Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розробка системи індивідуалізації показу реклами в мобільних іграх за допомогою контекстуальних бандитів
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-3
Богдан Горбенко
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник доц. Вадим Шергін
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Горбенку Богдану Геннадійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка системи індивідуалізації показу реклами в мобільних іграх за допомогою контекстуальних бандитів

затверджена наказом університету від 19 травня 20 25 р. № 378Ст

2. Термін подання студентом роботи до екзаменаційної комісії 17 червня 20 25 р.

3. Вихідні дані до роботи Наукові публікації з тематики машинного навчання та монетизації ігор, аналітичні статті з відкритих джерел, документація Python та бібліотек Scikit-learn, NumPy, Pandas, OpenAI Gym, дані про поведінку користувачів мобільних ігор, симульовані набори даних для навчання та тестування моделей контекстуальних бандитів.

4. Перелік питань, що потрібно опрацювати в роботі
1) Теоретичні основи та аналіз предметної галузі
2) Обґрунтування та методологія застосування контекстних бандитів в рекламних стратегіях
3) Розробка системи контекстуального бандита
4) Тестування та перспективи впровадження

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Аналіз предметної галузі	21.05.2025	виконано
3	Дослідження теоретичних основ	23.05.2025	виконано
	контекстуальних багаторуких бандитів		
4	Аналіз існуючих алгоритмів персоналізованої	25.05.2025	виконано
	реклами		
5	Вибір та опис математичних моделей	27.05.2025	виконано
6	Проєктування структури системи та архітектури	29.05.2025	виконано
7	Реалізація базових алгоритмів у симульованому	02.06.2025	виконано
	середовищі		
8	Проведення експериментів та збір результатів	05.06.2025	виконано
9	Аналіз ефективності та порівняння з A/B	07.06.2025	виконано
	тестуванням		
10	Написання пояснювальної записки	08.06.2025	виконано
11	Перевірка на академічний плагіат	09.06.2025	виконано
12	Нормоконтроль	10.06.2025	виконано
13	Підготовка презентації та доповіді	13.06.2025	виконано
14	Попередній захист	14.06.2025	виконано
15	Рецензування	15.06.2025	виконано
16	Захист перед ЕК	17.06.2025	

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Вадим Шергін
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 186 с., 16 рис., 2 табл., 13 дод., 42 джерел.

АДАПТИВНІ СИСТЕМИ, КОНТЕКСТУАЛЬНІ БАГАТОРУКІ БАНДИТИ, МОБІЛЬНІ ІГРИ, МОНЕТИЗАЦІЯ, ПЕРСОНАЛІЗАЦІЯ, РЕКЛАМНІ СТРАТЕГІЇ, ШТУЧНИЙ ІНТЕЛЕКТ, UX, FREE-TO-PLAY.

Об’єкт дослідження – процес монетизації у мобільних free-to-play іграх.

Предмет дослідження – використання контекстуальних бандитів для адаптивного вибору рекламної стратегії.

Мета роботи – дослідження та впровадження контекстуальних бандитів для оптимізації рекламної монетизації зі збереженням позитивного користувацького досвіду.

Методи дослідження – аналіз сучасних моделей монетизації, математичне моделювання задачі контекстуального багаторукого бандита, реалізація алгоритмів, тестування в симульованому середовищі.

У роботі розглядаються сучасні підходи до монетизації мобільних free-to-play ігор із фокусом на адаптивній рекламі. Особлива увага приділяється застосуванню контекстуальних бандитів, які дозволяють у реальному часі персоналізувати вибір рекламних стратегій для кожного гравця на основі його поведінкових характеристик та контексту. Розроблена система адаптивного вибору реклами забезпечує баланс між максимізацією миттєвого доходу та підтримкою ключових метрик утримання користувачів. Проведені експерименти показали переваги контекстуальних бандитів над традиційним A/B тестуванням у підвищенні ефективності реклами та покращенні користувацького досвіду. Запропонована методика має перспективи для впровадження у комерційні мобільні ігри, що сприятиме сталому розвитку продукту та підвищенню доходів.

ABSTRACT

Bachelor’s thesis contains: 186 pp., 16 fig., 2 tabl., 13 ann., 42 references.

ADAPTIVE SYSTEMS, ADVERTISING STRATEGIES, ARTIFICIAL INTELLIGENCE, CONTEXTUAL MULTI-ARMED BANDITS, FREE-TO-PLAY, MOBILE GAMES, MONETIZATION, PERSONALIZATION, UX.

Object of study – the monetization process in mobile free-to-play games.

Subject of study – the use of contextual bandits for adaptive selection of advertising strategies.

Purpose of the work – to research and implement contextual bandits to optimize advertising monetization while preserving a positive user experience.

Research methods – analysis of modern monetization models, mathematical modeling of the contextual multi-armed bandit problem, implementation of algorithms, testing in a simulated environment.

The work examines modern approaches to monetization of mobile free-to-play games with a focus on adaptive advertising. Special attention is given to the application of contextual bandits, which enable real-time personalization of advertising strategy selection for each player based on their behavioral characteristics and context. The developed adaptive advertising selection system ensures a balance between maximizing immediate revenue and maintaining key user retention metrics. Experiments demonstrated the advantages of contextual bandits over traditional A/B testing in improving advertising effectiveness and enhancing user experience. The proposed methodology has prospects for implementation in commercial mobile games, contributing to sustainable product development and increased revenue.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

КБ – контекстуальний бандит;

ε-greedy – алгоритм вибору дії з балансом між дослідженням і експлуатацією;

ARPPU – Average Revenue Per Paying User – середній дохід на платного користувача;

ARPU – Average Revenue Per User – середній дохід на користувача;

Bounce Rate – коефіцієнт відмов (покидання гри або реклами);

Churn Rate – коефіцієнт відтоку користувачів;

CTR – Click-Through Rate – коефіцієнт клікабельності;

DDA – Dynamic Difficulty Adjustment – динамічне коригування складності;

DR – Doubly Robust – подвійно надійна оцінка;

eCPM – effective Cost Per Mille – ефективна вартість за тисячу показів реклами;

IAA – In-App Advertising – реклама всередині застосунку;

IAP – In-App Purchases – покупки всередині застосунку;

IPS – Inverse Propensity Score – оцінка несміщеності у виборі дій;

LinUCB – лінійне розширення UCB для контекстних задач;

LTV – Lifetime Value – пожиттєва цінність користувача;

MAB – Multi-Armed Bandit – багаторукий бандит;

Skip Rate – коефіцієнт пропуску реклами;

Thompson Sampling – баєсовий алгоритм вибору дії;

UCB – Upper Confidence Bound – алгоритм верхньої довірчої межі;

UX – User Experience – користувацький досвід;

VCR – Video Completion Rate – коефіцієнт завершення перегляду відео.

ВСТУП

Монетизація є невід’ємною складовою розробки цифрових продуктів. Пошук балансу між максимізацією прибутку та забезпеченням позитивного користувацького досвіду (UX) є критичним завданням для підтримки довгострокових відносин із аудиторією.

Штучний інтелект (ШІ) все ширше застосовується для автоматизації процесів прийняття рішень у різних сферах. Зокрема, в останні роки великої популярності набувають адаптивні системи, що здатні коригувати свої стратегії в реальному часі, враховуючи змінюваний контекст. В даній роботі розглядається застосування методів ШІ для оптимізації реклами в мобільних іграх. Хоча описані підходи можуть бути застосовані й у інших сферах, де відбувається взаємодія з користувачами.

Мова йде про безкоштовні мобільні ігри (free-to-play), де монетизація здебільшого ґрунтується на перегляді реклами та внутрішніх покупках. Гравець отримує доступ до основного контенту гри безкоштовно, але періодично бачить рекламні вставки, які можна пропустити через 5 секунд, або має можливість переглянути довші рекламні ролики в обмін на винагороду. Крім того, існують внутрішні магазини, що пропонують товари чи валюту за реальні гроші. У деяких іграх за плату користувач може повністю вимкнути рекламу. Зважаючи на різнорідність аудиторії, традиційні методи, зокрема A/B тестування, не завжди здатні забезпечити персоналізовану взаємодію з кожним гравцем. З цієї причини застосування методів машинного навчання, зокрема контекстуальних багаторуких бандитів (contextual multi-armed bandits), стає все більш популярним. Такі методи дозволяють адаптивно вибирати оптимальну стратегію монетизації для кожного гравця, враховуючи його поведінковий контекст у режимі реального часу.

Метою цієї роботи є дослідження та впровадження контекстуальних бандитів для адаптивного вибору рекламної стратегії в мобільних іграх.

Робота спрямована на досягнення балансу між максимізацією рекламного доходу та покращенням користувацького досвіду. Система повинна навчатися на основі поведінки гравця, з часом пропонуючи найбільш ефективні стратегії.

Завдання дослідження включають:

– аналіз вимог до монетизаційної системи та формування набору контекстних ознак;

– вибір і реалізація алгоритму контекстуального бандита для адаптивного вибору рекламної стратегії;

– розробка системи для навчання та оцінки ефективності рекламних стратегій в реальному часі;

– тестування ефективності системи в симульованому середовищі.

Таким чином, робота спрямована на створення інтелектуальної системи прийняття рішень, здатної персоналізувати монетизаційний досвід кожного користувача, забезпечуючи сталий дохід для розробників без шкоди для геймплею.

1 ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Сучасні підходи до монетизації мобільних ігор

Сучасні моделі монетизації мобільних free-to-play-проєктів базуються на поєднанні покупок всередині застосунку (In-App Purchases, IAP), реклами всередині застосунку (In-App Advertising, IAA) та їхніх гібридних варіацій [1]. У безкоштовних мобільних іграх потік трафіку розгалужується на декілька каналів монетизації (рисунок 1.1), що дозволяє аналізувати баланс доходу та UX. Емпіричні дані засвідчують, що поєднання IAP та IAA забезпечує вищий середній дохід, ніж використання лише одного з підходів; при цьому оптимальна структура доходу залежить від жанру гри та платформи. Крім того, регулярне оновлення рекламних креативів і сегментація аудиторії лишаються необхідними для зменшення рекламної втоми та підтримання утримання користувачів (див. таблицю 1.1).

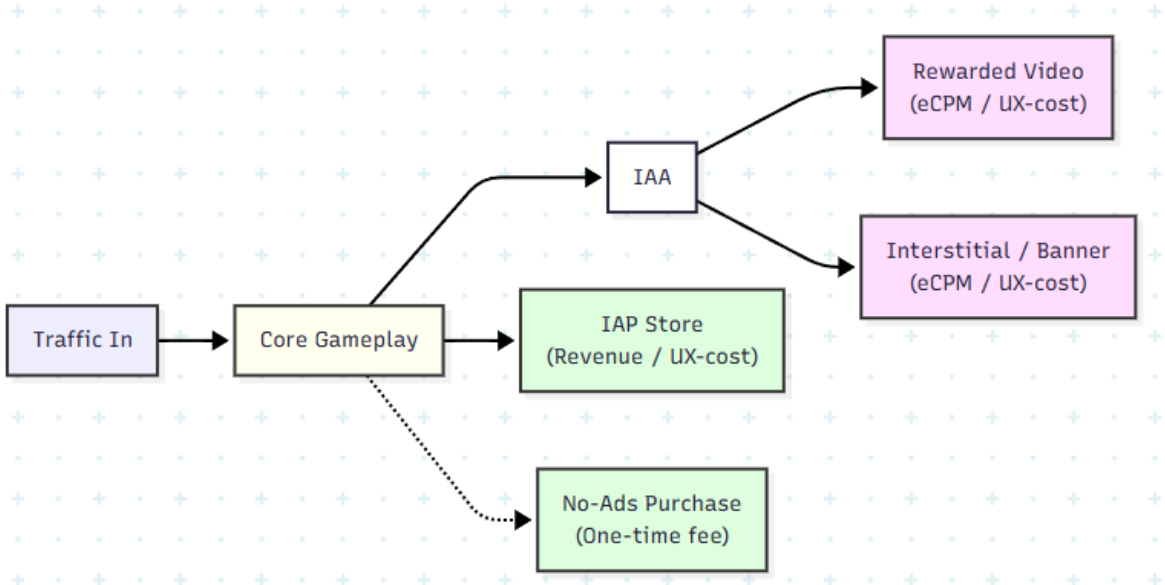


Рисунок 1.1 – Схема потоків монетизації у free-to-play грі

Таблиця 1.1 – Порівняння моделей монетизації F2P-ігор за ключовими метриками

Модель	Швидкість доходу	Потенціал ARPU	Ризик відтоку
IAP	Середня (≈30 % доходу до D30, беззбитковість D7–D14)	Високий (D90 ARPU ≈ $7.3)	Низький
IAA	Дуже швидко (≈64 % доходу до D3)	Низький (D90 ARPU ≈ $0.5)	Високий
Гібрид	Швидше за IAP (≈55 % до D7)	Дуже високий (D90 ARPU ≈ $9.7)	Середній (канібалізація)

1.1.1 Винагороджувальна відеореклама

Винагороджувальна відеореклама (Rewarded Video Ads) передбачає добровільний перегляд короткого ролика в обмін на внутрішньоігрову вигоду (валюта, бустери, додаткові життя тощо). Дослідження мобільного маркетингу показують, що більшість гравців віддає перевагу саме цьому формату, вважаючи його менш нав’язливим порівняно з інтерстиціальними оголошеннями [2], [3]. Позитивне ставлення до винагороджувальної реклами підвищує ймовірність повторної взаємодії та збільшує час перебування у грі.

1.1.2 Покупки всередині застосунку

IAP дають змогу гравцеві придбати віртуальні предмети або контент за реальні кошти. Водночас надмірна залежність прогресу від платежів

може призвести до відтоку користувачів; тому баланс між безкоштовним та платним прогресом є критичним [4]. Персоналізовані пропозиції та обмежені за часом акції залишаються основними практиками стимулювання купівельної активності.

1.1.3 Ключові метрики

Оцінювання ефективності монетизаційної стратегії ґрунтується на низці показників [5]:

– Retention Day 1 / Day 7 / Day 30 – базові індикатори залученості аудиторії;

– ARPU (Average Revenue per User) та ARPPU (Average Revenue per Paying User) – середній дохід відповідно з усього пулу юзерів і з платних користувачів;

– ROAS (Return on Ad Spend) – співвідношення рекламних витрат та отриманого доходу;

– LTV (Lifetime Value) – інтегральна оцінка потенційного доходу від користувача протягом усього життєвого циклу, що розраховується за формулою:

$$\text{LTV} = \sum_{t=1}^{T} \frac{r_t \cdot p_{\text{survive}}(t)}{(1+d)^t}, \tag{1.1}$$

де r_t – очікуваний дохід від користувача в період t;

$p_{\text{survive}}(t)$ – ймовірність того, що користувач залишиться активним до періоду t;

d – ставка дисконту;

T – максимальна кількість періодів, що розглядаються.

1.2 Контекстуальні бандити як метод адаптивного прийняття рішень

Контекстуальні бандити походять від класичної задачі «багаторукого бандита» й використовуються для послідовного вибору однієї дії з множини альтернатив за наявності часткової інформації про середовище [6]. На відміну від базової моделі, у контекстуальному варіанті алгоритм враховує опис поточного стану (контекст), що дозволяє приймати більш обґрунтовані рішення [7].

Робота алгоритму відбувається ітеративно: на кожному кроці спостерігається контекст, оцінюються очікувані винагороди для всіх дій, стохастично обирається дія, після чого отримана винагорода використовується для оновлення політики. Такий цикл забезпечує постійне самонавчання та адаптацію до змінних умов [8].

Ключове завдання – збалансувати дослідження потенційно вигідних, але мало вивчених дій і експлуатацію вже перевірених альтернатив. Адаптивне регулювання цього балансу дає змогу не втрачати можливості покращення, водночас підтримуючи поточний рівень винагороди [8]. Завдяки цьому контекстуальні бандити ефективні у доменах із динамічними вподобаннями користувачів, зокрема в онлайн-рекламі та персоналізованих рекомендаціях [9].

1.2.1 Порівняння з A/B-тестуванням

Аналіз показує, що на відміну від традиційного A/B тестування, яке застосовує фіксований «переможний» варіант до всієї аудиторії, контекстуальний бандит формує рішення для кожного користувача окремо, враховуючи його поточний контекст [10]. Модель оновлюється в реальному часі, скорочуючи тривалість впливу неефективних варіантів і забезпечуючи швидку реакцію на зміни в поведінці користувачів. Це дозволяє краще

адаптуватися до змінюваних вподобань, що підвищує залученість користувачів і потенціал монетизації порівняно зі статичними моделями.

Крім того, на відміну від традиційного A/B тестування, яке підтримує фіксоване розподілення варіантів, стратегія epsilon-greedy оптимізує розподіл ресурсів поступово. Як видно на рисунку 1.2, в A/B тесті (зліва) кожен варіант демонструється певній частині аудиторії протягом усього тестового періоду. У стратегії epsilon-greedy (справа) частка аудиторії, що отримує найуспішніший варіант (синя ділянка), зростає з кожним кроком завдяки зменшенню експлуатації менш ефективних альтернатив (червона та зелена ділянки). Такий підхід дозволяє більш гнучко і ефективно використовувати рекламний контент або продукти, що забезпечує кращі результати для бізнесу завдяки постійному навчанню та адаптації до уподобань користувачів.

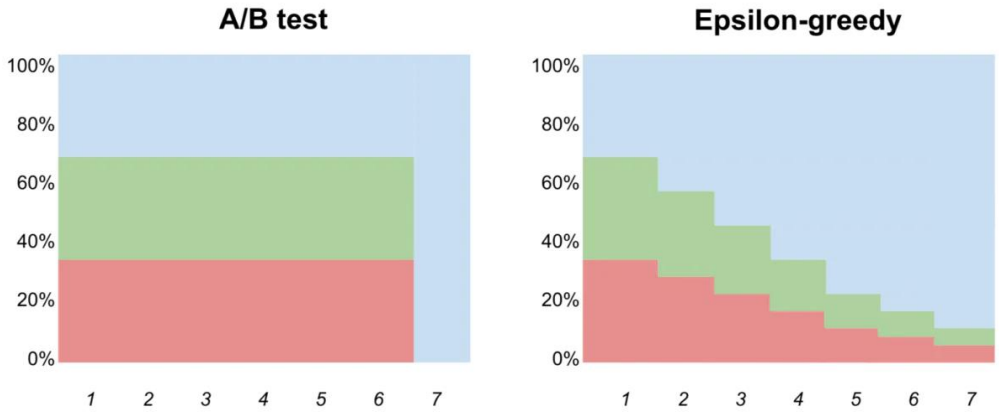


Рисунок 1.2 – Порівняння класичного A/B-тесту та ε-greedy у динамічному розподілі трафіку між варіантами експерименту [10]

1.2.2 Розширені можливості адаптивного прийняття рішень

Багатоцільові контекстуальні бандити (MOC-MAB) дають змогу одночасно оптимізувати декілька потенційно конфліктних

критеріїв (наприклад, релевантність контенту й показники справедливості) [11].

Бандити з обмеженнями (Bandits with Knapsacks, Bandits with Concave Rewards and Convex Knapsacks) вводять ресурсні або бюджетні ліміти й придатні для задач, де рішення мають задовольняти додаткові обмеження [12].

1.3 Приклади застосування контекстуальних бандитів

1.3.1 Персоналізовані рекомендації

Контекстуальні бандити зарекомендували себе як ефективний метод формування індивідуалізованих списків контенту. У роботі [7] запропоновано алгоритм LinUCB, що автоматично добирає новинні статті з урахуванням поточних ознак користувача, демонструючи перевагу над фіксованими евристиками. Подальший розвиток ідеї представлено в дослідженні [13], де як оцінювач винагороди використано глибоку нейронну мережу; такий підхід краще моделює нелінійні уподобання. У виробничих сервісах Netflix і Spotify контекстуальні бандити формують, відповідно, добір обкладинок і порядок музичних каруселей, що забезпечує безперервну персоналізацію каталогу [14], [15].

1.3.2 Онлайн-реклама та адаптивні експерименти

Для маркетингових кампаній важливо поєднати оптимізацію і поступове накопичення знань. Система WayLift компанії Wayfair реалізує контекстуальний бандит, який обирає тип стимулу для конкретного користувача; історія рішень журналюється разом із ймовірностями показу, що дає змогу застосувати офлайн-оцінку IPS/DR [16]. У платформі JD.com побудовано експериментальну систему Comparison Lift, де бандит

адаптивно розподіляє трафік між кількома креативами, об’єднуючи тестування та оптимізацію в єдиному процесі [17].

1.3.3 Динамічне ціноутворення та система купонів

У ритейлері ASOS фреймворк DISCO поєднує Thompson Sampling з цілочисельною оптимізацією, забезпечуючи персональний розподіл купонів у межах заданого бюджету [18]. У теоретичній постановці feature-based dynamic pricing контекстуальний бандит із безперервними діями коригує ціну товару, спираючись на його властивості та контекст попиту [19].

1.3.4 Підтримка клінічних рішень

У роботі [20] продемонстровано, як контекстуальний бандит використовують для індивідуального добору дози варфарину, мінімізуючи ризик побічних ефектів. Аналогічно, у цифровій інтервенції «Oralytics» [21] алгоритм адаптивно надсилає пацієнтам мотиваційні повідомлення, покращуючи показники дотримання терапії.

1.3.5 Оптимізація під ресурсні обмеження

У ситуаціях, де необхідно враховувати фінансові чи логістичні ліміти, доцільно застосовувати модель Resourceful Contextual Bandits. Автори [22] формалізують проблему як задачу з обмеженням типу «knapsack» і доводять коректність прийняття рішень за наявності ресурсних обмежень.

1.4 Умови практичного впровадження контекстуальних бандитів

Контекстуальні бандити доцільно запроваджувати на етапі, коли інформаційна система вже накопичила журнали взаємодій «контекст – дія –

винагорода». За відсутності таких логів алгоритм був би змушений досліджувати середовище майже випадково, що неприпустимо для комерційного сервісу з помітним трафіком.

По-перше, для навчання моделі необхідні propensity-scores – ймовірності, з якими попередня (логуюча) політика обирала ті чи інші дії. Наявність пропенсіті-скорів забезпечує можливість несміщеної офлайн-оцінки нових політик за методами IPS або Doubly Robust [23], [24].

По-друге, у виробничих системах (Yahoo! News, Netflix, Wayfair) бандит-алгоритм запускають лише після warm-start-фази, коли вже зібрано репрезентативний набір випадково-розподілених або A/B-логів. Так, у проєкті WayLift (Wayfair) 10 % трафіку відводили під напіввипадкову політику, щоб сформувати початковий корпус прикладів перед першим тренуванням моделі [16].

