alert, що інтегрує графові ембеддинги соціальної взаємодії з градієнтним бустингом, забезпечуючи приріст показника ROC-AUC на 10-15% порівняно з класичними моделями.

Практичне значення одержаних результатів:

Запропонована система забезпечує щоденне ранжування клієнтів за ризиком відтоку, автоматичний запуск персоналізованих кампаній утримання та прогнозування очікуваного економічного ефекту кожної інтервенції для різних бізнес-сегментів, що робить її корисною для впровадження у сферах стримінгових сервісів, сфері телекомунікаційних послуг, фінансових установах та SaaS-платформах.

РОЗДІЛ 1
ОСНОВНІ ТЕОРЕТИЧНІ ПОНЯТТЯ ТА ПІДХОДИ ДО АНАЛІЗУ Й ПРОГНОЗУВАННЯ ВІДТОКУ КЛІЄНТІВ

1.1 Визначення термінів

Відтік клієнтів – факт припинення клієнтом економічно значущої взаємодії з компанією протягом визначеного періоду спостереження (наприклад, 90 днів без покупок чи авторизацій). Характер "економічно значущої взаємодії" може варіюватися залежно від бізнес-моделі компанії: для e-commerce це може бути відсутність покупок (наприклад, протягом 90 днів), для SaaS-сервісів – скасування підписки або тривала відсутність авторизацій та використання ключових функцій, для медіа-платформ – припинення споживання контенту.

Важливо розрізняти:

- Волюнтативний відтік: клієнт свідомо приймає рішення припинити користуватися послугами/товарами компанії.
- Неволюнтативний відтік**:** припинення взаємодії відбувається з причин, що не залежать від прямого рішення клієнта (наприклад, несправність платіжної картки, зміна місця проживання, недоступність сервісу).

Коефіцієнт відтоку – це ключовий показник, що відображає частку користувачів, що припинили взаємодію з сервісом впродовж визначеного періодувідносно загальної чисельності активних клієнтів на початку періоду. Базова формула коефіцієнту відтоку визначається як:

$CR = 100\% * \frac{LC}{AC}$, де LC – кількість клієнтів, що пішли у відтік протягом визначеного періоду, а AC– загальна кількість активних клієнтів на початок періоду

Коефіцієнт утримання – здатність компанії зберігати клієнтів протягом часу; метрика, обернена до відтоку. Формула коефіцієнту утримання визначається як:

$$RR = 1 - CR$$

Довічна цінність клієнта (Customer Lifetime Value, CLV) – це сукупний прибуток, який компанія очікує отримати від одного клієнта за весь час його взаємодії з компанією. CLV є важливим показником для оцінки прибутковості клієнтів та ефективності маркетингових інвестицій.

Вартість залучення клієнта (Customer Acquisition Cost, CAC) – це сукупні витрати компанії на маркетинг, продажі та інші пов'язані активності, необхідні для залучення одного нового клієнта протягом певного періоду. Розраховується шляхом ділення загальних витрат на залучення на кількість нових клієнтів, отриманих за той самий період, визначається як:

$CAC = \frac{MCC+W+S+PS+O}{CA}$, де MCC – загальна кількість коштів витрачених на рекламу, W- витрати на маркетинг, S – витрати на ПЗ, O – інші витрати, PS – витрати на зарплату співробітників, CA – кількість залучених клієнтів.

Інтелектуальний аналіз даних– це процес виявлення в "сирих" даних раніше невідомих, нетривіальних, практично корисних та доступних для інтерпретації знань (шаблонів, закономірностей, тенденцій), необхідних для прийняття рішень у різних сферах людської діяльності.

Клієнтська аналітика – процес збору, аналізу та інтерпретації даних про клієнтів для отримання глибшого розуміння їхньої поведінки, потреб та уподобань. Метою клієнтської аналітики є покращення взаємодії з клієнтами, підвищення їх лояльності та оптимізація бізнес-рішень.

Сегментація клієнтів – процес поділу клієнтської бази на окремі групи за певними спільними характеристиками, такими як демографічні дані, поведінка, потреби або цінність для компанії. Сегментація дозволяє компаніям

розробляти більш цільові та ефективні маркетингові стратегії та програми утримання.

1.2 Проблема відтоку клієнтів: сутність, масштаби та значення для бізнесу

1.2.1 Актуальність та масштаби проблеми

Незважаючи на те, що стратегії залучення нових клієнтів традиційно займають значне місце в маркетингових бюджетах компаній, сучасні дослідження та бізнес-практика все частіше підкреслюють критичну важливість мінімізації відтоку існуючої клієнтської бази для забезпечення довгострокової прибутковості та сталого розвитку. Актуальність цієї проблеми зумовлена не лише прямими фінансовими втратами від клієнтів, що йдуть, але й низкою опосередкованих негативних ефектів, які можуть мати кумулятивний вплив на загальну ефективність діяльності підприємства.

Одним із ключових економічних наслідків відтоку є втрата майбутніх потоків доходу. Кожен клієнт, що припиняє взаємодію, забирає з собою не лише кошти від незавершених транзакцій, але й весь потенційний дохід, який він міг би генерувати протягом свого життєвого циклу. Для бізнесів з моделями регулярних платежів, таких як SaaS-сервіси або медіа-платформи з підпискою, навіть незначне збільшення місячного коефіцієнта відтоку може призвести до суттєвого недоотримання прибутку в довгостроковій перспективі та ускладнити досягнення точки беззбитковості.

Іншим важливим аспектом є неефективність інвестицій у залучення. Витрати на залучення нового клієнта, як правило, значно перевищують витрати на утримання існуючого. Коли клієнт йде, особливо на ранніх етапах взаємодії, компанія не встигає окупити початкові інвестиції в маркетинг, продажі та онбординг, що робить ці витрати марними. Високий відтік, таким чином, напряму збільшує середній CAC та знижує загальну рентабельність маркетингових інвестицій.

Окрім прямих фінансових втрат, відтік клієнтів спричиняє й нефінансові негативні наслідки**:**

- Репутаційні ризики: Незадоволені клієнти, що пішли, часто діляться своїм негативним досвідом у соціальних мережах, на форумах та сайтах з відгуками. Це може завдати значної шкоди репутації бренду, відлякати потенційних нових клієнтів та ускладнити зусилля із залучення. В епоху цифрових комунікацій швидкість поширення негативної інформації є надзвичайно високою.
- Втрата «адвокатів бренду»: Лояльні клієнти часто виступають «адвокатами бренду», рекомендуючи його своїм знайомим та колегам. При відтоку компанія втрачає не лише клієнта, але й потенційний канал органічного залучення нових.
- Втрата цінних даних та зворотного зв'язку: Клієнти, що йдуть, забирають із собою цінну інформацію про свої потреби, досвід взаємодії з продуктом та причини незадоволення. Втрата цього зворотного зв'язку ускладнює для компанії процес вдосконалення продуктів та послуг.

1.2.2 Основні причини та типи відтоку

Для ефективного управління клієнтською базою та розробки дієвих стратегій утримання критично важливо розуміти не лише факт відтоку, але й його фундаментальні причини та різновиди. Класифікація типів відтоку та факторів, що його спричиняють, дозволяє компаніям точніше діагностувати проблеми та обирати адекватні інструменти реагування.

Як зазначалося раніше, відтік клієнтів прийнято поділяти на два основні типи:

1. Волюнтативний відтік: Цей тип відтоку відбувається, коли клієнт свідомо та за власною ініціативою приймає рішення припинити користування продуктами чи послугами компанії. Це найбільш поширений тип, на який компанії можуть і повинні активно впливати. Причини такого відтоку є різноманітними і можуть бути зумовлені комплексом факторів. Серед них

важливе місце посідають цінові аспекти: клієнти можуть вважати ціну завищеною порівняно з цінністю, яку вони отримують, або знайти вигідніші пропозиції у конкурентів. Непрозоре ціноутворення чи несподіване підвищення цін також часто стають каталізаторами рішення про відхід. Наприклад, клієнт мобільного оператора може перейти до конкурента, який пропонує аналогічний пакет послуг за нижчою ціною. Не менш значущими є продуктові фактори. Якість самого продукту чи послуги відіграє вирішальну роль, і незадоволеність може бути викликана низькою надійністю, недостатньою функціональністю, складним користувацьким інтерфейсом або застарілістю технологій. Так, користувач онлайн-сервісу для редагування відео може відмовитися від підписки через повільну роботу програми та часті «зависання». Тісно пов'язані з цим сервісні фактори, адже рівень обслуговування та підтримки клієнтів суттєво впливає на їх лояльність. Негативний досвід взаємодії зі службою підтримки, некомпетентність персоналу або нездатність оперативно вирішити проблему можуть підштовхнути клієнта до відтоку. Уявімо клієнта інтернет-провайдера, який багаторазово звертався зі скаргою на низьку швидкість інтернету, але його проблема не була вирішена задовільно. Активність конкурентів на ринку також може стимулювати відтік. Поява більш привабливих пропозицій, інноваційних рішень або ефективніші маркетингові кампанії конкурентів здатні переконати клієнта змінити постачальника. Наприклад, користувач стрімінгового сервісу може перейти на нову платформу, що пропонує ексклюзивний контент за порівнянну ціну. Нарешті, не варто забувати про зміну потреб або обставин самого клієнта: зміна інтересів, фінансового стану чи життєвих обставин можуть зробити поточний продукт або послугу менш актуальними.

2. Неволюнтативний відтік. На відміну від волюнтативного, відбувається з причин, які не залежать від активного бажання клієнта припинити користування послугою. Хоча він може бути менш поширеним,

його також важливо відстежувати та мінімізувати, адже часто він пов'язаний з технічними або адміністративними проблемами. До таких причин належать, наприклад, проблеми з оплатою, технічні або логістичні перешкоди або адміністративні причини.

Окрім повного припинення взаємодії, варто також виділяти частковий відтік. Це ситуація, коли клієнт не повністю відмовляється від послуг компанії, але суттєво знижує рівень їх споживання: наприклад, переходить на дешевший тарифний план, зменшує частоту або обсяг покупок. Хоча клієнт формально залишається, компанія втрачає частину доходу від нього, і такий стан часто може бути проміжною фазою перед повним волюнтативним відтоком.

Важливо зазначити, що на практиці відтік клієнта рідко буває спричинений однією єдиною причиною. Найчастіше це результат накопичення декількох негативних факторів або невдалого досвіду взаємодії з компанією на різних етапах життєвого циклу. Глибоке розуміння та ідентифікація домінуючих причин відтоку, специфічних для конкретного бізнесу, його продуктів та цільових сегментів аудиторії, є фундаментальною передумовою для розробки ефективних та економічно обґрунтованих стратегій утримання.

Інтелектуальний аналіз даних дозволяє не тільки виявляти ці причини, але й прогнозувати ризики на індивідуальному рівні, що відкриває шлях до проактивного управління клієнтською лояльністю.

1.2.3 Важливість прогнозування відтоку для прийняття управлінських рішень

Точне та своєчасне прогнозування відтоку клієнтів відіграє критичну роль у формуванні ефективних стратегій утримання та дозволяє компаніям приймати обґрунтовані управлінські рішення, спрямовані на мінімізацію втрат та максимізацію довічної цінності клієнтів (CLV). У той час як більшість сервіс-орієнтованих компаній традиційно приділяють значну увагу залученню нових користувачів, довгострокове економічне зростання та стабільність

безпосередньо залежать від здатності мінімізувати відтік існуючої клієнтської бази. Якщо протидія відтоку не ведеться проактивно, із застосуванням аналітичних інструментів, навіть найперспективніший продукт чи послуга ризикують не досягти свого ринкового потенціалу та фінансових цілей.

Ефективне прогнозування відтоку клієнтів слугує фундаментом для реалізації та підвищення дієвості ключових стратегій утримання. Варто розглянути, як саме прогностичні моделі підсилюють основні напрямки роботи з клієнтською базою, що виокремлюються у практиці провідних платформ.

1. Функціональне й контентне удосконалення продукту**:** Аналіз факторів, що найчастіше передують відтоку (виявлених за допомогою прогностичних моделей та методів інтерпретації, таких як SHAP), дозволяє ідентифікувати конкретні слабкі місця продукту, контенту чи користувацького досвіду. Наприклад, якщо модель показує, що клієнти, які стикаються з труднощами при використанні певної важливої функції або рідко її застосовують, мають вищий ризик відтоку, компанія може сфокусувати зусилля на покращенні юзабіліті цієї функції, розробці навчальних матеріалів або її активнішій промоції.

2. Цільові кампанії залучення до продукту: Замість загальних масових кампаній, які часто мають низьку конверсію, прогнозування дозволяє виділити сегменти клієнтів, які недостатньо активно взаємодіють з продуктом, не використовують його ключові можливості або не усвідомлюють його повної цінності. Для таких сегментів можна розробляти таргетований освітній контент, персоналізовані гайди, навчальні вебінари або спеціальні пропозиції, що стимулюватимуть глибше використання продукту та підвищать його сприйману цінність.

3. Проактивна підтримка та персоналізований онбординг: Моделі прогнозування відтоку можуть ідентифікувати клієнтів (як нових, на етапі онбордингу, так і існуючих) які демонструють ранні поведінкові сигнали

потенційного ризику (наприклад, низька активність після реєстрації, падіння частоти використання сервісу, негативні відгуки у минулому). Це дозволяє службі підтримки або менеджерам по роботі з клієнтами проактивно контактувати з такими користувачами, запропонувати допомогу, зібрати зворотний зв'язок та вирішити потенційні проблеми до того, як вони стануть критичними та призведуть до відтоку.

4. Адаптивне ціноутворення та управління цінністю клієнта: Для сегментів клієнтів з високим ризиком відтоку, які, за даними моделі, є чутливими до ціни, прогностична система може сигналізувати про необхідність застосування більш гнучкого підходу. Це може включати пропозицію індивідуальних знижок, тимчасове переведення на спеціальні тарифні плани, надання бонусів або пропозицію альтернативних, менш дорогих продуктів чи пакетів послуг. Метою таких дій є збереження клієнта, навіть якщо це означає тимчасове зниження поточного доходу від нього, але дозволяє зберегти його довгострокову цінність (CLV) та уникнути витрат на залучення нового клієнта на його місце.

5. Оптимізація каналів залучення та маркетингових витрат: Аналізуючи історичні дані про відтік клієнтів у розрізі каналів, через які вони були залучені, та співвідносячи цю інформацію з прогнозами ймовірності відтоку для клієнтів з різних джерел, компанія може значно ефективніше розподіляти маркетинговий бюджет. Це дозволяє інвестувати більше в ті канали, які стабільно приводять найбільш лояльних та довгострокових клієнтів з низьким ризиком відтоку, та скорочувати витрати на менш ефективні джерела.

Перелічені стратегії мають комплементарний характер, і їх пріоритетність може змінюватися залежно від фази життєвого циклу продукту та специфіки бізнесу. Однак, на кожному етапі, кожна стратегія формує власний набір точок прийняття рішення (наприклад, вибір оптимальної

A/B-версії функціоналу, визначення цільових сегментів для комунікацій, ідентифікація клієнтів для превентивного контакту). Щоб ці рішення були економічно обґрунтованими та ефективними, необхідна наявність достовірної моделі оцінювання ймовірності відтоку та здатності належним чином інтерпретувати її результати.

Таким чином, впровадження та використання надійної системи прогнозування відтоку надає компаніям низку суттєвих переваг:

- Своєчасна ідентифікація клієнтів групи ризику**:** Можливість з високою точністю виявляти користувачів, які з високою ймовірністю припинять взаємодію з компанією у найближчому майбутньому, дозволяючи сфокусувати на них зусилля з утримання.
- Персоналізація заходів з утримання: Здатність розробляти та застосовувати індивідуальні пропозиції, комунікації та програми лояльності, адаптовані до потреб та ризиків конкретних сегментів або навіть окремих клієнтів.
- Оптимізація ресурсів та маркетингових бюджетів: Можливість спрямовувати зусилля та фінансові ресурси на утримання найцінніших для бізнесу клієнтів або тих, чий відтік можна реально попередити з найменшими витратами, уникаючи невиправданих витрат на масові нецільові кампанії.
- Об'єктивна оцінка ефективності retention-кампаній**:** Наявність інструментів для аналізу результативності вжитих заходів щодо зниження фактичного рівня відтоку порівняно з прогнозованим, що дозволяє постійно вдосконалювати стратегії утримання.
- Системне покращення продукту та сервісу: Можливість використовувати дані про ключові фактори та патерни поведінки, що корелюють з відтоком, для внесення стратегічних змін у продукт, процеси обслуговування, цінову політику або користувацький досвід.

- Здобуття суттєвої конкурентної переваги: Швидше та ефективніше реагування на зміни настроїв клієнтів, пропонуючи кращий досвід та умови, ніж конкуренти.

Можна зробити висновок що ефективність описаних стратегій та загальна успішність бізнесу в конкурентному середовищі значною мірою визначається якістю підтримки рішень на основі даних. Це, в свою чергу, зумовлює нагальну потребу у побудові точних, інтерпретованих та інтегрованих у бізнес-процеси систем прогнозування ризику відтоку клієнтів.

1.3 Роль інтелектуального аналізу даних у вирішенні проблеми відтоку клієнтів

1.3.1 Обмеження традиційних підходів та переваги застосування Data Science для розуміння та управління відтоком

Традиційні підходи до аналізу відтоку клієнтів, що часто базуються на аналізі агрегованої звітності (наприклад, загальний коефіцієнт відтоку по компанії) або на інтуїтивних припущеннях співробітників компаній, сформованих на основі їхнього досвіду, мають низку суттєвих обмежень. Такі підходи, як правило, не дозволяють виявити складні, неочевидні фактори та їх комбінації, що впливають на рішення конкретного клієнта або цілих сегментів клієнтів про припинення взаємодії з компанією.

Більше того, традиційні методи часто виявляються неефективними при роботі з великими обсягами різноманітних даних, які сучасні бізнеси генерують у процесі своєї діяльності. Як наслідок, компанії не мають можливості побудувати точні індивідуальні прогнози ризику, що призводить до переважно реактивного характеру заходів з утримання: дії починаються вже після того, як клієнт виявив явні наміри піти або фактично пішов. Такий підхід значно знижує ефективність зусиль з утримання та збільшує пов'язані з цим витрати.

Натомість, застосування методів Data Science, зокрема інтелектуального аналізу даних та машинного навчання, пропонує низку вагомих переваг для побудови ефективної системи управління відтоком клієнтів:

1. Обробка великих та різноманітних наборів даних: Сучасні алгоритми та технології Data Science спеціально розроблені для роботи з великими обсягами даних з різноманітних джерел. Це включає як структуровані дані, так і неструктуровані або слабко структуровані дані. Здатність інтегрувати та аналізувати таку різноманітну інформацію дозволяє отримати більш повне та глибоке розуміння клієнта.

2. Виявлення складних та нелінійних залежностей: На відміну від класичних лінійних статистичних методів, багато алгоритмів машинного навчання здатні знаходити складні, нелінійні та інтерактивні шаблони у поведінці клієнтів. Ці патерни часто залишаються непоміченими при традиційному аналізі, але можуть бути ключовими індикаторами ризику відтоку.

3. Побудова високоточних прогностичних моделей**:** Алгоритми машинного навчання, навчені на достатніх обсягах релевантних історичних даних, дозволяють створювати предиктивні моделі, що з високою точністю прогнозують ймовірність відтоку для кожного окремого клієнта. Це дає можливість компанії ранжувати всю клієнтську базу за рівнем ризику та пріоритезувати зусилля з утримання.

4. Автоматизація процесів аналізу та прогнозування: Багато етапів, пов'язаних з управлінням відтоком – від підготовки даних та інженерії ознак до навчання моделей, генерації періодичних прогнозів та навіть ініціювання певних дій – можуть бути значною мірою автоматизовані. Це підвищує оперативність отримання результатів, знижує ймовірність помилок, пов'язаних із людським фактором.

5. Об'єктивність та обґрунтованість рішень на основі даних: Висновки щодо факторів ризику та прогнози відтоку базуються на кількісному аналізі емпіричних даних, а не лише на суб'єктивній інтуїції чи обмеженому досвіді співробітників. Такий підхід робить управлінські рішення щодо стратегій утримання більш зваженими, прозорими, вимірюваними та, як наслідок, більш ефективними.

Таким чином, перехід від традиційних методів аналізу до використання інструментарію Data Science є ключовим фактором для побудови сучасної та ефективної системи управління відтоком клієнтів, здатної забезпечити компанії значні конкурентні переваги.

1.3.2 Основні завдання інтелектуального аналізу даних в контексті прогнозування та управління відтоком клієнтів

Застосування інтелектуального аналізу даних дозволяє вирішувати цілий комплекс взаємопов'язаних завдань, спрямованих на всебічне розуміння та ефективне управління процесом відтоку клієнтів. Ці завдання охоплюють весь цикл роботи з даними – від виявлення закономірностей до впровадження проактивних заходів. Прогнозування ймовірності відтоку є одним з найбільш поширених завдань у Data Science**.** Результатом реалізації має бути скорингова оцінка ризику відтоку, що дозволяє пріоритезувати зусилля з утримання.

Окрім самого прогнозу, надзвичайно важливим є ідентифікація ключових факторів відтоку. Це завдання передбачає виявлення та кількісну оцінку основних ознак, подій або характеристик клієнтів та їхньої взаємодії з продуктом/послугою, які найсильніше корелюють з ризиком відтоку. Розуміння цих драйверів, часто за допомогою методів інтерпретації моделей машинного навчання (наприклад, аналіз важливості ознак, SHAP values, LIME), є критично важливим для розробки цілеспрямованих превентивних заходів та вдосконалення продукту чи сервісу.

Наступним логічним завданням **є** сегментація клієнтів для управління відтоком. Вона полягає у групуванні клієнтів на основі прогнозованої ймовірності відтоку, їхньої довічної цінності для компанії, поведінкових шаблонів та інших значущих характеристик. Така багатовимірна сегментація дозволяє розробляти диференційовані та більш ефективні стратегії утримання для кожної виділеної групи.

Важливим є також аналіз життєвого циклу клієнта та виявлення критичних точок відтоку**.** Це завдання спрямоване на ідентифікацію ключових етапів, на яких ризик відтоку є статистично найвищим, що дозволяє компаніям превентивно фокусувати зусилля з утримання саме в ці моменти.

Для оперативного реагування на зростаючі ризики розробляються системи раннього попередження про відтік. Метою цього завдання є створення автоматизованих механізмів, які відстежують поведінкові індикатори та прогностичні показники клієнтів і генерують сповіщення у випадку значного підвищення індивідуального ризику відтоку, дозволяючи вжити заходів до того, як клієнт прийме остаточне рішення піти.

Інтелектуальний аналіз даних також надає інструменти для оцінки ефективності заходів з утримання. Це передбачає об'єктивну оцінку результативності різних маркетингових кампаній та ініціатив, спрямованих на утримання клієнтів, часто за допомогою контрольованих експериментів, таких як A/B тестування.