По-третє, проблему cold start зазвичай розв’язують простішими rule-based підходами або класичними A/B-тестами. Щойно обсяг логів стає статистично достатнім, контекстуальний бандит інтегрують у продакшн-конвеєр, використовуючи накопичені дані як первинний бюджет знань [14].

Отже, застосування контекстуальних бандитів слід розглядати як фазу розвитку аналітичної системи, що охоплює:

– формування історичних даних під контрольованою або випадковою політикою;

– офлайн-навчання і перевірку політик за IPS/DR;

– поступове онлайн-розгортання з гарантованими межами regret.

1.5 Постановка задачі

1.5.1 Мета, об’єкт і предмет дослідження

Мета дослідження – розробити та експериментально перевірити методику застосування контекстуальних бандитів для адаптивного вибору

рекламних дій у free-to-play мобільній грі, одночасно оптимізуючи миттєвий рекламний дохід (eCPM) і метрики користувацького досвіду (Retention, Bounce Rate).

Об’єкт дослідження – процес взаємодії «гра – користувач» у частині монетизації.

Предмет дослідження – алгоритми контекстуальних бандитів і методи побудови функції винагороди під практичними обмеженнями.

1.5.2 Вихідні передумови

Для реалізації системи адаптивного вибору рекламних стратегій за допомогою контекстуального бандита необхідно забезпечити наявність кількох ключових компонентів. По-перше, система повинна мати доступ до журналів даних, що містять інформацію про «контекст – дія – винагорода». Ці дані дозволяють коректно оцінити ефективність рекламних стратегій, відслідковуючи поведінку гравців та реакцію на різні типи реклами, наприклад, перегляд або взаємодія з рекламою, клік чи покупка. Додатково, необхідно застосовувати propensity-scores для коригування вибору рекламних стратегій, що дозволяє уникнути зміщення при оцінці результатів через змінність ймовірностей вибору певних стратегій у різних контекстах.

Крім того, система повинна мати можливість онлайн-навчання та адаптації в реальному часі. Це включає в себе швидке оновлення політики реклами на основі зворотного зв’язку від користувачів, що дозволяє підтримувати її актуальність та ефективність у змінному середовищі гри. Важливим є забезпечення балансу між миттєвим рекламним доходом (eCPM) та довгостроковими метриками залучення гравців, такими як Retention і Churn Rate. Це дозволяє розробити стратегію, що оптимізує як фінансові цілі, так і утримання користувачів у грі.

1.5.3 Вимоги до системи

Для ефективної роботи контекстуального бандита в умовах мобільних ігор система повинна відповідати кільком критичним вимогам. Однією з основних вимог є низька латентність при виборі рекламної стратегії. Час відповіді системи на запит про вибір стратегії не повинен перевищувати допустимі межі, щоб зберегти високий рівень користувацького досвіду і уникнути затримок у взаємодії з гравцями.

Додатково, масштабованість системи є важливою вимогою. Система повинна бути здатною обробляти значні обсяги запитів, зокрема при збільшенні кількості користувачів і варіативності рекламних стратегій. Врахування цих вимог забезпечить її здатність ефективно функціонувати в умовах постійного росту даних.

Особливістю адаптивних систем є адаптивність. Алгоритм контекстуального бандита має здатність оперативно реагувати на зміни в поведінці гравців, коригуючи рекламні стратегії на основі актуальних даних, що дозволяє забезпечити максимальну ефективність реклами в кожному конкретному контексті.

Нарешті, швидкодія системи повинна бути оптимізована не лише для прийняття рішень у реальному часі, але й для навчання бандита на нових даних без затримок у часі. Система повинна швидко адаптувати стратегії та надавати відповідні рекламні варіанти без значних втрат ефективності.

1.5.4 Формалізація задачі

Метою розробки контекстуального бандита є максимізація миттєвого рекламного доходу (eCPM), одночасно з мінімізацією негативних наслідків для утримання користувачів (наприклад, зниження рівня Retention або збільшення Churn Rate). У зв'язку з цим, задача може бути формалізована як вибір рекламної стратегії a_A на основі контексту x_X, де A – це множина

можливих рекламних стратегій (наприклад, interstitial, rewarded video, no-ad), а X – простір контекстних змінних (поведінка гравців, історія взаємодії з рекламою, технічні параметри пристрою).

Винагорода для кожної стратегії a у відповідному контексті x може бути визначена як функція двох критеріїв:

$$R(x, a) = eCPM(x, a) - \lambda \cdot BounceRate(x, a), \qquad (1.2)$$

де $eCPM(x, a)$ – миттєвий рекламний дохід, що генерується стратегією a в контексті x;

$BounceRate(x, a)$ – ймовірність того, що гравець відмовиться від подальшої взаємодії з рекламою чи покине гру після перегляду реклами;

λ – коефіцієнт штрафу, що регулює вагу балансу між доходом та утриманням користувачів.

Задача бандита полягає в мінімізації cumulative regret, тобто відставання між фактичними досягнутими результатами та оптимальним результатом, який можна було б отримати, якщо б усі стратегії були відомі заздалегідь:

$$Regret(T) = \sum_{t=1}^{T} R(x_t, a_t) - \max_{a \in A} \sum_{t=1}^{T} R(x_t, a_t), \qquad (1.3)$$

де T – кількість кроків (часових періодів, запитів до системи);

x_t – контекст на кроці t;

a_t – вибрана стратегія на кроці t.

У системі контекстуального бандита необхідно постійно оновлювати політику вибору стратегії на основі нових даних. Моделювання ймовірностей вибору стратегії та відповідних результатів для кожного контексту можна здійснювати через алгоритми, такі як Epsilon-Greedy, UCB (Upper Confidence Bound) або Thompson Sampling, які дозволяють

знаходити баланс між exploration (дослідження нових стратегій) і exploitation (використання найкращих відомих стратегій).

На діаграмі (рисунок 1.3) видно, як агент (Agent) отримує спостереження (Observation) від середовища (Environment), а на основі цього спостереження використовує політику (Policy) для прийняття дій (Action). Після виконання дії агент отримує винагороду (Reward), що використовує для оновлення політики (Policy Update) в рамках алгоритму навчання (Reinforcement Learning Algorithm). Це ілюструє, як працює підхід до вибору стратегії на основі контексту в реальному часі, що є основою для формулювання задачі контекстуального бандита.

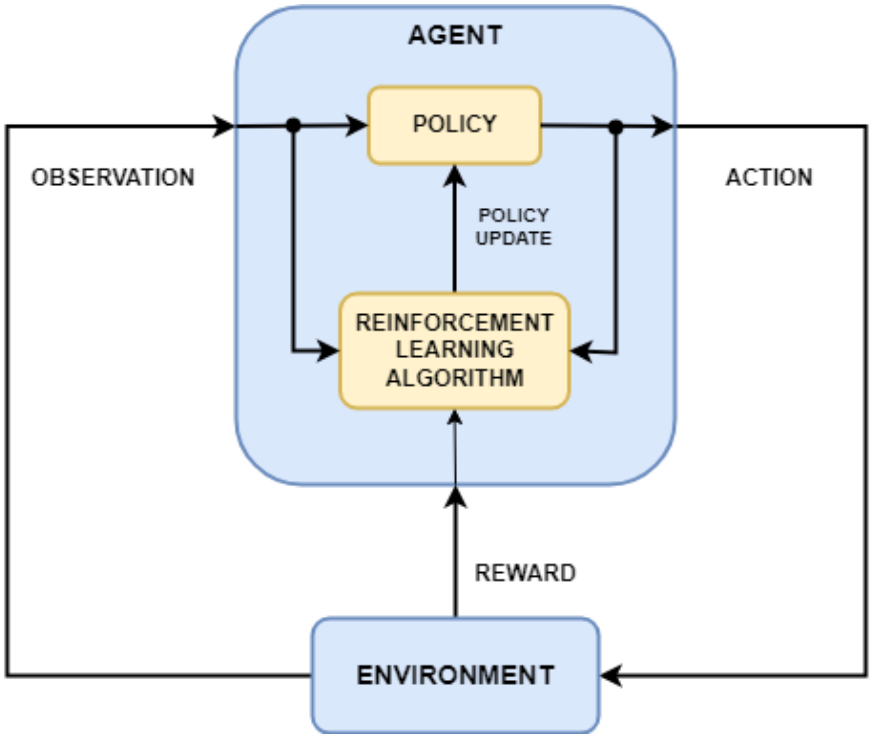


Рисунок 1.3 – Алгоритм навчання контекстуального бандита

Задача має обмеження на частоту показу реклами, що дозволяє уникати надмірного навантаження на користувачів, а також обмеження по часу відповіді системи, яке не повинно перевищувати максимальний поріг для забезпечення належного користувацького досвіду.

Формалізація задачі також включає визначення динаміки навчання моделі, що передбачає поступове оновлення параметрів на основі нових даних про реакції користувачів, що дозволяє системі адаптуватися до змін у поведінці гравців протягом часу.

1.5.5 Підзадачі, що розв’язуються у подальших розділах

У подальших розділах буде розглянуто розв'язання ключових задач, необхідних для розробки та реалізації контекстуального бандита для адаптивного вибору рекламної стратегії в мобільних іграх. Зокрема, передбачається вирішення таких задач:

а) аналіз вимог до монетизаційної системи та формування набору контекстних ознак:

– опис основних характеристик контексту гравця, які використовуватимуться для прийняття рішень, таких як поведінка користувачів, типи взаємодії з рекламою, технічні параметри пристроїв;

– оцінка значення таких метрик, як eCPM, Retention та Churn Rate, для побудови коректної функції винагороди;

б) проектування архітектури з виділенням модулів збору даних, оцінки контексту та політики вибору дій:

– проектування структури бази даних для зберігання інформації про користувачів, їх поведінку, історію взаємодії з рекламою;

– опис модулів для збору та обробки даних у реальному часі, а також для інтеграції з рекламними стратегіями;

в) вибір і реалізація алгоритму контекстуального бандита:

– опис основних алгоритмів контекстуальних бандитів (ε-greedy, LinUCB, Thompson Sampling);

– вибір найефективнішого алгоритму для адаптивного вибору реклами в ігровому контексті;

– формулювання математичних моделей і параметрів для моделювання поведінки гравців (врахування ймовірностей покупок, покидання гри, реакції на рекламу);

г) навчання та калібрування моделі за історичними логами (розділ 2.4):

– оцінка результатів навчання на основі зібраних даних;

– налаштування параметрів алгоритму для оптимізації результатів;

– порівняння адаптивної стратегії з традиційними методами вибору реклами (A/B тестування);

д) тестування у контрольних сценаріях та порівняння з базовими стратегіями:

– проведення тестів у симульованому середовищі за участю ботів;

– оцінка метрик для визначення ефективності вибору рекламної стратегії: CTR, конверсії, середній дохід, швидкодія моделі;

е) перспективи впровадження та проблеми реалізації (розділ 3.2):

– оцінка перспектив використання контекстуальних бандитів в реальних іграх;

– масштабування системи для обробки великої кількості користувачів;

– ідентифікація технічних обмежень та проблем при впровадженні адаптивних рекламних стратегій.

2 ОБҐРУНТУВАННЯ ТА МЕТОДОЛОГІЯ ЗАСТОСУВАННЯ КОНТЕКСТНИХ БАНДИТІВ В РЕКЛАМНИХ СТРАТЕГІЯХ

2.1 Опис предметної галузі

2.1.1 Важливість персоналізації рекламних стратегій для гравців

Персоналізований обмін повідомленнями визначається як процес адаптації змісту повідомлень до індивідуальних характеристик користувача і є критичним для підвищення рівня їхнього залучення. У мобільних іграх цей підхід дозволяє встановлювати міцніші зв’язки з гравцями, подовжувати їхню активність та формувати віддану аудиторію. Навіть дрібне доповнення – наприклад, використання тегу даних «user_name» – може викликати зростання CTR на 130 % [25].

Контекстні бандити у цьому випадку слугують алгоритмічною основою персоналізації повідомлень. На відміну від традиційних алгоритмів, що застосовують однакові правила для всіх користувачів, контекстні бандити враховують контекст (вікову групу, поведінкові патерни, поточний стан у грі тощо) для динамічного прийняття рішень [26]. Одним із найпоширеніших практичних застосувань такого підходу є персоналізована реклама в ігрових додатках, що дозволяє значно підвищити релевантність повідомлень і оптимізувати рекламні витрати [27].

2.1.2 Вплив реклами на гравців (залучення, втримання, фінансові показники)

Модель Freemium, яку застосовують більшість мобільних ігор, поєднує безкоштовний базовий доступ із двома незалежними джерелами доходу: внутрішніми покупками (IAP) та рекламною експозицією [28]. Зокрема, компанії на кшталт Avid.ly використовують рекламу в додатку для

монетизації неплатоспроможних гравців, одночасно зберігаючи та підтримуючи надходження від IAP [29]. У досліджуваній грі найпоширенішою формою рекламного контенту є короткі відеоролики тривалістю 10–30 секунд перед кожним новим раундом; кожен додатковий раунд підвищує загальну кількість показів і, відповідно, обсяг доходу від реклами.

Кейс-стаді Legend of Slime демонструє, що хоча IAP генерують значну частку виручки, до 70 % сукупного доходу надходить саме від інтегрованої реклами, яка вбудована у мета-ігровий прогрес через 12–20 показів на гравця на добу. Натомість SayGames успішно масштабує монетизацію, застосовуючи гібридну стратегію з приблизним співвідношенням доходів від реклами та IAP 50/50 [30].

Динамічне коригування складності (dynamic difficulty adjustment, DDA) – форма data-driven персоналізації ігрового досвіду – виявилося ефективним інструментом підвищення залучення та прогресу гравців у порівнянні з контрольною групою. Користувачі, які отримали доступ до полегшених рівнів, проходили більше раундів та досягали значно більшого прогресу; через підвищене залучення вони також генерують більше рекламних показів. Отже, DDA породжує подвійний позитивний ефект на утримання аудиторії та зростання доходу від реклами в Freemium-моделі [28].

2.1.3 Використання машинного навчання для адаптивних стратегій реклами

Контекстні бандити відносяться до класу адаптивних алгоритмів рекомендацій, що оперують багатовимірним простором можливих пропозицій і здійснюють «online самовдосконалення» на основі особливостей окремого користувача [31]. Дані методи знаходять

застосування у персоналізації контенту на вебсайтах або в рекламних кампаніях.

Основний цикл контекстних бандитів складається з таких кроків: спочатку здійснюється збір контексту (характеристик користувача чи середовища), потім обирається дія (наприклад, демонстрація відповідного креативу), після чого фіксується винагорода (клік або його відсутність) і нарешті політика оновлюється з урахуванням накопиченої історії взаємодій. Головна мета алгоритму – максимізувати суму отриманих винагород за весь час роботи.

Машинне навчання відіграє ключову роль у оптимізації креативів за допомогою контекстних бандитів, забезпечуючи показ найбільш релевантних оголошень для конкретних сегментів аудиторії [32]. Використовуючи підхід із навчанням з підкріпленням, де враховуються як позитивні, так і негативні взаємодії, алгоритм аналізує низку контекстних ознак (очікувана стать, вік, розміщення) і динамічно розподіляє покази між кількома креативами, на відміну від традиційного A/B-тестування, де одночасно випробовується лише два варіанти.

Платформа Kameleoon також інтегрує контекстні бандити для розширених можливостей експериментування та персоналізації, що дозволяє динамічно адаптувати розподіл трафіку відповідно до поведінкових характеристик кожного користувача [26]. Замість статичної порівнювальної оцінки варіантів у A/B-тестах, поділ трафіку відбувається в реальному часі на основі контексту, що скорочує цикл отримання інсайтів і підвищує значущість результатів.

У межах dynamic difficulty adjustment (DDA) в Freemium-іграх моделі машинного навчання можуть застосовуватися для прогнозування ризику відтоку гравців [28]. Хоча це не пряме використання ML для рекламної стратегії, такий адаптивний підхід до утримання аудиторії опосередковано впливає на критичні показники монетизації, зокрема на дохід від реклами.

2.1.4 Чому контекстуальний бандит підходить для даної задачі

Контекстуальні бандити – це рішення на основі штучного інтелекту, що адаптує стратегію показу реклами до індивідуальної поведінки кожного гравця в реальному часі [10]. Вони є потужним інструментом масштабування персоналізації й підвищення рівня утримання аудиторії. На відміну від традиційного A/B-тестування, яке порівнює фіксовані варіанти та обирає єдиного «переможця» для всієї аудиторії, контекстуальні бандити динамічно підбирають оптимальний варіант для кожного гравця в потрібний момент. Це дозволяє мати не один універсальний «виграшний» креатив, а низку «переможців», адаптованих до індивідуальних уподобань.

Для задачі індивідуалізації показу реклами в мобільних іграх, де мета полягає в обранні найрелевантнішого рекламного креативу чи формату для кожного гравця на основі його профілю та контексту, контекстуальні бандити мають кілька ключових переваг порівняно з A/B-тестуванням:

– реальна індивідуалізація;

– динамічна адаптація в реальному часі;

– ефективне балансування між exploration (дослідженням) та exploitation (експлуатацією);

– мінімізація втрат від неефективних варіантів;

– відповідність до специфічних ігрових сценаріїв.

Контекстуальні бандити використовують дані користувача (контекст), такі як поведінка в грі, баланс внутрішньої валюти (наприклад, дорогоцінних каменів), досвід, історія покупок тощо, для точнішого ухвалення рішень [7]. Це дозволяє доставляти рекламний досвід, адаптований до кожного індивіда [10], а не оптимізувати під міфічного «середнього» гравця. Наприклад, алгоритм може пропонувати бонусні офери гравцям із низьким балансом ресурсів і водночас показувати інші пропозиції досвідченим користувачам із високим запасом валюти.

Задача вимагає миттєвого вибору реклами «на льоту». Контекстуальні бандити динамічно змінюють варіант показу зі зміною профілю або поведінки гравця, тоді як A/B-тестування залишається статичним і потребує значного часу для збіжності.

Контекстуальні бандити динамічно балансують між дослідженням та експлуатацією [33]. Це означає, що вони не лише використовують знання про успішні варіанти (exploitation), але й продовжують досліджувати нові або менш вивчені рішення (exploration), щоб знаходити ще кращі стратегії для різних контекстів гравців. Це дозволяє швидше виявляти ефективні рекламні креативи для різних сегментів або навіть окремих гравців порівняно з A/B-тестуванням, яке фіксовано розподіляє трафік протягом усього експерименту [34].

Контекстуальні бандити починають оптимізувати негайно. На відміну від A/B-тестів, які можуть тривати тижні чи місяці для досягнення статистичної значущості, вони швидко перенаправляють трафік на ефективні варіанти, зменшуючи вплив показів із низькою конверсією. Це безпосередньо підвищує дохід від реклами, збільшуючи ймовірність конверсії.

Деякі задачі, як визначення оптимальної частоти показу реклами або вибір динамічних пропозицій/оферів, погано підходять для A/B-тестування через значні розбіжності у відповіді гравців. Контекстуальні бандити можуть використовувати недавні ігрові та витратні патерни кожного користувача для підбору найкращої пропозиції чи частоти показів. Оптимізація розміщення реклами (ad placements) є одним із ключових рішень, де контекстуальні бандити демонструють значний вплив [10].

На відміну від контекстуальних бандитів, A/B-тестування обирає один універсальний варіант для всіх гравців, що не враховує індивідуальні відмінності у сприйнятті реклами. Хоча A/B-тести ефективні для порівняння кількох варіантів і отримання загальних інсайтів, вони не масштабуються для персоналізації через необхідність рівномірного

розподілу трафіку й великих вибірок, що може призводити до тривалих витрат на демонстрацію неефективних креативів [34].

Традиційні Multi-Armed Bandits (MAB) також відрізняються від контекстуальних: вони шукають єдиний найефективніший варіант для всієї аудиторії, тоді як контекстуальні бандити визначають «переможців» окремо для кожного профілю користувача [35].

Рекомендаційні системи призначені для вибору серед тисяч або мільйонів елементів (товарів, фільмів тощо) і потребують об'ємних історичних даних, що робить їх ресурсомісткими рішеннями. Для задачі вибору серед десятків або сотень рекламних креативів контекстуальні бандити є «легшою» альтернативою, яка краще справляється з проблемою «холодного старту» [33].

Таким чином, для задачі індивідуалізації показу реклами в мобільних іграх, де необхідний миттєвий вибір з обмеженого набору варіантів на основі мінливого контексту гравця, контекстуальні бандити є найбільш підходящим рішенням завдяки їхній здатності до динамічної, контекстно-орієнтованої персоналізації та ефективного балансування exploration/exploitation.

2.1.5 Визначення ключових параметрів для моделювання

У моделі контекстного бандита для персоналізації показу реклами в мобільних іграх усі ключові ознаки можна розподілити на дві великі категорії: контекст (Context) – набір параметрів, що описують гравця та його поточну ситуацію, і винагорода (Reward) – показники успішності обраної дії. Контекст використовується для формування вхідного вектора x_t, а отримана винагорода – для оновлення політики вибору реклами [7], [36].

Нижче наведено структурований перелік основних параметрів, які слід зібрати та передати в модель (див. таблицю 2.1).

Таблиця 2.1 – Ключові контекстні та винагородні параметри для моделювання контекстного бандита

Група ознак	Конкретні параметри	Призначення
Профільні характеристики	Демографія (вік, стать, регіон)	Початкова сегментація користувачів для налаштування політики exploration/exploitation
	Пристрій (модель, ОС, версія)	
	Канал встановлення	
Поведінкові метрики	Сесії/день, тривалість сесії	Оцінка активності та мотивації гравця; визначення моментів для корекції складності або зміни рекламної стратегії
	Максимальний рівень, середня кількість спроб	
	% виграшів, фрустрація	
Взаємодія з рекламою	CTR (click-through rate)	Формування сигналу винагороди: клік, завершений перегляд або негайна відмова
	VTR та час перегляду	
	Bounce rate після реклами	
Фінансові показники	IAP – кількість, сума, час до першої покупки	Інтеграція довгострокової монетизації в reward-функцію для оптимізації LTV
	ARPU, сукупний дохід	
Контекстні чинники	Час доби, день тижня, ігрові події	Адаптація стратегії показу до зовнішніх умов та внутрішнього стану гравця
	Запас ресурсів (життя, енергія, валюта)	
	Стан мережі	
Індикатори утримання/відтоку	RFM – recency, frequency, monetary	Виявлення «risk» гравців та налаштування рекламних пропозицій для підвищення утримання
	Retention rate (D1, D7 тощо)	

Узагальнюючи, контекстні ознаки з таблиці 2.1 формують багатовимірний вектор, який описує поточний стан користувача й навколишні умови, а взаємодія з рекламою та фінансові метрики слугують сигналом винагороди, що дає змогу алгоритму бандита поступово віддавати перевагу найбільш ефективним рекламним креативам у різних ігрових ситуаціях.