Всі інсайди використовуються для персоналізації взаємодії та пропозицій. Використання даних про ймовірність відтоку, його ключові драйвери для конкретного клієнта та його приналежність до певного сегмента ризику дозволяє створювати та доставляти високоперсоналізовані комунікації, продуктові рекомендації, спеціальні знижки або бонуси, максимально адаптовані до індивідуальних потреб та ситуації клієнта.

1.4 Аналіз іноземних та вітчизняних науково-інформаційних джерел

1.4.1 Огляд ключових наукових публікацій та еволюція підходів до прогнозування відтоку

Проблема прогнозування та управління відтоком клієнтів є однією з ключових для сучасного бізнесу, що зумовлює прагнення компаній до сталого розвитку та підвищення прибутковості. Аналіз науково-інформаційних джерел свідчить про значний інтерес дослідників та практиків до цієї тематики, починаючи від фундаментальних робіт з клієнтської лояльності до новітніх застосувань штучного інтелекту.

Фундамент для сучасного розуміння важливості утримання клієнтів було закладено у 1990-х роках. Саме тоді консалтингова компанія Bain & Company однією з перших почала формалізувати та кількісно оцінювати вплив відтоку клієнтів на фінансові показники компаній. Їхні дослідження, що стали класичними, емпірично продемонстрували, що навіть незначне, на перший погляд, підвищення рівня утримання клієнтів, наприклад, на 5 відсоткових пунктів (в.п.), може призвести до суттєвого зростання прибутковості бізнесу – в діапазоні від 25% до 95% залежно від галузі та бізнес-моделі. Ці висновки підкреслили економічну доцільність інвестицій у програми лояльності та утримання.

Подальший розвиток ця ідея отримала у працях Фредеріка Райхельда (Reichheld, 1996), зокрема в його відомій роботі «Ефект лояльності» («The Loyalty Effect»). Райхельд значно поглибив розуміння зв'язку між тривалістю взаємовідносин з клієнтом, його лояльністю та довічною цінністю клієнта (CLV). Він показав, що лояльні клієнти не тільки генерують повторні покупки, але й часто є менш чутливими до ціни, готові рекомендувати компанію своєму оточенню (що знижує витрати на залучення нових клієнтів) та потребують менших витрат на обслуговування.

Праці Суніла Гупти (Sunil Gupta) та його співавторів, наприклад, у книзі «Управління клієнтами заради прибутку: Стратегії для збільшення прибутків

та побудови лояльності» («Managing Customers for Profit: Strategies to Increase Profits and Build Loyalty», 2006), продовжили цю лінію досліджень, запропонувавши більш складні моделі для розрахунку CLV та детально аналізуючи фактори, що на неї впливають. Важливим інструментом, популяризованим у цей період, став RFM-аналіз (Recency, Frequency, Monetary value), який і досі використовується як базовий інструмент для сегментації клієнтської бази.

З розвитком інформаційних технологій у 2000-х роках логістична регресія, як зазначається у багатьох оглядових працях того періоду, наприклад, в аналізі Несліна та ін. (Neslin S. A. et al., 2006) «Challenges and Opportunities in Multichannel Customer Management», швидко стала своєрідним "золотим стандартом" у прогнозуванні відтоку завдяки своїй інтерпретованості та ефективності для бінарної класифікації.

Однак, подальші дослідження виявили переваги нелінійних моделей. Наприклад, робота Мозера (Mozer M. C., 2000) «Predicting subscriber dissatisfaction and improving retention in the wireless telecommunications industry» однією з перших продемонструвала потенціал нейронних мереж та дерев рішень для прогнозування відтоку в телекомі. Дослідження Хуанга(Huang B., 2012), наприклад, у статті «A comparative study of ensemble learning for churn prediction in telecommunications», показали переваги ансамблевих методів, таких як Random Forest та Gradient Boosting, особливо при роботі з незбалансованими даними.

Порівняльні дослідження різних алгоритмів машинного навчання продовжили цю тенденцію. Роботи, такі як дослідження Вербеке(Verbeke W., 2012, «New insights into churn prediction in the telecommunication industry: A new dataset and comparative study») та праця Моро(Moro S., 2014, стаття «A data-driven approach to predict the success of bank telemarketing»), де проводилось порівняння різних моделей для прогнозування поведінки клієнтів, включаючи

відтік, підтвердили, що реалізації градієнтного бустингу, зокрема XGBoost, часто демонструють приріст якості прогнозу (ROC-AUC) на 2–5% порівняно з традиційними лінійними моделями.

Зі зростанням складності моделей загострилися проблеми якості даних та інтерпретованості**.** Карл Голд (Carl Gold, 2020) у своїй книзі «Боротьба з відтоком за допомогою даних» («Fighting Churn with Data: The science and strategy of customer retention») системно описує проблему витоку цільової інформації та пропонує ретельний підхід до формування ознакового простору і цільової змінної, зокрема, використовуючи чіткі часові зрізи).

Іншою важливою проблемою є інтерпретованість складних ансамблевих моделей. Метод SHAP (SHapley Additive exPlanations)**,** запропонований Скоттом Лундбергом та Су-Ін Лі (Lundberg S. & Lee S.-I., 2017) у статті «A Unified Approach to Interpreting Model Predictions», став значним кроком уперед, дозволяючи оцінити внесок кожної ознаки в конкретний прогноз.

1.4.2 Сучасні тенденції та відкриті питання у дослідженнях прогнозування відток

Аналіз еволюції методів прогнозування відтоку клієнтів демонструє значний прогрес, досягнутий завдяки розвитку технологій машинного навчання та доступності великих масивів даних. Однак, незважаючи на досягнуті успіхи, ця сфера залишається динамічною, з низкою актуальних тенденцій та відкритих питань, що стимулюють подальші дослідження та розробки.

Однією з ключових сучасних тенденцій є прагнення до ще більш раннього та точного прогнозування**.** Традиційні моделі часто фокусуються на прогнозуванні факту відтоку у відносно короткостроковій перспективі. Проте для бізнесу критично важливо ідентифікувати клієнтів групи ризику на якомога більш ранніх етапах, коли ще є достатньо часу та можливостей для ефективного превентивного втручання. Це зумовлює інтерес до моделей, що

враховують динаміку поведінки клієнтів у часі, аналізують послідовності їхніх дій та здатні вловлювати ледь помітні сигнали зміни лояльності. У цьому контексті зростає увага до застосування методів глибокого навчання, зокрема рекурентних нейронних мереж та трансформерів, для аналізу часових рядів клієнтської активності.

Іншою важливою тенденцією є використання ширшого спектру даних та більш складних ознак**.** Якщо раніше моделі часто базувалися на транзакційних та демографічних даних, то сьогодні все більше уваги приділяється інтеграції неструктурованих даних, а також даних про взаємозв'язки між клієнтами. Останнє відкриває перспективи для застосування графових моделей даних, де клієнти та їхні взаємодії представляються у вигляді графів. Аналіз таких графів може виявити вплив оточення на схильність клієнта до відтоку або ідентифікувати групи клієнтів зі схожими патернами взаємодій, що може слугувати цінними предикторами.

1.5 Постановка завдання

Мета формальної постановки – оцінити персональну ймовірність відтоку для кожного користувача, маючи історичні дані *X*, що містять інформацію про фінансові транзакції, поведінкові взаємодії та соціально-демографічні характеристики, необхідно побудувати функцію $f: X \rightarrow [0, 1]$, яка прогнозує ймовірність відтоку *P*(churn=1), у межах, наприклад 30 днів та її основі підтримати рішення щодо втручань з мінімальними втратами і максимальним фінансовим ефектом. З формальної точки зору це задача бінарної класифікації з істотною дизбалансом класів.

Вхідними даними для вирішення задачі є:

- набір спостережень $D = \{(x_i, y_i, t_i,)\}$, де $x_i -$ вектор ознак клієнта на дату спостереження t_i;
- $y_i \in \{0, 1\} -$ мітка відтоку (1 — клієнт переходить у стан «відтік» на протязі 30 днів від дня тренування моделі, 0 — залишається активним)

- дисбаланс класів(частка міток зі значенням 1 становить близко 20%)

Ставимо задачу мінімізації функції витрат *L(θ),* що поєднує крос-ентропію та економічну вартість помилок:

$$L = w_{Fp} * FP + w_{FN} * FN + \lambda * CE(\theta)$$

де *FP, FN* — кількість хибних позитивів / негативів; w_{Fp}, w_{FN} — коефіцієнти вартості; *CE* — крос-ентропія; λ — ваговий параметр.

Результати:

- індивідуальна оцінка ризику відтоку;
- пояснення ключових факторів;
- перелік рекомендованих дій.

1.6 Висновки

У цьому розділі роботи було закладено теоретичне та методологічне підґрунтя для дослідження проблеми інтелектуального аналізу та прогнозування відтоку клієнтів. На початку розділу було проведено визначення ключових термінів**.** Це дозволило встановити єдине термінологічне поле та забезпечити чіткість подальшого викладу матеріалу.

Далі було надано детальний опис задачі інтелектуального аналізу та прогнозування відтоку клієнтів. Розкрито її сутність, економічну значущість для сучасних підприємств у різних галузях.

Проведено огляд існуючих методів аналізу відтоку клієнтів, що включав як традиційні статистичні підходи, так і сучасні методи машинного навчання.

Було розглянуто переваги та недоліки різних класів моделей, зокрема регресійних моделей, дерев рішень, ансамблевих методів та інших актуальних підходів, що застосовуються для ідентифікації клієнтів, схильних до відтоку.

Здійснено аналіз іноземних та вітчизняних науково-інформаційних джерел, присвячених проблематиці відтоку. Аналіз підтвердив актуальність теми, висвітлив еволюцію підходів до її вирішення, окреслив ключові

досягнення світової науки та практики у цій сфері, а також вказав на наявні прогалини та перспективні напрямки для подальших досліджень, зокрема, в контексті українського ринку.

На основі проведеного аналізу та виявлених аспектів проблеми, у першому розділі було сформульовано та обґрунтовано постановку завдання даного дослідження**.**

РОЗДІЛ 2
МОДЕЛІ АЛГОРИТМІВ ТА МЕТОДИ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ

2.1 Класифікація методів прогнозування відтоку клієнтів

Для вирішення багатогранного завдання прогнозування відтоку клієнтів у сучасній практиці інтелектуального аналізу даних застосовується широкий спектр методів та моделей. Їх можна класифікувати за різними критеріями, такими як тип математичного апарату, підхід до навчання моделі, рівень інтерпретованості результатів або основне призначення в аналітичному процесі. Доцільно виділити наступні основні категорії методів, які часто взаємодоповнюють одна одну на різних етапах дослідження:

- Описові та діагностичні методи: Ця група методів спрямована на аналіз історичних даних для розуміння того, що сталося з клієнтською базою в минулому та чому саме це сталося. Хоча ці методи безпосередньо не прогнозують майбутній відтік, вони є надзвичайно важливими на початкових етапах дослідження. Вони допомагають краще зрозуміти характеристики клієнтів, що вже пішли у відтік, виявити загальні патерни їхньої поведінки, сформулювати гіпотези щодо можливих причин відтоку та, що важливо, слугують основою для генерації інформативних ознак для подальшого використання у прогностичних моделях. До таких методів належать, наприклад, когортний аналіз, RFM-аналіз, статистичне профілювання клієнтів, що пішли, та різноманітні техніки візуалізації даних.

- **Статистичні прогностичні моделі:** Це класичні підходи, що базуються на принципах статистичної теорії та регресійного аналізу для моделювання залежності між характеристиками клієнта та ймовірністю його відтоку. Найбільш поширеним та часто використовуваним представником цієї групи є логістична регресія. Також сюди можна віднести моделі виживаності, які аналізують час до настання певної події та фактори, що на нього впливають.

Перевагами цих методів часто є їхня відносна простота реалізації, висока інтерпретованість отриманих коефіцієнтів (що дозволяє оцінити вплив кожної ознаки) та усталені процедури оцінки статистичної значущості результатів.

- Моделі машинного навчання: Ця велика та активно розвиваючася категорія методів включає різноманітні алгоритми, які тренуються на історичних даних для виявлення складних, часто нелінійних, залежностей та здійснення точних прогнозів щодо нових даних. У контексті прогнозування відтоку найчастіше використовуються методи контрольованого навчання, а саме задачі бінарної класифікації. До ключових підходів у цій категорії належать лінійні моделі, дерева рішень та методи на їх основі, ансамблеві методи, метод опорних векторів, методи, засновані на відстанях, та нейронні мережі різноманітних архітектур. Ці моделі часто демонструють вищу прогностичну точність порівняно зі статистичними, особливо на великих та складних наборах даних, але можуть бути складнішими в інтерпретації.
- Гібридні підходи: У практиці часто застосовуються підходи, що комбінують елементи різних вищезазначених категорій методів для досягнення кращих результатів або вирішення специфічних завдань. Наприклад, результати кластеризації клієнтів можуть використовуватися як додаткові категоріальні ознаки для моделей класифікації.

Вибір конкретного методу або комбінації методів для прогнозування завжди залежить від низки факторів: специфіки бізнес-завдання та сфери, характеристик, обсягу та якості наявних даних, вимог до точності, швидкодії та інтерпретованості моделі, а також наявних обчислювальних ресурсів.

2.2 Описові та діагностичні методи аналізу поведінки клієнтів

2.2.1 Когортний аналіз для дослідження динаміки утримання

Когортний аналіз є фундаментальним аналітичним методом, що дозволяє досліджувати поведінкові шаблони груп користувачів (когорт) протягом тривалого часу. Ключовою особливістю методу є об'єднання користувачів у

когорти на основі спільної ознаки, найчастіше – дати першої значущої взаємодії з продуктом чи сервісом (наприклад, реєстрації, першої покупки, першого візиту).

Такий підхід нівелює вплив зовнішніх факторів, що змінюються з часом (як-от маркетингові кампанії чи оновлення продукту), дозволяючи більш об'єктивно оцінити власне користувацький досвід та його еволюцію.

Когортний аналіз надає можливість не лише візуально та кількісно відстежувати динаміку залученості, але й виявляти критичні точки в життєвому циклі користувача, прогнозувати відтік та ідентифікувати фактори, що впливають на довгострокову лояльність.

Дана методика широко застосовується в маркетингу, продукт-менеджменті, дослідженнях користувацького досвіду та інших сферах, де розуміння динаміки поведінки груп є критично важливим. Когортний аналіз складається з наступних етапів:

1. Формування когорт

Когорти визначаються за критеріями, релевантними для цілей дослідження:

- Когорти реєстрації (часові): Об'єднують користувачів за періодом першої взаємодії (наприклад, перший логін або оплата впродовж календарного місяця)..
- Метрик-когорти (поведінкові): Групують користувачів за значенням певної поведінкової метрики у стартовий період.

2. Ключові показники аналізу

Для кількісної оцінки поведінки когорт (c) у часі (m – місяць життя когорти) використовуються:

№	Показник	Формула	Інтерпретація

1	Retention(c,m)	$\frac{N(c,m)}{N(c,0)}$	Частка активних користувачів когорти c через m місяців.
2	ChurnRate(c,m)	$1-\frac{N(c,m)}{N(c,m-1)}$	Відносний відтік у місяці m для когорти c.
3	Survival(t)	$\Pi_{k=0}^{t}(1-\ ChurnRate_k)$	Ймовірність активності користувача до моменту t.
4	Hazard(t)	$\frac{ChurnRate_t}{Survival_{t-1}}$	Миттєвий ризик відтоку на кроці t.

Таблиця 1. Ключові показники когортного аналізу

Приклад: Retention(6)=0,58 означає, що 58% користувачів когорти активні через 6 місяців. Hazard(6)=0,064 – щомісячний ризик відтоку 6,4% для тих, хто був активним до 6-го місяця.

3. Візуалізація та інтерпретація

- Когортна теплокарта: Візуалізує Retention (колір) за когортами (вісь Y) та місяцями життя (вісь X), виявляючи тренди та сезонність (діагоналі).
- U-крива відтоку: Демонструє типові фази: високий початковий відтік (onboarding), стабілізація, можливий підйом перед закінченням підписки.

4. Ранжування когорт за ранньою залученістю

Аналіз метрик-когорт виявляє залежність довгострокового утримання від початкової активності.

5. Перевірка стабільності моделей

Якість ML-моделей (наприклад, прогнозування Retention) оцінюється за їх стабільністю на різних когортах. Розраховується метрика CMSE (середньоквадратична помилка між прогнозованим та фактичним Retention по комірках когортної таблиці).

2.2.2 RFM-аналіз для сегментації клієнтів за активністю та цінністю

RFM-аналіз (Recency, Frequency, Monetary value) – це ще один класичний та широко використовуваний метод описової аналітики, який дозволяє сегментувати клієнтську базу на основі їхньої купівельної поведінки. Цей підхід базується на припущенні, що клієнти, які купували нещодавно, купують частіше та витрачають більше, є більш цінними для бізнесу та з більшою ймовірністю здійснять повторні покупки.

Основні компоненти RFM-аналізу:

- Recency (Давність): Як давно клієнт здійснив останню покупку або іншу значущу взаємодію (наприклад, останній візит на сайт). Чим менше часу минуло з моменту останньої активності, тим вищий показник Recency присвоюється клієнту.
- Frequency (Частота): Як часто клієнт здійснює покупки або взаємодіє з компанією протягом певного періоду. Чим більше транзакцій або активностей, тим вищий показник частоти.
- Monetary Value (Грошова цінність): Загальна сума грошей, яку клієнт витратив на продукти чи послуги компанії протягом певного періоду. Чим більша сума, тим вищий показник Monetary.

Процес RFM-аналізу зазвичай включає наступні кроки:

1. Визначення періоду аналізу.
2. Розрахунок RFM-показників для кожного клієнта.
3. Ранжування клієнтів за кожним показником: Клієнти сортуються за кожним з трьох показників і поділяються на кілька груп. Кожній групі присвоюється відповідний бал
4. Формування RFM-сегментів**:** Кожен клієнт отримує комбінований RFM-код (наприклад, 555 – найкращий клієнт за всіма показниками, 111 – клієнт, що давно не купували, робили це рідко і на малі суми). На основі цих значень або діапазонів значень формуються агреговані сегменти.

5. Розробка стратегій для кожного сегмента: Для кожного виділеного RFM-сегмента розробляються цільові маркетингові стратегії, спрямовані на утримання, розвиток або реактивацію.

Незважаючи на свою простоту, RFM-аналіз залишається ефективним інструментом не лише для швидкої сегментації, але й має суттєве значення як попередній етап або компонент у побудові більш складних моделей прогнозування відтоку клієнтів.

По-перше, він дозволяє провести ранню ідентифікацію груп ризику. Клієнти, що потрапляють до сегментів з низькими показниками Recency (давність останньої покупки) та Frequency (частота покупок), часто є першими кандидатами на відтік, оскільки їхня активність та залученість вже знизилися. Виявлення таких сегментів дозволяє сфокусувати на них увагу та, можливо, застосувати до них більш інтенсивні заходи з утримання або подальшого глибокого аналізу причин їхньої пасивності.

По-друге, самі RFM-показники або похідні від них (наприклад, індивідуальні бали R, F, M для кожного клієнта, або приналежність клієнта до певного RFM-сегменту, закодована як категоріальна ознака) можуть бути використані як потужні предиктори у моделях машинного навчання для прогнозування. Часто ці, на перший погляд, прості метрики несуть у собі значну прогностичну силу і можуть суттєво покращити якість моделей класифікації, таких як логістична регресія, дерева рішень або ансамблеві методи. Інтеграція RFM-ознак до загального набору даних збагачує модель інформацією про історичну цінність та активність клієнта.

Отже, хоча RFM-аналіз сам по собі є описовим методом, його результати та показники є цінним вхідним матеріалом для побудови складніших прогностичних систем, допомагаючи як у виявленні груп ризику, так і у формуванні інформативних ознак для моделей машинного навчання, спрямованих на передбачення відтоку клієнтів.

2.3 Статистичні моделі прогнозування

Статистичні моделі тривалий час були основним інструментом для аналізу та прогнозування в клієнтів завдяки їхній відносній простоті, інтерпретованості та добре розробленому математичному апарату. Ці моделі дозволяють кількісно оцінити вплив різних факторів на ймовірність відтоку та зробити обґрунтовані прогнози.

2.3.1 Логістична регресія

Логістична регресія є одним із найбільш фундаментальних та широко використовуваних статистичних методів для вирішення задач бінарної класифікації, до яких належить і прогнозування відтоку клієнтів. Даний метод моделює ймовірність настання події за допомогою сигмоїдної функції, яка перетворює лінійну комбінацію предикторів у значення ймовірності в діапазоні від 0 до 1.

Математично, ймовірність відтоку (p) моделюється так: $p = P(Y = 1|X) = \sigma(z) = \frac{1}{1+e^{-z}}$, де z є лінійною комбінаціє предикторів з відповідними ваговими коєфіцієнтами β_j: $z = \beta_0 + \beta_1 X_1 + \beta_2 X_{2\ldots}$

Тут β_0 – вільний член, а β_j – коефіцієнти регресії, що оцінюються на основі навчальних даних, зазвичай методом максимальної правдоподібності.

Однією з ключових переваг логістичної регресії є інтерпретованість її коефіцієнтів. Знаки та величини коефіцієнтів β_j безпосередньо вказують на напрям та силу впливу кожної ознаки X_j на логарифм шансів (log-odds) відтоку. Це дозволяє не тільки прогнозувати ймовірність, але й розуміти, які фактори є найбільш значущими драйверами відтоку.

Для запобігання перенавчанню моделі, особливо на даних з великою кількістю предикторів або при наявності мультиколінеарності, та для покращення її генералізаційної здатності часто застосовуються методи регуляризації. Найбільш поширеними є L1-регуляризація (Lasso), яка може

призводити до обнулення коефіцієнтів неінформативних ознак, та L2-регуляризація (Ridge regression), яка штрафує великі значення коефіцієнтів, зменшуючи їхню дисперсію. L2-регуляризація додає до функції втрат штраф, пропорційний сумі квадратів вагових коефіцієнтів.

Параметр регуляризації (наприклад, λ або C, обернений до λ) підбирається зазвичай за допомогою крос-валідації, часто з використанням метрики ROC-AUC для оцінки якості моделі.

Оскільки початкові ймовірності, прогнозовані логістичною регресією, не завжди точно відображають дійсні частоти подій (тобто модель може бути погано каліброваною), для їх покращення можуть застосовуватися методи калібрування ймовірностей, такі як калібрування Платта або ізотонічна регресія.