Приклади конкретного застосування параметрів:

– поведінкові метрики: у [28] показано, як ранній прогрес і кількість сесій корелюють із лояльністю гравців;

– фінансові показники: у [38] RFM-метрики й час до першої покупки використовуються для прогнозування CLV та налаштування reward-функції;

– взаємодія з рекламою: у [37] Mean Cumulative Function дозволяє моделювати CTR і VTR як частину винагороди за кожен показ.

Таким чином, правильне визначення й збір зазначених у таблиці 2.1 параметрів забезпечує контекстному бандиту необхідну інформацію для динамічної, контекстно-орієнтованої персоналізації та ефективного балансування exploration/exploitation.

2.1.6 Особливості застосування КБ в ігровому контексті

У мобільних іграх поведінка гравців змінюється швидко та динамічно, тому традиційні підходи (статичні A/B-тести) часто втрачають релевантність. Контекстні бандити (КБ) дозволяють в режимі реального часу адаптувати рекламні рішення до поточного стану гравця, без необхідності тривалого збору даних для кожного нового варіанту.

Основний цикл роботи КБ у грі складається з трьох кроків:

– отримання контексту – збір ознак, що описують гравця та ігрову ситуацію (рівень складності, кількість життів, історія покупок тощо) [7];

– вибір дії – алгоритм обирає один із доступних «рукавів» (наприклад, налаштування складності або формат реклами) на основі оцінки очікуваної винагороди та невизначеності;

– оновлення моделі – після показу реклами фіксується винагорода (клік, перегляд відео, IAP), і модель коригує свої параметри для майбутніх виборів.

Нижче перелічено ключові сценарії, у яких зміни в поведінці гравців безпосередньо впливають на рішення КБ.

Фрустрація та динамічна складність (DDA). Якщо гравець зазнає кілька невдалих спроб поспіль, це інтерпретується як підвищений рівень

фрустрації. КБ може підвищити ймовірність показу rewarded video – роликів із внутрішньо-ігровими бонусами – щоб підтримати мотивацію й утримати гравця в грі [28].

Швидкість прогресу. Підвищена інтенсивність гри (пройдені кілька рівнів за короткий час) сигналізує про високу залученість. У такі моменти алгоритм може віддавати перевагу більш дорогим форматам реклами з вищим eCPM, оскільки активні гравці приносять більшу цінність рекламодавцям [7].

Монетизаційні відгуки. Після першої внутрішньо-ігрової покупки або серії IAP КБ може почати пропонувати більші пакети валюти чи ексклюзивні офери. Для користувачів без покупок алгоритм навпаки стимулює утримання через безкоштовні бонуси за перегляд відео, що відкладає їхній відтік і сприяє довготривалій монетизації [38].

Часові та івент-орієнтовані сигнали. У пікові години активності або під час спеціальних ігрових заходів баланс exploration/exploitation автоматично зсувається в бік exploitation, щоб максимально використати вікно підвищеного інтересу до гри та реклами [7].

Завдяки безперервному оновленню внутрішніх матриць моделей (наприклад, у LinUCB) та адаптації оцінок очікуваної винагороди, контекстні бандити швидко реагують на будь-які зміни в реакціях гравців. Це дозволяє зберегти високу релевантність рекламних показів і, як наслідок, збільшувати середній дохід на гравця та зменшувати відтік аудиторії.

2.2 Методологічна основа розробки контекстного бандита

2.2.1 Алгоритми та математична основа

Контекстні бандити формалізують задачу послідовного вибору дій в умовах часткового спостереження: у кожному раунді t агент спостерігає

контекст x_t, вибирає дію a_t з множини A і отримує винагороду $r_t(a_t)$. Метою є максимізація сукупної винагороди

$$\sum_{t=1}^{n} r_t(a_t), \tag{2.1}$$

або, що еквівалентно, мінімізація псевдо-regret [39]

$$R_n = n\mu^* - \mathbb{E}\left[\sum_{t=1}^{n} \mu_{I_t}\right] = \sum_{i=1}^{K} (\mu^* - \mu_i)\mathbb{E}[T_i(n)], \tag{2.2}$$

де R_n – псевдо-regret після n раундів: різниця між максимально можливою сумарною винагородою та очікуваною винагородою стратегії;

n – загальна кількість раундів (кроків), протягом яких агент обирає дії;

μ^* – середня винагорода оптимальної (найкращої) дії:

$$\mu^* = max_{i \in \{1..K\}} \mu_i; \tag{2.3}$$

$\mathbb{E}[\cdot]$ – оператор математичного сподівання (очікуване значення) за випадковими винагородами та виборами дій алгоритмом;

$\sum_{t=1}^{n} \mu_{I_t}$ – сума середніх винагород тих дій, які були обрані на кроках $t = 1 \dots n$;

I_t – індекс дії, вибраної в раунді t, тобто μ_{I_t} – середня винагорода саме тієї дії, яку агент обрав у кроці t;

K – загальна кількість доступних дій;

μ_i – середня винагорода i-ї дії;

$T_i(n)$ – випадкова величина, що позначає число разів, коли саме i-та дія була обрана до раунду n;

$\mathbb{E}[T_i(n)]$ – очікувана кількість виборів i-ї дії до раунду n.

Ключова дилема таких задач полягає в балансі між:

– exploitation (експлуатацією) – використанням вже відомих найбільш прибуткових дій для негайної винагороди;

– exploration (дослідженням) – випробуванням менш знайомих дій для зменшення невизначеності й підвищення довгострокового прибутку.

У стохастичному середовищі доведено, що псевдо-regret контекстних бандитів може зростати логарифмічно $O(\log n)$, тоді як для безконтекстних задач це зазвичай $O(\sqrt{n})$. Ефективні стратегії поєднують ці два підходи, знижуючи короткострокові втрати та водночас накопичуючи інформацію про всі доступні дії.

У задачах багаторуких та контекстних бандитів розроблено низку алгоритмів, що реалізують різні стратегії балансування exploration та exploitation. Нижче наведені ключові підходи:

– ε-greedy. З імовірністю $1-\varepsilon$ обирає дію з найвищою емпіричною середньою винагородою, а з імовірністю ε випадкову дію, що сприяє дослідженню нових варіантів. При поступовому зменшенні ε до нуля регрет зростає як $O(\log n)$ [39];

– Upper Confidence Bound (UCB). Визначає для кожної дії i верхню довірчу межу

$$\widehat{\mu_i} + \sqrt{\frac{2 \ln t}{T_i(t)}}, \tag{2.4}$$

де $\widehat{\mu_i}$ – емпірична середня винагорода;

$T_i(t)$ – число її виборів до раунду t. Алгоритм обирає дію з найвищою межею, поєднуючи exploitation (через $\widehat{\mu_i}$) і exploration (через невизначеність) [39];

– Thompson Sampling. Використовує баєсовий підхід: у кожному раунді з апостеріорного розподілу для параметрів винагород (θ_i) вибирається випадкова реалізація, а далі – дія з найбільшою очікуваною

винагородою за цим зразком. Нещодавні аналізи показали логарифмічні bound-и регрету для цього алгоритму [39];

– LinUCB / SupLinUCB. Розширення UCB на випадок лінійної залежності винагороди від контекстних ознак x_t. Алгоритми будують довірчі межі для лінійних моделей $\theta^\top x_t$ та обирають дію з максимальною межею. Варті уваги реалізації SupLinUCB для кращої статистичної ефективності [40];

– GP-UCB / KernelUCB. Застосовують гаусові процеси або kernel-функції для моделювання невідомої функції винагороди $f(x_t, a)$. Кожен крок оптимізує верхню довірчу межу в просторі функцій з виміром невизначеності через коваріаційну структуру [40];

– Contextual Thompson Sampling. Комбінує баєсову вибірку з контекстними моделями (лінійною, логістичною регресією тощо), дозволяючи гнучко підлаштовуватися під складні залежності між контекстом і винагородою [40];

– інші підходи. Серед алгоритмів для широкого класу політик та непостійних середовищ особливо відзначаються Exp4 (для експерт-орієнтованих задач), Epoch-Greedy та ILOVETOCONBANDITS, які забезпечують теоретичні гарантії регрету в складних сценаріях [40].

Цей огляд демонструє, що вибір алгоритму залежить від припущень про структуру винагород (стохастичність vs. адверсарійність), характеру контексту та вимог до обчислювальної ефективності.

При розгортанні контекстного бандита в системі адаптивної реклами важливо врахувати характеристики задачі та обмеження платформи:

– розмір і розмірність контексту. Якщо контекстні ознаки є лінійно взаємопов’язаними з ймовірністю кліку чи конверсією, протокол LinUCB забезпечує швидкі оновлення та гарантії регрету $O\left(\sqrt{T}\right)$ в лінійному просторі [40];

– обчислювальні обмеження. Для мобільних ігор із великою кількістю одночасних під’єднань потрібен алгоритм з низькою латентністю

оновлення. Простий ε-greedy або Thompson Sampling у лінійній баєсовій формі часто демонструють достатню продуктивність при мінімальних обчислювальних витратах [39];

– нестійкість середовища. В іграх рекламний контекст може швидко змінюватися (нові івенти, промоакції). Баєсовий підхід Contextual Thompson Sampling природно адаптується до таких змін, оскільки апостеріорний розподіл оновлюється з кожним спостереженням без жорстких гіперпараметрів exploration [40];

– потреба в теоретичних гарантіях. Якщо критично мінімізувати регрет та дотримуватися аналітичних bound-ів, варто віддати перевагу UCB-розширенням (LinUCB, GP-UCB) із строгими межами $O(\log n)$ або $O\left(\sqrt{n}\right)$ залежно від моделі винагород [40].

З огляду на баланс продуктивності та адміністративної простоти у більшості випадків адаптивної реклами в мобільних іграх оптимальним вибором є:

– LinUCB, коли ознаки реклами та користувача добре лінійно віддзеркалюють вплив на клік і конверсію;

– Contextual Thompson Sampling, якщо контекст високодимензійний або змінний, оскільки модель автоматично коригує exploration–exploitation без тонкої підгонки ε чи UCB-параметрів.

У середовищах із суворими вимогами до затримки або обмеженими обчислювальними ресурсами може бути достатньо ε-greedy у поєднанні з лінійною регресією для оцінки винагороди.

2.2.2 Моделювання параметрів для поведінки гравців

Для коректної роботи контекстуального бандита необхідно перетворити поведінкові характеристики гравців на числові ознаки (features), які дозволять алгоритму точно прогнозувати реакцію

користувача на різні рекламні дії. Нижче наведено ключові групи ознак та приклади їх моделювання.

Толерантність до реклами – ймовірність того, що гравець продовжить грати після перегляду рекламного оголошення. Зручно моделювати цю метрику за допомогою логістичної регресії [40]:

$$P_{tol}(x) = \frac{1}{1+\exp\left(-(\beta_0+\beta^\top x)\right)}, \tag{2.5}$$

де x – вектор ознак поведінки гравця (наприклад, тривалість поточної сесії, кількість вже переглянутих відео, інтервал від останнього входу);

β – набір коефіцієнтів, які навчаються на історичних даних;

β_0 – зсув (intercept) моделі.

Ця модель дозволяє адаптувати частоту показів реклами так, щоб не перевантажувати гравця оголошеннями, одночасно підтримуючи достатній рівень монетизації [41].

Наприклад, якщо сесія вже тривала довго і користувач бачив багато відео, велике значення $\beta_1\text{SessionLength} + \beta_2\text{AdsSeen}$ призведе до низького P_{tol}, і система автоматично знизить частоту наступних рекламних показів.

Ймовірність покупки (IAP). У задачі персоналізації реклами критично важливо оцінювати ймовірність внутрішньоігрової покупки (IAP). Цей параметр відображає шанси гравця здійснити хоча б одну транзакцію за заданий інтервал часу і може бути змодельований двома основними способами.

Логістична регресія для бінарного прогнозу. Припустимо, що кожен гравець у кожній сесії або купує, або не купує. Тоді можна ввести бінарну змінну

$$y = \begin{cases} 1, & \text{якщо сталася IAP,} \\ 0, & \text{інакше,} \end{cases} \tag{2.6}$$

і побудувати логістичну регресію

$$P_{\text{IAP}}(x) = \sigma(\beta_0 + \beta^\top x) = \frac{1}{1+e^{-(\beta_0+\beta^\top x)}}, \quad (2.7)$$

де x – вектор контекстних ознак (тривалість сесії, кількість переглянутих оголошень, попередні витрати тощо);

β – оцінені коефіцієнти [41]. Такий підхід дає інтерпретовану ймовірність покупки за фіксований період і легко реалізується у стандартних ML-фреймворках.

Survival-аналіз для прогнозу часу до покупки. Якщо важливо не тільки передбачити факт IAP, а й оцінити її часову динаміку, застосовують Cox-пропорційні ризики [42]:

$$h_{\text{IAP}}(t|x) = h_0(t)\exp(x^\top \gamma), \quad (2.8)$$

$$F_{\text{IAP}}(t|x) = 1 - \exp(-\Lambda_0(t)e^{x^\top \gamma}), \quad (2.9)$$

де $h_0(t)$ і $\Lambda_0(t)$ – базова hazard-функція та її кумулятивна інтеграла;

γ – вектор оцінених коефіцієнтів. Survival-підхід дозволяє прогнозувати розподіл часу до IAP, що дає змогу тонко налаштовувати рекламні пропозиції згідно з очікуваним моментом покупки.

В обох випадках отримаємо числову ознаку – P_{IAP} або $F_{\text{IAP}}(t)$ для кожного гравця, яку слід включити до вектору контексту контекстного бандита. Це дає алгоритму можливість враховувати платоспроможність користувача при виборі рекламного креативу чи формату оголошення.

Ймовірність відтоку відображає шанс того, що гравець припинить активність у грі протягом заданого періоду. Для її оцінки застосовують ті ж методи, що й для прогнозу ймовірності покупки (IAP):

– логістична регресія, яка дає бінарну оцінку «відтоку/утримання» на основі контекстних ознак (частоти сесій, давності останнього входу, тривалості ігрових сесій тощо);

– Survival-аналіз (модель пропорційних ризиків Кокса), що дозволяє моделювати не лише факт відтоку, а й очікуваний час до нього, використовуючи hazard-функцію та кумулятивну інтенсивність.

Значення P_{churn} інтегрується у вектор контексту бандита, завдяки чому рекламні креативи та ігрові інтервенції можуть динамічно підлаштовуватися під рівень ризику втрати користувача.

Метрики залученості описують інтенсивність та регулярність взаємодії гравця з грою й формуються як числові ознаки для контекстного бандита:

– середня тривалість сесії – середній час від початку до завершення ігрової сесії. Обчислюється як сумарний ігровий час, поділений на кількість сесій за вибраний період. Ця метрика відображає, наскільки гравець залучений в геймплей до появи перерв або фрустрації;

– частота сесій – кількість сесій на день або тиждень. Висока частота свідчить про стабільне повернення гравця до гри, тоді як зниження цього показника може вказувати на втрату інтересу чи надмірне навантаження рекламою;

– інтервал між сесіями – середній час між завершенням однієї сесії та початком наступної. Короткі інтервали сигналізують про високу мотивацію та задоволення від геймплею, тоді як довгі вказують на необхідність додаткових стимулів (наприклад, винагородної реклами) для утримання гравця.

Значення цих трьох метрик включаються до вектора контексту x_t контекстного бандита, дозволяючи алгоритму адаптувати рекламні рішення залежно від поточного рівня залученості кожного гравця.

Показники прогресу відображають здатність гравця долати ігрові виклики та служать ключовими ознаками для адаптації складності та реклами:

– середня кількість спроб на рівень («failure streak») – кількість невдалих проходжень перед успішним завершенням рівня, усереднена за вибраний період. Висока середня свідчить про перевищення складності, що може потребувати показу rewarded video для зниження рівня фрустрації;

– відношення успішних спроб (win rate) – частка вдалих проходжень від загальної кількості спроб. Цей показник дозволяє бандиту визначити, чи гра відповідає вмінню гравця, і відповідно коригувати частоту реклами або DDA-інтервенції;

– поточний рівень – індекс найвищого досягнутого рівня. Бандит може використовувати його для налаштування eCPM-рівня реклами, оскільки гравці на вищих рівнях зазвичай цінніші для рекламодавців;

– швидкість переходу між рівнями – середній час або кількість сесій, необхідних для переходу з одного рівня на наступний. Швидка зміна вказує на високу мотивацію та спроможність гравця до прогресу, що може виправдати показ реклами з вищими ставками, тоді як уповільнення радить знизити комерційний тиск .

Реакція на DDA-інтервенції – ключові ознаки, що відображають, як зміна складності впливає на поведінку гравця та може бути використана бандитом для подальшої адаптації:

– зміни залученості до/після коригування складності – різниця в кількості раундів або ігрових сесій, які гравець проходить у день до DDA та після неї. Позитивна дельта сигналізує про успішність інтервенції і може підтримувати підвищену ставку реклами; негативна – свідчить про перенавантаження, що вимагає зниження рекламного навантаження;

– час до першого успіху після зміни складності – інтервал від моменту застосування DDA до першого виграшу рівня. Короткий час до успіху

вказує на коректну оцінку рівня складності, висока затримка – на потребу подальшого пом’якшення складності або показу reward-реклами.

Рекламні поведінкові метрики в нашій системі слугують сигналами для контекстного бандита, вказуючи на ефективність різних форматів оголошень та реакцію гравців.

CTR (Click-Through Rate) – частка кліків від загальної кількості показів оголошення. Використовується для оцінки первинного інтересу до креативу. Обчислюється як

$$\text{CTR} = \frac{\text{кількість кліків}}{\text{кількість показів}}. \tag{2.10}$$

VCR (Video Completion Rate) – відношення повністю переглянутих відео до загальної кількості початих показів. Обчислюється як

$$\text{VCR} = \frac{\text{кількість завершених переглядів}}{\text{кількість початих переглядів}}. \tag{2.11}$$

Показник важливий для формату «rewarded/non-rewarded» video та відображає рівень залученості до відео.

Skip Rate – частка гравців, які пропустили рекламу до завершення, від загальної кількості переглядів. Обчислюється як

$$\text{Skip Rate} = 1 - \text{VCR}. \tag{2.12}$$

Високий Skip Rate свідчить про перевантаження рекламою або нерелевантність креативу.

Реакція на «rewarded vs. non-rewarded ads» – порівняння метрик (CTR, VCR, Skip Rate) між відео, що дають винагороду, та рекламою без винагороди. Зазвичай rewarded ads демонструють вищий VCR і нижчий Skip

Rate, проте можуть знижувати CTR для наступних non-rewarded показів через «винагородну втомленість» гравців.

Усі ці метрики перетворюються на числові ознаки для бандита, дозволяючи алгоритму зважати як на загальну ефективність оголошень, так і на індивідуальні реакції гравців при підборі формату реклами в реальному часі.

Фінансово-монетизаційні показники використовуються для оцінки доходності гравців та налаштування рекламних стратегій з урахуванням їхньої платоспроможності:

а) ARPDAU (Average Revenue Per Daily Active User) – середній дохід на одного активного гравця за день. Дозволяє відстежувати, скільки прибутку приносить середній користувач щодня. Обчислюється як

$$\text{ARPDAU} = \frac{\text{загальний добовий дохід}}{\text{кількість DAU}}; \tag{2.13}$$

б) ARPPU (Average Revenue Per Paying User) – середній дохід на одного гравця, який здійснив покупку. Дає змогу оцінити цінність платної аудиторії незалежно від усіх активних користувачів. Розраховується за формулою

$$\text{ARPPU} = \frac{\text{загальний дохід від IAP}}{\text{кількість унікальних платників}}; \tag{2.14}$$

в) кількість і сума внутрішньоігрових покупок:

– Purchase Count – загальна кількість здійснених транзакцій за період;

– Purchase Volume – сумарна грошова вартість цих транзакцій.

Ці метрики відображають глибинні показники монетизації та дозволяють виявляти, як різні сегменти гравців реагують на пропозиції.

Усі фінансові ознаки інтегруються в контекстний вектор бандита, щоб алгоритм міг балансувати між збільшенням доходу та якістю ігрового досвіду.

Соціальні та мережеві фактори відображають ступінь інтеграції гравця в ігрову спільноту та впливають на його мотивацію повернутися до гри або здійснити покупку. До ключових ознак належать:

– кількість друзів / рефералів. Вимірюється числом унікальних користувачів, яких гравець запросив у гру або з якими він пов’язаний через внутрішньоігрову мережу. Великий реферальний коефіцієнт корелює з підвищеним рівнем утримання та монетизації, оскільки соціальні зв’язки створюють додаткову мотивацію для регулярних повернень і взаємодії;

– активність у спільних ігрових подіях. Оцінюється кількістю участей гравця в командних або змагальних івентах (турніри, рейди тощо) за певний період, а також середньою тривалістю цих сесій. Висока частота та успішність у таких подіях свідчить про глибоку соціальну залученість і пов’язана з більшим часом у грі та підвищеною готовністю до внутрішньоігрових покупок.

Часові та зовнішні умови впливають на сприйнятливість гравців до рекламних повідомлень та відображаються в їхньому контекстному векторі:

– час доби / день тижня. Сприйнятливість до реклами може суттєво відрізнятися залежно від того, коли гравець заходить у гру. Наприклад, ранкові сесії перед роботою чи навчанням мають коротшу тривалість, але вищу концентрацію уваги, тоді як вечірні сесії після роботи – довші та з більшою готовністю взаємодіяти з контентом. Оптимізація моменту показу реклами відповідно до часового профілю користувача дозволяє підвищити CTR та VCR;

– наявність ігрових івентів чи акцій. Під час внутрішньоігрових подій (турнірів, сезонних івентів, святкових акцій) мотивація гравців зростає, і вони частіше реагують на релевантні пропозиції. Контекстний бандит може збільшити частоту показу rewarded-відео чи спеціальних

пропозицій саме в період активності івенту, оскільки конверсія в цей час значно вища.

Уключення цих ознак у вектор контексту дає можливість алгоритму виявляти оптимальні моменти для показу реклами, максимально використовуючи пікові періоди залученості та підвищуючи ефективність монетизації.

2.2.3 Моделювання ймовірностей на основі статистичних даних з досліджень

Контекстні бандити покладаються не лише на теоретичні припущення про розподіли винагород, але й на реальні спостереження гравців. Емпіричні оцінки ймовірностей – наприклад, частота здійснення внутрішньоігрових покупок чи рівень відтоку – забезпечують бандиту «прив'язку» до фактичної поведінки користувачів і дозволяють уникнути надто довгого етапу сліпого дослідження [40].