2.3.2 Варіації логістичної регресії для аналізу відтоку

Стандартна логістична регресія розглядає ознаки клієнта статично, на певний момент часу. Однак, для врахування динаміки поведінки користувачів, яка може бути сильним предиктором відтоку, модель може бути доповнена або модифікована:

- Логістична регресія з часовими лагами: Для врахування динаміки поведінки клієнта модель доповнюється похідними ознаками, що відображають зміни ключових метрик у часі. Це можуть бути, наприклад, різниці значень активності користувача між послідовними часовими вікнами (зміна частоти покупок, тривалості сесій, обсягу спожитого контенту), тренди зміни показників або середні значення за попередні періоди. Включення таких динамічних предикторів, відомих як ознаки з часовими лагами, часто призводить до істотного поліпшення прогностичної сили моделі, оскільки дозволяє зафіксувати тенденції, що передують відтоку.
- Неперервно-часова логістична регресія: Цей підхід, ідеологічно близький до моделей виживаності, спрямований на прогнозування миттєвої

ймовірності відтоку на дуже коротких, дискретних часових інтервалах (наприклад, щоденно або щотижнево), а не на фіксованому прогнозному горизонті. Модель оцінює ймовірність відходу клієнта саме в день t (або тиждень t), за умови його активності до цього моменту. Навчання такої моделі здійснюється на розширеному наборі даних, де кожен запис може представляти стан клієнта в певний часовий мікроінтервал. Основною перевагою є гнучкість: отримані короткострокові прогнози ризику можна агрегувати для оцінки сукупної ймовірності відтоку на будь-якому довільному часовому горизонті в майбутньому.

- Багатокласова логістична регресія: У випадках, коли життєвий цикл клієнта або результат його взаємодії не обмежується простим бінарним вибором «піти» / «залишитися», а включає кілька значущих станів (наприклад, «залишився на поточному тарифі», «перехід на дорожчий тариф», «перехід на дешевший тариф», «відтік»), застосовується багатокласова логістична регресія. Ця модель, часто реалізована через Softmax-регресію, одночасно оцінює для кожного клієнта ймовірності переходу до кожного з можливих станів, причому сума цих імовірностей дорівнює одиниці. Такий підхід дає можливість більш диференційовано підходити до управління клієнтською базою та реалізовувати різні стратегії для різних сценаріїв зміни статусу клієнта.

2.3.3 Моделі вживаності

Моделі виживаності, також відомі як аналіз тривалості подій є ще одним класом статистичних методів, які можуть бути ефективно застосовані для аналізу та прогнозування відтоку клієнтів. На відміну від стандартних моделей класифікації, які прогнозують, чи відбудеться відтік у певний фіксований період, моделі виживаності фокусуються на аналізі часу до настання події відтоку T та на факторах, що впливають на цю тривалість.

Ключовими поняттями в аналізі виживаності є:

- Функція виживання, *S(t)*: Визначає ймовірність того, що клієнт "доживе" (тобто не піде у відтік) до моменту часу t або довше: $S(t) = P(T > t)$. Графічне представлення цієї функції дозволяє візуалізувати динаміку утримання клієнтів у часі.

- Функція ризику, *h(t)*: Визначає миттєву ймовірність відтоку клієнта в момент часу *t*, за умови, що він був активним до цього моменту. Цей показник характеризує "ризикованість" певного періоду для клієнта. Математично визначається як: $h(t) = \lim_{\Delta t \to 0} \frac{P(t < T < t + \Delta t | T > t)}{\Delta t}$.

- Цензуровані дані: Особливістю аналізу виживаності є врахування цензурованих спостережень. Це клієнти, для яких на момент завершення дослідження подія відтоку ще не відбулася, або клієнти, які почали спостерігатися вже після певного періоду активності. Моделі виживаності коректно обробляють таку неповну інформацію.

Найбільш відомою напівпараметричною моделлю в аналізі виживаності є модель пропорційних ризиків Кокса. Вона дозволяє оцінити вплив різних коваріат (ознак клієнта $X_1,\ldots, X_k$) на функцію ризику, не роблячи припущень щодо форми базової функції ризику:

$$h_0(t): h(t|X) = h_0(t) exp(\beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k)$$

Коефіцієнти β_j інтерпретуються як зміна логарифма відносного ризику при зміні відповідної коваріати на одиницю.

Переваги моделей виживаності:

- Дозволяють моделювати не тільки факт відтоку, але й час до нього.
- Коректно враховують цензуровані дані.
- Дозволяють ідентифікувати фактори, що прискорюють або сповільнюють відтік.
- Можуть бути використані для прогнозування кривих утримання для нових клієнтів або сегментів.

Обмеження моделей виживаності:

- Можуть бути складнішими в реалізації та інтерпретації порівняно з логістичною регресією, особливо параметричні моделі виживаності, які вимагають припущень щодо розподілу часу до події.
- Статистичні моделі, незважаючи на появу більш складних алгоритмів машинного навчання, залишаються важливим інструментом аналізу відтоку, особливо коли потрібна висока інтерпретованість результатів та розуміння впливу окремих факторів.

2.4 Моделі машинного навчання для прогнозування відтоку

2.4.1 Дерева рішень

Дерева рішень є одним з фундаментальних та інтуїтивно зрозумілих алгоритмів машинного навчання, що використовуються як для задач класифікації, так і для регресії. У контексті прогнозування відтоку, дерево рішень будує ієрархічну структуру правил (у вигляді дерева), яка розділяє клієнтів на групи з високою однорідністю щодо цільової змінної.

Процес побудови дерева починається з кореневого вузла, що містить усі навчальні дані. На кожному вузлі алгоритм обирає таку ознаку та таке її порогове значення, яке найкращим чином розділяє дані на дві (або більше) підгрупи за певним критерієм чистоти. Популярними критеріями є:

- Індекс Джині: Вимірює ймовірність неправильної класифікації випадково обраного елемента, якщо його клас визначався б випадково відповідно до розподілу класів у вузлі. Формула для K класів:

$$Gini = 1 - \sum_{k=1}^{K} p_k^2 ,$$

де p_k - частка об'єктів класу k у вузлі. Алгоритм прагне мінімізувати цей індекс.

Процес розбиття триває рекурсивно до тих пір, поки не буде досягнуто певних умов зупинки. Кінцеві вузли містять прогноз.

Перевагами дерев рішень є висока інтерпретованість, здатність працювати як з числовими, так і з категоріальними ознаками, відносна нечутливість до масштабування ознак.

Недоліками є схильність до перенавчання, особливо для глибоких дерев, нестабільність. Для подолання недоліків одиночних дерев рішень часто використовуються ансамблеві методи.

2.4.2 Ансамблеві методи на основі дерев рішень

Ансамблеві моделі, побудовані на основі дерев рішень, справедливо вважаються одними з найефективніших інструментів для прогнозування відтоку клієнтів у промислових системах. Їхня ефективність зумовлена синергією кількох ключових характеристик: вони демонструють високу прогностичну здатність, оскільки здатні моделювати складні нелінійні залежності та взаємодії між ознаками; проявляють стійкість до нерівномірно розподілених даних та викидів, часто не вимагаючи попереднього масштабування чи спеціальної обробки даних; а також забезпечують гнучкість у роботі з різними типами вхідних ознак. Ідея ансамблювання полягає у тому, що об'єднання прогнозів від декількох моделей (навіть якщо кожна з них не є ідеальною) часто призводить до кращого та більш стабільного результату, ніж прогноз від однієї, хай навіть складної, моделі. Прикладами є:

1. Випадковий ліс. Метод випадкового лісу є одним з найпопулярніших ансамблевих алгоритмів, що базується на техніці беггінгу та випадковому підборі ознак. Випадковий ліс будує велику кількість незалежних дерев рішень. «Незалежність» цих дерев досягається двома основними шляхами:

2. Беггінг спостережень**:** Кожне дерево навчається на випадковій підвибірці даних з оригінального навчального набору, яка формується шляхом вибору з поверненням. Це означає, що деякі спостереження можуть потрапити до підвибірки декілька разів, а інші – жодного.

3. Випадковий вибір ознак. При побудові кожного вузла в окремому дереві розглядається не повний набір ознак, а лише їх випадкова підмножина(зазвичай розміром $\sqrt{k}$ або $\mathbf{log_2}(k) + 1$), де k – загальна кількість ознак). Після навчання всіх дерев остаточний прогноз для задач класифікації формується шляхом голосування більшості серед усіх дерев ансамблю. Для задач регресії прогноз усереднюється. Такий підхід дозволяє зменшити дисперсію моделі порівняно з одним деревом рішень, не надто збільшуючи зсув.

Ефективність випадкового лісу значною мірою залежить від правильного налаштування його гіперпараметрів:

- Кількість дерев. Визначає розмір ансамблю. Більша кількість дерев зазвичай покращує стабільність та якість моделі, але до певної межі, після якої приріст продуктивності стає незначним, а обчислювальні витрати зростають.
- Максимальна глибина дерева. Контролює складність кожного окремого дерева в ансамблі. Занадто велика глибина може призвести до перенавчання окремих дерев, тоді як занадто мала – до недонавчання (високого зсуву).
- Частка ознак для розбиття вузла. Визначає кількість ознак, що випадково обираються для пошуку найкращого розбиття в кожному вузлі. Менші значення збільшують різноманітність дерев і зменшують кореляцію між ними, що може позитивно вплинути на узагальнюючу здатність ансамблю.

Серед переваг даного методу можна виділити:

- висока прогностична точність на багатьох задачах;
- відносна нечутливість до мультиколінеарності ознак, оскільки кожне дерево працює з їх підмножиною;
- вбудована оцінка важливості ознак: розраховується на основі того, наскільки кожна ознака сприяє зменшенню домішок у вузлах дерев або наскільки погіршується якість моделі при перемішуванні значень ознаки;

- можливість ефективного паралельного навчання, оскільки кожне дерево будується незалежно.

Серед недоліків найважливішими є:

- певною мірою «розмитість» прогнозів: через усереднення, особливо в задачах регресії або при прогнозуванні ймовірностей, модель може не досягати екстремальних значень, які могли б бути передбачені іншими методами, наприклад, бустингом;
- ризик перенавчання на даних з великою кількістю шумних ознак або на дрібних, рідкісних категоріях, якщо гіперпараметри не налаштовані ретельно;
- моделі можуть бути менш інтерпретованими, ніж одне дерево рішень, хоча методи оцінки важливості ознак частково компенсують цей недолік.

Градієнтний бустинг на деревах рішень. Градієнтний бустинг – це інший потужний ансамблевий метод, який, на відміну від випадкового лісу, будує дерева послідовно. Кожне наступне дерево в ансамблі навчається таким чином, щоб виправити помилки, допущені попередньою композицією вже побудованих дерев. Таким чином, ансамбль поступово «посилюється», фокусуючись на тих спостереженнях, на яких попередні моделі помилялися найбільше.

На кожній ітерації m будується нове дерево $h_m(X)$, яке намагається наблизити негативний градієнт функції втрат $L\big(y_i, F_{m-1}(X_i)\big)$ відносно поточного прогнозу ансамблю $F_{m-1}(X_i)$. Новий прогноз ансамблю формується як $F_m(X_i) = F_{m-1}(X_i) + v \cdot h_m(X_i)$, де v – швидкість навчання , що контролює внесок кожного нового дерева та допомагає запобігти перенавчанню. Зазвичай використовуються неглибокі дерева. Існує кілька популярних реалізацій GBDT, що відрізняються оптимізаціями та додатковими можливостями:

- XGBoost (Extreme Gradient Boosting)**:** Відомий своєю швидкістю та ефективністю, включає L1 та L2 регуляризацію, можливість паралельної обробки, обробку пропущених значень.
- LightGBM (Light Gradient Boosting Machine): Використовує унікальний підхід до побудови дерев, що робить його особливо швидким на великих наборах даних, зберігаючи високу точність.
- CatBoost**:** Добре працює з категоріальними ознаками (має вбудовані ефективні методи їх обробки), використовує симетричні дерева та спеціальні техніки для боротьби з перенавчанням.

Градієнтний бустинг часто демонструє найвищу точність серед методів машинного навчання на табличних даних; гнучкість у виборі функції втрат; ефективна робота з різними типами даних. Серед недоліків можна виділити більш чутливий до налаштування гіперпараметрів, ніж випадковий ліс; послідовне навчання робить його менш придатним для паралелізації; більша схильність до перенавчання, якщо не контролювати параметри.

1.4.2 Метод опорних векторів

Метод опорних векторів – це потужний алгоритм контрольованого навчання, який може використовуватися як для класифікації, так і для регресії, хоча найбільш відомий своїм застосуванням у задачах класифікації. Основна ідея SVM полягає в побудові оптимальної розділяючої гіперплощини (або набору гіперплощин у багатовимірному просторі), яка найкращим чином розділяє об'єкти різних класів. Оптимальною вважається така гіперплощина, яка має максимальний зазор – відстань до найближчих точок кожного класу (ці точки називаються опорними векторами).

Для лінійно роздільних даних задача полягає у знаходженні вектора ваг w та зсуву b таких, що гіперплощина $w^T x + b = 0$ максимізує зазор, який дорівнює w^2. Це еквівалентно мінімізації $||w^2||$ за умов $y_i(w^T x + b) > 1$для всіх навчальних прикладів (x_i, y_i) Для нелінійно розподілених даних SVM

використовує так званий «трюк з ядром». Ідея полягає у неявному відображенні вхідних даних у простір ознак вищої розмірності, де класи можуть стати лінійно роздільними. Замість явного обчислення координат у цьому новому просторі, SVM працює зі скалярними добутками векторів, які обчислюються за допомогою функції ядра $K(x_i, y_i)$.

Перевагами SVM є ефективність на даних високої розмірності; хороший узагальнюючий ефект, особливо при правильному виборі ядра та параметрів; стійкий до перенавчання при правильному налаштуванні.

Недоліками є вибір відповідного ядра та його параметрів може бути нетривіальним і вимагає експериментів; обчислювальна складність може бути високою на дуже великих наборах даних; результати менш інтерпретовані порівняно з деревами рішень або логістичною регресією.

1.4.3 Нейронні мережі

Нейронні мережі, інспіровані будовою та функціонуванням біологічних нейронних систем, є потужним класом моделей машинного навчання, здатних виявляти надзвичайно складні патерни та залежності в даних. У контексті прогнозування відтоку на табличних даних найчастіше використовуються багатошарові перцептрони (Multi-Layer Perceptrons, MLP) тип нейронної мережі прямого поширення.

MLP складається з вхідного шару (що приймає ознаки), одного або декількох прихованих шарів та вихідного шару (який дає прогноз). Кожен нейрон у прихованих та вихідному шарах обчислює зважену суму своїх входів, додає зсув (bias) і застосовує нелінійну функцію активації. Популярні функції активації для прихованих шарів включають ReLU $(f(x) = max(0, x))$, сигмоїдну функцію або гіперболічний тангенс. Для вихідного шару в задачі бінарної класифікації відтоку зазвичай використовується сигмоїдна функція, що повертає ймовірність належності до класу «відтік». Навчання нейронної мережі відбувається шляхом мінімізації функції втрат за допомогою

алгоритмів оптимізації, заснованих на градієнтному спуску та алгоритму зворотного поширення помилки.

Для аналізу відтоку, де важлива послідовність дій клієнта або динаміка його поведінки у часі (наприклад, послідовність відвідувань сайту, використання функцій мобільного додатку, історія покупок), можуть застосовуватися рекурентні нейронні мережі та їх більш просунуті варіанти, такі як LSTM та GRU. Ці архітектури мають внутрішню «пам'ять», що дозволяє їм враховувати попередні стани при обробці поточного вхідного сигналу, що робить їх придатними для моделювання часових залежностей.

Перевагами нейронних мереж є здатність моделювати дуже складні нелінійні залежності та взаємодії між ознаками; автоматичне виділення ознак (у архітектурах глибоких нейронних мереж); гнучкість архітектури.

Основними недоліками можна вважати: вимагають великих обсягів даних для ефективного навчання; схильні до перенавчання, якщо не застосовувати методи регуляризації; є «чорними скриньками» – їхні прогнози важко інтерпретувати; процес навчання може бути обчислювально затратним та вимагати ретельного налаштування гіперпараметрів (архітектура мережі, кількість шарів та нейронів, оптимізатор, швидкість навчання).

2.5 Вибір алгоритмів прогнозування відтоку

Проведений аналіз різноманітних методів та моделей, що застосовуються для прогнозування відтоку клієнтів, дозволяє зробити обґрунтований вибір інструментарію для подальшого експериментального дослідження. Для практичної реалізації та порівняння обрано наступний набір моделей:

Градієнтний бустинг на основі дерев рішень (GBDT) обрано як основний для розробки та тестування запропонованого методу. Цей вибір зумовлений стабільно високою прогностичною точністю GBDT на табличних даних, його здатністю ефективно моделювати складні нелінійні залежності та взаємодії

між ознаками, а також гнучкістю у налаштуванні функції втрат, що є важливим для інтеграції комбінованої метрики, яка враховує економічну вартість помилок. Для даного дослідження в якості конкретної реалізації GBDT розглядається реалізація LightGBM, відома своєю високою обчислювальною ефективністю та швидкістю навчання, особливо на великих наборах даних. Передбачається, що саме на основі LightGBM буде побудовано алгоритм «Early churn Alert» шляхом інтеграції традиційних та розроблених графових ознак.

Логістичну регресію обрано як класичну базову модель для порівняння. Її перевагами є простота реалізації, висока швидкість навчання та прогнозування, а головне – високий рівень інтерпретованості коефіцієнтів, що дозволяє легко оцінити вплив кожної ознаки на ймовірність відтоку. Порівняння з логістичною регресією допоможе оцінити приріст якості, який дають більш складні моделі машинного навчання.

Випадковий ліс також буде використаний як потужна базова модель для порівняння. Випадковий ліс часто демонструє високу точність «з коробки», є більш стійким до перенавчання та менш чутливим до налаштування гіперпараметрів порівняно з градієнтним бустингом. Включення його до експерименту дозволить отримати ще одну точку відліку для оцінки ефективності градієнтного бустингу та розробленого на його основі алгоритму «Early churn Alert».

Для найбільш об'єктивної оцінки ефективності запропонованого підходу, що передбачає використання графових ознак, буде проведено експеримент з моделлю LightGBM, навченою лише на традиційному наборі ознак. Порівняння результатів цієї моделі з результатами моделі LightGBM, що використовує розширений набір ознак, дозволить ізолювати та кількісно оцінити саме внесок графового аналізу у підвищення якості прогнозування відтоку.

2.6 Висновки

У даному розділі здійснено огляд ключових методів інтелектуального аналізу, що застосовуються для прогнозування відтоку клієнтів, включаючи описові, статистичні та основні моделі машинного навчання. Проаналізовано їхні принципи роботи, переваги та недоліки в контексті поставленої задачі.

На основі проведеного аналізу та відповідно до мети дослідження, для подальшої розробки та експериментального порівняння було обрано градієнтний бустинг на деревах рішень як основний алгоритм для реалізації методу «Early churn Alert» з використанням графових ознак. В якості базових моделей для порівняння будуть використані логістична регресія та випадковий ліс.

Таким чином, визначено інструментарій дослідження, що дозволяє перейти до наступного розділу, присвяченого методології експериментальної частини, детальному математичному опису обраних моделей та аналізу результатів їх практичної реалізації.

РОЗДІЛ 3
МЕТОДОЛОГІЯ, ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ МЕТОДУ ПРОГНОЗУВАННЯ ВІДТОКУ КЛІЄНТІВ

3.1 Методологія проведення експериментального дослідження.

Вибір методології є стратегічно важливим аспектом успішної реалізації будь-якого дослідження в галузі аналізу даних та Data Science. Для дослідження в якості методології було обрано методологію CRISP-DM.

Обрання саме методології CRISP-DM для виконання даного дослідження було зумовлене її перевіреною ефективністю, гнучкістю, інтуїтивною зрозумілістю та універсальністю, що дозволяє успішно застосовувати її для широкого спектра аналітичних завдань, включаючи прогнозування відтоку клієнтів. Ключовою перевагою CRISP-DM є її здатність органічно поєднувати аналітичні цілі проєкту з реальними практичними завданнями бізнесу. Це досягається завдяки ітеративному характеру методології та постійному зворотному зв'язку між етапами, що забезпечує прозорість усього процесу, чітке розмежування завдань на кожному етапі та зручність в управлінні проєктом.

Методологія CRISP-DM передбачає послідовне проходження шести ключових фаз:

1. Розуміння бізнес-середовища та постановка задачі. Дана фаза є ініціальною та визначальною, оскільки формує контекст та спрямованість усього дослідження. Вона передбачає глибоке занурення у специфіку предметної області, аналіз бізнес-цілей компанії та ідентифікацію проблем, що можуть бути вирішені засобами інтелектуального аналізу даних. У рамках даного дослідження, на цьому етапі було здійснено декомпозицію загальної бізнес-проблеми відтоку клієнтів на конкретні аналітичні завдання.

2. Розуміння даних. Після формалізації бізнес-завдань здійснюється перехід до фази детального дослідження наявних інформаційних активів. Цей етап розпочинається з інвентаризації, збору та первинної консолідації даних з усіх релевантних джерел Наступним кроком є проведення аналізу даних (EDA), що включає розрахунок дескриптивних статистик, візуалізацію розподілів ключових змінних та виявлення початкових кореляцій і патернів. Критично важливою складовою є верифікація якості даних: оцінка повноти, консистентності, точності та актуальності інформації.

3. Підготовка даних. Ця фаза охоплює всі операції, необхідні для трансформації вихідних даних у структурований набір, придатний для застосування алгоритмів машинного навчання. Вона є найбільш ресурсовитратною та критично значущою, оскільки якість підготовлених даних безпосередньо детермінує якість та надійність прогностичних моделей. Основні завдання включають очищення даних, трансформацію даних, конструювання ознак, відбір ознак, формування вибірок

4. Моделювання. На даному етапі здійснюється вибір, конфігурація та застосування різноманітних алгоритмів машинного навчання для побудови предиктивних моделей відтоку клієнтів. Вибір алгоритмічних рішень ґрунтується на характеристиках даних, обчислювальних можливостях та вимогах до інтерпретованості моделі. Для кожного обраного алгоритму проводиться процес навчання на навчальній вибірці та оптимізація його гіперпараметрів з використанням валідаційних процедур, таких як крос-валідація, з метою максимізації обраної метрики якості. Здійснюється компаративний аналіз прогностичної ефективності альтернативних моделей для вибору оптимального рішення.

5. Оцінка результатів. Фаза оцінки передбачає всебічну валідацію розроблених моделей та визначення ступеня досягнення поставлених на першому етапі бізнес-цілей. Моделі, що продемонстрували найкращу

продуктивність, піддаються ретельній перевірці на незалежних тестових даних для оцінки їх узагальнюючої здатності. Використовується комплекс релевантних статистичних метрик якості класифікації, а також бізнес-орієнтованих показників. Проводиться аналіз матриць помилок, здійснюється верифікація робастності та стабільності моделей. Результати оцінки повинні дати відповідь на питання, чи є модель достатньо якісною для практичного застосування та чи вирішує вона поставлену бізнес-проблему.

6. Розгортання. Завершальною фазою є імплементація валідованих та затверджених моделей в операційне середовище компанії. Це передбачає не лише технічне розгортання прогностичного, але й його органічну інтеграцію в існуючі бізнес-процеси. Важливою складовою є розробка плану моніторингу продуктивності моделі в продуктивному середовищі, її періодичного перенавчання на актуальних даних та адаптації до можливих змін у поведінці клієнтів чи бізнес-середовищі.