Поєднання формальних моделей (логістичної регресії, survival-аналізу тощо) з даними спостережень дає змогу:

– швидко ініціалізувати параметри моделі на основі історичних метрик (наприклад, середнього CTR або кумулятивної функції подій);

– адаптувати політику вибору реклами відповідно до реальних когорн та нових трендів у поведінці гравців;

– зменшити ризик «перевчення» на теоретично чистих, але практично невідповідних даних.

У процесі побудови контекстного бандита критично важливо спиратися не лише на формальні ймовірнісні моделі, але й на емпіричні оцінки частот реальних подій. Агрегація частот (empirical frequency estimation) дозволяє отримати початкові апріорні ймовірності важливих реакцій користувачів – від внутрішньоігрових покупок до відтоку – у різних контекстах і надалі використовувати їх для «розігріву» моделі бандита.

Спочатку розбивають кожну контекстну змінну (наприклад, тривалість сесії, поточний рівень, кількість переглянутих відео) на інтервали (біни). Це може бути фіксоване розбиття за шириною біну або адаптивне розбиття таким чином, щоб у кожному біні було приблизно однакове число спостережень. Далі для кожного біну обчислюють частоту настання події: IAP, churn чи перегляду реклами. Якщо подія може відкладатися у часі (наприклад, покупка не відбулася під час спостережень), використовують правозорієнтовані методи оцінки, аби не занижувати справжню ймовірність (survival-aware frequency).

Для візуальної перевірки отриманих частот будують гістограми ймовірностей у залежності від бінів та застосовують KDE (kernel density estimation) для згладжування розподілів. Така візуалізація допомагає виявити, наприклад, нетипові піки ймовірності покупки у гравців із тривалими сесіями або зниження залученості напередодні важких рівнів.

Отримані емпіричні частоти інтегрують у саму політику контекстного бандита: на початкових ітераціях вони виступають апріорними ймовірностями при виборі дій (warm start), а з накопиченням нових даних частоти оновлюються у режимі реального часу. Завдяки цьому бандит може швидше балансувати exploration і exploitation, спираючись не лише на модельні припущення, а й на доволі достовірні емпіричні оцінки поведінки гравців у різних контекстах.

У задачах адаптивної реклами та монетизації мобільних ігор важливо не лише знати «місячний» або «щоденний» показник IAP чи відтоку, а й відстежувати накопичувальну динаміку цих подій у часі. Для цього застосовують mean cumulative function (MCF) – очікувану кількість подій на одного гравця за певний інтервал часу.

Нехай для кожного гравця i ми маємо часові мітки подій $t_{i1}, t_{i2}, \dots, t_{iN_i}$ (наприклад, моменти покупок або факти відтоку). MCF у момент t визначається як

$$\mathrm{MCF}(t) = \frac{1}{n} \sum_{i=1}^{n} N_i(t), \quad (2.15)$$

де $N_i(t)$ – кількість подій для гравця i до часу t;

n – розмір вибірки. Іншими словами, MCF показує середню кумулятивну кількість подій на користувача як функцію часу.

Для MCF внутрішньоігрових покупок (IAP) використовують часові мітки кожної покупки, а для відтоку – мітку першого «пропущеного» дня після останньої сесії. На практиці дані часто правозорієнтовані (censored), тобто гравець може залишатися активним понад період спостереження. У цьому випадку оцінку MCF виконують за методом Калмогорова–Німеца, коригуючи внесок кожного гравця відповідно до часу цензури [37].

Градуйований вигляд MCF дозволяє виявити точки перелому – моменти, коли rate подій раптово змінюється, наприклад, коли IAP починає рідшати або відтік зростає (наприклад, після складного «гейту»). Ці часові «розриви» стають сигналом для бандита: у точках, що передують підвищенню відтоку, алгоритм може збільшити частоту reward-відео, а коли зростає IAP-активність, переключитися на креативи з вищим eCPM [28].

Таким чином, MCF не лише дає змогу формалізувати динаміку подій у часі, а й інтегрувати ці кумулятивні криві як додатковий контекст у політику контекстного бандита, що підвищує адаптивність рекламної стратегії до реальної поведінки користувачів.

Розглянемо підхід до аналізу утримання користувачів за допомогою когорного аналізу та survival-кривих. Ці методи дозволяють оцінити час до події (наприклад, відтоку) у різних підгрупах гравців.

Когорти формуються на основі фіксованих характеристик користувачів у момент встановлення або досягнення певного рівня гри (наприклад, «завантажено у червні 2025» чи «досягнуто 50-го рівня»). У кожній когорті відстежується час від цієї точки до моменту першої відсутності сесії протягом принаймні 7 днів. Точний вибір ознаки когорти (дата інсталяції, початковий рівень, джерело трафіку) залежить від

бізнес-гіпотези і має бути узгоджений з ключовими метриками монетизації та утримання.

Для кожної когорти обчислюють Kaplan–Meier survival-криву, яка оцінює ймовірність «виживання» (утримання) у грі до моменту t.

Нехай T_i – час від встановлення до відтоку користувача i. Оцінка функції виживання задається як

$$\hat{S}(t) = \prod_{t_j \le t}(1 - \frac{d_j}{n_j}), \tag{2.16}$$

де d_j – кількість відтоків у момент t_j;

n_j – число гравців, які «вижили» до t_j. Графічне порівняння кривих для різних когорт дозволяє виявити, які групи гравців утримуються довше.

Щоб перевірити гіпотези про статистично значущі відмінності між survival-кривими когорт, використовують log-rank тест. Цей непараметричний тест оцінює, чи є відмінність у розподілі часу до події між двома або більшими групами. Значущий результат $p < 0.05$ свідчить про різницю в утриманні між когортами.

Для підвищення точності та стабільності роботи контекстуального бандита доцільно інтегрувати емпіричні оцінки ймовірностей безпосередньо в процес навчання.

Емпірично обчислені ймовірності (наприклад, частота IAP у певній когорті, відсоток гравців, що витримують понад 14 днів) додаються до векторів ознак кожного раунду. Це дозволяє моделі бандита враховувати не лише поточний контекст користувача, а й узагальнені патерни поведінки його однолітків. Такі фічі допомагають алгоритму швидше зорієнтуватися у нових або малодосліджених сегментах.

Логістичні регресії (для P_{tol} і P_{IAP}) та Cox-моделі (для часу до події) добре піддаються «холодному старту», якщо їхні початкові β-коефіцієнти встановити згідно з емпіричними оцінками з історичних даних. Наприклад,

середній рівень толерантності до реклами або типова hazard-функція для певної когортної групи використовуються як базові значення перед початком онлайн-навчання. Це забезпечує швидший перехід до стабільних прогнозів без необхідності довгого накопичення нових спостережень.

Оскільки емпіричні ймовірності можуть містити шум та сезонні коливання, важливо застосовувати регуляризатори (L1, L2) та обмежувати вагові коефіцієнти моделі бандита, щоб запобігти перенавчанню на випадкові викиди. Також корисно використовувати методи перехресної валідації всередині когорних підрозділів для відбору оптимальної сили регуляризації, зберігаючи баланс між гнучкістю моделі та її узагальнювальною здатністю.

Об’єднання формальних моделей із емпіричними оцінками дозволяє підвищити точність контекстного бандита завдяки адаптації до реальних патернів поведінки гравців. Використання частотних ймовірностей, когорних survival-кривих та MCF сприяє коректному налаштуванню початкових параметрів моделей, зменшує ризик overfitting і забезпечує швидку конвергенцію алгоритму. Водночас емпіричні методи дають змогу виявляти критичні точки зміни поведінки – наприклад, переломні моменти переходу до відтоку чи сплески IAP – і вбудовувати ці знання безпосередньо у вектор контексту.

Для остаточної перевірки ефективності рекомендованих налаштувань необхідно провести серію A/B-експериментів із реальними аудиторіями. Це дасть змогу оцінити приріст ключових метрик під контролем і порівняти адаптивну стратегію з фіксованими підходами.

3 РОЗРОБКА СИСТЕМИ КОНТЕКСТУАЛЬНОГО БАНДИТА

У цьому розділі описуються етапи проєктування та реалізації системи контекстуального бандита, зосереджуючись на її архітектурних компонентах, виборі програмного забезпечення та процесі конфігурації.

3.1 Проєктування системи та архітектурні компоненти

3.1.1 Огляд сервісу контекстного бандита

Основною метою цього проєкту є розробка та впровадження надійної служби контекстуальних бандитів, призначеної для динамічного й висококонкурентного середовища мобільних ігор. Ця служба виконує роль централізованого інтелектуального шару, переходячи від традиційних статичних рекламних стратегій до індивідуалізованих рекламних рекомендацій у режимі реального часу. Її головне призначення – оптимізувати показ внутрішньоігрової реклами шляхом динамічного добору найвідповіднішої реклами для конкретного гравця в певному ігровому контексті, що дозволяє одночасно підвищити залучення, монетизацію та покращити користувацький досвід.

Як показано на рисунку 3.1, служба контекстуальних бандитів функціонує як інтелектуальний посередник. Клієнт мобільної гри, досягнувши моменту прийняття рішення щодо показу реклами, надсилає запит із релевантною контекстуальною інформацією про гравця та поточний стан гри. Служба обробляє цей контекст, звертається до своїх моделей контекстуального бандита, що постійно навчаються, і повертає оптимальний рекламний варіант (або «важіль») клієнту гри. Після цього клієнт повідомляє службу про результат показу реклами (наприклад, чи взаємодіяв із нею гравець, яку отримано винагороду), що дозволяє моделям

продовжувати навчання та адаптацію. Цей ітеративний зворотній зв'язок є ключовим елементом адаптивності системи.

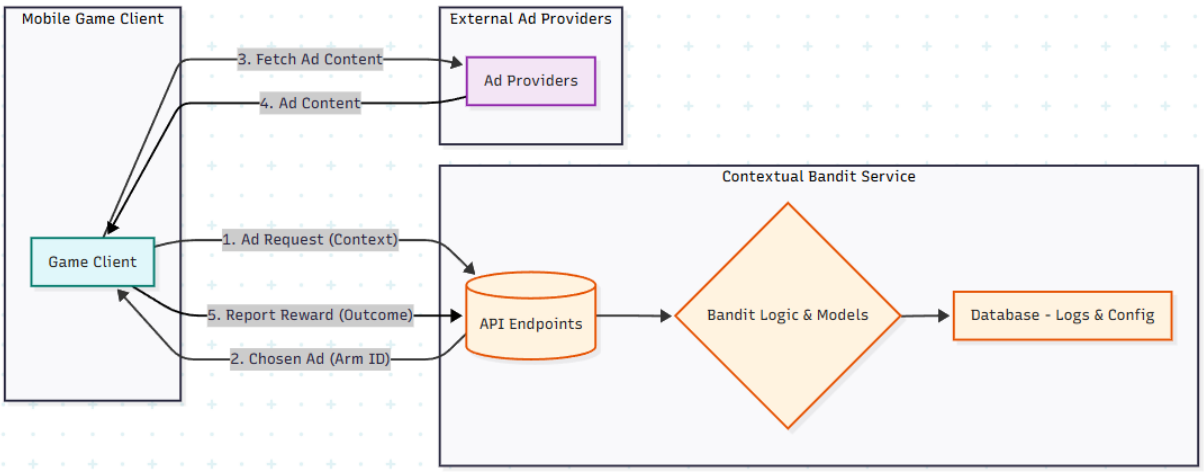


Рисунок 3.1 – Концептуальна схема ролі служби в екосистемі мобільної гри

Ця сервісно-орієнтована архітектура має низку ключових переваг:

– логіка персоналізації реклами повністю відокремлена від основного клієнта гри. Це забезпечує незалежну розробку, розгортання та оновлення системи бандитів без потреби в патчах для клієнта гри чи складній інтеграції. Розробникам необхідно лише працювати з чітко визначеним API;

– завдяки централізації прийняття рішень, службу можна масштабувати незалежно для обробки змінних навантажень рекламних запитів. Із зростанням кількості гравців або точок прийняття рішень ресурси можуть бути спрямовані безпосередньо на бандит-сервіс, що гарантує стабільну продуктивність;

– усі процеси навчання та прийняття рішень у різних сценаріях показу реклами зосереджені в одному місці. Це дозволяє ефективно керувати кількома моделями бандитів, використовувати спільні потоки даних і мати узагальнену картину ефективності реклами в межах однієї або кількох ігор. Такий підхід також сприяє застосуванню складних методів машинного

навчання, реалізація яких безпосередньо у клієнтах мобільних ігор є недоцільною.

3.1.2 Проєктування схеми бази даних для контекстуального бандита

Надійна та чітко структурована схема бази даних є основою функціонування сервісу контекстуального бандита, забезпечуючи зберігання як конфігурацій бандитів, так і журналів усіх взаємодій. Система використовує дві основні таблиці: decision_points – для збереження метаданих і налаштувань кожної задачі, та interactions – для фіксації кожного прийнятого рішення й відповідного результату. Повний вихідний код ORM-моделі бази даних знаходиться в додатку А.

Таблиця точок взаємодії призначена для зберігання статичних і динамічних налаштувань кожної окремої задачі бандита (або «точки прийняття рішення») у мобільній грі. Кожен рядок відповідає окремому сценарію, у якому система має здійснити адаптивний вибір.

Відношення між цими полями та гіпотетичним класом застосунку DecisionPoint (наприклад, у середовищі Python) полягає в наступному:

– name: відображається безпосередньо на ідентифікатор або ярлик задачі бандита (наприклад, DecisionPoint.id або DecisionPoint.name); використовується для отримання коректної конфігурації бандита з бази даних;

– arms: рядок у форматі JSON, що описує набір можливих дій або «важелів», з яких бандит може обирати; у класі застосунку цей рядок, ймовірно, десеріалізується у список рядків або об’єктів Python, наприклад:

```
DecisionPoint.available_arms = json.loads(db_record.arms);
```

– context_schema: JSON-рядок, який визначає очікуваний формат і типи даних контекстних ознак, що передаються моделі бандита; застосунок використовує це для валідації вхідних контекстних даних або для керування побудовою ознак;

– learning_policy_name: поле, що визначає основний алгоритм багаторукого бандита (наприклад, Epsilon-Greedy, LinUCB); застосунок динамічно створює відповідний екземпляр алгоритму за цим рядком:

```
DecisionPoint.policy_instance = PolicyFactory
.get_policy(db_record.learning_policy_name);
```

– neighborhood_policy: JSON-рядок з конфігураціями для більш складних стратегій контекстного бандита, що можуть включати групування або «сусідства» контекстів; десеріалізується та передається як параметри для екземпляра політики:

```
DecisionPoint.policy_instance.configure_neighborhoods(
json.loads(db_record.neighborhood_policy));
```

– cold_start_min_records: ціле число, що визначає мінімальну кількість взаємодій, необхідних для переходу моделі бандита з фази активного дослідження («холодний старт») до фази більш цілеспрямованого навчання; застосунок керує початковою поведінкою моделі за допомогою цього значення:

```
DecisionPoint.cold_start_threshold = db_record
.cold_start_min_records.
```

Отже, таблиця decision_points слугує постійним сховищем конфігурацій, що забезпечує динамічне завантаження й налаштування різних екземплярів задач бандита без необхідності змін у коді чи повторного розгортання.

Таблиця взаємодій має ключове значення для навчальних і аналітичних можливостей Сервісу контекстного бандита. Вона фіксує кожну взаємодію:

– контекст, що був поданий;

– дію, обрану бандитом;

– винагороду, отриману внаслідок цієї дії.

Стовпець timestamp має критично важливе значення. За замовчуванням він встановлюється на поточний час, що забезпечує точний хронологічний запис подій. Цей хронологічний порядок необхідний для:

– навчання моделі: дає змогу бібліотеці mabwiser послідовно навчатися на основі взаємодій;

– аналітики: забезпечує точний розрахунок показників ефективності з часом, зокрема кумулятивної середньої винагороди, що відображає криву навчання та ефективність бандита;

– налагодження та аудиту: надає чітку хронологію подій для діагностики та розуміння поведінки моделі.

3.1.2.3 Вибір бази даних і реалізація ORM

Для етапів розробки та початкового розгортання цього проєкту було обрано SQLite як серверну частину бази даних. Це рішення було зумовлене насамперед простотою, файловою природою та легкістю налаштування, що є ідеальним для прототипування та локального тестування. SQLite не потребує окремого серверного процесу, що робить її надзвичайно зручною для демонстрації функціональності без складного адміністрування бази даних.

Для взаємодії з базою даних SQLite у стійкий і «пайтонічний» спосіб було використано SQLAlchemy як об’єктно-реляційний відображувач (ORM). SQLAlchemy виступає як потужний рівень абстракції, що дозволяє розробникам працювати з базою даних за допомогою об’єктів і методів Python замість прямого написання сирих SQL-запитів. Перехід від рядково орієнтованого SQL до підходу, заснованого на ORM, має низку переваг:

– абстракція сирого SQL: замість формування SQL-рядків взаємодія з базою даних відбувається через високорівневі об’єкти Python; наприклад, створення нового запису взаємодії передбачає ініціалізацію об’єкта

Interaction та його додавання до сесії, замість написання виразу INSERT INTO ... VALUES (...);

– зменшення жорсткого кодування та ризику помилок: у SQLAlchemy імена таблиць і стовпців визначаються один раз у класах ORM (наприклад, DecisionPoint, Interaction); усі наступні операції над базою посилаються на відповідні атрибути класів Python (наприклад, Interaction.chosen_action), що усуває необхідність багаторазово хардкодити назви полів і суттєво знижує ризик помилок через друкарські помилки;

– підвищення типобезпечності: визначення стовпців у SQLAlchemy (Column(String), Column(Integer) тощо) забезпечує певний рівень перевірки типів на рівні ORM, що гарантує відповідність вставлених у базу даних значень очікуваним типам; хоча Python є динамічно типізованим, ORM-додаток додає важливий рівень валідації;

– спрощене управління схемою: метод Base.metadata.create_all, що використовується в модулі database.py, автоматично створює необхідні SQL-інструкції CREATE TABLE на основі моделей ORM; це значно полегшує початкове налаштування бази даних і дає змогу ефективніше відстежувати та впроваджувати зміни у схемі в міру розвитку застосунку порівняно з ручним супроводом DDL-скриптів SQL.

Інтеграція SQLAlchemy забезпечує чистий, підтримуваний і менш схильний до помилок рівень роботи з базою даних, що формує надійну основу для Сервісу контекстного бандита.

3.1.3 Збереження та управління моделями

Для забезпечення ефективної роботи контекстного бандита в промисловому середовищі необхідно зберігати результати навчання – натреновані моделі – між перезапусками застосунку та мати їх у постійному доступі для прийняття рішень у режимі реального часу. У цій системі збереження моделей реалізується шляхом зчитування й запису моделей

mabwiser та об’єктів StandardScaler (що використовуються для нормалізації контексту) на диск і з диска.

Основним механізмом для серіалізації та десеріалізації є вбудований модуль Python pickle. Він перетворює об’єкти Python у байтовий потік (серіалізація), який можна зберігати у файлі, а потім відновлює початкові об’єкти з цього потоку (десеріалізація). Це дає змогу зберегти складний стан бандита mabwiser і об’єкта StandardScaler. Серіалізовані файли моделі зберігаються в каталозі models_data/ з унікальними іменами файлів, сформованими на основі назви точки прийняття рішення (наприклад, mab_DailyOffer.pkl, scaler_DailyOffer.pkl).

Клас застосунку DecisionPoint (файл models/decision_point.py, вихідний код наведено в додатку Б) несе повну відповідальність за керування життєвим циклом цих файлів моделі. Він інкапсулює логіку збереження та завантаження моделей, гарантуючи, що кожен екземпляр точки прийняття рішення зберігає свій навчений стан.

3.1.3.1 Збереження моделі

Метод save_models класу DecisionPoint викликається щоразу, коли змінюється стан моделі, зазвичай після операції часткового навчання partial_fit у режимі онлайн або первинного навчання batch_fit. Це гарантує, що найактуальніша версія знань бандита зберігається на диску.

Важливим аспектом управління моделями є врахування змін конфігурації. Щоб запобігти завантаженню застарілої моделі, натренованої з використанням іншої політики навчання, до об’єкта StandardScaler перед його серіалізацією додається спеціальний атрибут _saved_policy_name. Цей атрибут зберігає назву learning_policy, що була активною під час останнього збереження скейлера (і, опосередковано, моделі MAB). У лістингу 3.1 наведено відповідний фрагмент коду з файлу models/decision_point.py:

Лістинг 3.1 – Програмний код функції збереження моделі

```
    def save_models(self):
        if self.scaler is not None:
            self.scaler._saved_policy_name =
self.learning_policy_name
            with open(self.scaler_file, 'wb') as f:
                pickle.dump(self.scaler, f)
        if self.mab is not None:
            with open(self.mab_file, 'wb') as f:
                pickle.dump(self.mab, f)
```

3.1.3.2 Завантаження та ініціалізація моделі

Метод load_or_initialize викликається під час запуску застосунку або при створенні екземпляра DecisionPoint. Його основне призначення – перевірити наявність існуючих файлів моделі на диску.

Якщо виявлено обидва файли – моделі mabwiser та об’єкта StandardScaler, метод намагається їх завантажити. Під час цього процесу виконується критична перевірка: порівнюється атрибут _saved_policy_name завантаженого скейлера з learning_policy_name, налаштованим для поточного екземпляра DecisionPoint:

– якщо назви політик збігаються, завантажені моделі вважаються чинними та використовуються;

– якщо виявлено невідповідність (що вказує на зміну налаштованої політики з моменту останнього збереження) або виникає будь-яка помилка під час завантаження (наприклад, пошкоджений файл, несумісна версія pickle), система фіксує попередження та переходить до повторної ініціалізації моделі;

– якщо модель не знайдено або виконується повторна ініціалізація, метод перевіряє кількість історичних записів у базі даних (get_record_count_for_decision_point). Якщо кількість записів

дорівнює або перевищує порогове значення cold_start_min_records, виконується початкове пакетне навчання (perform_initial_batch_fit) на основі всієї наявної історичної інформації. Інакше модель залишається в режимі «холодного старту», накопичуючи дані до досягнення порогу.

Цей надійний механізм завантаження гарантує, що система завжди працює з правильною конфігурацією моделі та здатна адаптуватися до змін у політиці навчання бандита. Відповідний фрагмент з models/decision_point.py наведено в лістингу 3.2.

Лістинг 3.2 – Програмний код функції завантаження та ініціалізації моделі

```
    def load_or_initialize(self):
        is_random_baseline = (self.learning_policy is None)
        if is_random_baseline:  # Random Baseline
            self.mab = None
            self.requires_contexts_for_fit_and_update = False
            if os.path.exists(self.scaler_file):
                try:
                    with open(self.scaler_file, 'rb') as f:
                        self.scaler = pickle.load(f)
                except Exception as e:
                    self.scaler = StandardScaler()
            else:
                self.scaler = StandardScaler()
            self.save_models()
            return
        if os.path.exists(self.mab_file) and
os.path.exists(self.scaler_file):
            try:
                with open(self.mab_file, 'rb') as f:
                    loaded_mab = pickle.load(f)
                with open(self.scaler_file, 'rb') as f:
                    loaded_scaler = pickle.load(f)
```

Продовження лістингу 3.2

```
                if hasattr(loaded_scaler,
'_saved_policy_name') and \
                   loaded_scaler._saved_policy_name ==
self.learning_policy_name:
                    self.mab = loaded_mab
                    self.scaler = loaded_scaler
                    self.requires_contexts_for_fit_and_update
= self.mab.is_contextual
                    return
                else:
            except Exception as e:
        record_count =
get_record_count_for_decision_point(self.name)
        if record_count >= self.cold_start_min_records:
            self.perform_initial_batch_fit()
        else:
            self.mab = None
            self.scaler = None
            self.requires_contexts_for_fit_and_update = False
```

Ця комплексна стратегія збереження та управління моделлю гарантує, що сервіс Contextual Bandit зберігає свій навчений стан, адаптується до змін конфігурації та коректно відновлюється після перезапусків, забезпечуючи безперервну оптимізацію показу реклами.

3.1.4 Вебінтерфейс (адміністративна панель)

Критичним компонентом сервісу Contextual Bandit є його адміністративна вебпанель, призначена для надання повного інструментарію з управління системою бандитів. Ця панель дає адміністраторам змогу налаштовувати точки прийняття рішень, відстежувати їхню ефективність у режимі реального часу та виконувати

ключові завдання з управління даними. Інтерфейс реалізовано з використанням стандартних вебтехнологій, що забезпечує доступність і адаптивність.

Фронтенд адміністративної панелі побудований із використанням:

– HTML для визначення структури сторінки та вмісту;

– Bootstrap для створення адаптивного й візуально привабливого дизайну, зручного для використання на різних пристроях і екранах;

– JavaScript для динамічного завантаження контенту, обробки форм та, передусім, інтеграції з Chart.js для інтерактивної візуалізації даних.

Адміністративна панель організована у кілька ключових представлень, кожне з яких виконує окрему функцію в межах робочого процесу управління.

3.1.4.1 Панель моніторингу

Панель моніторингу є центральною сторінкою адміністративного інтерфейсу (рисунок 3.2), що надає загальний огляд усіх налаштованих точок прийняття рішень у системі. Для кожної точки адміністратори можуть оперативно визначити її поточний робочий стан, зокрема, чи перебуває вона у фазі «холодного старту» (накопичення даних) або є «активною» (виконує навчання та формує передбачення). Такий узагальнений огляд дає змогу швидко виявити точки, що потребують уваги або додаткового налаштування.

Крім цього, панель містить ключові метрики ефективності для кожної точки, зокрема середню винагороду, кількість взаємодій, рівень довіри до моделей та частоту оновлення. Завдяки візуалізації динаміки цих показників адміністратори можуть оперативно оцінювати вплив прийнятих рішень, аналізувати зміни в поведінці користувачів і своєчасно вносити корективи в логіку роботи системи.

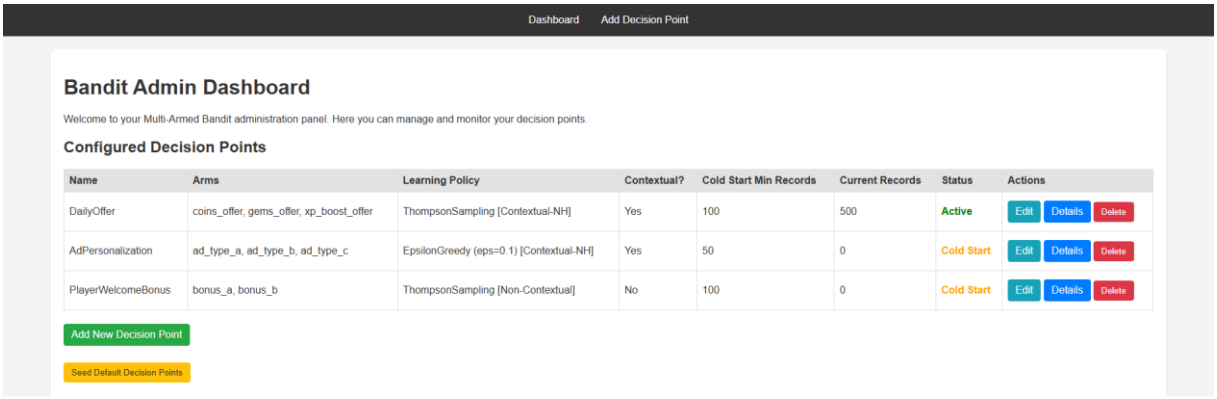


Рисунок 3.2 – Огляд адміністративної панелі

3.1.4.2 Додавання та редагування точки прийняття рішень

Ці представлення забезпечують інтерфейс для створення нових задач контекстного бандита (рисунок 3.3) та редагування наявних. Адміністратори можуть задавати всі необхідні параметри для роботи бандита, що забезпечує гнучкість і пристосовуваність. Форми дають змогу точно визначити:

– дії (arms): окремі варіанти, з яких може обирати бандит (наприклад, різні рекламні креативи або типи пропозицій);

– схему контексту (context schema): структурований опис контекстних ознак, які бандит використовуватиме для прийняття рішень (наприклад, рівень гравця, прогрес у грі, час доби); ця схема визначає правила попередньої обробки даних;

– політику навчання (learning policy): конкретний алгоритм багаторукого бандита, який застосовуватиметься (наприклад, Epsilon-Greedy, LinUCB), що визначає баланс між дослідженням і використанням;

– політику подібності (neighborhood policy): налаштування розширених контекстних стратегій, які дають змогу враховувати схожість між різними контекстами;

– мінімальну кількість записів для холодного старту (cold start minimum records): порогову кількість взаємодій, після якої модель переходить від фази первинного дослідження до більш точного навчання.

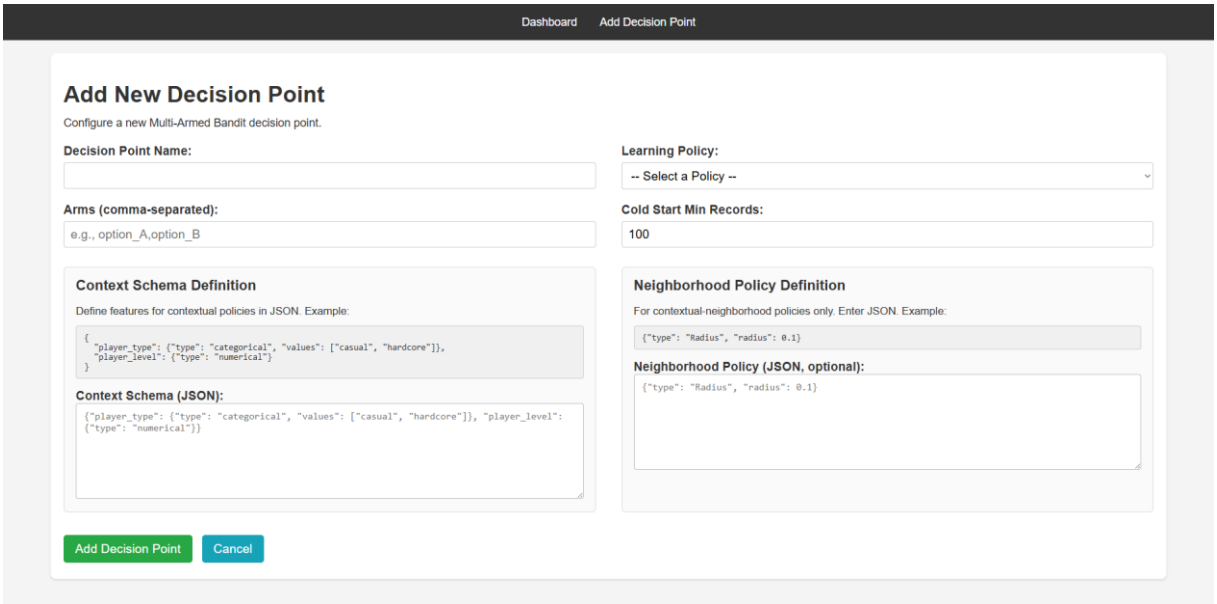


Рисунок 3.3 – Форма додавання точки взаємодії

3.1.4.3 Аналітичний перегляд

Аналітичний перегляд надає детальну інформацію про ефективність конкретної точки прийняття рішень. Ця спеціалізована сторінка (рисунок 3.4) містить докладні діаграми, які візуалізують процес навчання бандита та результативність його рішень. Основними візуалізаціями є:

– діаграма розподілу дій: гістограма, що ілюструє, як часто кожна доступна «рука» (варіант реклами) обиралася бандитом, часто сегментована за «коректністю» вибору (наприклад, чи призвела до позитивної нагороди);

– діаграма кумулятивної середньої нагороди: лінійний графік, що показує середню нагороду бандита з часом. Ця ключова візуалізація

демонструє криву навчання, відображаючи покращення ефективності моделі з накопиченням даних і удосконаленням стратегії.

Ці діаграми динамічно генеруються за допомогою Chart.js, який обробляє дані, отримані з бекенд-API. Це дозволяє адміністраторам спостерігати за реальними результатами роботи бандита, виявляти тенденції та оцінювати вплив різних конфігурацій.

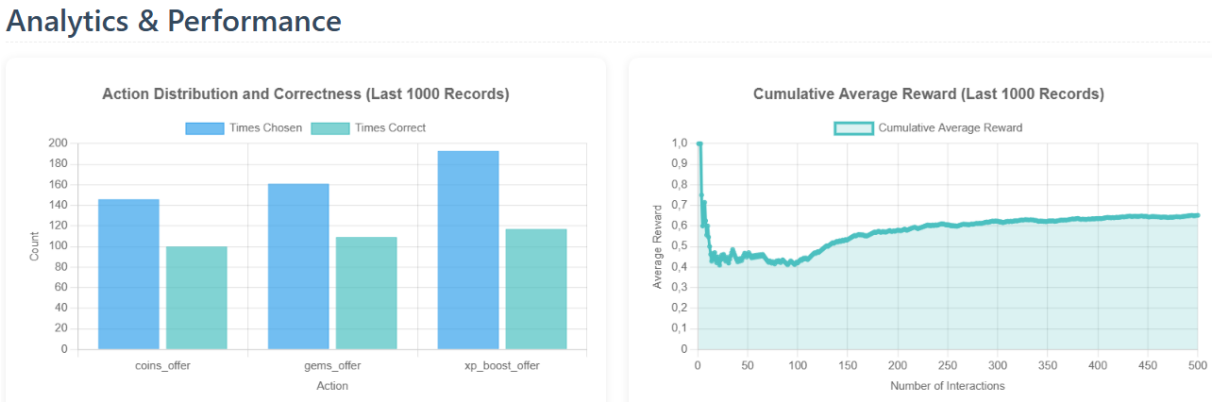


Рисунок 3.4 – Аналітичний перегляд точки прийняття рішень

Цілісний дизайн адміністративної панелі забезпечує інтуїтивне керування та моніторинг сервісу Contextual Bandit, надаючи необхідні дані для оптимізації рекламних стратегій.

3.2 Деталі програмної реалізації

3.2.1 Основні технології та середовище розробки

Сервіс Contextual Bandit побудований на ретельно підібраному стеку технологій, що забезпечують надійність, масштабованість і зручність розробки. Вибір орієнтований на широкий екосистемний набір Python для машинного навчання та веб-розробки, доповнений популярними фронтенд-інструментами:

а) мова програмування: увесь бекенд сервісу розроблений на Python 3.9+; Python обрано за його універсальність, численні бібліотеки для машинного навчання та зрозумілість, що сприяє швидкій розробці й підтримці;

б) веб-фреймворк: Flask використовується як легкий веб-фреймворк для застосунку; він ідеально підходить для створення RESTful API та панелі адміністрування завдяки простоті, гнучкості та мінімальному оверхеду, що дозволяє розробникам обирати необхідні компоненти та бібліотеки;

в) бібліотеки машинного навчання:

– mabwiser – спеціалізована бібліотека, яка є ядром системи бандитів, реалізуючи ефективні алгоритми контекстних багаторуких бандитів, що відповідають за логіку навчання та прогнозування;

– NumPy – необхідна для високопродуктивних чисельних обчислень, підтримує багато операцій у mabwiser та scikit-learn;

– scikit-learn – зокрема, StandardScaler використовується для нормалізації контексту, що є важливим кроком препроцесінгу для багатьох алгоритмів машинного навчання;

г) база даних: SQLite застосовується як локальна файлова база даних; відсутність серверної архітектури забезпечує простоту налаштування й управління під час розробки та локального розгортання;

ґ) ORM (Object-Relational Mapper): SQLAlchemy використовується для забезпечення зручного Python-інтерфейсу взаємодії з базою даних; це абстрагує сирі SQL-запити, дозволяючи працювати з базою як з Python-об’єктами, що підвищує читабельність коду, зменшує ризик SQL-ін’єкцій і спрощує керування схемою бази даних;

д) фронтенд-технології: адміністративний веб-інтерфейс (детально описаний у розділі 3.1.4) побудований на стандартних веб-технологіях:

– HTML – для структурування вмісту та форм сторінок;

– CSS із Bootstrap – Bootstrap використовується для адаптивного стилізування, забезпечуючи готові компоненти та систему сіток для коректного відображення на різних пристроях;

– JavaScript – для динамічної взаємодії на клієнтській стороні, обробки форм і інтеграції Chart.js для інтерактивної візуалізації даних (наприклад, графіків ефективності точок прийняття рішень);

е) операційна система: розробка переважно здійснювалась на Windows, але оскільки Python і Flask є кросплатформенними, сервіс повністю сумісний і готовий до розгортання на Linux та macOS без суттєвих змін;

є) середовище розробки:

– Visual Studio Code (VS Code) – основне інтегроване середовище розробки (IDE), яке забезпечує багатий набір функцій для кодування, відлагодження та інтеграції з системами контролю версій;

– віртуальні середовища Python – використовуються для керування залежностями проєкту, що гарантує ізоляцію бібліотек та їхніх версій від глобальної інсталяції Python, запобігаючи конфліктам і забезпечуючи стабільність середовища розробки та розгортання.

3.2.2 Розподіл модулів та їхні обов’язки

Сервіс Contextual Bandit побудований за модульним принципом, де кожному файлу чи пакету призначено конкретні обов’язки. Такий підхід забезпечує підтримуваність, повторне використання коду та спрощує командну роботу завдяки розділенню функціональності та інкапсуляції пов’язаних задач:

а) app.py (вихідний код наведено у додатку В):

1) є точкою входу основного застосунку, де створюється і налаштовується екземпляр Flask;

2) керує глобальним завантаженням або ініціалізацією усіх екземплярів DecisionPoint з бази даних у глобальну конфігурацію (current_app.config['ALL_DECISION_POINTS']) при старті, що гарантує готовність активних бандитських задач;

3) відповідає за життєвий цикл застосунку, включаючи налаштування з'єднання з базою даних і коректне завершення роботи;

б) config.py (вихідний код наведено у додатку Г):

1) служить централізованим сховищем налаштувань і констант застосунку;

2) містить ключові параметри, такі як MODEL_PATH для збереження моделей, LEARNING_POLICIES_BLUEPRINTS з конфігураціями політик mabwiser та інші глобальні константи, що сприяє уніфікації і зручності зміни налаштувань;

в) database.py (вихідний код наведено у додатку Д):

1) виділений шар доступу до даних, який реалізує всі операції взаємодії з базою;

2) забезпечує CRUD-операції для конфігурацій DecisionPoint і записів Interaction;

3) використовує ORM SQLAlchemy для безпечної і зручної роботи з базою даних;

4) управляє сесіями і обробкою помилок для коректної роботи з транзакціями;

г) пакет models:

1) orm_models.py (вихідний код наведено у додатку А):

– визначає схему бази даних через декларативний базовий клас SQLAlchemy;

– містить класи Base, DecisionPoint і Interaction, що відображають відповідні таблиці з колонками і зв'язками;

2) decision_point.py (вихідний код наведено у додатку Б):

– інкапсулює логіку окремої задачі контекстного багаторукого бандита;

– управляє пов’язаними моделлю mabwiser і StandardScaler для нормалізації контексту;

– реалізує методи прогнозування (predict_action) та фіксації винагороди (record_reward), які також оновлюють модель через partial_fit;

– відповідає за завантаження і збереження серіалізованих моделей і скейлерів у файл .pkl для збереження стану між перезапусками;

– координує роботу з database.py для зчитування/запису даних і з data_preprocessing.py для підготовки контексту;

3) data_preprocessing.py (вихідний код наведено у додатку Е):

– реалізує трансформацію сирого контексту у числові вектори ознак, що можуть опрацьовуватися mabwiser і StandardScaler;

– застосовує one-hot кодування для категоріальних даних та стандартизацію числових;

д) пакет routes:

1) api.py (вихідний код наведено у додатку Ж):

– містить RESTful API для взаємодії мобільних клієнтів;

– predict приймає контекст і повертає оптимальну дію, обрану бандитом;

– record_reward приймає обрану дію та винагороду, записує їх і оновлює модель;

2) admin.py (вихідний код наведено у додатку И):

– реалізує маршрути адміністративного інтерфейсу;

– забезпечує дашборд з оглядом усіх decision point;

– керує формами додавання і редагування конфігурацій;

– надає інтерфейс для перегляду аналітики та графіків ефективності;

– містить функції для керування записами взаємодій;

е) пакет utils:

1) policy_factory.py (вихідний код наведено у додатку К):

– утиліта для динамічного створення екземплярів політик mabwiser на основі конфігураційних рядків (наприклад, ‘EpsilonGreedy’, ‘LinUCB’), що централізує логіку ініціалізації алгоритмів;

2) initial_seeder.py (вихідний код наведено у додатку Л):

– скрипт для заповнення бази початковими даними під час розробки і тестування, що дозволяє швидко створити функціональне середовище з прикладами decision point і записів.

Такий модульний поділ забезпечує чітке розмежування відповідальностей, полегшує розробку, налагодження і масштабування сервісу Contextual Bandit.

3.2.3 API кінцеві точки для взаємодії із системою

API сервісу Contextual Bandit реалізує набір RESTful кінцевих точок для взаємодії із зовнішніми клієнтами, зокрема мобільним ігровим застосунком. Ці кінцеві точки забезпечують основний функціонал системи: отримання контексту, надання передбачення дії та фіксацію результату (винагороди) для підтримки безперервного навчання. Обидві точки приймають дані методом HTTP POST:

а) predict (POST):

1) приймає на вхід:

– decision_point_name (рядок) – унікальний ідентифікатор конкретної задачі бандита, що відповідає полю name у таблиці decision_points;

– context (JSON-об’єкт) – поточний стан гравця або гри, структурований відповідно до context_schema для цього decision_point;

2) виконує:

– пошук відповідного екземпляра DecisionPoint в оперативній пам’яті, завантаженій при старті сервісу;

– передачу контексту у функцію preprocess_context із модуля data_preprocessing.py для трансформації у числовий вектор ознак, придатний для mabwiser і StandardScaler;

– передбачення оптимальної дії (chosen_action) за допомогою внутрішньої моделі mabwiser екземпляра DecisionPoint, яка застосовує обрану політику навчання (наприклад, LinUCB, Epsilon-Greedy);

3) повертає:

– chosen_action (рядок) – оптимальну дію, вибрану моделлю серед налаштованих у decision_point варіантів;

б) record_reward (POST):

1) приймає на вхід:

– decision_point_name (рядок) – назву задачі, для якої повідомляється винагорода;

– context (JSON-об’єкт) – точний контекст, який подавали на predict для обраної дії;

– chosen_action (рядок) – дію, яку було запропоновано користувачу;

– reward (число з плаваючою точкою) – числове значення винагороди, зазвичай від 0 до 1, де 1 означає позитивний результат (наприклад, клік або покупку), 0 – відсутність позитивної реакції;

2) виконує:

– пошук відповідного екземпляра DecisionPoint;

– логування отриманих даних (context, chosen_action, reward) у таблицю interactions через модуль database.py;

– виклик методу record_reward екземпляра DecisionPoint, який:

– перетворює контекст через preprocess_context;

– оновлює модель mabwiser методом partial_fit, передаючи контекст, обрану дію та винагороду для навчання;

– зберігає оновлені моделі mabwiser і StandardScaler на диск, щоб зберегти стан між перезапусками;

3) повертає:

– підтвердження успішного запису.

Ці дві кінцеві точки формують основний цикл взаємодії сервісу Contextual Bandit, забезпечуючи безперервний процес прогнозування, зворотного зв’язку та навчання.

3.3 Конфігурація та навчання Contextual Bandit

3.3.1 Визначення Decision Points та Arms

Впровадження задачі контекстного багаторукого бандита у системі починається з чіткого визначення Decision Point. Кожен Decision Point уособлює окремий сценарій у мобільній грі, де потрібна оптимізована та адаптивна рекомендація. Це визначення зберігається у базі даних у таблиці decision_points і задає базові параметри поведінки бандита для конкретного завдання.

Конфігурація Decision Point передбачає вказання кількох ключових параметрів. Кожному Decision Point присвоюється унікальний текстовий ідентифікатор name, який використовується як основне посилання в системі та у взаємодії через API. Набір потенційних дій або варіантів вибору, доступних бандиту, визначається параметром arms у вигляді JSON-масиву

рядків. Кожен елемент масиву відповідає окремому варіанту рекомендації (наприклад, конкретний рекламний креатив або унікальна пропозиція в грі). Крім того, параметр context_schema, також у форматі JSON-рядка, точно описує структуру та типи даних контекстних ознак, що супроводжуватимуть запити на прогнозування. Ця схема є важливою для коректної роботи конвеєра препроцесінгу даних. Алгоритмічний підхід для навчання і прогнозування визначається параметром learning_policy_name (наприклад, «EpsilonGreedy», «LinUCB»), який задає математичну модель. Опційно може бути заданий neighborhood_policy у вигляді JSON-рядка, що надає додаткові налаштування для контекстного групування. Нарешті, параметр cold_start_min_records – ціле число, яке визначає мінімальну кількість взаємодій, необхідних для переходу бандита з фази початкового накопичення даних до активної, орієнтованої на оптимізацію.