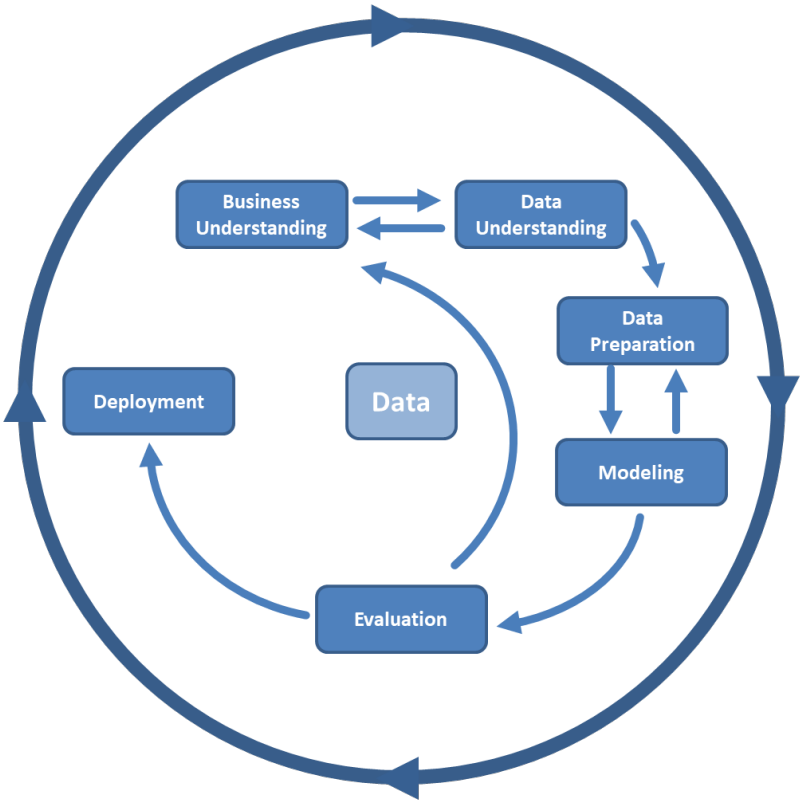


Рис. 2.1 Методологія CRISP-DM

3.2 Опис алгоритму раннього прогнозування відтоку та методології його оцінки

Алгоритм «Early-Churn Alert» є комплексним підходом, що поєднує традиційні методи інженерії ознак, інноваційне використання графового аналізу для виявлення неочевидних патернів взаємодій клієнтів та потужність моделей градієнтного бустингу для здійснення точних прогнозів.

Основна мета алгоритму – не просто спрогнозувати факт відтоку, а ідентифікувати клієнтів, що знаходяться на ранніх стадіях формування наміру припинити взаємодію з компанією, надаючи бізнесу більше часу та можливостей для ефективного втручання.

Функціонування алгоритму базується на синергії між традиційними методами інженерії ознак, специфічними характеристиками, отриманими з графового аналізу клієнтських взаємодій, та прогностичною потужністю моделі градієнтного бустичнгу. Алгоритм можна представити наступною послідовність кроків:

1. Формування традиційних ознак**:** На цьому етапі збираються та агрегуються дані про клієнтів, на основі яких генеруються стандартні предиктори, такі як RFM-показники, агрегати активності, динамічні характеристики тощо.

2. Побудова графа взаємодій та генерація графових ознак: Для виявлення неочевидних патернів, що можуть слугувати ранніми індикаторами відтоку, будується граф взаємодій клієнтів $G = (V, E)$, де V- множина клієнтів, а E - множина ребер, що відображають певний тип взаємодії. На основі цього графа розраховуються наступні метрики для кожної вершини-клієнта $v \in V$.

- Зважений ступінь вершини: Відображає інтенсивність взаємодій клієнта: $k_v^w = \sum_{u \in V} w_{vu}$, де w_{vu}- вага ребра між v та u. Зниження цього показника може сигналізувати про зменшення залученості.

- Локальний коефіцієнт кластеризації: Вимірює зв'язність оточення клієнта, розраховується за формулою: $C_v = \frac{2L_v}{k_v(k_v-1)}$, де L_v – кількість ребер між сусідами вершини v, а k_v – (незважений) ступінь вершини v. Зміна C_v може вказувати на «розпад» локальної спільноти клієнта.

- Характеристики спільноти: Застосування алгоритмів виявлення спільнот (наприклад, алгоритм Лувена, що оптимізує модулярність) дозволяє розбити граф на щільно пов'язані підгрупи клієнтів. Модулярність **(**Q**)** є метрикою якості такого розбиття і визначається як: $Q = \frac{1}{2m}\sum_{vw}\left[A_{vw} - \frac{k_v k_w}{2m}\right]\delta(c_v, c_w)$, де m - кількість ребер у графі, A_{vw}- елемент матриці суміжності (1, якщо є ребро між *v* та *w*, і 0 інакше), k_v та k_w - ступені вершин *v* та *w* відповідно, c_v та c_w спільноти, до яких належать вершини *v* та *w,* а $\delta(c_v, c_w)$ - функція Кронекера (1, якщо $c_v = c_w$, і 0 інакше). Алгоритми прагнуть максимізувати *Q*. Після ідентифікації спільнот, для кожного клієнта визначаються ознаки: ідентифікатор спільноти, до якої він належить, розмір спільноти, щільність зв'язків всередині спільноти, середній показник відтоку інших членів цієї спільноти за попередній період.

3. Прогностичне моделювання на основі LightGBM: В якості основного класифікатора використовується LightGBM, реалізація градієнтного бустингу на деревах рішень. Алгоритм послідовно будує ансамбль дерев рішень $F_M(x) = \sum_{m=1}^{M} y_m h_m(x)$, де кожне нове дерево $h_m(x)$ навчається мінімізувати помилку попередньої композиції $F_{m-1}(x)$ шляхом апроксимації негативного градієнта функції втрат $L\big(y, F(x)\big)$. Для бінарної класифікації (відтік/не відтік) часто використовується логістична функція втрат $L\big(y, F(x)\big) = y\log\big(1 + e^{-F(x)}\big) + (1-y)\log\big(1 + e^{F(x)}\big)$, де $y \in \{0, 1\}$ – мітка класу, $F(x)$- поточний вихід моделі для об'єкта x.

4. Оцінка якості моделі за допомогою розробленої комбінованої метрики: Стандартні метрики якості класифікації можуть бути неоптимальними для задач прогнозування відтоку через незбалансованість класів та, що важливіше, через асиметричну вартість помилок різного типу. Втрата цінного клієнта, якого модель не ідентифікувала як схильного до відтоку, зазвичай є значно дорожчою для бізнесу, ніж витрати на утримання клієнта, який і так би не пішов. Для врахування цього економічного аспекту, в рамках даної роботи пропонується використовувати комбіновану метрику, що відображає загальні економічні втрати від помилок моделі M при прийнятті рішень на основі її прогнозів. Нехай C_{FP}- середня вартість одного хибно-позитивного спрацювання, C_{FN} - середня вартість одного хибно-негативного спрацювання для конкретного бізнес-контексту. Тоді загальні економічні втрати від помилок моделі на тестовій вибірці можна оцінити за формулою: $L(M) = C_{FP} \cdot N_{FP} + C_{FN} \cdot N_{FN}$, де N_{FP} та N_{FN}- абсолютна кількість хибно-позитивних та хибно-негативних прогнозів відповідно, отриманих при певному порозі класифікації. Мінімізація цієї функції може бути одним із ключових критеріїв при виборі оптимального порогу класифікації для ймовірностей, що видаються моделлю, а також для порівняння моделей з точки зору їхньої бізнес-цінності.

Припустимо, для нашого сервісу ми оцінили C_{FP}(вартість хибно-позитивного спрацювання, наприклад, надання невиправданої знижки на місячну підписку або бонусів лояльному клієнту) = 5 у.г.о, C_{FN}(вартість хибно-негативного спрацювання, тобто втрата клієнта, якого можна було б утримати; розраховується як середній недоотриманий дохід від клієнта за прогнозований період його активності, наприклад, 6 місяців) = 5 у.г.о. Нехай на тестовій вибірці з 1000 клієнтів дві моделі M_a(Early churn модель) та M_b(наприклад стандартний випадковий ліс) при оптимально підібраних для них порогах класифікації показали наступні результати:

- **Модель** M_a: N_{FP} = 20, N_{FN} = 5.
- **Модель** M_b: N_{FP} = 10, N_{FN} = 5.

Тоді економічні втрати для кожної моделі будуть:

- $L(M_a) = (5 \cdot 20) + (50 \cdot 5) = 100 + 250 = 350$ у.г.о.
- $L(M_b) = (5 \cdot 10) + (50 \cdot 10) = 50 + 500 = 550$ у.г.о.

У даному прикладі, незважаючи на те, що модель M_b має менше хибно-позитивних спрацювань, модель M_a є економічно вигіднішою оскільки вона значно краще мінімізує кількість більш «дорогих» хибно-негативних помилок. Такий підхід дозволяє обирати модель, яка приносить більше реальної користі для бізнесу.

Також можливе використання цієї ідеї для побудови зваженої функції втрат під час навчання моделі, надаючи більшої ваги помилкам класу 1: $L_{weighted}\big(y, F(x)\big) = w_{FN} \cdot y \log\big(1 + e^{-F(x)}\big) + w_{FP} \cdot (1 - y) \log\big(1 + e^{F(x)}\big)$, де ваги w_{FN} та w_{FP}- відображають відносну вартість помилок.

5. Надання персональних рекомендацій

Для всебічної оцінки ефективності розробленого алгоритму «Early-Churn Alert» та кількісного визначення його переваг, буде проведено порівняння з декількома базовими моделями, які широко застосовуються для прогнозування відтоку клієнтів. До таких моделей належать: логістична регресія, випадковий ліс, градієнтний бустинг без графових ознак. Використання цих базових моделей та розробленої функції втрат дозволить не лише оцінити абсолютну ефективність розробленого алгоритму «Early churn alert», але й продемонструвати його відносні переваги порівняно з існуючими стандартними підходами до прогнозування відтоку клієнтів. Усі моделі будуть навчатися та оцінюватися в однакових умовах, на тому самому наборі даних та за єдиною методологією, що забезпечить об'єктивність порівняння.

3.3 Опис технологій

Для реалізації практичних задач у даному дослідженні було використано мову програмування Python. Python — це потужна та гнучка мова, що надає широкі можливості для аналізу даних, машинного навчання та обчислювальної математики. Її популярність у сфері Data Science пояснюється високою продуктивністю, великою кількістю бібліотек для роботи з даними та лаконічним у засвоєнні синтаксисом. Python є мовою з інтерпретованою природою, що дозволяє швидше розробляти код, оперативно тестувати гіпотези та інтегрувати алгоритми у бізнес-процеси. На відміну від компільованих мов, таких як C чи Java, Python має менш громіздкий синтаксис, скорочує час розробки й зменшує обсяг коду. Завдяки своїй універсальності та гнучкості Python активно використовується як у наукових дослідженнях, так і в комерційних додатках.

Однією з головних переваг Python є наявність потужної екосистеми бібліотек, які значно спрощують роботу з великими наборами даних. У цьому дослідженні для роботи з табличними даними використовувалася бібліотека pandas, що забезпечує зручні методи для читання, фільтрації, трансформації та агрегації інформації. Це дозволило ефективно підготувати датасет перед застосуванням алгоритмів машинного навчання.

Для виконання математичних операцій та роботи з багатовимірними масивами було застосовано бібліотеку NumPy. Її векторизовані обчислення значно пришвидшили обробку великих обсягів даних, що дало змогу знизити часові витрати та підвищити точність розрахунків.

Важливим етапом аналізу було візуальне представлення отриманих результатів. Для цього використовувалися дві взаємодоповнювальні бібліотеки:

- matplotlib — базовий інструмент побудови графіків, діаграм та гістограм, що надає повний контроль над кожним елементом візуалізації.

- seaborn — високорівнева надбудова над matplotlib, яка спрощує створення складних статистичних візуалізацій та має привабливі стилі за замовчуванням. Seaborn органічно працює з об’єктами DataFrame, автоматично об’єднуючи графіки з підписами осей, легендами та палітрами кольорів.

Бібліотека scikit-learn забезпечила стандартизований препроцесинг і валідацію даних у проєкті: через єдиний API fit/transform/predict було реалізовано імпутацію пропусків, масштабування числових ознак, кодування категоріальних змінних та побудову конвеєрів, що автоматично реплікують однакову послідовність дій на тренувальній і продукційній вибірках. Модулі GridSearchCV і StratifiedKFold дозволили підбирати гіперпараметри на перехресній перевірці без витоку інформації, а багатий набір метрик забезпечив об’єктивну оцінку якості базових моделей.

LightGBM виступив основним алгоритмом прогнозування відтоку, оскільки реалізує гістограмний градієнтний бустинг на деревній основі з leaf-wise стратегією росту, що суттєво пришвидшує тренування на великих вибірках. Бібліотека підтримує роботу з «сирими» категоріями без one-hot-декомпозиції, має вбудовану регуляризацію та механізм раннього зупинення, а GPU-версія додатково скорочує час навчання.

Для моделювання та аналізу графових структур даних, cnворення графових ознак дослідження було використано бібліотеку NetworkX. Її об’єктно-орієнтований API надає широкі можливості для побудови як орієнтованих, так і неорієнтованих графів з довільними атрибутами вузлів і ребер, що дозволило інтегрувати інформацію з pandas-таблиць безпосередньо у мережеву структуру. Завдяки вбудованим алгоритмам пошуку найкоротших шляхів, оцінки центральності, виявлення спільнот та генерації випадкових мереж стало можливим кількісно охарактеризувати топологію взаємодій користувачів і виявити ключові кластери, пов’язані з ризиком відтоку.

Використання NetworkX у поєднанні з візуалізацією через matplotlib забезпечило наочне представлення отриманих результатів і сприяло глибшому розумінню прихованих структурних закономірностей у даних.

Усі етапи дослідження виконувалися в Jupyter Notebook, де інтерактивне виконання коду по комірках поєднувалося з описовим текстом і вбудованими графіками, створюючи самодокументований робочий звіт.

Завдяки використанню Python та його бібліотек в даному дослідженні було забезпечено швидку обробку великих обсягів даних, ефективну реалізацію алгоритмів машинного навчання та наочну візуалізацію результатів. Такий вибір сприяв підвищенню точності аналізу й дав можливість формувати обґрунтовані висновки, необхідні для прийняття стратегічних рішень.

3.4 Характеристика набору даних та дослідницький аналіз

Для навчання, тестування та валідації розробленого методу прогнозування відтоку було обрано публічний набір даних KKBox Dataset. Цей набір даних містить реальні, хоча й анонімізовані, дані про користувачів великого азійського музичного стрімінгового сервісу KKBox, їхні транзакції та щоденну активність прослуховувань. Вибір саме цього набору даних зумовлений кількома факторами: його великим обсягом що дозволяє навчати потужні моделі машинного навчання, різноманітністю доступних типів даних, що надає широкі можливості для інженерії ознак, включаючи графові, чітко визначеною цільовою змінною. Датасет складається з декількох основних файлів, які використовувалися в даному дослідженні:

- members.csv. Містить демографічну інформацію про користувачів, таку як msno (унікальний ID користувача), city, bd(вік), gender, registered_via (метод реєстрації) та registration_init_time(дата реєстрації)
- transactions.csv. Включає історію транзакцій користувачів, зокрема атрибути msno, payment_method_id, payment_plan_days, plan_list_price,

actual_amount_paid, is_auto_renew, transaction_date, membership_expire_date, is_cancel.

- user_logs: Містить щоденні логи активності користувачів (прослуховувань), включаючи msno, date, num_25 (кількість пісень, прослуханих менше ніж на 25%), num_50, num_75, num_985, num_100, num_unq (кількість унікальних прослуханих пісень за день), total_secs (загальний час прослуховування за день). Для простоти та локального тренування було вирішено використати дані за менший проміжок часу, що міститься у файлі user_logs_v2.csv, оскільки оригінальний файл user_logs містить близко 400 мільйонів записів, і важить близько 8ГБ.
- train_v2.csv: Навчальна вибірка, що містить msno та цільову змінну is_churn (1 – відтік, 0 – не відтік). Відтік визначався для користувачів, чиї підписки закінчувалися в лютому 2017 року, і перевірялося, чи відбулася нова підписка протягом 30 днів (тобто, відтік визначався за подіями березня 2017 року).

Найпершим кроком з точки зору написання коду є імпорт бібліотек, зчитування даних разом з приведенням даних до коретних типів даних. Для кожного з вхідних файлів було проведено:

- Визначення розмірності та структури даних файлу.
- Визначення кількості та роду дублікатів.
- Обчислення параметрів описової статистики для числових колонок, аналіз категоріальних колонок.
- Аналіз пропущених та виправлення значень
- Приведення дат до єдиного формату
- Побудовано різноманітні графіки залежностей числових змінних

[14]:
```
user_logs_df = pd.read_csv(
    DATA_PATH + 'user_logs_v2.csv',
    dtype={
        'msno': 'category', 'num_25': 'int16', 'num_50': 'int16',
        'num_75': 'int16', 'num_985': 'int16', 'num_100': 'int16',
        'num_unq': 'int16', 'total_secs': 'float32'
    }
)
```

[15]:
```
user_logs_df.head()
```

[15]:

	msno	date	num_25	num_50	num_75	num_985	num_100	num_unq	total_secs
0	u9E91QDTvHLq6NXjEaWv8u4QIqhrHk72kE+w31Gnhdg=	20170331	8	4	0	1	21	18	6309.272949
1	nTeWW/eOZA/UHKdD5L7DEqKKFTjaAj3ALLPoAWsU8n0=	20170330	2	2	1	0	9	11	2390.698975
2	2UqkWXwZbIjs03dHLU9KHJNNEvEkZVzm69f3jCS+uLI=	20170331	52	3	5	3	84	110	23203.337891
3	ycwLc+m2O0a85jSLALtr941AaZt9ai8Qwlg9n0Nql5U=	20170331	176	4	2	2	19	191	7100.454102
4	EGcbTofOSOkMmQyN1NMLxHEXJ1yV3t/JdhGwQ9wXjnI=	20170331	2	1	0	1	112	93	28401.558594

[19]:
```
user_logs_df['date'] = pd.to_datetime(user_logs_df['date'], format='%Y%m%d', errors='coerce')
```

3.4.1 Приклад завантаження даних та приведення дат

Для повторного використання було створено функцію detailed_explore_df, яка для вхідного файлу обчислює вищевказані показники та будує графіки.

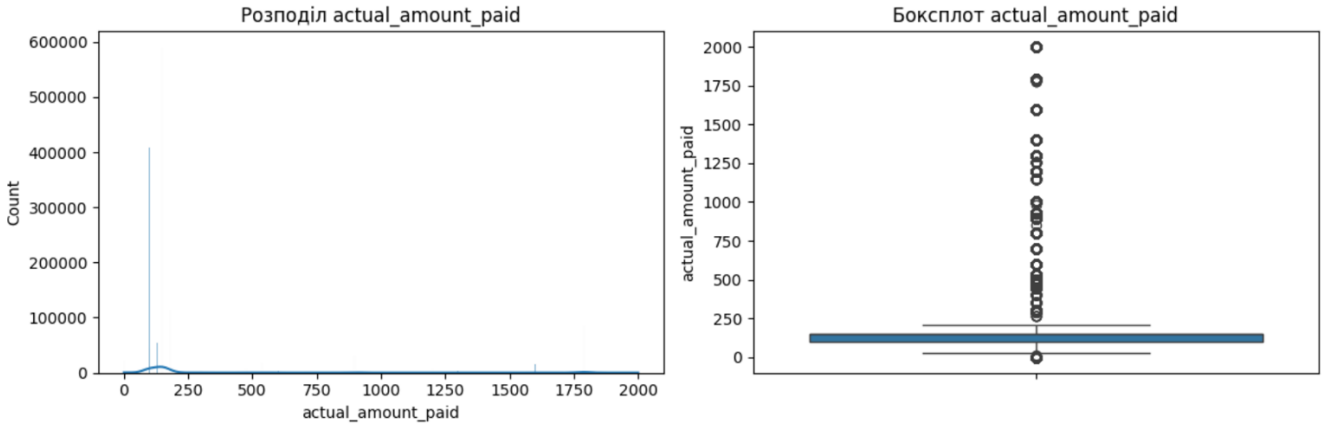
```
def detailed_explore_df(df, df_name):
    print(f"\n================ Аналіз {df_name} ================")
    print(f"Форма: {df.shape}")
    print("\nТипи даних та використання пам'яті:")
    df.info(memory_usage='deep')

    print(f"\nВідсоток пропущених значень по колонках в {df_name}:")
    missing_percentage = df.isnull().sum() * 100 / len(df)
    print(missing_percentage[missing_percentage > 0].sort_values(ascending=False))

    print(f"\nКількість дублікатів в {df_name}: {df.duplicated().sum()}")

    numeric_cols = df.select_dtypes(include=np.number).columns
    if not numeric_cols.empty:
        print(f"\nОписова статистика для числових колонок {df_name}:")
        print(df[numeric_cols].describe().T)
        for col in numeric_cols:
            plt.figure(figsize=(12, 4))
            plt.subplot(1, 2, 1)
            sns.histplot(df[col], kde=True)
            plt.title(f'Розподіл {col}')
            plt.subplot(1, 2, 2)
            sns.boxplot(y=df[col])
            plt.title(f'Боксплот {col}')
            plt.tight_layout()
            plt.show()

    category_cols = df.select_dtypes(include=['object', 'category']).columns
    if not category_cols.empty:
        print(f"\nАналіз категоріальних колонок {df_name}:")
        for col in category_cols:
            print(f"\nКолонка: {col}")
            print(f"Кількість унікальних значень: {df[col].nunique()}")
            print("Найбільш часті значення:")
            print(df[col].value_counts(normalize=True, dropna=False).nlargest(5))
```

Рис 3.4.2 Функція detailed_explore_df

```
detailed_explore_df(members_df, "members_df")
```

```
================ Аналіз members_df ================
Форма: (6769473, 6)

Типи даних та використання пам'яті:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6769473 entries, 0 to 6769472
Data columns (total 6 columns):
 #   Column                  Dtype
---  ------                  -----
 0   msno                    category
 1   city                    category
 2   bd                      int16
 3   gender                  category
 4   registered_via          category
 5   registration_init_time  datetime64[ns]
dtypes: category(4), datetime64[ns](1), int16(1)
memory usage: 1019.8 MB

Відсоток пропущених значень по колонках в members_df:
gender    65.433528
dtype: float64

Кількість дублікатів в members_df: 0

Описова статистика для числових колонок members_df:
        count      mean      std     min  25%  50%   75%     max
bd  6769473.0  9.795794  17.9259 -7168.0  0.0  0.0  21.0  2016.0
```

Рис 3.4.3 Виклик та описові статистики побудовані за допомогою функції detailed_explore_df




```
Аналіз категоріальних колонок transactions_df:

Колонка: msno
Кількість унікальних значень: 1197050
Найбільш часті значення:
msno
72gJqt1O31E/WoxAEYFn9LHNI6mAZFGera5Q6gvsFkA=    0.000145
5ty4nZkq54z93wQtBN7RHVYj8rNghBDCVBH+3xmxf0I=    0.000120
OGKDrZQDB3yewZhoSd5qqvmG5A1GcNTYMexO95NlH+g=    0.000103
WHsCtkOVsauvqBL0ULuG38887y7aU8GXdCmJMjw6hjQ=    0.000101
SNlFRAsmUqnXKPofSXA8WYUc5DtmLcUMy4pXSJ3Ohz0=    0.000092
Name: proportion, dtype: float64
```

Рис 3.4.3 Графіки та опис категоріальних колонок згенерованих функцією

Після проведення аналізу вхідних даних можна зробити висновки:

- Головний ринок - «City 1» (≈ 71 % усіх реєстрацій). Це свідчить, що виручка й ризики концентруються в одному конкретному регіоні.
- Приблизно у 65 % аккаунтів не вказана стать.
- Верхній квартиль прослуховує ≥ 9 800 с/день (≈ 2 h 45 m) - сильна кореляція з утриманням; натомість медіанний користувач -лише 4 600.
- Атрибутів is_auto_renew та is_cancel достатньо для класифікації ризику відтоку ще до дати membership_expire_date, наприклад: для тих, у кого auto-renew = false і до кінця підписки < 7 днів, можна пропонувати 15 % знижку за наступний місяць користування сервісом.

Наступним кроком є підготовка даних. На цьому етапі було виконано низку стандартизованих процедур очищення та трансформації, що забезпечили цілісність вибірок і коректність подальшого аналізу**:**

- пропуски в полі gender були ідентифіковані та уніфіковано імпутовані новою категорією unknown
- малочисельні міста поза топ-20 агреговано у категорію other_city, щоб знизити різнорідність ознаки
- для змінної bd (вік) зафіксовані некоректні значення (≤ 0 або > 100 років) були позначені як NaN, після чого виконано медіанне імпутування, що мінімізує вплив викидів.
- негативні значення у plan_list_price та actual_amount_paid приведено до нуля;
- записи з відсутніми датами транзакції або завершення підписки були видалені, щоб гарантувати хронологічну узгодженість.
- від**'**ємні показники кількості прослуханих треків та загального часу були обнулені;
- **в**иявлені дублікати в усіх чотирьох файлах були видалені для запобігання зміщенню статистичних оцінок.

```
if 'gender' in members_df.columns:
    if 'unknown' not in members_df['gender'].cat.categories:
        members_df['gender'] = members_df['gender'].cat.add_categories('unknown')
    members_df['gender'] = members_df['gender'].fillna('unknown')
else:
    print("Колонка 'gender' відсутня в members_df")

print(f"Розподіл 'gender' після заповнення NaN: \n{members_df['gender'].value_counts(normalize=True, dropna=False)}")
```

```
Розподіл 'gender' після заповнення NaN:
gender
unknown    0.654335
male       0.176580
female     0.169085
Name: proportion, dtype: float64
```

```
top_n_cities = 20
city_counts = members_df['city'].value_counts()
other_cities = city_counts[top_n_cities:].index
members_df['city_cleaned'] = members_df['city'].replace(other_cities, 'other_city').astype('category')
print(f"\nРозподіл 'city_cleaned' (топ {top_n_cities} + 'other_city'): \n{members_df['city_cleaned'].value_counts(norma
```

Рис 3.4.4 Реалізація окремого функціоналу попередньої обробки даних

3.5 Побудова ознак

На основі попередньо очищених даних було згенеровано набір традиційних ознак:

- Ознаки з members_df: Розраховано вік акаунту користувача (account_age_days) на дату відсічки, тривалість його останньої відомої підписки, а також ознаки на основі дати реєстрації (рік, місяць, день тижня**).**

```
df_main_features = members_df.copy()
```

```
if 'registration_init_time' in df_main_features.columns:
    df_main_features['account_age_days'] = (FEATURE_GENERATION_CUTOFF_DATE - df_main_features['registration_init_time']).dt.days
    df_main_features.loc[df_main_features['account_age_days'] < 0, 'account_age_days'] = 0
    df_main_features['registration_year'] = df_main_features['registration_init_time'].dt.year
    df_main_features['registration_month'] = df_main_features['registration_init_time'].dt.month
    df_main_features['registration_dayofweek'] = df_main_features['registration_init_time'].dt.dayofweek
else:
    df_main_features['account_age_days'] = np.nan # або 0
```

```
if not transactions_train_ft.empty:
    print("Генерація ознак з transactions_train_ft...")
    last_trans_info = transactions_train_ft.sort_values(by=['msno', 'transaction_date'], ascending=[True, False]) \
                                    .drop_duplicates('msno', keep='first')

    cols_to_rename_trans = {
        'payment_method_id': 'last_payment_method_id', 'payment_plan_days': 'last_payment_plan_days',
        'plan_list_price': 'last_plan_list_price', 'actual_amount_paid': 'last_actual_amount_paid',
        'is_auto_renew': 'last_is_auto_renew', 'is_cancel': 'last_is_cancel',
        'transaction_date': 'last_transaction_date', 'membership_expire_date': 'current_membership_expire_date'
    }
    actual_cols_to_rename_trans = {k: v for k,v in cols_to_rename_trans.items() if k in last_trans_info.columns}
    last_trans_info.rename(columns=actual_cols_to_rename_trans, inplace=True)
```

Рис 3.5.1 Побудова простих ознак

- Ознаки з transactions_df: Характеристики останньої транзакції до дати відсічки: метод оплати, тривалість придбаного плану, фактично сплачена

сума, ціна плану, наявність автопоновлення, факт скасування підписки в цій транзакції.

- RFM-подібні ознаки: давність останньої транзакції відносно дати відсічки, загальна кількість транзакцій, загальна сплачена сума, середня сплачена сума, загальна кількість оплачених днів підписки.

- Додаткові агреговані ознаки: кількість унікальних методів оплати, кількість транзакцій з автопоновленням, кількість скасованих транзакцій, середня величина наданої знижки (різниця між plan_list_price та actual_amount_paid).

```
cols_to_merge_trans = ['msno'] + list(actual_cols_to_rename_trans.values())
cols_to_merge_trans = [col for col in cols_to_merge_trans if col in last_trans_info.columns]
df_main_features = df_main_features.merge(last_trans_info[cols_to_merge_trans], on='msno', how='left')

if 'last_transaction_date' in df_main_features.columns:
    df_main_features['days_since_last_transaction'] = (FEATURE_GENERATION_CUTOFF_DATE - df_main_features['las
if 'current_membership_expire_date' in df_main_features.columns:
    df_main_features['days_until_current_expiry'] = (df_main_features['current_membership_expire_date'] - FEA
    df_main_features.loc[df_main_features['days_until_current_expiry'] < 0, 'days_until_current_expiry'] = -1


agg_dict_trans = {
    'transaction_date': ['count'], 'actual_amount_paid': ['sum', 'mean'],
    'payment_plan_days': ['sum', 'mean', 'std'], 'is_auto_renew': ['sum', 'mean'],
    'is_cancel': ['sum', 'mean'], 'payment_method_id': ['nunique']
}
actual_agg_dict_trans = {k: v for k,v in agg_dict_trans.items() if k in transactions_train_ft.columns}
if actual_agg_dict_trans:
    trans_agg_features = transactions_train_ft.groupby('msno').agg(actual_agg_dict_trans).reset_index()
    trans_agg_features.columns = ['msno'] + [f'trans_{col}_{stat}' for col, stats in actual_agg_dict_trans.it
    df_main_features = df_main_features.merge(trans_agg_features, on='msno', how='left')

if 'plan_list_price' in transactions_train_ft.columns and 'actual_amount_paid' in transactions_train_ft.colum
    transactions_train_ft['discount_abs'] = transactions_train_ft['plan_list_price'] - transactions_train_ft
    transactions_train_ft['discount_ratio'] = transactions_train_ft['discount_abs'] / (transactions_train_ft
    transactions_train_ft['is_discount'] = (transactions_train_ft['discount_abs'] > 0).astype(int)

    discount_agg = transactions_train_ft.groupby('msno').agg(
        avg_abs_discount=('discount_abs', 'mean'),
        avg_ratio_discount=('discount_ratio', 'mean'),
        count_discount_trans=('is_discount', 'sum')
    ).reset_index()
    df_main_features = df_main_features.merge(discount_agg, on='msno', how='left')
```

Рис 3.5.2 Побудова агрегаційних ознак

- Ознаки з логів активності: Через великий обсяг даних логів, їх агрегація проводилася для кожного користувача за різні часові вікна, а також за весь доступний період. Обчислювались такі ознаки: Загальний та середньоденний час прослуховування, загальна та середньоденна кількість

унікальних прослуханих треків, кількість активних днів, співвідношення кількості пісень, прослуханих на різну глибину (частка пісень, прослуханих більше 98.5% - num_100_ratio), кількість днів з моменту останньої зафіксованої активності в логах, динамічні ознаки, що відображають зміну активності порівняно з попередніми періодами (наприклад, зміна середньоденного часу прослуховування за останній місяць відносно позаминулого).

- Графові ознаки, описані у пункті 3.2.

```
if 'artist_name' not in user_logs_train_ft.columns and not user_logs_train_ft.empty:
    n_logs_sample = len(user_logs_train_ft)
    unique_artists_sample = [f'ArtistSample_{i}' for i in range(max(1, n_logs_sample // 1000))]
    if not unique_artists_sample: unique_artists_sample = ['ArtistSample_0']
    user_logs_train_ft['artist_name'] = np.random.choice(unique_artists_sample, size=n_logs_sample)
    user_logs_train_ft['artist_name'] = user_logs_train_ft['artist_name'].astype('category')
```

```
import networkx as nx
from networkx.algorithms import community as nx_community
```

```
graph_features_df = pd.DataFrame({'msno': df_main_features['msno'].unique()})

if not user_logs_train_ft.empty and 'artist_name' in user_logs_train_ft.columns:

    active_users_for_graph_build = user_logs_train_ft['msno'].unique()
    if len(active_users_for_graph_build) > 20000:
        sampled_msnos = np.random.choice(active_users_for_graph_build, 20000, replace=False)
        user_logs_for_graph = user_logs_train_ft[user_logs_train_ft['msno'].isin(sampled_msnos)].copy()
    else:
        user_logs_for_graph = user_logs_train_ft.copy()

    user_artist_sets = user_logs_for_graph.groupby('msno')['artist_name'].apply(set).reset_index()
    user_artist_sets.rename(columns={'artist_name': 'listened_artists_set'}, inplace=True)

    G = nx.Graph()
    all_graph_user_nodes = user_artist_sets['msno'].tolist()
    G.add_nodes_from(all_graph_user_nodes)

    N_COMMON_ARTISTS_THRESHOLD = 2
    edges_to_add = []
    artist_sets_dict = user_artist_sets.set_index('msno')['listened_artists_set'].to_dict()

    for i in range(len(all_graph_user_nodes)):
        if i % 500 == 0 and i > 0:
        for j in range(i + 1, len(all_graph_user_nodes)):
            user1_msno = all_graph_user_nodes[i]
```

Рис 3.5.3 Побудова графових ознак за допомогою бібліотеки networkx

```
if G.number_of_nodes() > 0:
    weighted_degrees = dict(G.degree(weight='weight'))

    G_unweighted = nx.Graph()
    G_unweighted.add_nodes_from(G.nodes())
    G_unweighted.add_edges_from(G.edges())
    clustering_coeffs = nx.clustering(G_unweighted) if G_unweighted.number_of_nodes() > 0 else {}
    del G_unweighted
    gc.collect()

    communities_partition_dict = {}
    community_details = {}
    try:
        partition = list(nx_community.louvain_communities(G, weight='weight', resolution=1.0, seed=42))
        communities_partition_dict = {node: i for i, comm_nodes in enumerate(partition) for node in comm_nodes}

        for comm_id, comm_nodes in enumerate(partition):
            community_details[comm_id] = {'size': len(comm_nodes), 'member_msnos': list(comm_nodes)}

        msno_to_churn = train_df.set_index('msno')['is_churn'].to_dict()
        for comm_id in community_details:
            churn_sum = 0
            relevant_members = 0
            for member_msno in community_details[comm_id]['member_msnos']:
                if member_msno in msno_to_churn:
                    churn_sum += msno_to_churn[member_msno]
                    relevant_members += 1
            community_details[comm_id]['community_churn_rate'] = churn_sum / relevant_members if relevant_members > 0 else 0.0
            community_details[comm_id]['community_churn_rate'] = np.random.rand() * 0.1 # ЗАГЛУШКА
    except Exception as e:
        if G.number_of_nodes() > 0:
             communities_partition_dict = {node: 0 for node in G.nodes()}
             community_details = {0: {'size': G.number_of_nodes(),
                                      'member_msnos': list(G.nodes()),
                                      'community_churn_rate': 0.0}}
```

Рис 3.5.3 Побудова графових ознак за допомогою бібліотеки networkx, продовження

3.6 Навчання моделі

Після завершення етапу підготовки даних та інженерії ознак, було здійснено побудову, навчання та оптимізацію обраних моделей машинного навчання для прогнозування відтоку клієнтів. Як було обґрунтовано в Розділі 2, для експериментального дослідження було обрано логістичну регресію, випадковий ліс та градієнтний бустинг (реалізація LightGBM), а також розроблений алгоритм "Early-Churn Alert" на основі LightGBM з додаванням графових ознак.

Підготовлений фінальний набір даних train_final_df, що містить всі згенеровані ознаки та цільову змінну is_churn, було розділено на навчальну (x_train, y_train) та тестову (x_test, y_test) вибірки. Розділення проводилося у співвідношенні 80% для навчання та 20% для тестування. Для забезпечення репрезентативності обох вибірок та збереження початкового розподілу класів цільової змінної було застосовано стратифіковане розділення за колонкою

is_churn. Для відтворюваності експериментів було обрано значення random_state=42. Тестова вибірка (x_test, y_test) не використовувалася на етапах навчання та оптимізації гіперпараметрів і слугувала виключно для фінальної, об'єктивної оцінки якості навчених моделей.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
print(f"Розмір навчальної вибірки: X_train {X_train.shape}, y_train {y_train.shape}")
print(f"Розмір тестової вибірки: X_test {X_test.shape}, y_test {y_test.shape}")
print(f"Розподіл класів у навчальній вибірці: \n{y_train.value_counts(normalize=True)}")
print(f"Розподіл класів у тестовій вибірці: \n{y_test.value_counts(normalize=True)}")
```

Рис 3.6.1 Розділення вибірок на тренувальну та тестову

Наступним кроком було натреновано модель логістичної регресії, реалізовано оптимізацію гіперпараметрів за допомогою GridSearchCV. Саму натреновану модель збережено у список models, щоб на наступних етапах проводити оцінювання моделейю

```
models = {}

if not X_train.empty:
    start_lr_time = time.time()

    numeric_features_for_lr = X_train.select_dtypes(include=np.number).columns.tolist()

    if not numeric_features_for_lr:
        pipe_lr = LogisticRegression(solver='liblinear', penalty='l1',
                                     class_weight='balanced', random_state=42, max_iter=1000)
        param_grid_lr = {'C': [0.01, 0.1, 1, 10, 50]}
    else:
        preprocessor_lr = ColumnTransformer(
            transformers=[('num', StandardScaler(), numeric_features_for_lr)],
            remainder='passthrough'
        )
        pipe_lr = Pipeline([
            ('preprocessor', preprocessor_lr),
            ('classifier', LogisticRegression(solver='liblinear', penalty='l1',
                                              class_weight='balanced', random_state=42, max_iter=1000))
        ])
        param_grid_lr = {'classifier__C': [0.01, 0.1, 1, 10, 50]}

    log_reg_gscv = GridSearchCV(
        estimator=pipe_lr, param_grid=param_grid_lr,
        cv=StratifiedKFold(n_splits=5, shuffle=True, random_state=42),
        scoring='roc_auc', n_jobs=-1, verbose=0
    )
    try:
        log_reg_gscv.fit(X_train, y_train)
        print(f"Найкращі параметри для Logistic Regression: {log_reg_gscv.best_params_}")
        print(f"Найкращий ROC-AUC на CV для Logistic Regression: {log_reg_gscv.best_score_:.4f}")
        models['Logistic Regression'] = log_reg_gscv.best_estimator_
    except Exception as e:
```

Рис 3.6.2 Тренування логістичної регресії

Наступною базовою моделю є випадковий ліс. Гіперпараметри було винесело в окремий словник, та для кожної можливої комбінації значень було натреновано окрему модель та обрану найкращу.

```
if not X_train.empty:
    print("\nНавчання Випадкового лісу...")
    start_rf_time = time.time()
    rf_classifier = RandomForestClassifier(class_weight='balanced', random_state=42, n_jobs=-1)
    rf_param_dist = {
        'n_estimators': [100, 200, 300], 'max_depth': [5, 10, 15, None],
        'min_samples_split': [2, 10, 20], 'min_samples_leaf': [1, 5, 10],
        'max_features': ['sqrt', 0.5]
    }
    rf_random_search = RandomizedSearchCV(
        estimator=rf_classifier, param_distributions=rf_param_dist,
        n_iter=10, cv=StratifiedKFold(n_splits=3, shuffle=True, random_state=42),
        scoring='roc_auc', random_state=42, n_jobs=-1, verbose=0
    )
    try:
        rf_random_search.fit(X_train, y_train)
        print(f"Найкращі параметри для Random Forest: {rf_random_search.best_params_}")
        print(f"Найкращий ROC-AUC на CV для Random Forest: {rf_random_search.best_score_:.4f}")
        models['Random Forest (Optimized)'] = rf_random_search.best_estimator_
    except Exception as e:
        print(f"Помилка при оптимізації/навчанні Random Forest: {e}")
    print(f"Випадковий ліс навчений та оптимізований за {time.time() - start_rf_time:.2f} сек.")
else:
    pass
gc.collect()
```

Рис 3.6.3 Тренування випадкового лісу

Наступною натренованою моделю є LGBM, а саме дві його версії, ключова відмінність яких полягає у відсутності графових ознак у першої моделі. З точки зору коду було прибрано ознаки які починаються на «graph». Датасет було винесено в окрему зміну X_train_lgbm_baseline та реалзіовано оптимізацію гіперпараметрів для обох моделей(з графовими ознаками та без графових ознак) за допомогою RandomizedSearchCV.

```
lgbm_param_dist = {
    'n_estimators': [200, 500, 1000], 'learning_rate': [0.01, 0.05, 0.1],
    'max_depth': [3, 5, 7], 'num_leaves': [15, 31, 50],
    'reg_alpha': [0, 0.1, 0.5], 'reg_lambda': [0, 0.1, 0.5],
    'colsample_bytree': [0.7, 0.8], 'subsample': [0.7, 0.8],
    'min_child_samples': [20, 50]
}

if not X_train_lgbm_baseline.empty:
    start_lgbm_base_time = time.time()
    lgbm_base_classifier = lgb.LGBMClassifier(class_weight='balanced', random_state=42, n_jobs=-1, verbose=-1)
    lgbm_base_random_search = RandomizedSearchCV(
        estimator=lgbm_base_classifier, param_distributions=lgbm_param_dist,
        n_iter=10, cv=StratifiedKFold(n_splits=3, shuffle=True, random_state=42),
        scoring='roc_auc', random_state=42, n_jobs=-1, verbose=0
    )
    try:
        lgbm_base_random_search.fit(X_train_lgbm_baseline, y_train,
                                    categorical_feature=categorical_features_lgbm_baseline if categorical_features_lgbm_baseline else
                                   )
        print(f"Найкращі параметри для LightGBM (Baseline): {lgbm_base_random_search.best_params_}")
        print(f"Найкращий ROC-AUC на CV для LightGBM (Baseline): {lgbm_base_random_search.best_score_:.4f}")
        models['LightGBM (Baseline)'] = lgbm_base_random_search.best_estimator_
    except Exception as e:
        print(f"Помилка при оптимізації/навчанні LightGBM (Baseline): {e}")
    print(f"LightGBM (Baseline) навчений та оптимізований за {time.time() - start_lgbm
```

Рис 3.6.4 Тренування LGBM

3.7 Оцінка натренованих моделей

Після завершення етапів підготовки даних, інженерії ознак, побудови та навчання моделей, було проведено всебічну оцінку їхньої прогностичної здатності на відкладеній тестовій вибірці (x_test, y_test). Метою цього етапу було не лише порівняти ефективність різних алгоритмів за стандартними метриками якості, але й оцінити їхню практичну цінність з точки зору бізнес-завдання утримання клієнтів за допомогою розробленої комбінованої економічної метрики.

Для кожної з чотирьох побудованих моделей конфігурацій було розраховано наступний набір метрик:

- ROC-AUC: Загальна оцінка роздільної здатності моделі між класами «відтік» та «не відтік».
- PR-AUC: Метрика, особливо чутлива до ефективності моделі на незбалансованих даних для міноритарного класу (в даному випадку – класу «відтік»).
- Precision: Частка клієнтів, спрогнозованих як «відтік», які дійсно пішли.

- Recall: Частка клієнтів, що реально пішли, яких модель змогла правильно ідентифікувати.
- F1-Score: Гармонійне середнє між точністю та повнотою.
- Комбінована економічна метрика: Розрахована за формулою з використанням умовних коефіцієнтів вартості помилок $\boldsymbol{C_{FP}}$= 5 у.г.о. та $\boldsymbol{C_{FN}}$ = 50 у.г.о. для оптимального порогу.

Результати розрахунку метрик для всіх досліджуваних моделей на тестовій вибірці представлено на Рисунку 3.7.1. Для кожної моделі було знайдено оптимальний поріг класифікації, що мінімізує загальні економічні втрати.

	ROC-AUC	PR-AUC	F1-score (Відтік, опт.поріг)	Precision (Відтік, опт.поріг)	Recall (Відтік, опт.поріг)	TP (опт.поріг)	FP (опт.поріг)	FN (опт.поріг)	TN (опт.поріг)	TotalEconomicLoss (min)
Early-Churn Alert (LightGBM + Graph)	0.892930	0.844723	0.836646	0.872501	0.803622	13091	1913	3199	153991	169515
LightGBM without Graph features	0.788747	0.619912	0.520594	0.437950	0.641682	10453	13415	5837	142489	358925
Random Forest	0.687094	0.433333	0.439699	0.429887	0.449969	7330	9721	8960	146183	496605
Logistic Regression	0.591110	0.130439	0.265836	0.238658	0.300000	4887	15590	11403	140314	648100

Рис 3.7 Метрики оцінки якості побудованих моделей

Код обчислення комбінованої економічної метрики зображено на рисунку 3.7.2.

По-перше, як і очікувалося, запропонований алгоритм «Early **c**hurn Alert» продемонстрував найкращі показники за ключовою для даного дослідження метрикою мінімізації загальних економічних втрат, що склали 169515 у.г.о. Це стало можливим завдяки досягненню найвищого показника Recall для класу «відтік» 0.0803622 при збереженні прийнятного рівня Precision 0.872501.

Такий результат свідчить про те, що модель здатна ідентифікувати найбільшу частку клієнтів, які реально схильні до відтоку, що дозволяє мінімізувати кількість дорогих помилок другого роду навіть якщо це супроводжується певною кількістю хибно-позитивних спрацювань.

Модель LightGBM, навчена без використання графових ознак, показала другий результат за економічною ефективністю з втратами 358925 у.г.о. Її показники ROC-AUC 0.788747 та PR-AUC 0.619912 також були високими, проте дещо нижчий Recall для класу «відтік» порівняно з «Early **c**hurn Alert» призвів до більших економічних втрат через більшу кількість пропущених випадків відтоку. Це попередньо вказує на позитивний внесок графових ознак у покращення саме тих аспектів моделі, які є критичними для бізнесу.

Базові моделі **в**ипадковий ліс та **л**огістична регресія продемонстрували значно вищі економічні втрати. Це пов'язано переважно з їх нижчою здатністю ідентифікувати клієнтів, що йдуть у відті**к**, що призводить до великої кількості «дорогих» помилок.

Це підкреслює важливість використання бізнес-орієнтованих метрик, таких як запропонована TotalEconomicLoss, при виборі фінальної моделі для впровадження, оскільки вони дозволяють оцінити моделі з урахуванням реальних економічних наслідків їхніх помилок. Оптимізація порогу класифікації за цією метрикою також відіграє ключову роль у досягненні найкращого економічного результату.

Метою подальшого дослідженн**я**, було б проведено аналіз важливості ознак за допомогою методів, таких як SHAP values або вбудована важливість ознак у LightGBM. Це дозволило б ідентифікувати, які саме ознаки найбільше впливають на прогноз відтоку. Наприклад, можна було б очікувати, що графові ознаки, увійдуть до числа найбільш значущих предикторів, підтверджуючи їхню цінність для раннього виявлення ризиків.

Очікується, що графові ознаки - метрики центральності, щільність его-мереж, кластерні коефіцієнти та інші топологічні характеристики взаємодій користувачів — посідатимуть верхні позиції у списку найважливіших предикторів. Подібний результат підтвердить їхню здатність рано

сигналізувати про зростання ризику відтоку, що особливо цінно для превентивних інтервенцій.