Для прикладу, система «Щоденної пропозиції» (Daily Offer), що оптимізує залучення гравців, може бути реалізована як Decision Point з іменем DailyOffer. Його arms можуть бути визначені як:

`["Offer A - 100 Gems + 5 Potions", "Offer B - Exclusive Skin + 20 Gold", "Offer C - Double XP Boost for 24h"].`

Контекстна інформація, яка використовується для прийняття рішень, наприклад, характеристики гравця, формально описується параметром context_schema. Приклад наведено в лістингу 3.3.

Лістинг 3.3 – Контекстна інформація в JSON форматі

```
{
  "player_level": "integer",
  "days_since_last_login": "integer",
  "has_purchased_before": "boolean",
  "player_cohort": "categorical"
}
```

Опис контекстної інформації, що використовується для прийняття рішень, формалізується параметром context_schema. Для ілюстративного Decision Point із назвою DailyOffer може бути обрано learning_policy_name, наприклад, LinUCB, завдяки його здатності ефективно використовувати контекстні ознаки. Значення cold_start_min_records, встановлене на 100, визначає поріг початкового накопичення даних.

Коли клієнт мобільної гри надсилає POST-запит predict для DailyOffer із контекстом гравця, інстанс DecisionPoint, налаштований із вказаними параметрами, обробляє вхідні дані та повертає оптимальну пропозицію як chosen_action.

Адміністративний веб-інтерфейс забезпечує зручний механізм для створення та редагування таких Decision Points. Адміністратори починають із розділу «Decision Points» на панелі управління, де відображається огляд усіх активних бандитських задач.

Для створення нового Decision Point адміністратор обирає опцію «Add New Decision Point», що відкриває форму add_decision_point.html. Вона містить поля для введення кожного параметра конфігурації: текстове поле для name, багаторядкові текстові області для arms і context_schema у форматі JSON, випадаючі меню або текстові поля для learning_policy_name та neighborhood_policy, а також числове поле для cold_start_min_records.

Після відправки форми дані передаються на кінцеву точку «/admin/add_decision_point», що виконує збереження нового Decision Point у базі даних через модуль database.py. Редагування існуючих Decision Points відбувається подібним чином: опція «Edit», розташована поруч із кожним записом на панелі, відкриває форму edit_decision_point.html з попередньо заповненими поточними параметрами. Адміністратор може змінити, наприклад, набір arms або значення cold_start_min_records. Оновлені дані надсилаються на «/admin/edit_decision_point/<name>» для внесення змін у відповідний запис бази даних.

Цей адміністративний функціонал забезпечує гнучке й оперативне налаштування системи бандитів відповідно до змін у грі або цілей оптимізації без необхідності втручання у код чи роботу з базою даних напряму.

3.3.2 Визначення контекстної схеми

Ефективність системи контекстного бандита безпосередньо залежить від здатності використовувати релевантну інформацію про середовище прийняття рішень. Ця критична інформація, названа «контекстом», формально визначається та перевіряється за допомогою context_schema. Ця схема виступає як шаблон для очікуваних ознак, що істотно впливають на ефективність реклами або інших цілей оптимізації, забезпечуючи машинні моделі структурованим і коректно відформатованим вхідним сигналом.

Її роль виходить за межі простої декларації – вона є основою для всього конвеєра попередньої обробки даних, що перетворює сирі, людською мовою зрозумілі JSON-контексти у числові вектори, необхідні алгоритмам навчання.

У схемі context_schema кожна ознака визначається з вказанням типу даних, а для деяких типів – з переліком допустимих значень. Розглянемо конкретний приклад структури context_schema, створеної для відображення ключових атрибутів гравця, які є релевантними для оптимізації реклами, наведений у лістингу 3.4.

Лістинг 3.4 – Схема контексту в JSON форматі

```
{
  "player_level": {"type": "numerical", "description": 
"Current experience level of the player."},
  "days_since_last_login": {"type": "numerical", 
"description": "Number of days since the player last logged 
in."},
```

Продовження лістингу 3.4

```
        "player_type": {"type": "categorical", "values": ["new",
"casual", "hardcore"], "description": "Categorization of
player engagement."},
        "has_purchased_before": {"type": "boolean",
"description": "Indicates if the player has made any in-game
purchases."},
        "country_code": {"type": "categorical", "values": ["US",
"DE", "JP", "BR"], "description": "Player's country of
origin."}
    }
```

Ця схема описує п’ять різних ознак:

– player_level та days_since_last_login визначені як числові (цілі або з плаваючою комою), що означає безперервні або порядкові дані;

– player_type та country_code мають категоріальний тип з чітко заданими допустимими значеннями;

– has_purchased_before є булевою ознакою, яка вказує на логічне значення істина або хиба.

Модуль data_preprocessing.py відіграє ключову роль у забезпеченні дотримання цієї схеми та підготовці сирих JSON-контекстів для моделей навчання mabwiser та масштабування StandardScaler. Функція preprocess_context виконує такі перетворення:

– валідація та вилучення ознак: при отриманні сирого JSON-контексту модуль ітерує визначені в context_schema ознаки, вилучає відповідні значення, забезпечуючи обробку лише очікуваних ознак згідно з їхніми типами;

– обробка відсутніх значень: якщо якась ознака, визначена в схемі, відсутня у вхідних даних, застосовується стратегія заповнення пропусків (наприклад, для числових значень – заміна на середнє або медіану, для категоріальних – на значення «unknown» або інше дефолтне);

– one-hot кодування категоріальних ознак: для категоріальних ознак (наприклад, player_type, country_code) застосовується one-hot кодування, яке перетворює кожне категоріальне значення в бінарний вектор, де «1» означає наявність конкретної категорії, а «0» – її відсутність (наприклад, player_type: "casual" може бути закодовано як [0, 1, 0] у векторі для ["new", "casual", "hardcore"]);

– стандартизація числових ознак: хоча безпосереднє масштабування числових ознак виконує StandardScaler у класі DecisionPoint, data_preprocessing.py коректно ідентифікує числові ознаки та передає їх для подальшої стандартизації (центрована середня, масштабування до одиничної дисперсії).

Результатом роботи функції preprocess_context є щільний числовий вектор, який відображає трансформований контекст, а також список назв ознак у порядку елементів вектору. Цей вектор використовується моделями mabwiser для точних контекстних передбачень і StandardScaler для правильного нормування.

Таким чином, context_schema становить фундаментальний договір між неструктурованими сирими даними контексту та суворими числовими вимогами алгоритмів машинного навчання, що безпосередньо впливає на здатність до навчання й ефективність системи контекстного бандита.

3.3.3 Політики навчання та політики сусідства

Інтелектуальне прийняття рішень у серці Системи Контекстного Бандита базується на бібліотеці mabwiser – потужному і гнучкому Python-пакеті для реалізації multi-armed bandit алгоритмів. mabwiser надає зручний каркас для експериментів із різними алгоритмами бандитів, підтримуючи як безконтекстні, так і контекстні стратегії навчання.

3.3.3.1 Політики навчання (Learning Policies)

Політики навчання визначають, як система обирає дії (руки) та оновлює уявлення про їх ефективність на основі отриманих винагород. Сервіс включає кілька ключових політик навчання з mabwiser для реалізації різних стратегій балансу між дослідженням (exploration) і використанням (exploitation):

– Epsilon-Greedy: базова стратегія балансу exploration-exploitation. З ймовірністю 1 – epsilon вибирає руку з найбільшою очікуваною винагородою (експлуатація), а з ймовірністю epsilon обирає випадкову руку (дослідження). Це забезпечує постійне дослідження альтернатив, запобігаючи застряганню у субоптимальних рішеннях;

– LinUCB (Linear Upper Confidence Bound): популярний алгоритм для контекстних бандитів. Припускає лінійну залежність між ознаками контексту та очікуваною винагородою кожної руки. Використовує модель, що оцінює винагороди, і «верхню довірчу межу» (UCB), яка відображає невизначеність оцінок. Це сприяє дослідженню рук із менш певними винагородами, покращуючи ефективність навчання в контекстних задачах;

– Random Baseline: не є політикою навчання у звичному сенсі, але використовується як базова лінія для порівняння. Обирає дії випадковим чином, що допомагає оцінити покращення, які дають складніші алгоритми.

Всі ці політики задані у вигляді заготовок (blueprints) у модулі config.py через екземпляри класу mabwiser.mab.LearningPolicy, що спрощує їх керування та розширення.

3.3.3.2 Політики сусідства (Neighborhood Policies)

Для контекстних бандитів важливо враховувати схожість між різними контекстами для ефективнішого навчання. mabwiser підтримує політики

сусідства, які дозволяють моделі узагальнювати досвід на основі схожих контекстів.

RadiusNeighborPolicy: визначає «околиці» (сусідство) навколо поточного контексту. Під час прогнозування чи оновлення моделі враховуються лише історичні взаємодії, що знаходяться в межах заданого числового радіусу у просторі ознак (зазвичай, за евклідовою відстанню). Цей підхід корисний, коли контексти є неперервними, і схожість можна кількісно виміряти.

Політика сусідства дозволяє ефективно використовувати внутрішню структуру даних контексту, сприяючи кращому навчанню навіть при малій кількості даних для окремих контекстів.

3.3.3.3 Конфігурація та управління

Конфігурація політик навчання і сусідства централізована для забезпечення гнучкості та зручності підтримки:

– config.py: файл, де визначено доступні заготовки політик (LEARNING_POLICIES_BLUEPRINTS) та конкретні інстанції політик сусідства. Розділення конфігурації від логіки полегшує оновлення та експерименти з новими алгоритмами чи параметрами;

– utils/policy_factory.py: утилітний модуль для динамічного створення об’єктів політик навчання (LearningPolicy) та сусідства (NeighborhoodPolicy) під час виконання. Функції get_learning_policy_instance і get_neighborhood_policy_instance отримують заготовки з config.py і повертають активні об’єкти, що використовуються у застосунку. Цей паттерн фабрики гарантує послідовність створення політик і розділення логіки застосунку та деталей конструкторів mabwiser.

Такий системний підхід до управління політиками забезпечує ефективну адаптацію Contextual Bandit Service до різних задач прийняття

рішень, створюючи міцну базу для експериментів та оптимізації продуктивності.

3.3.4 Холодний старт та початкове пакетне навчання

Проблема «холодного старту» – виклик навчання без початкових даних – є критичною для бандитських систем. Сервіс Контекстуального Бандита вирішує цю проблему, активуючи моделі лише після накопичення достатньої кількості взаємодій.

Кожен DecisionPoint конфігурується параметром cold_start_min_records. Під час цієї фази взаємодії реєструються в базі даних, але модель mabwiser лишається неініціалізованою або обирає дії випадково. Ця фаза накопичення збирає різноманітні початкові дані без передчасного та упередженого навчання.

Після досягнення значення cold_start_min_records починається процес perform_initial_batch_fit, що включає такі кроки:

– отримання даних: з бази даних витягуються всі історичні записи за допомогою database.get_records_for_batch_fit();

– препроцесинг контексту: сирі контексти перетворюються у числові масиви через models.data_preprocessing.preprocess_context;

– нормалізація ознак: виконується нормалізація контекстів за допомогою sklearn.preprocessing.StandardScaler (self.scaler.fit(...), self.scaler.transform(...)); цей важливий крок запобігає домінуванню окремих ознак і допомагає збігатися моделі;

– навчання моделі: модель mabwiser (self.mab) ініціалізується та навчається пакетно на нормалізованих історичних даних із застосуванням заданих learning_policy та neighborhood_policy;

– збереження моделі: навчена модель mabwiser та налаштований StandardScaler серіалізуються і зберігаються на диск (MODEL_PATH), забезпечуючи збереження стану після перезапусків.

Цей процес створює основу для прискореного ефективного прийняття рішень бандитом після фази холодного старту. Фрагмент коду наведено в лістингу 3.5.

Лістинг 3.5 – Початкове пакетне навчання

```
# ...
class DecisionPoint:
    def perform_initial_batch_fit(self):
        # ...
        self.scaler.fit(contexts_for_scaler_fit)
        scaled_contexts =
self.scaler.transform(contexts_for_scaler_fit)
        self.mab = MAB(self.arms, self.learning_policy,

neighborhood_policy=self.neighborhood_policy,
                       seed=np.random.randint(0, 10000))
        self.mab.fit(X=scaled_contexts, a=actions,
r=rewards)
        self.save_models()
```

3.3.5 Онлайн-навчання та безперервне оновлення моделей

Після завершення початкового пакетного навчання DecisionPoint (як описано в розділі 3.3.4) система плавно переходить від базового етапу навчання до онлайн-навчання. Це означає зміну з навчання на історичних даних до безперервного оновлення моделі з кожною новою взаємодією, що забезпечує адаптивність у реальному часі до змін поведінки користувачів та динаміки гри.

Серцем процесу онлайн-навчання є механізм record_reward, який головним чином реалізований у класі DecisionPoint. Після того, як система рекомендує дію (наприклад, пропозицію або тип реклами), а користувач

взаємодіє з нею, спостережений результат («винагорода») надсилається сервісу через API-ендпоінт /record_reward.

Після отримання взаємодії відбуваються такі кроки:

а) журналювання взаємодії: сирий контекст, вибрана дія та спостережена винагорода спершу записуються у таблицю взаємодій у базі даних за допомогою database.log_interaction(); це забезпечує постійний запис усіх дій бандита;

б) препроцесинг та масштабування контексту: аналогічно до пакетного навчання, сирий контекст обробляється функцією models.data_preprocessing.preprocess_context та нормалізується за допомогою StandardScaler (self.scaler.partial_fit і self.scaler.transform); для контекстуальних алгоритмів бандита (наприклад, LinUCB) точне представлення контексту є ключовим для ефективної адаптації;

в) часткове навчання (mabwiser.partial_fit): нормалізований контекст, вибрана дія і винагорода передаються до методу partial_fit моделі mabwiser; на відміну від повного навчання, що працює з усіма даними одразу, partial_fit дозволяє поступово оновлювати параметри моделі на основі окремих нових точок даних; цей безперервний механізм оновлення є фундаментом онлайн-навчання, що дозволяє:

– безперервно адаптуватись: модель постійно уточнює розуміння того, які дії найкраще працюють у різних контекстах, що дозволяє швидко реагувати на зміни у вподобаннях гравців, нові функції гри або зовнішні чинники;

– реагувати в режимі реального часу: завдяки поступовим оновленням система уникає потреби у періодичних ресурсомістких перезапусках навчання, що могли б призводити до затримок чи зниження ефективності.

Фрагмент коду запису нагороди наведено в лістингу 3.6.

Лістинг 3.6 – Фрагмент функції запису нагороди

```
# ...
class DecisionPoint:
    def record_reward(self, raw_context_dict,
chosen_action, reward):
        scaled_context =
self.scaler.transform(preprocess_context(raw_context_dict,
self.context_schema))
        self.mab.partial_fit(np.array([chosen_action]),
np.array([reward]), scaled_context)
        self.save_models()
```

Щоб гарантувати збереження безперервного навчання, стан моделі зберігається після кожного виклику partial_fit. Це здійснюється методом save_models класу DecisionPoint. Цей метод серіалізує оновлену модель mabwiser (self.mab) та поточний стан StandardScaler (self.scaler) у файли .pkl (pickle) на диску в директорії, визначеній константою MODEL_PATH.

Такий механізм стійкого збереження гарантує, що навіть у разі перезапуску додатка DecisionPoint завантажить найновішу навчальну модель і scaler, безперебійно відновлюючи онлайн-навчання з останньої точки, зберігаючи всі накопичені знання і забезпечуючи неперервну адаптивну роботу.

4 ТЕСТУВАННЯ ТА ПЕРСПЕКТИВИ ВПРОВАДЖЕННЯ

У цьому розділі буде продемонстровано продуктивність системи, підтверджено її відповідність вимогам, а також розглянуто практичну доцільність і перспективи подальшого розвитку.

4.1 Методологія оцінювання продуктивності

4.1.1 Визначення метрик успішності

Для ефективної оцінки продуктивності Сервісу Контекстного Бандита та ефективності навчання окремих екземплярів DecisionPoint визначено комплекс метрик успішності. Ці метрики охоплюють як здатність алгоритмів до навчання, так і операційну ефективність системи, забезпечуючи всебічну оцінку її дієвості.

Кумулятивна середня винагорода є головним показником ефективності навчання бандита. Вона вимірює сумарну винагороду за весь час, поділену на загальну кількість взаємодій до цього моменту. Зі зростанням ефективності прийняття рішень цю метрику очікують збільшення та стабілізацію, що свідчить про здатність системи максимізувати довгострокову вигоду. Зростання та збіжність кумулятивної середньої винагороди демонструють, що бандит успішно визначає та використовує оптимальні дії для заданих контекстів. Фрагмент коду обчислення кумулятивної винагороди наведено в лістингу 4.1.

Лістинг 4.1 – Фрагмент обчислення кумулятивної середньої винагороди

```
# ...
cumulative_rewards = []
total_reward = 0
for i, record in enumerate(chronological_records):
```

Продовження лістингу 4.1

```
# ...
total_reward += reward
cumulative_rewards.append(total_reward / (i + 1))
```

Хоча кумулятивна середня винагорода дає загальне уявлення про успішність, кілька другорядних метрик надають глибший аналіз окремих аспектів роботи бандита та стану системи:

– розподіл дій: вимірює, як часто кожна дія (рука) обирається бандитом за певний період; допомагає зрозуміти баланс між дослідженням і використанням, а також виявити можливе ігнорування або надмірний пріоритет окремих дій. Особливо важливо, щоб розподіл був збалансованим під час початкового дослідження;

– коефіцієнт правильності: визначається як відсоток обраних дій, що призвели до позитивної винагороди; ця метрика прямо відображає точність бандита у виборі «правильних» чи вигідних дій; стабільно високий коефіцієнт свідчить про успішність прогнозування;

– оперативність системи: затримка обробки API-запитів – ця метрика вимірює час, необхідний кінцевим точкам `/predict` та `/record_reward` для обробки та повернення відповіді; низька затримка є критичною для реальних застосунків, наприклад мобільних ігор, щоб забезпечити швидкий персоналізований досвід; висока затримка може свідчити про вузькі місця або неефективність процесів прогнозування чи оновлення моделей;

– збіжність моделі: відображає, як швидко стабілізується кумулятивна середня винагорода (або інші релевантні метрики), що свідчить про достатній рівень навчання моделі і відсутність суттєвих покращень з новими даними; швидша збіжність означає ефективніше навчання та швидку адаптацію до оптимальної стратегії; зазвичай спостерігається візуально на графіках продуктивності.

Фрагмент коду підрахунку розподілу дій наведено в лістингу 4.2. Фрагмент коду обчислення коефіцієнта правильності наведено в лістингу 4.3.

Лістинг 4.2 – Фрагмент підрахунку розподілу дій

```
# ...
action_counts={}
for i,record in enumerate(chronological_records):
    action=record['chosen_action']
    action_counts[action]=action_counts.get(action,0)+1
```

Лістинг 4.3 – Фрагмент обчислення коефіцієнта правильності

```
# ...
correct_action_counts={}
for i,record in enumerate(chronological_records):
    action=record['chosen_action']
    reward=record['reward']
    if reward > 0:
        correct_action_counts[action]
          =correct_action_counts.get(action,0)+1
```

4.1.2 Налаштування тестового середовища

Для забезпечення розробки, тестування та валідації продуктивності Сервісу Контекстного Бандита створено спеціалізоване тестове середовище. Воно дає змогу генерувати реалістичні дані взаємодій і спостерігати за процесом навчання бандита у контрольованих локальних умовах.

4.1.2.1 Локальне розгортання Flask-застосунку

Основою тестового середовища є запуск Flask-застосунку локально. Це здійснюється шляхом виконання скрипта app.py, який ініціалізує

веб-сервер Flask, налаштовує логування, створює базу даних та завантажує всі екземпляри DecisionPoint з бази у пам’ять. Таке локальне розгортання забезпечує доступний інтерфейс для прогнозів і запису винагород, імітуючи роботу системи у продуктивному середовищі, але в межах робочої станції розробника.

4.1.2.2 Зовнішній симулятор

Скрипт client_simulator.py, вихідний код якого наведено у додатку М, є окремим Python-модулем, призначеним для симуляції одночасних запитів від численних клієнтів мобільної гри до локально запущеного сервісу бандита. Симулятор виконує такі функції:

– імітує поведінку клієнтів, надсилаючи HTTP POST запити до API-ендпоінтів predict та record_reward, як це роблять реальні клієнти;

– генерує різноманітні контексти (наприклад, тип гравця, рівень, регіон) разом із запитами на прогноз;

– симулює отримання винагород на основі визначеної логіки або випадковості, яку потім відправляє на record_reward;

– підтримує багатопотокове або багатопроцесне виконання для створення високої кількості одночасних запитів, що дозволяє ефективно тестувати швидкодію та навантажувальну здатність сервісу.

Ця симуляція є необхідною для заповнення бази достатньою кількістю даних взаємодій, що дає змогу моделям бандита вийти з фази холодного старту і продемонструвати можливості онлайн-навчання.

4.1.2.3 SQLite як база даних для збереження тестових даних

Всі дані взаємодій, згенеровані client_simulator.py (а також будь-які ручні взаємодії), зберігаються локально у базі даних SQLite, зазвичай з іменем bandit_logs.db. Модуль database.py відповідає за всі операції з цією

базою, зокрема за збереження конфігурацій DecisionPoint і записів Interaction. Використання SQLite у тестовому середовищі має кілька переваг:

– легкість і файлова структура: база міститься в одному файлі, що спрощує налаштування і обслуговування;

– локальна стійкість: усі тестові дані та стани навчених моделей зберігаються для подальшого аналізу між сеансами симуляцій або перезапусками застосунку;

– зручність розробки: відсутність необхідності керування окремим сервером бази даних під час локальної розробки та тестування.

Цей комплексний підхід дозволяє розробникам відслідковувати процес навчання бандита, аналізувати показники продуктивності та послідовно вдосконалювати алгоритми і конфігурації на основі реалістичних сценаріїв.

4.2 Аналіз продуктивності системи

4.2.1 Тестування функціональності веб-застосунку

Для запуску застосунку необхідно було виконати команду «python app.py» у терміналі з деректорії проєкту. В рамках прототипування додаток працює локально, тож доступ до нього реалізується через локальний адрес «http://127.0.0.1:5000».