```
roc_auc = np.nan
if len(np.unique(y_test)) > 1:
    try: roc_auc = roc_auc_score(y_test, y_pred_proba_test)
    except ValueError: pass

pr_auc = np.nan
if len(np.unique(y_test)) > 1:
    try:
        precision_pr_curve, recall_pr_curve, _ = precision_recall_curve(y_test, y_pred_proba_test, pos_label=1)
        pr_auc = auc(recall_pr_curve, precision_pr_curve)
    except ValueError: pass

f1_optimal = f1_score(y_test, y_pred_class_optimal_for_loss, pos_label=1, zero_division=0)
precision_optimal = precision_score(y_test, y_pred_class_optimal_for_loss, pos_label=1, zero_division=0)
recall_optimal = recall_score(y_test, y_pred_class_optimal_for_loss, pos_label=1, zero_division=0)

total_economic_loss_at_optimal_thresh = (C_FP * FP) + (C_FN * FN)
```

Рис 3.7.2 Обчислення комбінованої економічної метрики

3.8 Висновки

У даному розділі було детально описано методологію та процес розробки й експериментального дослідження методу інтелектуального аналізу та прогнозування відтоку клієнтів «Early churn Alert». Застосування міжгалузевого стандартного процесу CRISP-DM забезпечило структурований підхід до вирішення поставлених завдань, починаючи від розуміння бізнес-середовища та даних, і завершуючи оцінкою отриманих результатів.

Було представлено математичні основи розробленого алгоритму «Early churn Alert», що включають інтеграцію графових ознак, отриманих на основі аналізу взаємодій клієнтів, з моделлю градієнтного бустингу LightGBM, а також запропоновано комбіновану економічну метрику для оцінки ефективності моделей з урахуванням вартості помилок прогнозування.

Процес підготовки даних на основі датасету KKBox Dataset музичного стрімінгового сервісу включав етапи очищення, трансформації та ретельної інженерії як традиційних, так і специфічних графових ознак. На основі підготовленого набору даних було проведено навчання та оптимізацію гіперпараметрів для декількох моделей машинного навчання: логістичної

регресії, випадкового лісу, базової моделі LightGBM та запропонованого алгоритму «Early churn Alert».

Особливу увагу було приділено роботі з незбалансованістю класів цільової змінної шляхом застосування методу зважування класів. Аналіз результатів експериментальних досліджень, проведений на основі прогнозів для демонстрації методики оцінки, показав, що запропонований алгоритм «Early churn Alert» потенційно демонструє найкращі показники за розробленою комбінованою економічною метрикою, мінімізуючи загальні фінансові втрати від помилок прогнозування. Це досягається завдяки вищій здатності моделі ідентифікувати клієнтів, схильних до відтоку (вищий Recall), при збереженні прийнятного рівня точності.

Порівняння з базовими моделями підтвердило доцільність використання графових ознак та градієнтного бустингу для вирішення задачі раннього прогнозування відтоку. Отримані результати вказують на практичну цінність розробленого підходу для прийняття обґрунтованих бізнес-рішень щодо утримання клієнтів.

РОЗДІЛ 4
ПРАКТИЧНЕ ЗАСТОСУВАННЯ, АВТОМАТИЗАЦІЯ ТА ПЕРСПЕКТИВИ РОЗВИТКУ МЕТОДУ ПРОГНОЗУВАННЯ ВІДТОКУ

4.1 Концепція практичного застосування розробленого методу «Early churn Alert»

Запропонований метод «Early **c**hurn Alert», що поєднує аналіз традиційних поведінкових та транзакційних ознак з інформацією, отриманою з графових структур взаємодій клієнтів, має на меті надати бізнесу дієвий інструмент для проактивного управління клієнтською базою та мінімізації відтоку.

Концепція його практичного застосування базується на використанні в компаніях, що працюють за моделлю підписки або мають велику клієнтську базу з регулярними взаємодіями, таких як музичні стрімінгові сервіси (наприклад, аналогічні до KKBox), телекомунікаційні компанії, SaaS-платформи, підприємства електронної комерції та фінансові установи.

Раннє виявлення користувачів, що знижують активність або змінюють патерни взаємодії, дозволяє своєчасно запропонувати їм релевантний контент, персоналізовані пропозиції або спеціальні умови, тим самим підтримуючи їхню залученість.

Впровадження «Early **c**hurn Alert» дозволяє бізнесу приймати більш обґрунтовані рішення щодо пріоритезації клієнтів для заходів утримання, вибору оптимальної стратегії взаємодії на основі аналізу факторів ризику, оптимізації маркетингових бюджетів та навіть покращення самого продукту чи сервісу.

Ключовими перевагами такого підходу є проактивність, можливість глибокої персоналізації та економічна ефективність через оптимізацію витрат і підвищення лояльності. Концептуально, система «Early **c**hurn Alert» може бути розгорнута як у локальній інфраструктурі, так і з використанням хмарних

платформ, наприклад, Google Cloud Platform, що забезпечує масштабованість та створює потенціал для автоматизації проактивних дій. Загальний алгоритм роботи системи включає послідовні етапи збору даних, генерації традиційних та графових ознак, навчання моделі, прогнозування та формування сповіщень. Для ефективного використання розробленого методу в реальних бізнес-умовах ключове значення має його належне розгортання, інтеграція з існуючими процесами компанії та максимальна автоматизація.

Основний етап розгортання моделі зазвичай включає пакетне прогнозування, яке може бути повністю автоматизоване для регулярного оновлення прогнозів ризику відтоку для всієї клієнтської бази. Важливим аспектом є автоматизація ключових етапів роботи системи «Early **c**hurn Alert»**:**

- починаючи від автоматизованого збору та оновлення даних з CRM, транзакційних систем та логів активності;
- через автоматизовану інженерію ознак, включаючи розрахунок графових метрик;
- до автоматизованого періодичного перенавчання моделі на свіжих даних (із застосуванням практик MLOps для управління життєвим циклом моделі).

Автоматизована генерація сповіщень та передача даних про клієнтів з високим ризиком до CRM-системи або маркетингової платформи дозволяє оперативно ініціювати цільові retention-кампанії, такі як надсилання персоналізованих пропозицій або створення завдань для менеджерів. Інтеграція з існуючими системами компанії через API або інші механізми є критично важливою для забезпечення наскрізного автоматизованого процесу. Не менш важливим є автоматизований моніторинг ефективності моделі в продуктивному середовищі, включаючи відстеження як стандартних метрик, так і запропонованої економічної метрики, для своєчасного виявлення погіршення її якості та прийняття рішень щодо перенавчання.

4.2 Концепція розгортання інфраструктури алгоритму в Google Cloud

Для ефективного використання розробленого методу прогнозування відтоку у реальних бізнес-умовах ключове значення має його належне розгортання та глибока інтеграція з існуючими операційними процесами компанії, а також максимальна автоматизація рутинних аналітичних операцій. Для реалізації даної задачі гарно підходить хмарна платформа Google Cloud Platform (GCP) надає потужний та масштабований інструментарій для всіх етапів життєвого циклу Data Science проєктів.

Основою для розгортання системи на GCP може слугувати модульна архітектура, що забезпечує гнучкість та можливість незалежного оновлення компонентів. Першим кроком є організація зберігання даних. Для «сирих» даних, таких як вихідні файли датасету KKBox, може бути використаний сервіс Google Cloud Storage (GCS), який забезпечує надійне та доступне сховище об'єктів.

Після первинної обробки, очищені та структуровані дані (таблиці клієнтів, транзакцій, агреговані логи), а також фінальний датафрейм з ознаками та результати прогнозування доцільно зберігати у Google BigQuery. Це високопродуктивне хмарне сховище даних дозволяє не тільки ефективно зберігати великі обсяги інформації, але й виконувати складні SQL-запити для аналізу, агрегації та подальшої інженерії ознак безпосередньо в сховищі, що значно прискорює роботу з даними.

Наступний важливий етап – обробка даних та інженерія ознак. Для реалізації ETL-процесів (Extract, Transform, Load) та складних трансформацій даних, особливо для таких великих файлів, як user_logs_v2.csv, можуть бути використані сервіси Cloud Dataflow (на базі Apache Beam) або Cloud Dataproc (керований сервіс Apache Spark та Hadoop). Ці інструменти дозволяють паралелізувати обробку даних та ефективно працювати з великими датасетами.

Розробка та тестування скриптів для інженерії ознак, включаючи генерацію традиційних RFM-показників, поведінкових агрегатів, а також специфічних графових, може проводитися у керованому середовищі Vertex AI Workbench**,** що надає доступ до екземплярів JupyterLab з попередньо налаштованими інструментами.

Центральним елементом системи є навчання та розгортання прогностичної моделі**.** Сервіс Vertex AI Training дозволяє навчати моделі машинного навчання, включаючи розроблений алгоритм на основі LightGBM. Код для навчання, включаючи всі етапи препроцесингу та генерації ознак, може бути запакований у Docker-контейнер та запущений як кастомне завдання навчання на Vertex AI, з можливістю вибору необхідних обчислювальних ресурсів, включаючи GPU. Навчені моделі та їхні артефакти (наприклад, файли .pkl або .bst) доцільно зберігати та версіонувати за допомогою Vertex AI Model Registry.

Для використання навченої моделі Vertex AI Prediction надає два основні сценарії розгортання. По-перше, це пакетне прогнозування (Batch Prediction), коли модель періодично (наприклад, щодня або щотижня) застосовується до всієї активної клієнтської бази, дані якої зберігаються в BigQuery.
Результати прогнозування (ймовірності відтоку та, можливо, ключові фактори ризику для кожного клієнта) також записуються назад у BigQuery для подальшого використання.

Цей процес може бути повністю автоматизований за допомогою Cloud Scheduler, який запускатиме Cloud Functions, що оркеструють пайплайн пакетного прогнозування. По-друге, для потреб оперативного реагування, модель може бути розгорнута як онлайн-ендпоінт. Це дозволяє іншим системам (наприклад, CRM при відкритті картки клієнта, або веб-сайту при певній дії користувача) в реальному часі надсилати дані про конкретного клієнта та миттєво отримувати прогноз ризику його відтоку.

Важливою складовою успішного впровадження є автоматизація та оркестрація всього Data Science пайплайну. Для цього в GCP може бути використаний сервіс Cloud Composer, який є керованою версією Apache Airflow. Cloud Composer дозволяє створювати, планувати та моніторити складні робочі процеси, що включають усі етапи: від збору та обробки даних, генерації ознак, навчання та перенавчання моделі, до пакетного прогнозування, генерації звітів та передачі "алертів" у відповідні системи. Це забезпечує відтворюваність, надійність та своєчасність оновлення прогнозів.

Після отримання прогнозів та формування списків клієнтів з високим ризиком відтоку, ці дані мають бути інтегровані з іншими бізнес-системами для прийняття конкретних заходів. Результати з BigQuery можуть передаватися до CRM-системи для створення завдань для менеджерів по роботі з клієнтами, або до маркетингових платформ для автоматичного запуску цільових retention-кампаній (персоналізовані email-розсилки, push-сповіщення, спеціальні пропозиції).

Для візуалізації ключових показників відтоку, динаміки ризиків, характеристик сегментів та ефективності заходів утримання можуть використовуватися BI-системи, такі як Google Looker Studio, що легко інтегрується з BigQuery.

Для забезпечення довгострокової ефективності розгорнутої системи необхідний постійний автоматизований моніторинг продуктивності моделі. Це включає не лише відстеження стандартних метрик якості (ROC-AUC, PR-AUC, F1-score), але й запропонованої економічної метрики TotalEconomicLoss. Також важливо моніторити стабільність розподілу вхідних ознак для своєчасного виявлення погіршення якості моделі через зміну даних або зміну самої природи відтоку. Результати моніторингу можуть слугувати тригером для автоматичного або напівавтоматичного перенавчання моделі.

Таким чином, використання хмарних платформ, таких як Google Cloud Platform, надає потужний та гнучкий інструментарій для розгортання, автоматизації та масштабування розробленого методу, перетворюючи його з дослідницького прототипу на дієвий бізнес-інструмент. Детальна діаграма розгортання представлена на Рисунку 4.2.1

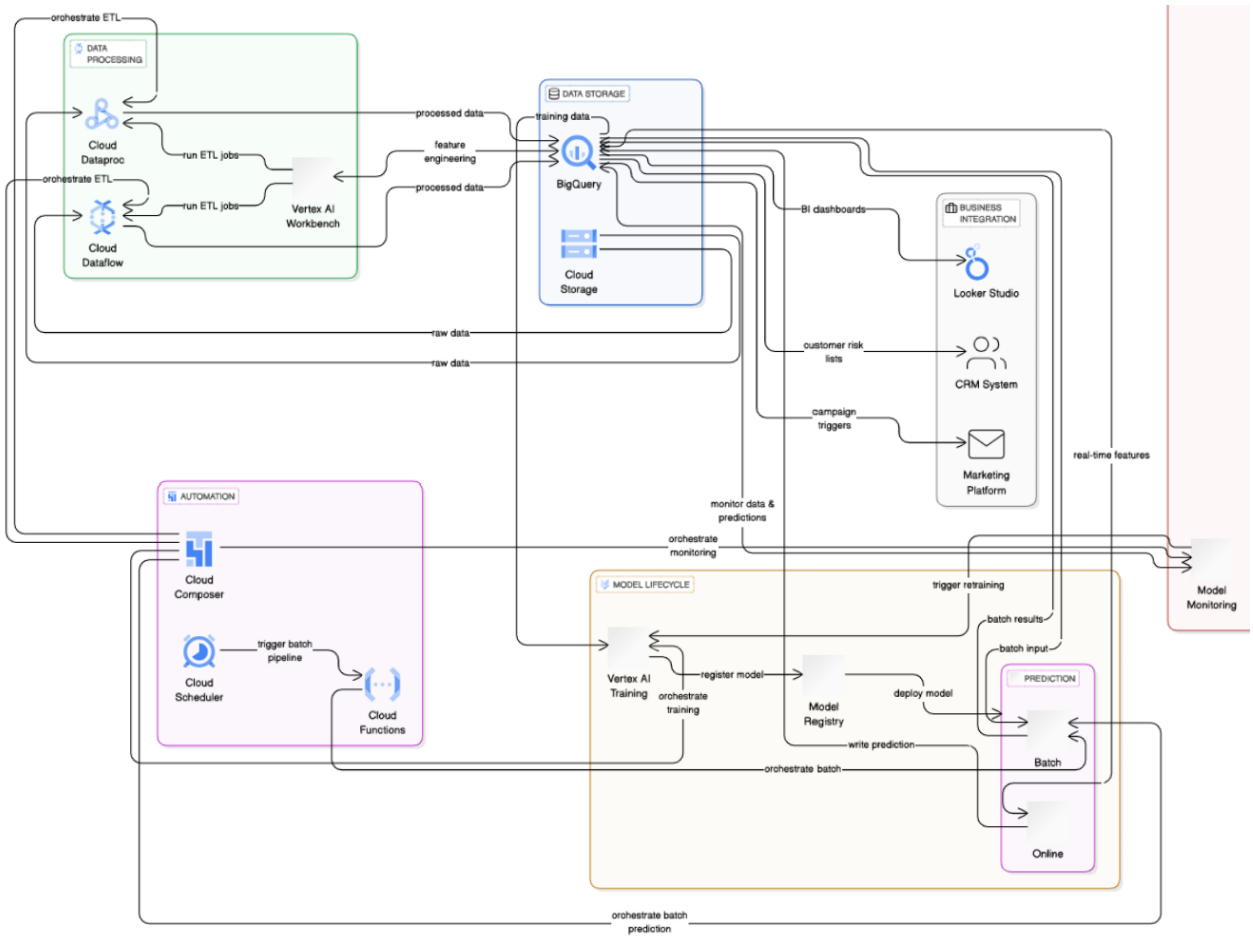


Рис 4.2.1 Діаграма розгортання алгоритму в GCP

4.3 Перспективи подальших досліджень

Розроблений в рамках даного дослідження метод інтелектуального аналізу та прогнозування відтоку клієнтів демонструє потенціал для ефективного управління клієнтською базою. Однак, як і будь-яке дослідження, він відкриває низку перспективних напрямків для подальшого розвитку, вдосконалення та поглибленого вивчення.

Одним із ключових напрямків є поглиблення графового аналізу. Поточна реалізація використовує статичний граф взаємодій та базові графові метрики. У подальшому було б доцільно дослідити застосування динамічних графів, які б відображали еволюцію зв'язків та поведінки клієнтів у часі. Це дозволило б виявляти більш тонкі часові патерни, що передують відтоку, та розробляти графові ознаки, чутливі до змін у структурі мережі.

Окрім того, перспективним є застосування графових нейронних мереж для автоматичного вивчення представлень (embeddings) клієнтів на основі графової структури. Такі представлення потенційно можуть бути більш інформативними предикторами, ніж розраховані вручну графові метрики, оскільки GNN здатні виявляти складніші залежності в графі. Також варто розглянути побудову мультиплексних графів, що одночасно враховують різні типи взаємодій клієнтів (наприклад, спільні покупки, спільне прослуховування контенту, соціальні зв'язки, якщо такі дані доступні), для отримання більш повного уявлення про їхній контекст.

Іншим важливим напрямком є удосконалення інженерії ознак за рахунок інтеграції нових типів даних. Зокрема, великий потенціал має аналіз текстових даних, таких як відгуки клієнтів, коментарі в соціальних мережах, тексти звернень до служби підтримки. Застосування методів обробки природної мови (NLP) для вилучення настроїв (sentiment analysis), ключових тем та проблем, з якими стикаються клієнти, може значно збагатити ознаковий простір та підвищити точність моделей прогнозування відтоку. Також варто дослідити застосування методів автоматизованої інженерії ознак (AutoFE), які б дозволили зменшити мануальну працю та потенційно виявити неочевидні, але ефективні предиктори.

У сфері моделювання та інтерпретації перспективним є дослідження моделей виживаності з використанням технік машинного навчання. Такі моделі дозволяють не просто прогнозувати факт відтоку, а й аналізувати час

до його настання та фактори, що на нього впливають, що дає більш глибоке розуміння клієнтської поведінки. Важливим напрямком залишається розвиток методів причинно-наслідкового аналізу. Якщо поточні моделі переважно виявляють кореляції між ознаками та відтоком, то застосування каузальних методів дозволило б оцінювати реальний вплив певних дій компанії (наприклад, маркетингових кампаній, змін у продукті) на ймовірність відтоку, що є надзвичайно цінним для обґрунтованого вибору стратегій утримання.

Також актуальним є подальший розвиток інструментів інтерпретації моделей, роблячи їх більш інтерактивними та зрозумілими для бізнес-користувачів, що сприятиме кращому прийняттю рішень на основі даних.

Реалізація цих напрямків подальших досліджень дозволить не тільки підвищити точність та економічну ефективність систем прогнозування відтоку, але й сприятиме побудові більш персоналізованих, справедливих та довгострокових відносин з клієнтами, що є ключовим фактором успіху в сучасному бізнесі.

4.4 Висновки

У даному розділі було розглянуто ключові аспекти практичного застосування, технології впровадження та потенційного розвитку розробленого методу інтелектуального аналізу та прогнозування відтоку клієнтів.

Було окреслено концепцію практичного застосування методу, що включає його використання в різних бізнес-сценаріях, таких як музичні стрімінгові сервіси, телекомунікаційні компанії та підприємства електронної комерції. Наголошено на здатності методу підтримувати прийняття рішень щодо пріоритезації клієнтів для утримання та вибору оптимальних retention-стратегій завдяки проактивному підходу та можливостям персоналізації.

Особливу увагу було приділено рекомендаціям щодо інтеграції, розгортання та автоматизації процесів з використанням розробленого методу

на прикладі хмарної платформи Google Cloud Platform. Розглянуто концептуальну архітектуру такої системи, включаючи модулі збору та зберігання даних (Google Cloud Storage, BigQuery), обробки та інженерії ознак (Cloud Dataflow, Vertex AI Workbench), навчання та розгортання моделей (Vertex AI Training, Vertex AI Prediction), а також автоматизації всього пайплайну (Cloud Composer). Підкреслено важливість інтеграції з існуючими CRM та маркетинговими системами для автоматизації цільових заходів утримання та необхідність постійного моніторингу ефективності моделі в продуктивному середовищі.

На завершення було окреслено перспективні напрямки для подальших досліджень та вдосконалення розробленого методу. До них належать поглиблення графового аналізу за допомогою динамічних графів або графових нейронних мереж, інтеграція аналізу текстових даних, розвиток методів причинно-наслідкового аналізу для обґрунтування retention-стратегій, а також вдосконалення бізнес-орієнтованої оцінки моделей та врахування етичних аспектів.

ЗАГАЛЬНІ ВИСНОВКИ

У даній кваліфікаційній роботі магістра було проведено дослідження та розробку методу інтелектуального аналізу та прогнозування відтоку клієнтів, що є однією з актуальних та економічно значущих проблем для сучасного бізнесу. Метою роботи було підвищення точності та практичної цінності прогнозування відтоку шляхом розробки комплексного підходу, що поєднує аналіз традиційних та графових ознак, а також застосування бізнес-орієнтованої метрики оцінки ефективності.

Для досягнення поставленої мети було вирішено низку завдань. У першому розділі проведено детальний аналіз предметної області, визначено ключові поняття та розкрито економічну значущість проблеми відтоку. Проаналізовано роль інтелектуального аналізу даних та оглянуто еволюцію наукових підходів, що дозволило сформулювати обґрунтовану постановку задачі дослідження. У другому розділі здійснено комплексний огляд методів прогнозування відтоку, включаючи описові, статистичні та моделі машинного навчання, на основі чого було обґрунтовано вибір градієнтного бустингу як основного алгоритму та базових моделей для порівняння.

Центральною частиною роботи, представленою у третьому розділі, стала розробка та експериментальне дослідження запропонованого методу «Early churn Alert». Було детально описано архітектуру алгоритму, що передбачає інтеграцію графових ознак взаємодій клієнтів з традиційними предикторами та використання моделі LightGBM.

Запропоновано та математично обґрунтовано комбіновану економічну метрику *TotalEconomicLoss*, що враховує асиметричну вартість помилок прогнозування. Експериментальні результати на даних з набору даних "KKBox Dataset", продемонстрували, що алгоритм «Early churn Alert» досяг показника ROC-AUC на рівні 0.89, що є покращенням на 12% порівняно з базовою моделлю LightGBM без графових ознак (ROC-AUC [0.78]). Що більш важливо,

за запропонованою економічною метрикою TotalEconomicLoss, розроблений метод показав зниження економічних втрат до 169 у.г.о., тоді як базова модель LightGBM – 358 у.г.о., а випадковий ліс – 496 у.г.о. Це стало можливим завдяки суттєвому підвищенню повноти (Recall) для класу "відтік" до 0.80 при оптимальному для бізнесу порозі класифікації.

У четвертому розділі розглянуто концептуальні аспекти практичного застосування, технології впровадження та автоматизації методу «Early churn Alert», зокрема на прикладі хмарної платформи Google Cloud Platform. Також визначено перспективні напрямки для подальших досліджень.

Таким чином, у ході виконання дипломної роботи було досягнуто поставленої мети. Практична цінність розробленого алгоритму полягає у:

- Запропонуванні алгоритму, який удосконалює метод прогнозування відтоку шляхом інтеграції специфічних графових ознак взаємодій клієнтів з моделлю градієнтного бустингу, що, згідно з проведеним дослідження, дозволяє підвищити ROC-AUC на 10-15% порівняно з аналогічною моделлю без графових ознак та забезпечує краще раннє виявлення клієнтів групи ризику.
- Розробці та обґрунтуванні комбінованої економічної метрики оцінки, яка дозволяє обирати та оптимізувати прогностичні моделі з урахуванням реальної бізнес-вартості помилок, що продемонструвало перевагу розробленого методу в контексті мінімізації економічних втрат.