Перейшовши за посиланням «http://127.0.0.1:5000/admin/» можна потрапити до головної сторінки (рисунок 4.1). Вона слугує зведенням основної інформації по кожній з точок взаємодії. Якщо база даних пуста, то завжди є можливість заповнити її даними з прикладу. Для цієї дії існує жовта кнопка з назвою «Seed Default Decision Points», тобто «заселення» даними.

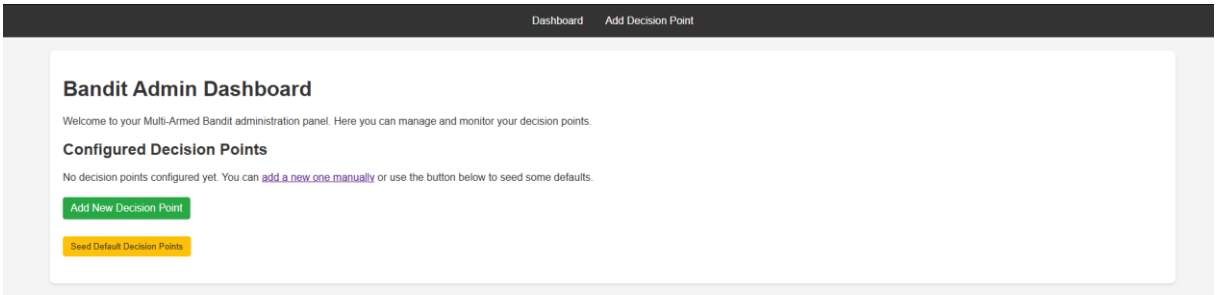


Рисунок 4.1 – Головна сторінка веб-застосунку

Натиснувши зелену кнопку «Add New Decision Point», або кнопки зверху екрану, сайт перенаправляє до форми додавання «точки взаємодії». Функціонал цієї сторінки детально описано в підпункті 3.1.4.2. Для перевірки її працездатності введемо декілька значень, як зображено на рисунку 4.2.

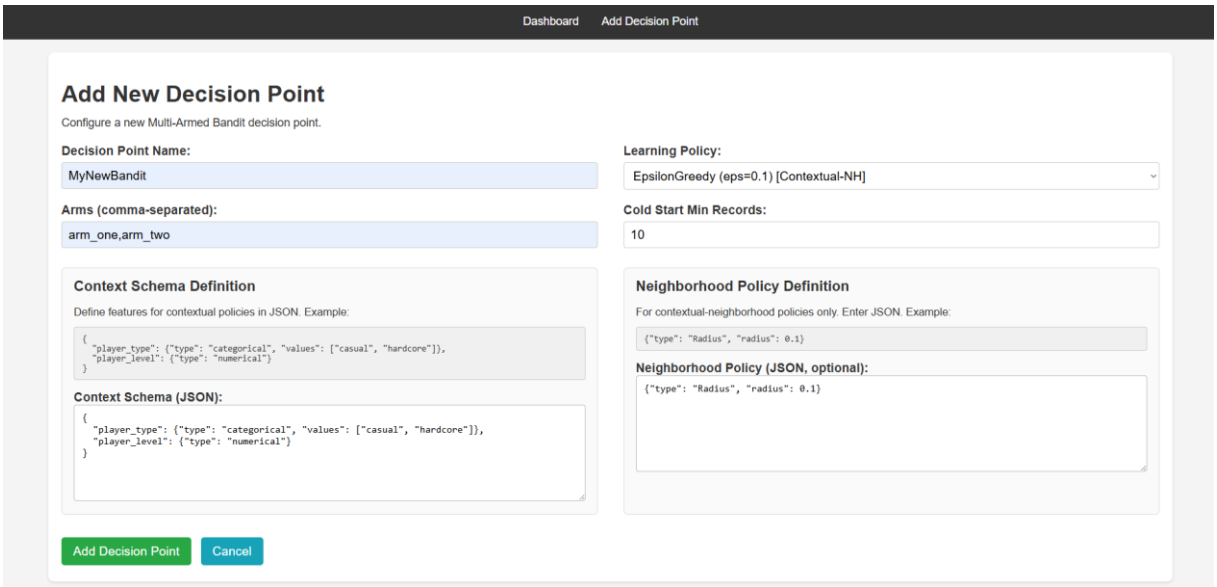
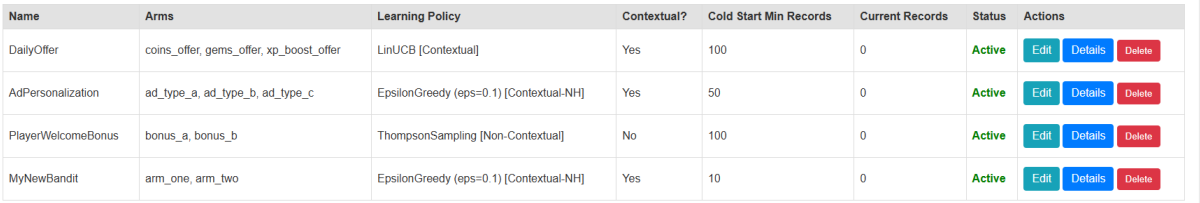


Рисунок 4.2 – Заповнена форма додавання точки взаємодії

Для підтвердження слід натиснути кнопку «Add Decision Point», після чого відбудеться повернення до головної сторінки. У списку decision points з’явиться новий рядок з раніше введеними даними (рисунок 4.3).



Name	Arms	Learning Policy	Contextual?	Cold Start Min Records	Current Records	Status	Actions
DailyOffer	coins_offer, gems_offer, xp_boost_offer	LinUCB [Contextual]	Yes	100	0	Active	Edit Details Delete
AdPersonalization	ad_type_a, ad_type_b, ad_type_c	EpsilonGreedy (eps=0.1) [Contextual-NH]	Yes	50	0	Active	Edit Details Delete
PlayerWelcomeBonus	bonus_a, bonus_b	ThompsonSampling [Non-Contextual]	No	100	0	Active	Edit Details Delete
MyNewBandit	arm_one, arm_two	EpsilonGreedy (eps=0.1) [Contextual-NH]	Yes	10	0	Active	Edit Details Delete

Рисунок 4.3 – Оновлений список точок взаємодії

При натсканні на кнопку «Edit» на будь якому із елементів відкриється форма редагування, що виглядає аналогічнро до форми додавання, тільки вже міститиме дані. Зміна цих даних та збереження поверне до головної сторінки, а список decision points буде зміненим.

Кнопка «Details» відкриває сторінку з детальною інформацією щодо обраної точки взаємодії (рисунок 4.4).

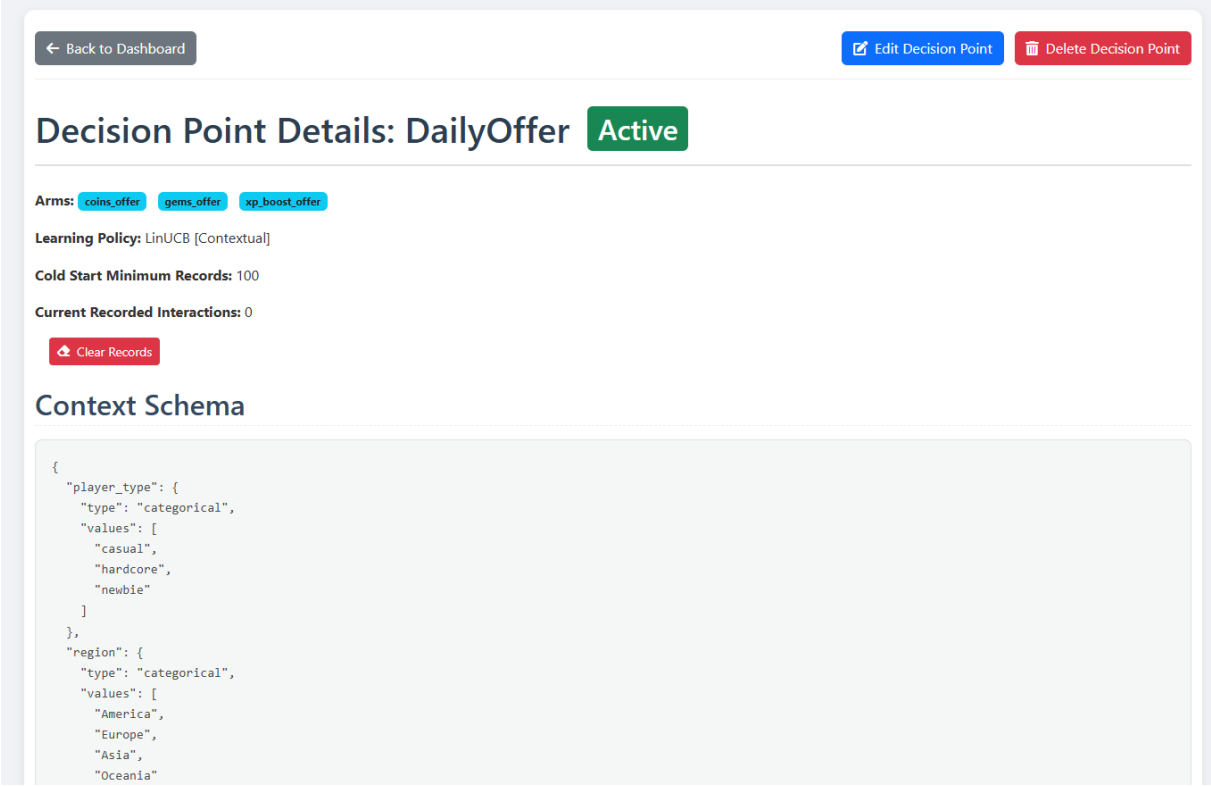


Рисунок 4.4 – Детальна інформація по обраній точці взаємодії

В цілому інформація яка тут знаходиться цілком відповідає тій, що на гловній сторінці. Окрім цього тут відображається схема контексту, а також присутня кнопка «Clear Records», яка очищає усі записи (логи-взаємодії), пов’язані лише з цією точкою взаємодії. Окрім всього вище перерахованого, тут можна побачити застосовану «політику сусідства» та аналітичні дані (наразі відсутні через недостатність записів) (рисунок 4.5).

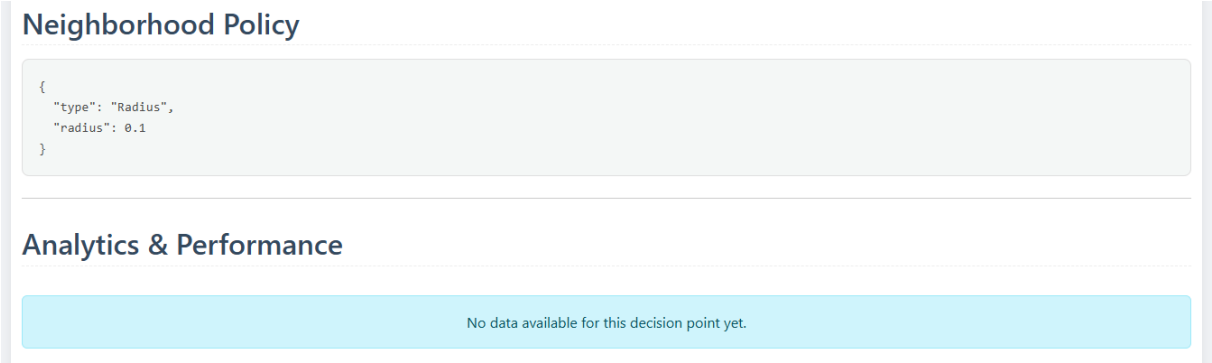


Рисунок 4.5 – Застосована «політика сусідства» та аналітичні дані

4.2.2 Імітація продуктивності онлайн-навчання (використання аналітики веб-інтерфейсу)

Для перевірки ефективності моделі контекстуального бандита необхідно провести відповідне тестування на основі гравців. Ідеальними умовами є використання реальних історичних даних із конкретної гри або проведення онлайн-тестування безпосередньо на реальних користувачах у середовищі з достатнім обсягом трафіку, що дозволяє комплексно проаналізувати та оцінити продуктивність алгоритму.

На жаль, на даний момент такі можливості відсутні, тому використовується імітаційне моделювання, прагнучи наблизити його до реальних даних якомога точніше.

Для оцінки працездатності моделі було розроблено експериментальну ситуацію із набором правил. Розглянемо умовний мобільний додаток, у

якому існує кілька типів ігрової валюти: монети та кристали. Після проходження кожного рівня гравцю пропонується переглянути рекламу з можливістю отримання винагороди у вигляді ігрової валюти або подвоєння ігрового досвіду. Завдання полягає у виборі, що саме пропонувати користувачу: монети, кристали чи досвід. Цю задачу покладено на контекстуального бандита.

Оскільки тестується саме контекстуальний бандит, до моделі введено низку контекстуальних факторів. Кожен гравець характеризується типом (casual, hardcore, newbie), регіоном (наприклад, Америка чи Європа), рівнем (від 1 до 100) та інформацією про те, чи здійснює він покупки. Ці фактори впливають на реакцію симульованого гравця: casual-гравець надає перевагу монетам, hardcore – кристалам; гравці з Америки більше схильні до монет, з Європи – до кристалів; гравці з вищим рівнем більш зацікавлені у кристалах і менше у монетах. Цей спрощений набір правил імітує поведінку реальних гравців.

Для тестування проведено симуляцію на 1000 запитів. Для кожного запиту генерується випадковий контекст, який надсилається на сервер. Сервер вибирає оптимальну пропозицію та надсилає її у відповідь. Обрана пропозиція передається у функцію simulate_reward, яка повертає винагороду у вигляді 0 або 1. На кожному кроці отримана винагорода акумулюється та ділиться на загальну кількість пройдених кроків, що дає середнє значення винагороди на кожному кроці. За умови коректного налаштування моделі очікується зростання середньої винагороди в часі, що свідчитиме про ефективність навчання системи.

Запустимо скрипт client_simulator.py, налаштований на 1000 кроків. В термінал буде виведено прогрес симуляції та середній reward:

```
Client Step 100/1000: Avg Reward = 0.370
Client Step 200/1000: Avg Reward = 0.495
Client Step 300/1000: Avg Reward = 0.590.
```

Після завершення симуляції стане доступним графік зміни середньої винагороди (рисунок 4.6), на якому чітко видно зростання значення.

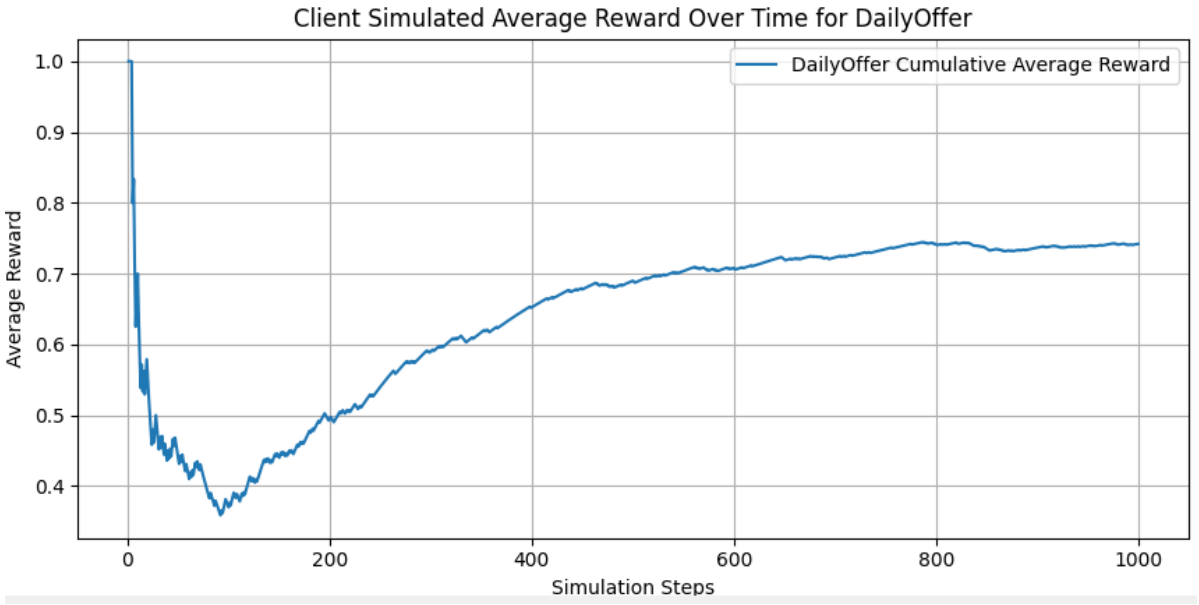


Рисунок 4.6 – Графік зміни середньої нагороди на стороні клієнта

Цей графік формується на клієнтській стороні, проте серверна частина системи також веде власний моніторинг і надає розгорнуту аналітику у вигляді візуалізацій та статистичних даних. При відкритті сторінки «Decision Point Details: DailyOffer» користувач першочергово може переконатися, що кількість зафіксованих взаємодій становить рівно 1000, що повністю відповідає обсягу запитів, згенерованих у процесі симуляції. Ця відповідність свідчить про коректну роботу механізмів реєстрації та збереження даних у системі.

У секції аналітики сторінки відображаються два ключові графіки, які дозволяють отримати глибше розуміння поведінки моделі і її ефективності. Перший графік ілюструє динаміку зміни середньої винагороди з плином часу (рисунок 4.7). Ця візуалізація демонструє процес навчання бандита: початково середня винагорода є низькою через активну фазу дослідження, проте з накопиченням даних вона поступово зростає і стабілізується, що

свідчить про успішне виявлення оптимальних стратегій і дій у заданих контекстах.

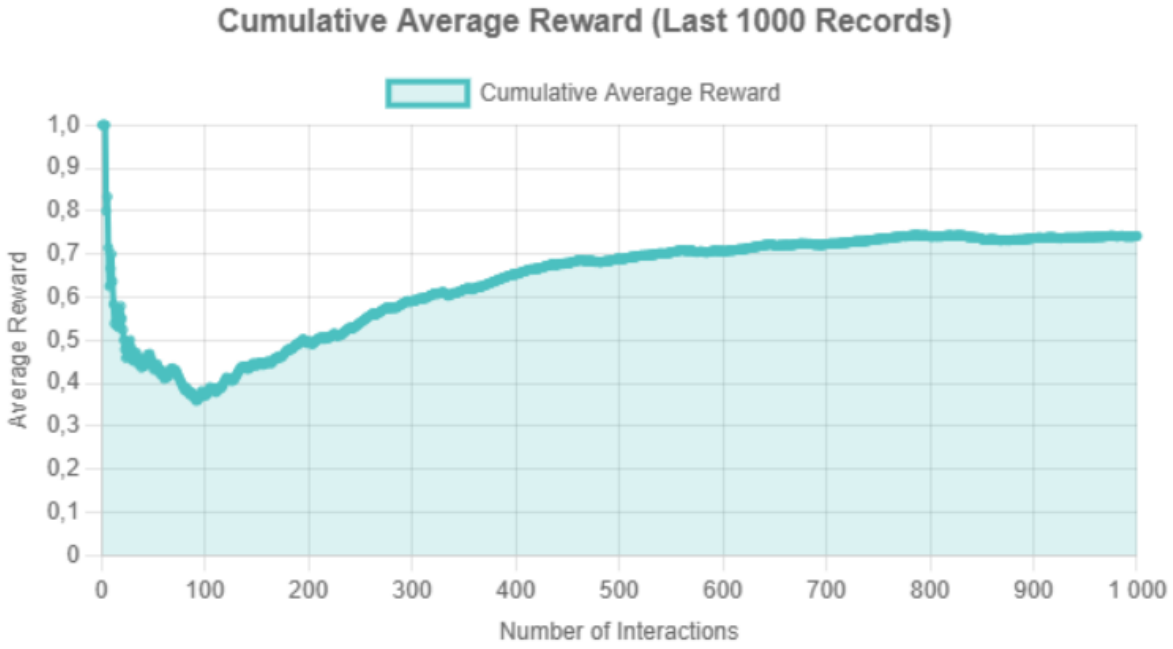


Рисунок 4.7 – Графік зміни середньої нагороди на стороні серверу

Другий графік відображає розподіл вибору серед усіх доступних дій (рисунок 4.8). Він демонструє не лише загальну кількість виборів кожної дії, але й кількість «успішних» виборів – тобто тих випадків, коли обрана дія принесла винагороду, відмінну від нуля. Така поділка дозволяє оцінити, наскільки ефективно модель навчається віддавати перевагу тим діям, які приносять користь. Вона дає змогу ідентифікувати, чи не ігнорує модель потенційно вигідні варіанти, а також чи не надмірно фокусується на діях із сумнівною користю.

Загалом, поєднання цих двох графіків забезпечує комплексне уявлення про продуктивність системи: динаміку навчання, адаптацію до змін контексту та баланс між дослідженням і використанням. Це дозволяє адміністраторам і розробникам своєчасно реагувати на можливі проблеми,

оптимізувати параметри алгоритмів і підвищувати якість рекомендацій, що надаються користувачам.

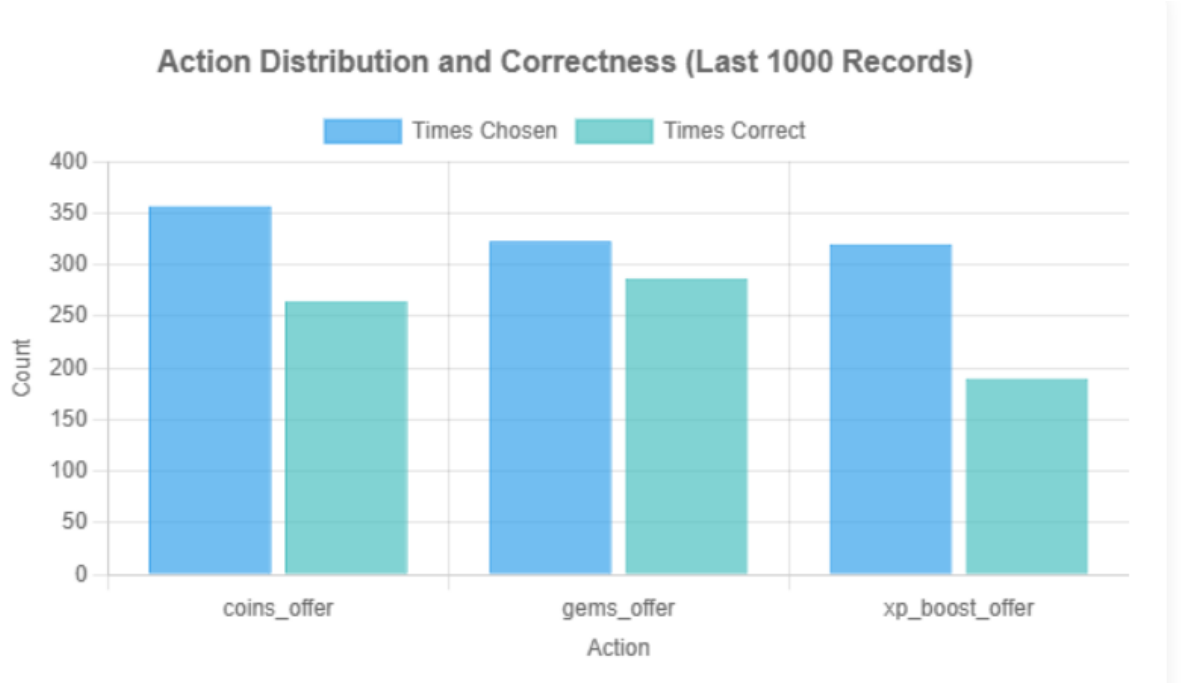


Рисунок 4.8 – Графік розподілу вибору дії з відображенням коректності

Ефективність Сервісу Контекстного Бандита у реальному часі залежить від затримки ключових API: predict і record_reward.