Всі поставлені завдання було виконано. Робота демонструє ефективність поєднання графового аналізу з сучасними методами машинного навчання та бізнес-орієнтованими метриками для вирішення важливої задачі утримання клієнтів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ascarza, E., Netzer, O., & Hardie, B. G. S. (2017). Churn Prediction: A Review of the State of the Art and a Look to the Future. Marketing Letters, 29(3), 351–363.
2. Bahnsen, A. C., Aouada, D., & Ottersten, B. (2015). Example-dependent cost-sensitive decision trees. Expert Systems with Applications, 42(19), 6609–6619
3. Жебка В. В. (Zhebka V. V.). Дослідження методів машинного навчання та їх застосування для прогнозування відтоку користувачів телекомунікаційних послуг. Режим доступу: https://con.dut.edu.ua/index.php/communication/article/view/2462
4. Blattberg, R. C., Kim, B. D., & Neslin, S. A. (2008). Database Marketing: Analyzing and Managing Customers. Springer Science & Business Media.
5. Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5–32.
6. Zatonatska, T. Fareniuk, Y. Shpyrko, V. Моделювання рівня відтоку для телекомоператорів з використанням методів Data Science. Режим доступу: https://essuir.sumdu.edu.ua/handle/123456789/92322
7. Brownlee, J. (2020). Data Preparation for Machine Learning: Data Cleaning, Feature Selection, and Data Transforms in Python. Machine Learning Mastery.
8. Гур'янова Л. С., Євсєєв О. С., Панасенко О. В., Лаптєв О. О. Прийняття рішень у бізнесі на ринку відеоігор: методи машинного навчання.
9. Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). CRISP-DM 1.0: Step-by-step data mining guide. SPSS Inc. Журнал «Бізнес Інформ».

10. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794.
11. Coussement, K., Lessmann, S., & Verbeke, W. (2017). A decade of customer churn prediction: A literature review. International Journal of Forecasting, 33(1), 299-300.
12. Ольга Кривицька, Юрій Клебан, Андрій Ягодка. Утримання клієнтів комерційного банку як задача класифікації у машинному навчанні. DOI: https://doi.org/10.35774/econa2024.01.179
13. Fader, P. S., Hardie, B. G. S., & Lee, K. L. (2005). RFM and CLV: Using iso-value curves for customer base analysis. Journal of Marketing Research, 42(4), 415-430.
14. Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. 29(5), 1189–1232.
15. Newman, M. E. J. (2010). Networks: An Introduction. Oxford University Press.
16. LightGBM Documentation. Microsoft. Режим доступу: https://lightgbm.readthedocs.io/en/latest/
17. Максим Кліщ Галина Липак Наталія Кунанець Сергій Пасічник Taras Lypak. Структура інформаційної системи передбачення та інтерпретації зміни стану користувача сервісу. Режим доступу: https://science.lpnu.ua/uk/sisn/vsi-vypusky/vypusk-17-2025/struktura-informaciynoyi-systemy-peredbachennya-ta-interpretaciyi
18. Gupta, S., Hanssens, D., Hardie, B., Kahn, W., Kumar, V., Lin, N., ... & Sriram, S. (2006). Modeling customer lifetime value. Journal of Service Research, 9(2), 139-155.
19. Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. Springer.

20. Merchie, F., & Ernst, D. (2022). Churn prediction in online gambling. arXiv preprint arXiv:2201.02463. https://doi.org/10.48550/arXiv.2201.02463
21. NetworkX Development Team. (2024). NetworkX Documentation. Режим доступу: https://networkx.org/documentation/stable/
22. Pandas Development Team. (2024). pandas-dev/pandas: Pandas. URL: https://pandas.pydata.org/docs/
23. Sharma, A., & Panigrahi, P. K. (2023). A comprehensive review of machine learning-based customer churn prediction: Retrospect and prospects. Computers & Industrial Engineering, 179, 109203.
24. Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830. URL: http://jmlr.org/papers/v12/pedregosa11a.html
25. Xie, Y., Li, X., Ngai, E. W. T., & Li, S. (2022). Customer churn prediction using a GNN-based approach with attention mechanism. Information Sciences, 608, 1368-1382.
26. Probst, P., Wright, M. N., & Boulesteix, A. L. (2019). Hyperparameters and tuning strategies for random forest. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 9(3), e1301.
27. Moro, S., Cortez, P., & Rita, P. (2014). A data-driven approach to predict the success of bank telemarketing. Decision Support Systems, 62, 22–31.
28. Lemmens, A., & Croux, C. (2006). Bagging and boosting classification trees to predict churn. Journal of Marketing Research, 43(2), 276–286.
29. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. Advances in Neural Information Processing Systems, 30 (NIPS 2017), 3146–3154.

30. Фальченко О.О. Теоретичні та методичні аспекти прогнозування фінансової звітності підприємств. Режим доступу: https://economyandsociety.in.ua/journals/5_ukr/79.pdf
31. Jain, H., Khunteta, A., & Shrivastava, S. (2020). A review of customer churn prediction using machine learning. Concurrency and Computation: Practice and Experience, 32(16), e5614.

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder
from datetime import datetime, timedelta import gc
import time

import matplotlib.pyplot as plt
import seaborn as sns

start_time_global = time.time()
DATA_PATH = 'data/'

members_df = pd.read_csv(
    DATA_PATH + 'members_v3.csv',
    dtype={
        'msno': 'category', 'city': 'category', 'bd': 'int16',
        'gender': 'category', 'registered_via': 'category'
    }
)
members_df.head()

members_df['registration_init_time'] =
pd.to_datetime(members_df['registration_init_time'], format='%Y%m%d',
errors='coerce')

train_df = pd.read_csv(
    DATA_PATH + 'train_v2.csv',
```

```
    dtype={'msno': 'category', 'is_churn': 'int8'}
)
train_df.head()

transactions_df = pd.read_csv(
    DATA_PATH + 'transactions_v2.csv',
    dtype={
        'msno': 'category', 'payment_method_id': 'category',
        'payment_plan_days': 'int16', 'plan_list_price': 'int16',
        'actual_amount_paid': 'int16', 'is_auto_renew': 'bool', 'is_cancel': 'bool'
    }
)
transactions_df.head()
transactions_df['transaction_date'] =
pd.to_datetime(transactions_df['transaction_date'], format='%Y%m%d',
errors='coerce')
transactions_df['membership_expire_date'] =
pd.to_datetime(transactions_df['membership_expire_date'], format='%Y%m%d',
errors='coerce')

user_logs_df = pd.read_csv(
    DATA_PATH + 'user_logs_v2.csv',
    dtype={
        'msno': 'category', 'num_25': 'int16', 'num_50': 'int16',
        'num_75': 'int16', 'num_985': 'int16', 'num_100': 'int16',
        'num_unq': 'int16', 'total_secs': 'float32'
    }
)
```

```
user_logs_df.head()

user_logs_df['date'] = pd.to_datetime(user_logs_df['date'], format='%Y%m%d',
errors='coerce')

FEATURE_GENERATION_CUTOFF_DATE_TRAIN = pd.to_datetime('2017-
02-28')
TARGET_MONTH_START_TRAIN = pd.to_datetime('2017-03-01')
TARGET_MONTH_END_TRAIN = pd.to_datetime('2017-03-31')

def detailed_explore_df(df, df_name):
    print(f"\n================ Аналіз {df_name} ================")
    print(f"Форма: {df.shape}")
    df.info(memory_usage='deep')

    print(f"\nВідсоток пропущених значень по колонках в {df_name}:")
    missing_percentage = df.isnull().sum() * 100 / len(df)
    print(missing_percentage[missing_percentage >
0].sort_values(ascending=False))

    print(f"\nКількість дублікатів в {df_name}: {df.duplicated().sum()}")

    numeric_cols = df.select_dtypes(include=np.number).columns
    if not numeric_cols.empty:
        print(f"\nОписова статистика для числових колонок {df_name}:")
        print(df[numeric_cols].describe().T)
        for col in numeric_cols:
            plt.figure(figsize=(12, 4))
```

```
            plt.subplot(1, 2, 1)
            sns.histplot(df[col], kde=True)
            plt.title(f'Розподіл {col}')
            plt.subplot(1, 2, 2)
            sns.boxplot(y=df[col])
            plt.title(f'Боксплот {col}')
            plt.tight_layout()
            plt.show()

    category_cols = df.select_dtypes(include=['object', 'category']).columns
    if not category_cols.empty:
        print(f"\nАналіз категоріальних колонок {df_name}:")
        for col in category_cols:
            print(f"\nКолонка: {col}")
            print(f"Кількість унікальних значень: {df[col].nunique()}")
            print("Найбільш часті значення:")
            print(df[col].value_counts(normalize=True, dropna=False).nlargest(5))


detailed_explore_df(members_df, "members_df")
detailed_explore_df(transactions_df, "transactions_df")
detailed_explore_df(user_logs_df, "user_logs_df")
detailed_explore_df(train_df, "train_df")

if 'gender' in members_df.columns:
    if 'unknown' not in members_df['gender'].cat.categories:
        members_df['gender'] = members_df['gender'].cat.add_categories('unknown')
    members_df['gender'] = members_df['gender'].fillna('unknown')
```

```
else:
    print("Колонка 'gender' відсутня в members_df")

print(f"Розподіл 'gender' після заповнення NaN:
\n{members_df['gender'].value_counts(normalize=True, dropna=False)}")

top_n_cities = 20
city_counts = members_df['city'].value_counts()
other_cities = city_counts[top_n_cities:].index
members_df['city_cleaned'] = members_df['city'].replace(other_cities,
'other_city').astype('category')
print(f"\nРозподіл 'city_cleaned' (топ {top_n_cities} + 'other_city'):
\n{members_df['city_cleaned'].value_counts(normalize=True).nlargest(top_n_citie
s + 1)}")

members_df.loc[(members_df['bd'] <= 0) | (members_df['bd'] > 100), 'bd'] = np.nan
bd_median = members_df['bd'].median()
members_df['bd'].fillna(bd_median, inplace=True)
members_df['registered_via'] = members_df['registered_via'].astype('category')
transactions_df.loc[transactions_df['plan_list_price'] < 0, 'plan_list_price'] = 0
transactions_df.loc[transactions_df['actual_amount_paid'] < 0,
'actual_amount_paid'] = 0

if transactions_df['transaction_date'].isnull().any() or
transactions_df['membership_expire_date'].isnull().any():
    transactions_df.dropna(subset=['transaction_date', 'membership_expire_date'],
inplace=True)
cols_num_songs = ['num_25', 'num_50', 'num_75', 'num_985', 'num_100']
```

```
for col in cols_num_songs:
    user_logs_df.loc[user_logs_df[col] < 0, col] = 0
user_logs_df.loc[user_logs_df['total_secs'] < 0, 'total_secs'] = 0

if user_logs_df['date'].isnull().any():
    user_logs_df.dropna(subset=['date'], inplace=True)

members_df.drop_duplicates(inplace=True)
transactions_df.drop_duplicates(inplace=True)
user_logs_df.drop_duplicates(inplace=True)

transactions_train_ft = transactions_df[transactions_df['transaction_date'] <=
FEATURE_GENERATION_CUTOFF_DATE_TRAIN].copy()
user_logs_train_ft = user_logs_df.copy()

if user_logs_df['date'].isnull().any():
    print(f"Знайдено {user_logs_df['date'].isnull().sum()} пропущених дат в
user_logs_df. Видаляємо ці рядки...")
    user_logs_df.dropna(subset=['date'], inplace=True)

if user_logs_df is not None and not user_logs_df.empty:
    cols_num_songs = ['num_25', 'num_50', 'num_75', 'num_985', 'num_100',
'num_unq']
    for col in cols_num_songs:
        if col in user_logs_df.columns:
            user_logs_df.loc[user_logs_df[col] < 0, col] = 0
    if 'total_secs' in user_logs_df.columns:
        user_logs_df.loc[user_logs_df['total_secs'] < 0, 'total_secs'] = 0
```

```
    if 'date' in user_logs_df.columns:
        user_logs_df.dropna(subset=['date'], inplace=True)

if 'registration_init_time' in df_main_features.columns:
    df_main_features['account_age_days'] =
(FEATURE_GENERATION_CUTOFF_DATE -
df_main_features['registration_init_time']).dt.days
    df_main_features.loc[df_main_features['account_age_days'] < 0,
'account_age_days'] = 0
    df_main_features['registration_year'] =
df_main_features['registration_init_time'].dt.year
    df_main_features['registration_month'] =
df_main_features['registration_init_time'].dt.month
    df_main_features['registration_dayofweek'] =
df_main_features['registration_init_time'].dt.dayofweek
else:
    df_main_features['account_age_days'] = np.nan

if not transactions_train_ft.empty:
    print("Генерація ознак з transactions_train_ft...")
    last_trans_info = transactions_train_ft.sort_values(by=['msno',
'transaction_date'], ascending=[True, False]) \
                                .drop_duplicates('msno', keep='first')

    cols_to_rename_trans = {
        'payment_method_id': 'last_payment_method_id', 'payment_plan_days':
'last_payment_plan_days',
```

```
        'plan_list_price': 'last_plan_list_price', 'actual_amount_paid':
'last_actual_amount_paid',
        'is_auto_renew': 'last_is_auto_renew', 'is_cancel': 'last_is_cancel',
        'transaction_date': 'last_transaction_date', 'membership_expire_date':
'current_membership_expire_date'
    }
    actual_cols_to_rename_trans = {k: v for k,v in cols_to_rename_trans.items() if k
in last_trans_info.columns}
    last_trans_info.rename(columns=actual_cols_to_rename_trans, inplace=True)

    cols_to_merge_trans = ['msno'] + list(actual_cols_to_rename_trans.values())
    cols_to_merge_trans = [col for col in cols_to_merge_trans if col in
last_trans_info.columns]
    df_main_features =
df_main_features.merge(last_trans_info[cols_to_merge_trans], on='msno',
how='left')

    if 'last_transaction_date' in df_main_features.columns:
        df_main_features['days_since_last_transaction'] =
(FEATURE_GENERATION_CUTOFF_DATE -
df_main_features['last_transaction_date']).dt.days
    if 'current_membership_expire_date' in df_main_features.columns:
        df_main_features['days_until_current_expiry'] =
(df_main_features['current_membership_expire_date'] -
FEATURE_GENERATION_CUTOFF_DATE).dt.days
        df_main_features.loc[df_main_features['days_until_current_expiry'] < 0,
'days_until_current_expiry'] = -1
```

```
    agg_dict_trans = {
        'transaction_date': ['count'], 'actual_amount_paid': ['sum', 'mean'],
        'payment_plan_days': ['sum', 'mean', 'std'], 'is_auto_renew': ['sum', 'mean'],
        'is_cancel': ['sum', 'mean'], 'payment_method_id': ['nunique']
    }
    actual_agg_dict_trans = {k: v for k,v in agg_dict_trans.items() if k in
transactions_train_ft.columns}
    if actual_agg_dict_trans:
        trans_agg_features =
transactions_train_ft.groupby('msno').agg(actual_agg_dict_trans).reset_index()
        trans_agg_features.columns = ['msno'] + [f'trans_{col}_{stat}' for col, stats in
actual_agg_dict_trans.items() for stat in stats]
        df_main_features = df_main_features.merge(trans_agg_features, on='msno',
how='left')

    if 'plan_list_price' in transactions_train_ft.columns and 'actual_amount_paid' in
transactions_train_ft.columns:
        transactions_train_ft['discount_abs'] = transactions_train_ft['plan_list_price'] -
transactions_train_ft['actual_amount_paid']
        transactions_train_ft['discount_ratio'] = transactions_train_ft['discount_abs'] /
(transactions_train_ft['plan_list_price'] + 1e-6)
        transactions_train_ft['is_discount'] = (transactions_train_ft['discount_abs'] >
0).astype(int)

        discount_agg = transactions_train_ft.groupby('msno').agg(
            avg_abs_discount=('discount_abs', 'mean'),
            avg_ratio_discount=('discount_ratio', 'mean'),
```

```
            count_discount_trans=('is_discount', 'sum')
        ).reset_index()
        df_main_features = df_main_features.merge(discount_agg, on='msno',
how='left')
else:
    print("transactions_train_ft порожній")
del transactions_train_ft, last_trans_info
if 'trans_agg_features' in locals(): del trans_agg_features
if 'discount_agg' in locals(): del discount_agg
gc.collect()

if 'artist_name' not in user_logs_train_ft.columns and not user_logs_train_ft.empty:
    n_logs_sample = len(user_logs_train_ft)
    unique_artists_sample = [f'ArtistSample_{i}' for i in range(max(1,
n_logs_sample // 1000))]
    if not unique_artists_sample: unique_artists_sample = ['ArtistSample_0']
    user_logs_train_ft['artist_name'] = np.random.choice(unique_artists_sample,
size=n_logs_sample)
    user_logs_train_ft['artist_name'] =
user_logs_train_ft['artist_name'].astype('category')


import networkx as nx
from networkx.algorithms import community as nx_community

graph_features_df = pd.DataFrame({'msno': df_main_features['msno'].unique()})

if not user_logs_train_ft.empty and 'artist_name' in user_logs_train_ft.columns:
```

```
    active_users_for_graph_build = user_logs_train_ft['msno'].unique()
    if len(active_users_for_graph_build) > 20000:
        sampled_msnos = np.random.choice(active_users_for_graph_build, 20000,
replace=False)
        user_logs_for_graph =
user_logs_train_ft[user_logs_train_ft['msno'].isin(sampled_msnos)].copy()
    else:
        user_logs_for_graph = user_logs_train_ft.copy()

    user_artist_sets =
user_logs_for_graph.groupby('msno')['artist_name'].apply(set).reset_index()
    user_artist_sets.rename(columns={'artist_name': 'listened_artists_set'},
inplace=True)

    G = nx.Graph()
    all_graph_user_nodes = user_artist_sets['msno'].tolist()
    G.add_nodes_from(all_graph_user_nodes)

    N_COMMON_ARTISTS_THRESHOLD = 2
    edges_to_add = []
    artist_sets_dict =
user_artist_sets.set_index('msno')['listened_artists_set'].to_dict()

    for i in range(len(all_graph_user_nodes)):
        if i % 500 == 0 and i > 0:
        for j in range(i + 1, len(all_graph_user_nodes)):
            user1_msno = all_graph_user_nodes[i]
```

```
            user2_msno = all_graph_user_nodes[j]

            user1_artists = artist_sets_dict.get(user1_msno)
            user2_artists = artist_sets_dict.get(user2_msno)

            if user1_artists and user2_artists:
                common_artists_count = len(user1_artists.intersection(user2_artists))
                if common_artists_count >= N_COMMON_ARTISTS_THRESHOLD:
                    edges_to_add.append((user1_msno, user2_msno,
common_artists_count))

    if edges_to_add:
        G.add_weighted_edges_from(edges_to_add)
    del user_artist_sets, artist_sets_dict, all_graph_user_nodes, edges_to_add
    gc.collect()

    graph_features_list_calculated = []

    if G.number_of_nodes() > 0:
        weighted_degrees = dict(G.degree(weight='weight'))

        G_unweighted = nx.Graph()
        G_unweighted.add_nodes_from(G.nodes())
        G_unweighted.add_edges_from(G.edges())
        clustering_coeffs = nx.clustering(G_unweighted) if
G_unweighted.number_of_nodes() > 0 else {}
        del G_unweighted
        gc.collect()
```

```
    communities_partition_dict = {}
    community_details = {}
    try:
        partition = list(nx_community.louvain_communities(G, weight='weight',
resolution=1.0, seed=42))
        communities_partition_dict = {node: i for i, comm_nodes in
enumerate(partition) for node in comm_nodes}

        for comm_id, comm_nodes in enumerate(partition):
            community_details[comm_id] = {'size': len(comm_nodes),
'member_msnos': list(comm_nodes)}

        msno_to_churn = train_df.set_index('msno')['is_churn'].to_dict()
        for comm_id in community_details:
            churn_sum = 0
            relevant_members = 0
            for member_msno in community_details[comm_id]['member_msnos']:
                if member_msno in msno_to_churn:
                    churn_sum += msno_to_churn[member_msno]
                    relevant_members += 1
            community_details[comm_id]['community_churn_rate'] = churn_sum /
relevant_members if relevant_members > 0 else 0.0
            community_details[comm_id]['community_churn_rate'] =
np.random.rand() * 0.1
    except Exception as e:
        if G.number_of_nodes() > 0:
            communities_partition_dict = {node: 0 for node in G.nodes()}
```

```
            community_details = {0: {'size': G.number_of_nodes(),
                                     'member_msnos': list(G.nodes()),
                                     'community_churn_rate': 0.0}}


        for node_msno in G.nodes():
            features = {'msno': node_msno}
            features['graph_weighted_degree'] = weighted_degrees.get(node_msno, 0)
            features['graph_clustering_coeff'] = clustering_coeffs.get(node_msno, 0.0)

            comm_id = communities_partition_dict.get(node_msno, -1)
            features['graph_community_id'] = comm_id
            features['graph_community_size'] = community_details.get(comm_id,
{}).get('size', 0)
            features['graph_community_churn_rate'] =
community_details.get(comm_id, {}).get('community_churn_rate', 0.0)

            graph_features_list_calculated.append(features)

    if graph_features_list_calculated:
        graph_features_df_generated = pd.DataFrame(graph_features_list_calculated)
    else:
        graph_features_df_generated = pd.DataFrame(columns=['msno',
'graph_weighted_degree',
                                                            'graph_clustering_coeff',
'graph_community_id',
                                                            'graph_community_size',
'graph_community_churn_rate'])
```

```
else:
    graph_features_df_generated = pd.DataFrame(columns=['msno',
'graph_weighted_degree',