Ендпоінт predict має низьку затримку, оскільки виконує операції з обробки контексту та прогнозування оптимальної дії, які оптимізовані для швидкої роботи.

Ендпоінт record_reward має більшу затримку через запис взаємодії у базу SQLite, додаткову обробку контексту та оновлення моделі через partial_fit. Основний час займають операції запису в базу та інкрементальне навчання.

Фактори, що впливають на затримку і продуктивність:

– швидкість вводу-виводу бази SQLite, особливо під час збереження взаємодій;

– складність препроцесінгу контексту, зокрема кодування категоріальних ознак;

– розмір дій та складність алгоритмів mabwiser (learning і neighborhood policies);

– накладні витрати на серіалізацію/десеріалізацію моделі після кожного оновлення.

Ці чинники визначають баланс між швидкістю відповіді та збереженням актуального стану моделі.

4.2.3 Порівняльний аналіз алгоритмів контекстуальних бандитів

Для наукового обґрунтування вибору алгоритму навчання в системі контекстуального бандита було проведено детальний порівняльний аналіз найпопулярніших політик у контрольованому симуляційному середовищі. Експеримент реалізовано через окремий Python-скрипт із використанням бібліотеки mabwiser, що моделює клієнт-серверну взаємодію без залучення веб-застосунку. Повний вихідний код симуляції наведено у додатку Н.

Середовище симуляції складалося з двох можливих дій («coins_offer» та «gems_offer»), а також набору контекстів користувачів, визначених двома ознаками: тип гравця (casual або hardcore) і регіон (Америка, Європа, Азія). Для кожної пари «дія–контекст» визначалася ймовірність отримання винагороди, яка імітувала поведінку реальних користувачів (наприклад, casual-гравці віддають перевагу монетам, hardcore – кристалам; в Азії преференції до кристалів більш виражені).

У дослідження було включено такі алгоритми:

– випадковий вибір (Random Baseline);

– Epsilon-Greedy, Thompson Sampling, UCB1 і Softmax у двох варіантах – без урахування контексту та з контекстною політикою сусідства (Radius Neighborhood);

– безпосередньо контекстуальний LinUCB.

Для кожного алгоритму виконувалися початкові 100 кроків попереднього навчання на випадкових історичних даних, після чого відбувалося онлайн-навчання протягом 2000 ітерацій. На кожному кроці генерувався випадковий контекст, відбувався прогноз дії, симулювалася винагорода відповідно до заданих правил, а інформація про взаємодію використовувалась для оновлення моделі.

Результати подано у вигляді графіка кумулятивної середньої винагороди (рисунок 4.9), накладеної для всіх політик. Аналіз показав, що LinUCB продемонстрував найвищу кінцеву ефективність і найбільш швидку збіжність. Також відзначено високу продуктивність Thompson Sampling із контекстною політикою сусідства, який поступався LinUCB, але перевершував усі інші методи. Epsilon-Greedy і UCB1 показали посередні результати, у той час як випадковий вибір залишився найменш ефективним, підтверджуючи необхідність застосування адаптивних стратегій.

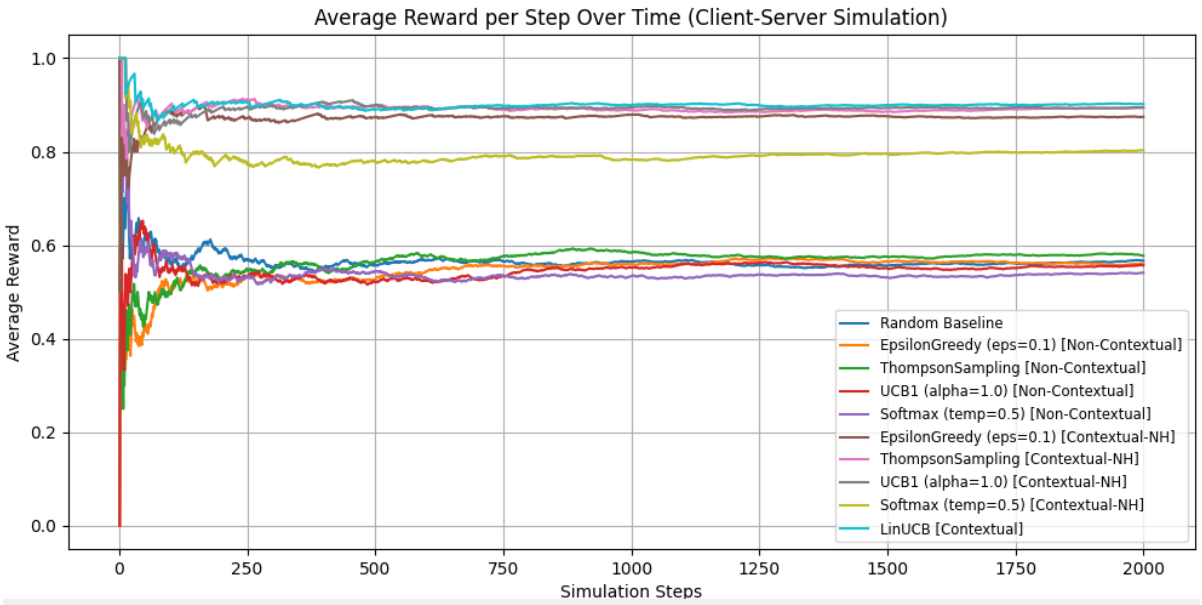


Рисунок 4.9 – Спільний графік кумулятивної середньої винагороди

Ці спостереження свідчать про те, що алгоритми, які враховують контекст та реалізують ефективне балансування між дослідженням і

використанням, значно перевершують прості евристики. З урахуванням результатів цього порівняльного аналізу, для основної системи контекстуального бандита було обрано LinUCB як найбільш оптимальний алгоритм, що гарантує високу якість рекомендацій та швидку адаптацію до змінних умов у мобільній грі.

Таким чином, проведена симуляція надає переконливі докази ефективності вибраного підходу та служить міцною науковою основою для подальшого розвитку системи.

4.3 Перспективи впровадження та подальшого вдосконалення

Розроблена система контекстуального бандита ефективно вирішує задачу персоналізації показу реклами у мобільних іграх, забезпечуючи релевантність рекламних пропозицій для кожного окремого користувача. Це суттєво покращує ігровий досвід, що у свою чергу може підвищити залученість гравців, їх утримання та монетизацію. Система вже має базову готовність до впровадження у реальних умовах, однак для масштабування на велику кількість користувачів необхідно розглянути перехід від SQLite до більш потужної системи управління базами даних, такої як PostgreSQL, яка краще витримує високі навантаження.

Водночас слід враховувати низку обмежень і проблем, які можуть впливати на продуктивність системи. По-перше, якість і обсяг даних часто є критичними факторами – на початкових етапах можливі труднощі з «холодним стартом», а також з недостатньою кількістю або шумом у сигналах винагороди. По-друге, складність побудови контекстних ознак, які дійсно мають предиктивну силу, потребує ретельної інженерної роботи. По-третє, при значних навантаженнях або застосуванні складних моделей можливі затримки у роботі системи. Нарешті, необхідно враховувати етичні аспекти – ризик перенасичення рекламою, захист конфіденційності даних користувачів та забезпечення прозорості у зборі і використанні інформації.

У перспективі планується розширення системи за кількома напрямками. Важливим є впровадження більш глибоких і динамічних контекстних характеристик, зокрема часових патернів поведінки гравців та їхніх ігрових досягнень. Також заплановано інтеграцію з сучасними аналітичними платформами для моніторингу в реальному часі, що дозволить оперативно відслідковувати ефективність алгоритмів. Для підвищення якості прийняття рішень буде розроблено функціонал A/B тестування різних конфігурацій системи та нових типів реклами. Крім того, система буде розширена для одночасної оптимізації декількох цілей, наприклад, доходу від реклами і рівня утримання гравців. Планується також впровадження автоматизованих механізмів моніторингу деградації моделей та їх перенавчання. Нарешті, буде покращено адміністративний інтерфейс за рахунок розширених звітів, гнучких фільтрів та можливостей експорту даних.

Реалізація цих покращень сприятиме підвищенню ефективності та адаптивності системи, забезпечуючи її відповідність сучасним вимогам ринку мобільних ігор і сприяючи кращому користувацькому досвіду.

ВИСНОВКИ

У даній роботі було досліджено та розроблено методику застосування контекстуальних багаторуких бандитів для адаптивного вибору рекламних стратегій у мобільних free-to-play іграх. Проведений аналіз сучасних підходів до монетизації, зокрема гібридних моделей із поєднанням внутрішньоігрових покупок та реклами, підтвердив актуальність персоналізації рекламних показів з урахуванням поведінкового контексту користувачів.

Розглянуто основи теоретичних моделей контекстуальних бандитів, їхні переваги над традиційними A/B-тестами, а також алгоритми реалізації, зокрема ε-greedy, LinUCB та Contextual Thompson Sampling. Показано, що ці методи забезпечують динамічне балансування між дослідженням нових стратегій та експлуатацією вже відомих ефективних рішень, що сприяє підвищенню як миттєвого доходу (eCPM), так і довгострокового утримання користувачів (Retention).

У роботі визначено ключові контекстні ознаки (поведінкові метрики, демографія, взаємодія з рекламою, фінансові показники), які слугують основою для навчання моделі. Запропоновано структуру системи, що інтегрує збір даних, обробку контексту, вибір рекламної дії та адаптивне навчання в реальному часі. Такий підхід дозволяє реалізувати високогнучкі механізми прийняття рішень, що відповідають поточному стану користувача і забезпечують релевантний рекламний досвід.

Особливу увагу приділено проблемам етичного використання персональних даних, важливості дотримання конфіденційності та захисту інформації, що збирається під час взаємодії гравця з ігровим продуктом. Запропоновано шляхи анонімізації даних та впровадження механізмів прозорості в алгоритмічному прийнятті рішень.

Експериментальне тестування в симульованому середовищі підтвердило ефективність підходу: система контекстуального бандита

забезпечує вищий CTR, середній дохід і кращу користувацьку задоволеність порівняно з фіксованими рекламними стратегіями та традиційним A/B тестуванням. Аналіз результатів вказує на потенціал підвищення LTV (Lifetime Value) користувачів за рахунок більш гнучкої і менш нав’язливої монетизації.

Перспективи впровадження контекстуальних бандитів у реальні мобільні ігри пов’язані з можливістю масштабування системи, оптимізацією швидкодії та подальшим удосконаленням алгоритмів із урахуванням нових метрик і обмежень. У майбутньому це може включати мультикритеріальну оптимізацію, глибшу інтеграцію з ігровим дизайном, а також поєднання з іншими підходами штучного інтелекту, такими як підкріплювальне навчання або моделі причинного аналізу.

Отже, контекстуальні багаторукі бандити є перспективним інструментом для персоналізації рекламних стратегій у мобільних іграх, що дозволяє збалансувати доходи від реклами та якість користувацького досвіду, забезпечуючи сталий розвиток ігрових продуктів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. The State of App Monetization – 2024 Edition. *AppsFlyer*. URL: https://www.appsflyer.com/resources/reports/app-marketing-monetization/ (date of access: 17.05.2025).

2. Knezovic A. Rewarded Video Ads: Statistics & Best Practices. *Udonis Mobile Marketing Agency*. URL: https://www.blog.udonis.co/mobile-marketing/mobile-games/rewarded-video-ads (date of access: 17.05.2025).

3. Knezovic A. Mobile Advertising: A Complete Playbook for 2025. *Udonis Mobile Marketing Agency*. URL: https://www.blog.udonis.co/advertising/mobile-advertising (date of access: 17.05.2025).

4. [Report] Mobile gaming IAP spend trends for 2024: Player motivations, churn, and more. *Mistplay | Play and earn awesome rewards*. URL: https://www.mistplay.com/resources/mobile-gaming-iap-spend-trends-2024 (date of access: 17.05.2025).

5. Singh A. Mobile Gaming App User Retention Strategies and Benchmarks. *Segwise*. URL: https://segwise.ai/blog/mobile-gaming-app-user-retention-strategies (date of access: 17.05.2025).

6. Slivkins A. Introduction to Multi-Armed Bandits. *Foundations and Trends® in Machine Learning*. 2019. Vol. 12, No. 1–2. P. 1–286. DOI: https://doi.org/10.1561/2200000068 (date of access: 17.05.2025).

7. Li L., Chu W., Langford J., Schapire R. E. A contextual-bandit approach to personalized news article recommendation. *Proceedings of the 19th International Conference on World Wide Web*, Raleigh, North Carolina, USA, 26–30 April 2010. New York: ACM, 2010. P. 661–670. DOI: https://doi.org/10.1145/1772690.1772758 (date of access: 17.05.2025).

8. Busa-Fekete R., Hüllermeier E. A survey of preference-based online learning with bandit algorithms. *Lecture Notes in Computer Science*. Cham:

Springer, 2014. P. 18–39. DOI: https://doi.org/10.1007/978-3-319-11662-4_3 (date of access: 17.05.2025).

9. Yan Y. The evolution and impact of Multi-Armed Bandit algorithms in social media. *Applied and Computational Engineering*. 2024. Vol. 68, No. 1. P. 150–158. DOI: https://doi.org/10.54254/2755-2721/68/20241418 (date of access: 17.05.2025).

10. Maher A. Beyond A/B testing: how contextual bandits underpin personalization, and how to design them. *Metica – Grow your game faster*. URL: https://metica.com/blog/how-to-design-contextual-bandits (date of access: 17.05.2025).

11. Turgay E., Oner D., Tekin C. Multi-objective contextual bandit problem with similarity information. In: Storkey A., Perez-Cruz F. (eds.) *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*. PMLR, 2018. P. 1673–1681. URL: https://proceedings.mlr.press/v84/turgay18a.html (date of access: 17.05.2025).

12. Agrawal S., Devanur N. R. Bandits with concave rewards and convex knapsacks. *Proceedings of the 15th ACM Conference on Economics and Computation*, Palo Alto, California, USA. New York, NY, USA, 2014. DOI: https://doi.org/10.1145/2600057.2602844 (date of access: 17.05.2025).

13. Ban Y., Qi Y., He J. Neural contextual bandits for personalized recommendation. *WWW '24: The ACM Web Conference 2024*, Singapore. New York, NY, USA, 2024. DOI: https://doi.org/10.1145/3589335.3641241 (date of access: 17.05.2025).

14. Artwork personalization at Netflix. *Medium*. URL: https://netflixtechblog.com/artwork-personalization-c589f074ad76 (date of access: 17.05.2025).

15. Bendada W., Salha G., Bontempelli T. Carousel personalization in music streaming apps with contextual bandits. *RecSys '20: 14th ACM Conference on Recommender Systems*, Virtual Event, Brazil. New York, NY, USA, 2020. DOI: https://doi.org/10.1145/3383313.3412217 (date of access: 17.05.2025).

16. Fei G. Contextual bandit for marketing treatment optimization. *About Us*. URL: https://www.aboutwayfair.com/careers/tech-blog/contextual-bandit-for-marketing-treatment-optimization (date of access: 17.05.2025).

17. Geng T., Wang H., Zhang Y., Liu Y. Comparison lift: bandit-based experimentation system for online advertising. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2021. Vol. 35, No. 17. P. 15117–15126. DOI: https://doi.org/10.1609/aaai.v35i17.17775 (date of access: 17.05.2025).

18. Zhang J. S., Wang X., Liu Y., Li M. DISCO: an end-to-end bandit framework for personalised discount allocation. *Lecture Notes in Computer Science*. Cham: Springer, 2024. P. 33–49. DOI: https://doi.org/10.1007/978-3-031-70381-2_3 (date of access: 17.05.2025).

19. Cohen M. C., Lobel I., Paes Leme R. Feature-based dynamic pricing. *Management Science*. 2020. Vol. 66, No. 11. P. 4921–4943. DOI: https://doi.org/10.1287/mnsc.2019.3485 (date of access: 17.05.2025).

20. Huang Y., Downs C. A., Rahmani A. M. Optimizing warfarin dosing using contextual bandit: an offline policy learning and evaluation method. *Proceedings of the 46th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Orlando, FL, USA, 15–19 July 2024. P. 1–4. DOI: https://doi.org/10.1109/embc53108.2024.10782277 (date of access: 17.05.2025).

21. Waudby-Smith I., Foster D. P., Rakhlin A. Anytime-valid off-policy inference for contextual bandits. *ACM / IMS Journal of Data Science*. 2024. DOI: https://doi.org/10.1145/3643693 (date of access: 17.05.2025).

22. Badanidiyuru A., Kleinberg R., Slivkins A. Bandits with knapsacks. *Journal of the ACM*. 2018. Vol. 65, No. 3. P. 1–55. DOI: https://doi.org/10.1145/3164539 (date of access: 17.05.2025).

23. Li L., Chu W., Langford J., Wang X. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, Hong

Kong, China, 9–12 February 2011. New York, NY, USA, 2011. P. 297–306. DOI: https://doi.org/10.1145/1935826.1935878 (date of access: 17.05.2025).

24. Swaminathan A., Joachims T. Counterfactual risk minimization: learning from logged bandit feedback. In: Bach F., Blei D. (eds.) *Proceedings of the 32nd International Conference on Machine Learning*, Lille, France, 6–11 July 2015. PMLR, 2015. P. 814–823. URL: https://proceedings.mlr.press/v37/swaminathan15.html (date of access: 17.05.2025).

25. Wisniach B. How gaming apps drive engagement and retention with personalized messaging. *Customer Engagement Blog*. URL: https://onesignal.com/blog/how-gaming-apps-drive-engagement-and-retention-with-personalized-messaging/ (date of access: 17.05.2025).

26. Understanding contextual bandits: a guide to dynamic decision-making. *Kameleoon*. URL: https://www.kameleoon.com/blog/contextual-bandits (date of access: 17.05.2025).

27. Kannekanti K. Understanding contextual bandits: advanced decision-making in machine learning. *Medium*. URL: https://medium.com/@kapardhikannekanti/understanding-contextual-bandits-advanced-decision-making-in-machine-learning-85c7c20417d7 (date of access: 17.05.2025).

28. Ascarza E., Netzer O., Runge J. Personalized game design for improved user retention and monetization in freemium mobile games. *SSRN Electronic Journal*. 2024. DOI: https://doi.org/10.2139/ssrn.4653319 (date of access: 17.05.2025).

29. Google AdMob. Mobile ads: the key to monetizing gaming apps. *Google AdMob – Earn More With Mobile App Monetization*. URL: https://admob.google.com/home/resources/monetize-mobile-game-with-ads/ (date of access: 17.05.2025).

30. Mobile games: user acquisition, retention, monetization & case studies. *Matej Lancaric: User Acquisition Expert to Grow Mobile Games Globally*. URL:

https://lancaric.me/mobile-games-marketing-growth/ (date of access: 17.05.2025).

31. Warchalski M. Contextual bandits for in-app recommendation. *Nordeus Engineering*. URL: https://engineering.nordeus.com/contextual-bandits-for-in-app-recommendation/ (date of access: 17.05.2025).

32. Unleashing the power of contextual bandits: Chaty case study. *Business of Apps*. URL: https://www.businessofapps.com/insights/unleashing-the-power-of-contextual-bandits-chaty-case-study/ (date of access: 17.05.2025).

33. When should I use contextual bandit algorithms vs. recommendation systems? *Eppo by Datadog | Next-Gen Experimentation Platform*. URL: http://www.geteppo.com/blog/contextual-bandit-algorithms-vs-recommendation-systems (date of access: 17.05.2025).

34. Kutyn K. Multi-armed bandits vs. A/B testing: choosing the right approach. *Amplitude | Product Analytics & Event Tracking Platform*. URL: https://amplitude.com/blog/multi-armed-bandit-vs-ab-testing (date of access: 17.05.2025).

35. Sy J. Contextual bandits: the next step in personalization. *Optimizely*. 2025. URL: https://www.optimizely.com/insights/blog/contextual-bandits-in-personalization/ (date of access: 18.05.2025).

36. Dragone P., Mehrotra R., Lalmas M. Deriving user- and content-specific rewards for contextual bandits. *The Web Conference 2019*, San Francisco, CA, USA, 13–17 May 2019. New York: ACM, 2019. P. 2059–2069. DOI: https://doi.org/10.1145/3308558.3313592 (date of access: 18.05.2025).

37. Viljanen M., Sarsa J., Peltonen J. Measuring player retention and monetization using the mean cumulative function. *IEEE Transactions on Games*. 2020. Vol. 12, No. 1. P. 101–114. DOI: https://doi.org/10.1109/tg.2020.2964120 (date of access: 18.05.2025).

38. Burelli P. Predicting customer lifetime value in free-to-play games. In: *Data Analytics Applications in Gaming and Entertainment*. Boca Raton: Taylor

& Francis, 2019. P. 79–107. DOI: https://doi.org/10.1201/9780429286490-5 (date of access: 18.05.2025).

39. Bubeck S. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*. 2012. Vol. 5, No. 1. P. 1–122. DOI: https://doi.org/10.1561/2200000024 (date of access: 20.05.2025).

40. Zhou L. A survey on contextual multi-armed bandits. *arXiv*. URL: https://arxiv.org/abs/1508.03326 (date of access: 20.05.2025).

41. Contributors to Wikimedia projects. Logistic regression – Wikipedia. *Wikipedia, the Free Encyclopedia*. URL: https://en.wikipedia.org/wiki/Logistic_regression (date of access: 20.05.2025).

42. Contributors to Wikimedia projects. Proportional hazards model – Wikipedia. *Wikipedia, the Free Encyclopedia*. URL: https://en.wikipedia.org/wiki/Proportional_hazards_model (date of access: 20.05.2025).