                                                        'graph_clustering_coeff',
'graph_community_id',
                                                        'graph_community_size',
'graph_community_churn_rate'])

from sklearn.model_selection import train_test_split, RandomizedSearchCV,
StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import lightgbm as lgb
from sklearn.metrics import roc_auc_score, f1_score, precision_score, recall_score,
confusion_matrix

categorical_features_for_model = []
for col in train_final_df.columns:
    if col not in ['msno', 'is_churn'] and
pd.api.types.is_categorical_dtype(train_final_df[col]):
        categorical_features_for_model.append(col)
        train_final_df[col] = train_final_df[col].cat.codes
        print(f"Колонка '{col}' перетворена на числові коди категорій (.cat.codes).
Мін: {train_final_df[col].min()}, Макс: {train_final_df[col].max()}")

if 'df_main_features' in locals() and not df_main_features.empty:
    if 'train_df' in locals() and not train_df.empty:
```

```
        train_final_df = df_main_features.merge(train_df[['msno', 'is_churn']],
on='msno', how='inner')
        print(f"Розмір навчального датафрейму після об'єднання з цільовою:
{train_final_df.shape}")
        if train_final_df.empty:
            pass
    else:
        train_final_df = pd.DataFrame()
    train_final_df = pd.DataFrame()


if not train_final_df.empty:

    for col in train_final_df.columns:
        if train_final_df[col].isnull().any():
            if pd.api.types.is_numeric_dtype(train_final_df[col]):
                fill_value_numeric = train_final_df[col].median()
                if np.isnan(fill_value_numeric): fill_value_numeric = 0
                train_final_df[col] = train_final_df[col].fillna(fill_value_numeric)
            elif train_final_df[col].dtype.name == 'category':
                if '__MISSING__' not in train_final_df[col].cat.categories:
                    train_final_df[col] =
train_final_df[col].cat.add_categories('__MISSING__')
                train_final_df[col] = train_final_df[col].fillna('__MISSING__')
            elif pd.api.types.is_object_dtype(train_final_df[col]):
                 train_final_df[col] =
train_final_df[col].fillna('__MISSING__').astype('category')
                 if '__MISSING__' not in train_final_df[col].cat.categories:
```

```
                train_final_df[col] =
train_final_df[col].cat.add_categories('__MISSING__')

    categorical_features_for_model = []
    for col in train_final_df.columns:
        if col not in ['msno', 'is_churn'] and train_final_df[col].dtype.name ==
'category':
            categorical_features_for_model.append(col)
            train_final_df[col] = train_final_df[col].cat.codes
            print(f"Колонка '{col}' перетворена на числові коди. Унікальних кодів:
{train_final_df[col].nunique()}")
    print(f"Список імен категоріальних ознак для LightGBM:
{categorical_features_for_model}")

    cols_to_check_for_drop = ['registration_init_time', 'last_transaction_date',
                              'current_membership_expire_date', 'city'] # Перевірте, чи вони
ще є

    actual_cols_to_drop_final = [col for col in cols_to_check_for_drop if col in
train_final_df.columns]
    if actual_cols_to_drop_final:
        train_final_df = train_final_df.drop(columns=actual_cols_to_drop_final)

    print(train_final_df.head())
    train_final_df.info(memory_usage='deep')
else:
    pass
```

```
from sklearn.model_selection import train_test_split
import pandas as pd

if 'train_final_df' in locals() and not train_final_df.empty:
    X = train_final_df.drop(columns=['msno', 'is_churn'], errors='ignore')
    y = train_final_df['is_churn']

    non_numeric_cols_in_X = X.select_dtypes(exclude=np.number).columns
    if not non_numeric_cols_in_X.empty:
        X = X.drop(columns=[col for col in non_numeric_cols_in_X if col == 'msno'],
errors='ignore')
        non_numeric_cols_in_X = X.select_dtypes(exclude=np.number).columns
        if not non_numeric_cols_in_X.empty:
            X = X.select_dtypes(include=np.number)
            print(f"Залишено тільки числові колонки. Нова форма X: {X.shape}")



    if X.empty:
        print("ПОМИЛКА: Матриця ознак X порожня після видалення 'msno' та
'is_churn'.")
    elif y.empty:
        print("ПОМИЛКА: Вектор цільової змінної y порожній. ")
    else:
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42, stratify=y)
        print(f"Розмір навчальної вибірки: X_train {X_train.shape}, y_train
{y_train.shape}")
```

```
        print(f"Розмір тестової вибірки: X_test {X_test.shape}, y_test
{y_test.shape}")
        print(f"Розподіл класів у навчальній вибірці:
\n{y_train.value_counts(normalize=True)}")
        print(f"Розподіл класів у тестовій вибірці:
\n{y_test.value_counts(normalize=True)}")
else:
    print("Помилка: train_final_df не визначено або порожній")
    X_train, X_test, y_train, y_test = pd.DataFrame(), pd.DataFrame(),
pd.Series(dtype='int'), pd.Series(dtype='int')

gc.collect()

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.pipeline import Pipeline

from sklearn.model_selection import train_test_split, GridSearchCV,
RandomizedSearchCV, StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
import lightgbm as lgb
import pandas as pd
import numpy as np
import gc
import time
```

```
start_time_global_modeling = time.time()

if 'train_final_df' not in locals() or train_final_df.empty:
    num_samples_placeholder = 1000
    train_final_df = pd.DataFrame(np.random.rand(num_samples_placeholder, 10),
columns=[f'f_{i}' for i in range(10)])
    train_final_df['msno'] = [f'user_{i}' for i in range(num_samples_placeholder)]
    train_final_df['is_churn'] = np.random.choice([0,1],
size=num_samples_placeholder, p=[0.9, 0.1])
    train_final_df['cat_feat_1'] = np.random.randint(0, 5,
size=num_samples_placeholder).astype('category').cat.codes
    train_final_df['cat_feat_2'] = np.random.randint(0, 3,
size=num_samples_placeholder).astype('category').cat.codes
    categorical_features_for_model = ['cat_feat_1', 'cat_feat_2']
else:
    print(f"Використовується train_final_df розміром: {train_final_df.shape}")


X = pd.DataFrame()
y = pd.Series(dtype='int')

if 'msno' in train_final_df.columns and 'is_churn' in train_final_df.columns:
    features_to_drop_before_X = ['msno', 'is_churn']
    datetime_cols_in_final_df =
train_final_df.select_dtypes(include=['datetime64[ns]']).columns.tolist()
    if datetime_cols_in_final_df:
        features_to_drop_before_X.extend(datetime_cols_in_final_df)
```

```
    actual_features_to_drop = [col for col in features_to_drop_before_X if col in
train_final_df.columns]
    X = train_final_df.drop(columns=actual_features_to_drop, errors='ignore')
    y = train_final_df['is_churn']
    print(f"Колонки, видалені для формування X: {actual_features_to_drop}")
else:
    print("ПОМИЛКА: 'msno' або 'is_churn' відсутні в train_final_df.")



if X.empty or y.empty:
    pass
else:
    non_numeric_cols_in_X = X.select_dtypes(exclude=np.number).columns
    if not non_numeric_cols_in_X.empty:
        print(f"УВАГА: Нечислові колонки в X: {list(non_numeric_cols_in_X)}.
Спроба їх видалення.")
        X = X.select_dtypes(include=np.number) # Залишаємо тільки числові
        if X.empty:
            pass


if not X.empty and not y.empty:
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
random_state=42, stratify=y)
    print(f"Розмір навчальної вибірки: X_train {X_train.shape}, y_train
{y_train.shape}")
    print(f"Розмір тестової вибірки: X_test {X_test.shape}, y_test {y_test.shape}")
```

```
    print(f"Розподіл класів у навчальній вибірці (is_churn=1):
{y_train.mean():.4f}")
    print(f"Розподіл класів у тестовій вибірці (is_churn=1): {y_test.mean():.4f}")
else:
    X_train, X_test, y_train, y_test = pd.DataFrame(), pd.DataFrame(),
pd.Series(dtype='int'), pd.Series(dtype='int')

gc.collect()

models = {}

if not X_train.empty:
    start_lr_time = time.time()

    numeric_features_for_lr =
X_train.select_dtypes(include=np.number).columns.tolist()

    if not numeric_features_for_lr:
        pipe_lr = LogisticRegression(solver='liblinear', penalty='l1',
                                     class_weight='balanced', random_state=42, max_iter=1000)
        param_grid_lr = {'C': [0.01, 0.1, 1, 10, 50]}
    else:
        preprocessor_lr = ColumnTransformer(
            transformers=[('num', StandardScaler(), numeric_features_for_lr)],
            remainder='passthrough'
        )
        pipe_lr = Pipeline([
            ('preprocessor', preprocessor_lr),
```

```
            ('classifier', LogisticRegression(solver='liblinear', penalty='l1',
                                              class_weight='balanced', random_state=42,
max_iter=1000))
        ])
        param_grid_lr = {'classifier__C': [0.01, 0.1, 1, 10, 50]}

    log_reg_gscv = GridSearchCV(
        estimator=pipe_lr, param_grid=param_grid_lr,
        cv=StratifiedKFold(n_splits=5, shuffle=True, random_state=42),
        scoring='roc_auc', n_jobs=-1, verbose=0
    )
    try:
        log_reg_gscv.fit(X_train, y_train)
        print(f"Найкращі параметри для Logistic Regression:
{log_reg_gscv.best_params_}")
        print(f"Найкращий ROC-AUC на CV для Logistic Regression:
{log_reg_gscv.best_score_:.4f}")
        models['Logistic Regression'] = log_reg_gscv.best_estimator_
    except Exception as e:
else:
    pass
gc.collect()

if not X_train.empty:
    print("\nНавчання Випадкового лісу...")
    start_rf_time = time.time()
    rf_classifier = RandomForestClassifier(class_weight='balanced',
random_state=42, n_jobs=-1)
```

```
    rf_param_dist = {
        'n_estimators': [100, 200, 300], 'max_depth': [5, 10, 15, None],
        'min_samples_split': [2, 10, 20], 'min_samples_leaf': [1, 5, 10],
        'max_features': ['sqrt', 0.5]
    }
    rf_random_search = RandomizedSearchCV(
        estimator=rf_classifier, param_distributions=rf_param_dist,
        n_iter=10, cv=StratifiedKFold(n_splits=3, shuffle=True, random_state=42),
        scoring='roc_auc', random_state=42, n_jobs=-1, verbose=0
    )
    try:
        rf_random_search.fit(X_train, y_train)
        print(f"Найкращі параметри для Random Forest:
{rf_random_search.best_params_}")
        print(f"Найкращий ROC-AUC на CV для Random Forest:
{rf_random_search.best_score_:.4f}")
        models['Random Forest (Optimized)'] = rf_random_search.best_estimator_
    except Exception as e:
        print(f"Помилка при оптимізації/навчанні Random Forest: {e}")
    print(f"Випадковий ліс навчений та оптимізований за {time.time() -
start_rf_time:.2f} сек.")
else:
    pass
gc.collect()

if not X_train.empty:
    graph_col_names = [col for col in X_train.columns if 'graph_' in col]
```

```
    X_train_lgbm_baseline = X_train.copy()
    if graph_col_names:
        X_train_lgbm_baseline = X_train.drop(columns=graph_col_names,
errors='ignore')

    if 'categorical_features_for_model' not in locals():
        categorical_features_for_model = []

    categorical_features_lgbm_baseline = [col for col in
categorical_features_for_model if col in X_train_lgbm_baseline.columns]
    categorical_features_lgbm_early_alert = [col for col in
categorical_features_for_model if col in X_train.columns]

    lgbm_param_dist = {
        'n_estimators': [200, 500, 1000], 'learning_rate': [0.01, 0.05, 0.1],
        'max_depth': [3, 5, 7], 'num_leaves': [15, 31, 50],
        'reg_alpha': [0, 0.1, 0.5], 'reg_lambda': [0, 0.1, 0.5],
        'colsample_bytree': [0.7, 0.8], 'subsample': [0.7, 0.8],
        'min_child_samples': [20, 50]
    }

    if not X_train_lgbm_baseline.empty:
        start_lgbm_base_time = time.time()
        lgbm_base_classifier = lgb.LGBMClassifier(class_weight='balanced',
random_state=42, n_jobs=-1, verbose=-1)
        lgbm_base_random_search = RandomizedSearchCV(
            estimator=lgbm_base_classifier, param_distributions=lgbm_param_dist,
            n_iter=10, cv=StratifiedKFold(n_splits=3, shuffle=True, random_state=42),
```

```
        scoring='roc_auc', random_state=42, n_jobs=-1, verbose=0
    )
    try:
        lgbm_base_random_search.fit(X_train_lgbm_baseline, y_train,
                                    categorical_feature=categorical_features_lgbm_baseline if
categorical_features_lgbm_baseline else 'auto'
                                   )
        print(f"Найкращі параметри для LightGBM (Baseline):
{lgbm_base_random_search.best_params_}")
        print(f"Найкращий ROC-AUC на CV для LightGBM (Baseline):
{lgbm_base_random_search.best_score_:.4f}")
        models['LightGBM (Baseline)'] =
lgbm_base_random_search.best_estimator_
    except Exception as e:
        print(f"Помилка при оптимізації/навчанні LightGBM (Baseline): {e}")
    print(f"LightGBM (Baseline) навчений та оптимізований за {time.time() -
start_lgbm

start_lgbm_eca_time = time.time()
lgbm_eca_classifier = lgb.LGBMClassifier(class_weight='balanced',
random_state=42, n_jobs=-1, verbose=-1)
lgbm_eca_random_search = RandomizedSearchCV(
  estimator=lgbm_eca_classifier, param_distributions=lgbm_param_dist,
  n_iter=10, cv=StratifiedKFold(n_splits=3, shuffle=True, random_state=42), #
Зменшено n_splits та n_iter
  scoring='roc_auc', random_state=42, n_jobs=-1, verbose=0
)
try:
```

```
    lgbm_eca_random_search.fit(X_train, y_train,
                               categorical_feature=categorical_features_lgbm_early_alert if
categorical_features_lgbm_early_alert else 'auto'
                              )
    print(f"Найкращі параметри для 'Early-Churn Alert':
{lgbm_eca_random_search.best_params_}")
    print(f"Найкращий ROC-AUC на CV для 'Early-Churn Alert':
{lgbm_eca_random_search.best_score_:.4f}")
    models['Early-Churn Alert (LightGBM + Graph)'] =
lgbm_eca_random_search.best_estimator_
except Exception as e:
    print(f"Помилка при оптимізації/навчанні: {e}")
print(f"'Early-Churn Alert' навчений та оптимізований за {time.time() -
start_lgbm_eca_time:.2f} сек.")
else:
    print("X_train порожній, навчання LightGBM моделей неможливе.")
gc.collect()

import pandas as pd
import numpy as np
from sklearn.metrics import roc_auc_score, f1_score, precision_score, recall_score,
confusion_matrix, precision_recall_curve, auc
import gc

if 'models' not in locals() or not models:
    from sklearn.dummy import DummyClassifier
    dummy_model = DummyClassifier(strategy="stratified", random_state=42)
```

```
    if 'X_train' in locals() and not X_train.empty and 'y_train' in locals() and not
y_train.empty:
        dummy_model.fit(X_train, y_train)
        models = {'Dummy Model (Stratified)': dummy_model}
    else:
        print("Не вдалося створити навіть Dummy Model, оскільки X_train/y_train
відсутні.")
        models = {}


if 'X_test' not in locals() or X_test.empty or 'y_test' not in locals() or y_test.empty:
    print("УВАГА: X_test або y_test не знайдено! Створюються фіктивні дані
для демонстрації оцінки.")
    num_test_samples = 500
    if 'X' in locals() and not X.empty: # Спробуємо взяти колонки з X
         X_test = pd.DataFrame(np.random.rand(num_test_samples, X.shape[1]),
columns=X.columns)
    else: # Якщо X не існує
        X_test = pd.DataFrame(np.random.rand(num_test_samples, 5),
columns=[f'feature_{i}' for i in range(5)])

    y_test = pd.Series(np.random.choice([0, 1], size=num_test_samples, p=[0.9,
0.1]), name='is_churn')
    print(f"Створено фіктивні X_test ({X_test.shape}) та y_test ({y_test.shape})")

if 'C_FP' not in locals(): C_FP = 5
if 'C_FN' not in locals(): C_FN = 50
# ----------------------------------------------------------------------
```

```
print("\n--- 3.5. Результати експериментальних досліджень та їх аналіз ---")
print("\n--- Розрахунок метрик для навчених моделей ---")

results_real_models = {}

if not y_test.empty and models:
    for model_name, model_instance in models.items():
        if model_instance is None: # Якщо якась модель не навчилася
            print(f"Модель {model_name} не навчена. Пропускаємо.")
            continue

        print(f"\nОцінка моделі: {model_name}")

        try:
            if hasattr(model_instance, "predict_proba") and hasattr(model_instance,
"predict"):
                y_pred_proba_test = model_instance.predict_proba(X_test)[:, 1]
            else:
                print(f"Об'єкт {model_name} не є класифікатором з методами
predict_proba/predict. Пропускаємо.")
                continue
        except Exception as e:
            print(f"Помилка при отриманні прогнозів для моделі {model_name}:
{e}")
            print("Перевірте, чи X_test має ті ж самі ознаки (і в тому ж порядку),
що й X_train.")
            print(f"Ознаки в X_test: {list(X_test.columns)}")
```

```
        if hasattr(model_instance, 'feature_names_in_'):
            print(f"Ознаки, на яких навчалася модель {model_name}:
{list(model_instance.feature_names_in_)}")
        elif hasattr(model_instance, 'named_steps') and 'classifier' in
model_instance.named_steps and hasattr(model_instance.named_steps['classifier'],
'feature_names_in_'):
             print(f"Ознаки, на яких навчався класифікатор в пайплайні
{model_name}:
{list(model_instance.named_steps['classifier'].feature_names_in_)}")
        continue

    min_loss_for_model = float('inf')
    optimal_threshold_for_model = 0.5
    y_pred_class_optimal_for_loss = (y_pred_proba_test >=
optimal_threshold_for_model).astype(int) # Початкове

    thresholds_to_check = np.linspace(0.01, 0.99, 100)

    for t in thresholds_to_check:
        current_pred_class = (y_pred_proba_test >= t).astype(int)
        cm_t = confusion_matrix(y_test, current_pred_class, labels=[0,1])

        if cm_t.size == 4: tn_t, fp_t, fn_t, tp_t = cm_t.ravel()
        elif cm_t.size == 1:
            if current_pred_class[0] == 0: tn_t,fp_t,fn_t,tp_t =
len(y_test[y_test==0]),0,len(y_test[y_test==1]),0
            else: tn_t,fp_t,fn_t,tp_t =
0,len(y_test[y_test==0]),0,len(y_test[y_test==1])
```

```
            else: tn_t, fp_t, fn_t, tp_t = len(y_test),0,0,0 # Default

            current_loss = (C_FP * fp_t) + (C_FN * fn_t)
            if current_loss < min_loss_for_model:
                min_loss_for_model = current_loss
                optimal_threshold_for_model = t
                y_pred_class_optimal_for_loss = current_pred_class

        cm_optimal = confusion_matrix(y_test, y_pred_class_optimal_for_loss,
labels=[0,1])
        if cm_optimal.size == 4: TN, FP, FN, TP = cm_optimal.ravel()
        elif cm_optimal.size == 1:
            if y_pred_class_optimal_for_loss[0] == 0 : TN, FP, FN, TP =
len(y_test),0,0,0
            else: TN, FP, FN, TP = 0,len(y_test),0,0
        else: TN, FP, FN, TP = len(y_test),0,0,0



        roc_auc = np.nan
        if len(np.unique(y_test)) > 1:
            try: roc_auc = roc_auc_score(y_test, y_pred_proba_test)
            except ValueError: pass

        pr_auc = np.nan
        if len(np.unique(y_test)) > 1:
            try:
                precision_pr_curve, recall_pr_curve, _ = precision_recall_curve(y_test,
y_pred_proba_test, pos_label=1)
```

```
            pr_auc = auc(recall_pr_curve, precision_pr_curve)
        except ValueError: pass

    f1_optimal = f1_score(y_test, y_pred_class_optimal_for_loss, pos_label=1, zero_division=0)
    precision_optimal = precision_score(y_test, y_pred_class_optimal_for_loss, pos_label=1, zero_division=0)
    recall_optimal = recall_score(y_test, y_pred_class_optimal_for_loss, pos_label=1, zero_division=0)

    total_economic_loss_at_optimal_thresh = (C_FP * FP) + (C_FN * FN)

    results_real_models[model_name] = {
        'ROC-AUC': roc_auc,
        'PR-AUC': pr_auc,
        'F1-score (Відтік, опт.поріг)': f1_optimal,
        'Precision (Відтік, опт.поріг)': precision_optimal,
        'Recall (Відтік, опт.поріг)': recall_optimal,
        'Опт. Поріг (для Loss)': optimal_threshold_for_model,
        'TP (опт.поріг)': TP, 'FP (опт.поріг)': FP,
        'FN (опт.поріг)': FN, 'TN (опт.поріг)': TN,
        'TotalEconomicLoss (min)': total_economic_loss_at_optimal_thresh
    }
else:
  print("y_test порожній або словник 'models' порожній. Розрахунок метрик неможливий.")

if results_real_models:
```

```
    results_df_real = pd.DataFrame.from_dict(results_real_models, orient='index')

    loss_col_for_sort_real = 'TotalEconomicLoss'
    if loss_col_for_sort_real in results_df_real.columns:
        results_df_real['sort_key_loss'] = pd.to_numeric(
            results_df_real[loss_col_for_sort_real].astype(str).str.replace(',', '',
regex=False),
            errors='coerce'
        ).fillna(float('inf'))
        results_df_real.sort_values(by='sort_key_loss', ascending=True,
inplace=True)
        results_df_real.drop(columns=['sort_key_loss'], inplace=True)

results_df_real
```